macromedia®

FLASH® 8

Referencia del lenguaje ActionScript 2.0

Agradecimientos

Dirección del proyecto: JuLee Burdekin

Redactores jefe: Francis Cheng, Robert Dixon, Shimul Rahim

Otros redactores: Jen deHaan, Thais Derich, Guy Haas, David Jacowitz, Jeff Swartz

Desarrolladores de muestras: Luke Bayes, Francis Cheng, Robert Dixon, Ali Mills, Jeff Swartz

Edición: Linda Adler, Geta Carlson, Evelyn Eldridge, John Hammett, Noreen Maher, Mark Nigara, Lisa Stanziano, Anne Szabla, Jessie Wood

Dirección de la producción: Patrice O'Neill

Producción y diseño multimedia: Adam Barnett, John Francis, Brett Jarvis, Mario Reynoso

Agradecimientos especiales: Peter deHaan, Gary Grossman, Lee Thomason ,y a todo el equipo de Flash Player

Primera edición: Septiembre de 2005

Macromedia, Inc.
601 Townsend St.
San Francisco, CA 94103, EE. UU.

Contenido

CAPÍTULO 1

Elementos del lenguaje ActionScript

1

En esta sección se proporciona la sintaxis, información de uso y ejemplos de código de funciones globales y propiedades (los elementos que no pertenecen a ninguna clase de ActionScript); directivas del compilador y constantes, operadores, sentencias y palabras clave utilizadas en ActionScript y definidas en el borrador de la especificación del lenguaje ECMAScript (ECMA-262) edición 4.

Directivas del compilador

Esta sección contiene las directivas que deben incluirse en el archivo de ActionScript para dirigir al compilador en el preprocesamiento de determinadas instrucciones. No inserte un punto y coma (;) al final de la línea que contiene la directiva.

Resumen de directivas del compilador

Directiva	Descripción
`#endinitclip`	Indica el final de un bloque de acciones de inicialización.
`#include`	Incluye el contenido del archivo especificado, como si los comandos del archivo formaran parte del script que realiza la llamada.
`#initclip`	Indica el principio de un bloque de acciones de inicialización.

Directiva #endinitclip

```
#endinitclip
```

Indica el final de un bloque de acciones de inicialización.

No inserte un punto y coma (;) al final de la línea que contiene la directiva #endinitclip.

Disponibilidad: Flash Player 6.0; ActionScript 1.0

Ejemplo

```
#initclip
...initialization actions go here...
#endinitclip
```

Directiva #include

```
#include "[path]filename.as":String
```

Incluye el contenido del archivo especificado, como si los comandos del archivo formaran parte del script que realiza la llamada. La directiva `#include` se invoca durante la compilación. Por lo tanto, si realiza modificaciones en un archivo externo deberá guardarlo y volver a compilar todos los archivos FLA que lo utilizan.

Si utiliza el botón Revisar sintaxis en un script que contiene sentencias `#include`, también se revisará la sintaxis de los archivos incluidos.

Puede utilizar `#include` en archivos FLA y en archivos de script externos, pero no en archivos de clase de ActionScript 2.0.

Puede elegir entre no especificar ninguna ruta, especificar una ruta relativa o especificar una ruta absoluta del archivo que debe incluirse. Si no especifica ninguna ruta, el archivo AS debe encontrarse en una de las siguientes ubicaciones:

- El mismo directorio que el archivo FLA. El mismo directorio que el script que contiene la sentencia `#include`
- El directorio Include global, cuya ruta puede ser:

  --Windows 2000 o Windows XP: C:\Documents and Settings*usuario* \Local Settings\ Datos de programa\Macromedia\Flash 8*idioma*\Configuration\Include

  --Macintosh OS X: Disco duro/Users/Library/Application Support/Macromedia/Flash 8/*idioma*/Configuration/Include
- El directorio *Programa Flash 8**idioma*\First Run\Include; si se guarda aquí un archivo, se copiará al directorio Include global la próxima vez que se inicie Flash.

Para especificar una ruta relativa del archivo AS, utilice un punto (.) para indicar el directorio actual, dos puntos (..) para indicar un directorio principal y barras diagonales (/) para indicar subdirectorios. Lea la siguiente sección de ejemplo.

Para especificar una ruta absoluta del archivo AS, utilice el formato compatible con su plataforma (Macintosh o Windows). Lea la siguiente sección de ejemplo. (No se recomienda este uso porque requiere tener la misma estructura de directorios en cualquier equipo que se utilice para compilar el script.)

Si coloca archivos en el directorio First Run/Include o en el directorio Include global, realice una copia de seguridad de dichos archivos. Si alguna vez necesita desinstalar Flash y volver a instalarlo, es posible que se eliminen y sobrescriban estos directorios.

No inserte un punto y coma (;) al final de la línea que contiene la directiva #include.

Disponibilidad: Flash Player 4.0; ActionScript 1.0

Parámetros

[path]filename.as:String - *filename.as*El nombre de archivo y la ruta opcional del script que se añadirán al panel Acciones o al script actual; *.as* es la extensión de nombre de archivo recomendada.

Ejemplo

Los ejemplos siguientes muestran varias formas de especificar una ruta para un archivo que se va a incluir en el script:

```
// Note that #include statements do not end with a semicolon (;)
// AS file is in same directory as FLA file or script
// or is in the global Include directory or the First Run/Include directory
#include "init_script.as"

// AS file is in a subdirectory of one of the above directories
// The subdirectory is named "FLA_includes"
#include "FLA_includes/init_script.as"
// AS file is in a subdirectory of the script file directory
// The subdirectory is named "SCRIPT_includes"
#include "SCRIPT_includes/init_script.as"
// AS file is in a directory at the same level as one of the above
  directories
// AS file is in a directory at the same level as the directory
// that contains the script file
// The directory is named "ALL_includes"
#include "../ALL_includes/init_script.as"

// AS file is specified by an absolute path in Windows
// Note use of forward slashes, not backslashes
#include "C:/Flash_scripts/init_script.as"

// AS file is specified by an absolute path on Macintosh
#include "Mac HD:Flash_scripts:init_script.as"
```

Directiva #initclip

```
#initclip [order:Number]
```

Indica el principio de un bloque de acciones de inicialización. Cuando se inicializan varios clips simultáneamente, se puede utilizar el parámetro `order` para especificar la inicialización que se ejecuta en primer lugar. Las acciones de inicialización se ejecutan cuando se define un símbolo de clip de película. Si el clip de película es un símbolo exportado, las acciones de inicialización se ejecutan antes que la acciones en el fotograma 1 del archivo SWF. De lo contrario, se ejecutan inmediatamente antes que las acciones del fotograma que contiene la primera instancia del símbolo de clip de película asociado.

Las acciones de inicialización sólo se ejecutan una vez que se reproduce un archivo SWF. Utilícelas para inicializaciones que se realizan una vez, como definición de clases y registro.

No inserte un punto y coma (;) al final de la línea que contiene la directiva #initclip.

Disponibilidad: Flash Player 6.0; ActionScript 1.0

Parámetros

order:`Number` [opcional] - Un entero no negativo que especifica el orden de ejecución de bloques de código `#initclip`. Este parámetro es opcional. Debe especificar el valor utilizando un literal entero (sólo se permiten valores decimales, no hexadecimales) y sin utilizar variables. Si incluye varios bloques `#initclip` en un solo símbolo de clip de película, el compilador utiliza el último valor `order` especificado en ese símbolo de clip de película para todos los bloques `#initclip` del mismo.

Ejemplo

En el ejemplo siguiente, se inserta código ActionScript en el fotograma 1 de una instancia de clip de película. En el mismo directorio se inserta el archivo de texto variables.txt.

```
#initclip

trace("initializing app");

var variables:LoadVars = new LoadVars();

variables.load("variables.txt");

variables.onLoad = function(success:Boolean) {

 trace("variables loaded:"+success);
```

```
 if (success) {
 for (i in variables) {
 trace("variables."+i+" = "+variables[i]);
 }
 }
};

#endinitclip
```

Constantes

Una constante es una variable que se utiliza para representar una propiedad cuyo valor no cambia nunca. En esta sección se describen las constantes globales disponibles en cada script.

Resumen de constantes

Modificadores	Constante	Descripción
	`false`	Valor booleano exclusivo que representa lo contrario de `true`.
	`Infinity`	Especifica el valor IEEE-754 que representa el infinito positivo.
	`-Infinity`	Especifica el valor IEEE-754 que representa el infinito negativo.
	`NaN`	Variable predefinida con el valor IEEE-754 para NaN (no es un número).
	`newline`	Inserta un carácter de retorno de carro (`\r`) que genera una línea en blanco en la salida de texto que genera el código.
	`null`	Valor especial que puede asignarse a las variables o ser devuelto por una función si no se ha proporcionado ningún dato.
	`true`	Valor booleano exclusivo que representa lo contrario de `false`.
	`undefined`	Valor especial que normalmente se utiliza para indicar que todavía no se ha asignado un valor a una variable.

Contante false

Valor booleano exclusivo que representa lo contrario de `true`.

Cuando la introducción automática de datos convierte `false` en un número, pasa a ser 0; cuando convierte `false` en una cadena, pasa a ser `"false"`.

Disponibilidad: Flash Player 5; ActionScript 1.0

Ejemplo

Este ejemplo muestra cómo la introducción automática de datos convierte `false` en un número y en una cadena:

```
var bool1:Boolean = Boolean(false);

// converts it to the number 0
trace(1 + bool1); // outputs 1

// converts it to a string
trace("String: " + bool1); // outputs String: false
```

Constante Infinity

Especifica el valor IEEE-754 que representa el infinito positivo. El valor de esta constante es igual que `Number.POSITIVE_INFINITY`.

Disponibilidad: Flash Player 5; ActionScript 1.0

Véase también

`POSITIVE_INFINITY (propiedad Number.POSITIVE_INFINITY)`

Constante -Infinity

Especifica el valor IEEE-754 que representa el infinito negativo. El valor de esta constante es igual que `Number.NEGATIVE_INFINITY`.

Disponibilidad: Flash Player 5; ActionScript 1.0

Véase también

`NEGATIVE_INFINITY (propiedad Number.NEGATIVE_INFINITY)`

Constante NaN

Variable predefinida con el valor IEEE-754 para NaN (no es un número). Para determinar si un número es NaN, utilice `isNaN()`.

Disponibilidad: Flash Player 5; ActionScript 1.0

Véase también

`Función isNaN`, `NaN (propiedad Number.NaN)`

Constante newline

Inserta un carácter de retorno de carro (`\r`) que genera una línea en blanco en la salida de texto que genera el código. Utilice `newline` para crear un espacio donde ubicar la información que recupera una función o sentencia del código.

Disponibilidad: Flash Player 4; ActionScript 1.0

Ejemplo

El ejemplo siguiente muestra cómo `newline` presenta el resultado de la sentencia `trace()` en varias líneas.

```
var myName:String = "Lisa", myAge:Number = 30;
trace(myName+myAge);
trace("-----");
trace(myName+newline+myAge);
// output:
Lisa30
-----
Lisa
30
```

Véase también

`Función trace`

Constante null

Valor especial que puede asignarse a las variables o ser devuelto por una función si no se ha proporcionado ningún dato. Puede utilizar `null` para representar los valores que faltan o que no tienen un tipo de datos definido.

Disponibilidad: Flash Player 5; ActionScript 1.0

Ejemplo

El mensaje siguiente comprueba los seis primeros valores de una matriz indexada y devuelve un mensaje si no hay ningún valor definido (si el valor == `null`):

```
var testArray:Array = new Array();
testArray[0] = "fee";
testArray[1] = "fi";
testArray[4] = "foo";

for (i = 0; i < 6; i++) {
 if (testArray[i] == null) {
 trace("testArray[" + i + "] == null");
 }
}
```

La salida es la siguiente:

```
testArray[2] == null
testArray[3] == null
testArray[5] == null
```

Constante true

Valor booleano exclusivo que representa lo contrario de `false`. Cuando la introducción automática de datos convierte `true` en un número, pasa a ser 1; cuando convierte `true` en una cadena, pasa a ser `"true"`.

Disponibilidad: Flash Player 5; ActionScript 1.0

Ejemplo

El ejemplo siguiente muestra el uso de `true` en una sentencia `if`:

```
var shouldExecute:Boolean;
// ...
// code that sets shouldExecute to either true or false goes here
// shouldExecute is set to true for this example:

shouldExecute = true;

if (shouldExecute == true) {
 trace("your statements here");
}

// true is also implied, so the if statement could also be written:
// if (shouldExecute) {
// trace("your statements here");
// }
```

El ejemplo siguiente muestra cómo la introducción automática de datos convierte `true` en el número 1:

```
var myNum:Number;
myNum = 1 + true;
trace(myNum); // output: 2
```

Véase también

`Contante false`, `Boolean`

Constante undefined

Valor especial que normalmente se utiliza para indicar que todavía no se ha asignado un valor a una variable. Una referencia a un valor undefined devuelve el valor especial `undefined`. El código ActionScript `typeof(undefined)` devuelve la cadena `"undefined"`. El único valor de tipo `undefined` es `undefined`.

En los archivos publicados para Flash Player 6 o anterior, el valor de `String(undefined)` es "" (una cadena vacía). En los archivos publicados para Flash Player 7 o posterior, el valor de `String(undefined)` es `"undefined"` (`undefined` se convierte en una cadena).

En los archivos publicados para Flash Player 6 o anterior, el valor de `Number(undefined)` es `0`. En los archivos publicados para Flash Player 7 o posterior, el valor de `Number(undefined)` es `NaN`.

El valor `undefined` es similar al valor especial `null`. Cuando `null` y `undefined` se comparan con el operador de igualdad (==), se comparan como iguales. Sin embargo, cuando `null` y `undefined` se comparan con el operador de igualdad estricta (===), se comparan como no iguales.

Disponibilidad: Flash Player 5; ActionScript 1.0

Ejemplo

En el ejemplo siguiente no se ha declarado la variable `x` y, por tanto, tiene el valor `undefined`.

En la primera sección del código, el operador de igualdad (==) compara el valor de `x` con el valor `undefined` y envía el resultado correspondiente al panel Salida.

En la segunda sección del código, el operador de igualdad (==) compara los valores `null` y `undefined`.

```
// x has not been declared
trace("The value of x is "+x);

if (x == undefined) {
```

```
 trace("x is undefined");
} else {
 trace("x is not undefined");
}

trace("typeof (x) is "+typeof (x));

if (null == undefined) {
 trace("null and undefined are equal");
} else {
 trace("null and undefined are not equal");
}
```

El resultado siguiente se muestra en el panel Salida.

```
The value of x is undefined
x is undefined
typeof (x) is undefined
null and undefined are equal
```

Funciones globales

Esta sección contiene un conjunto de funciones incorporadas que se encuentran disponibles en cualquier parte de un archivo SWF donde se utilice ActionScript. Estas funciones globales abarcan una gran variedad de tareas de programación frecuentes como, por ejemplo, trabajar con tipos de datos (`Boolean()`, i`int()`, etc.), proporcionar información de depuración (`trace()`) y comunicarse con Flash Player o con el navegador (`fscommand()`).

Resumen de funciones globales

Modificadores	Firma	Descripción
	`Array([numElements:Number], [elementN:Object])`	Crea una nueva matriz vacía o convierte determinados elementos en una matriz.
	`asfunction(function:String, parameter:String)`	Un protocolo especial para URL en campos de texto HTML que permite que un vínculo HREF llame a una función ActionScript.
	`Boolean(expression:Object)`	Convierte el parámetro *`expression`* en un valor booleano y devuelve `true` o `false`.
	`call(frame:Object)`	*Desfasada* desde Flash Player 5. Esta acción está desfasada y en su lugar debe utilizarse la sentencia `function`. Ejecuta el script en el fotograma llamado sin mover la cabeza lectora a ese fotograma.

Modificadores	Firma	Descripción
	`chr(number:Number)`	*Desfasada* desde Flash Player 5. Esta función está desfasada y en su lugar debe utilizarse `String.fromCharCode()`. Convierte números de código ASCII en caracteres.
	`clearInterval(intervalID:Number)`	Detiene la llamada a `setInterval()`.
	`duplicateMovieClip(target:Object, newname:String, depth:Number)`	Crea una instancia de un clip de película durante la reproducción del archivo SWF.
	`escape(expression:String)`	Convierte el parámetro en una cadena y la codifica con un formato URL codificado donde todos los caracteres no alfanuméricos se sustituyen por secuencias hexadecimales %.
	`eval(expression:Object)`	Accede a las variables, propiedades, objetos o clips de película por su nombre.
	`fscommand(command:String, parameters:String)`	Permite que el archivo SWF se comunique con Flash Player o con el programa que aloja Flash Player, por ejemplo, un navegador Web.
	`getProperty(my_mc:String, property)`	Devuelve el valor de la propiedad especificada para el clip de película *`my_mc`*.
	`getTimer()`	Devuelve el número de milisegundos transcurridos desde que se inició la reproducción del archivo SWF.
	`getURL(url:String, [window:String], [method:String])`	Carga un documento de una URL específica en una ventana o pasa variables a otra aplicación en una URL definida.
	`getVersion()`	Devuelve una cadena que contiene información de la versión de Flash Player y de la plataforma.
	`gotoAndPlay([scene:String], frame:Object)`	Envía la cabeza lectora al fotograma especificado en una escena y reproduce desde dicho fotograma.
	`gotoAndStop([scene:String], frame:Object)`	Envía la cabeza lectora al fotograma especificado en una escena y la detiene.

Modificadores	Firma	Descripción
	`ifFrameLoaded([scene:String], frame:Object)`	*Desfasada* desde Flash Player 5. Esta función está desfasada. Macromedia recomienda que utilice la propiedad `MovieClip._framesloaded`. Comprueba si el contenido de un fotograma específico está disponible localmente.
	`int(value:Number)`	*Desfasada* desde Flash Player 5. Esta función está desfasada y en su lugar debe utilizarse `Math.round()`. Convierte un número decimal en un valor de entero truncando el valor decimal.
	`isFinite(expression:Object)`	Evalúa *`expression`* y devuelve `true` si es un número finito o `false` si es infinito o infinito negativo.
	`isNaN(expression:Object)`	Evalúa el parámetro y devuelve `true` si el valor es `NaN` (no es un número).
	`length(expression:String, variable:Object)`	*Desfasada* desde Flash Player 5. Esta función, junto con todas las funciones string, está desfasada. Macromedia recomienda que utilice los métodos de la clase String y la propiedad `String.length` para realizar las mismas operaciones. Devuelve la longitud de la cadena o la variable especificada.
	`loadMovie(url:String, target:Object, [method:String])`	Carga un archivo SWF o JPEG en Flash Player durante la reproducción del archivo SWF original.
	`loadMovieNum(url:String, level:Number, [method:String])`	Carga un archivo SWF o JPEG en un nivel de Flash Player durante la reproducción del archivo SWF que se cargó originalmente.
	`loadVariables(url:String, target:Object, [method:String])`	Lee datos de un archivo externo, como un archivo de texto o texto generado por ColdFusion, un script CGI, páginas Active Server (ASP), script PHP o Perl, y establece los valores de las variables en un clip de película de destino.
	`loadVariablesNum(url:String, level:Number, [method:String])`	Lee datos de un archivo externo, como un archivo de texto o texto generado por ColdFusion, script CGI, páginas Active Server (ASP), script PHP o Perl, y establece los valores de las variables en un nivel de Flash Player.

Modificadores	Firma	Descripción
	`mbchr(number:Number)`	*Desfasada* desde Flash Player 5. Esta función está desfasada y en su lugar debe utilizarse el método `String.fromCharCode()`. Convierte un número de código ASCII en un carácter multibyte.
	`mblength(string:String)`	*Desfasada* desde Flash Player 5. Esta función está desfasada y en su lugar deben utilizarse los métodos y propiedades de la clase String. Devuelve la longitud de la cadena de caracteres multibyte.
	`mbord(character:String)`	*Desfasada* desde Flash Player 5. Esta función está desfasada y en su lugar debe utilizarse `String.charCodeAt()`. Convierte el carácter especificado en un número multibyte.
	`mbsubstring(value:String, index:Number, count:Number)`	*Desfasada* desde Flash Player 5. Esta función está desfasada y en su lugar debe utilizarse `String.substr()`. Extrae una cadena de caracteres multibyte nueva de la cadena de caracteres multibyte.
	`MMExecute(command:String)`	Permite emitir comandos de la API JavaScript de Flash (JSAPI) desde ActionScript.
	`nextFrame()`	Envía la cabeza lectora al siguiente fotograma.
	`nextScene()`	Envía la cabeza lectora al fotograma 1 de la siguiente escena.
	`Number(expression:Object)`	Convierte el parámetro *`expression`* en un número.
	`Object([value:Object])`	Crea un nuevo objeto vacío o convierte en un objeto el número, cadena o valor booleano especificado.
	`on(mouseEvent:Object)`	Especifica el evento de ratón o la pulsación de tecla que activa una acción.
	`onClipEvent(movieEvent:Object)`	Activa acciones definidas para una determinada instancia de un clip de película.
	`ord(character:String)`	*Desfasada* desde Flash Player 5. Esta función está desfasada y en su lugar deben utilizarse los métodos y propiedades de la clase String. Convierte caracteres en números de código ASCII.

Modificadores	Firma	Descripción
	`parseFloat(string:String)`	Convierte una cadena en un número de coma flotante.
	`parseInt(expression:String, [radix:Number])`	Convierte una cadena en un entero.
	`play()`	Mueve la cabeza lectora hacia delante en la línea de tiempo.
	`prevFrame()`	Envía la cabeza lectora al fotograma anterior.
	`prevScene()`	Envía la cabeza lectora al fotograma 1 de la escena anterior.
	`print(target:Object, boundingBox:String)`	Imprime el clip de película de destino (`target`) según los límites especificados en el parámetro (`bmovie`, `bmax` o `bframe`).
	`printAsBitmap(target:Object, boundingBox:String)`	Imprime el clip de película de destino como un mapa de bits (`target`) según los límites especificados en el parámetro (`bmovie`, `bmax` o `bframe`).
	`printAsBitmapNum(level:Number, boundingBox:String)`	Imprime un nivel en Flash Player como un mapa de bits según los límites especificados en el parámetro (`bmovie`, `bmax` o `bframe`).
	`printNum(level:Number, boundingBox:String)`	Imprime el nivel en Flash Player según los límites especificados en el parámetro `boundingBox` (`bmovie`, `bmax` o `bframe`).
	`random(value:Number)`	*Desfasada* desde Flash Player 5. Esta función está desfasada y en su lugar debe utilizarse `Math.random()`. Devuelve un entero aleatorio entre 0 y uno menos que el entero especificado en el parámetro ‹code›‹em›value‹/em›‹/code› .
	`removeMovieClip(target:Object)`	Elimina el clip de película especificado.
	`setInterval(functionReference:Function, interval:Number, [param:Object], objectReference:Object, methodName:String)`	Llama a una función, un método o un objeto en intervalos periódicos durante la reproducción de un archivo SWF.

Modificadores	Firma	Descripción
	`setProperty(target:Object, property:Object, expression:Object)`	Cambia un valor de propiedad de un clip de película durante la reproducción del clip de película.
	`showRedrawRegions(enable:Boolean, [color:Number])`	Ofrece la posibilidad al reproductor del depurador de resaltar las zonas de la pantalla que se van a redibujar.
	`startDrag(target:Object, [lock:Boolean], [left,top,right,bottom:Number])`	Permite arrastrar el clip de película *`target`* durante la reproducción de la película.
	`stop()`	Detiene el archivo SWF que se está reproduciendo.
	`stopAllSounds()`	Detiene todos los sonidos que se están reproduciendo en un archivo SWF sin detener la cabeza lectora.
	`stopDrag()`	Detiene la operación de arrastre actual.
	`String(expression:Object)`	Devuelve una representación de cadena del parámetro especificado.
	`substring(string:String, index:Number, count:Number)`	*Desfasada* desde Flash Player 5. Esta función está desfasada y en su lugar debe utilizarse `String.substr()`. Extrae parte de una cadena.
	`targetPath(targetObject:Object)`	Devuelve una cadena que contiene la ruta de destino de *`movieClipObject`*.
	`tellTarget(target:String, statement(s))`	*Desfasada* desde Flash Player 5. Macromedia recomienda el uso de notación con puntos (.) y la sentencia `with`. Aplica las instrucciones especificadas en el parámetro *`statements`* a la línea de tiempo especificada en el parámetro *`target`*.
	`toggleHighQuality()`	*Desfasada* desde Flash Player 5. Esta función está desfasada y en su lugar debe utilizarse `_quality`. Activa y desactiva el suavizado en Flash Player.
	`trace(expression:Object)`	Evalúa la expresión y produce el resultado.
	`unescape(string:String)`	Evalúa el parámetro *`x`* como una cadena, descodifica la cadena con formato URL codificado (convirtiendo todas las secuencias hexadecimales en caracteres ASCII) y devuelve la cadena.

Modificadores	Firma	Descripción
	`unloadMovie(target:Object)`	Elimina un clip de película que se cargó mediante `loadMovie()` desde Flash Player.
	`unloadMovieNum(level:Number)`	Elimina un archivo SWF o una imagen que se cargó mediante `loadMovieNum()` desde Flash Player.
	`updateAfterEvent()`	Actualiza la visualización cuando se llama en un controlador o mediante `setInterval()`.

Función Array

```
Array() : Array
Array(numElements:Number) : Array
Array(element0:Object, [element1, element2, ...elementN]) : Array
```

Crea una nueva matriz de longitud 0 o más, o una matriz con una lista de elementos especificados, posiblemente de distintos tipos de datos.

Use `Array()` para crear una de las matrices siguientes:

- Una matriz vacía
- Una matriz de longitud específica pero con elementos con valores no definidos (undefined).
- Una matriz cuyos elementos tienen valores específicos.

El uso de esta función es similar a la creación de una matriz con el constructor Array (véase "Constructor de la clase Array").

Puede pasar un número (`numElements`) o una lista de elementos que contenga uno o varios tipos (`element0, element1, ... elementN`).

Los parámetros que pueden aceptar más de un tipo de datos se muestran en la firma como tipo `Object`.

Disponibilidad: Flash Player 6; ActionScript 1.0

Parámetros

`numElements``:Number` [opcional] - Un entero positivo que especifica el número de elementos de la matriz. Puede especificar `numElements` o la lista de elementos, pero no los dos.

`elementN``:Object` [opcional] - uno o varios parámetros, `element0, element1, ... , elementN`, cuyos valores pueden ser de cualquier tipo. Los parámetros que pueden aceptar más de un tipo de datos se engloban en el tipo `Object`. Puede especificar numElements o la lista de elementos, pero no los dos.

Valor devuelto

`Array` - Una matriz.

Ejemplo

```
var myArray:Array = Array();
myArray.push(12);
trace(myArray); //traces 12
myArray[4] = 7;
trace(myArray); //traces 12,undefined,undefined,undefined,7
```

Sintaxis 2: El ejemplo siguiente crea una matriz de longitud 4 pero sin elementos definidos:

```
var myArray:Array = Array(4);
trace(myArray.length); // traces 4
trace(myArray); // traces undefined,undefined,undefined,undefined
```

Sintaxis 3: El ejemplo siguiente crea una matriz con tres elementos definidos:

```
var myArray:Array = Array("firstElement", "secondElement", "thirdElement");
trace (myArray); // traces firstElement,secondElement,thirdElement
Unlike the Array class constructor, the Array() function does not use the
  keyword new .
```

Véase también

`Array`

Protocolo asfunction

```
asfunction:function:Function, parameter:String
```

Un protocolo especial para URL en campos de texto HTML que permite que un vínculo HREF llame a una función ActionScript. En los campos de texto HTML es posible crear vínculos utilizando la etiqueta HTML `A`. El atributo HREF de la etiqueta `A` contiene una URL que utiliza un protocolo estándar como HTTP, HTTPS o FTP. El protocolo `asfunction` es un protocolo adicional específico de Flash que hace que el vínculo invoque una función ActionScript.

Disponibilidad: Flash Player 5; ActionScript 1.0

Parámetros

`function``:String` - Un identificador de una función.

`parameter``:String` - Una cadena que se pasa a la función designada en el parámetro *`function`*.

Ejemplo

En el ejemplo siguiente se define la función `playMP3()`. Se crea el objeto TextField list_txt y se establece de modo que pueda procesarse texto HTML. El texto Track 1 y Track 2 son vínculos dentro del campo de texto. Se llama a la función `playMP3()` cuando el usuario hace clic en uno de los vínculos y reproduce el MP3 que se pasa como parámetro de la función asfunction.

```
var myMP3:Sound = new Sound();
function playMP3(mp3:String) {
 myMP3.loadSound(mp3, true);
 myMP3.onLoad = function(success) {
 if (!success) {
 // code to handle errors here
 }
 };
}
this.createTextField("list_txt", this.getNextHighestDepth(), 0, 0, 200,
  100);
list_txt.autoSize = true;
list_txt.html = true;
list_txt.multiline = true;
list_txt.htmlText = "<a href=\"asfunction:playMP3, track1.mp3\">Track 1</
  a><br>";
list_txt.htmlText += "<a href=\"asfunction:playMP3, track2.mp3\">Track 2</
  a><br>";
```

Cuando se hace clic en un vínculo, el archivo de sonido MP3 se carga en Flash Player.

Véase también

`htmlText (propiedad TextField.htmlText)`

Función Boolean

```
Boolean(expression:Object) : Boolean
```

Convierte el parámetro `expression` en un valor booleano y devuelve un valor como se describe en la siguiente lista:

- Si `expression` es un valor booleano, el valor devuelto es `expression`.
- Si `expression` es un número, el valor devuelto es `true` si el número no es 0; en caso contrario, el valor devuelto es `false`.

Si `expression` es una cadena, el valor devuelto es el siguiente:

- En archivos publicados para Flash Player 6 y versiones anteriores, la cadena se convierte primero en número. El valor es `true` si el número no es 0; en caso contrario, el valor devuelto es `false`.

- En archivos publicados para Flash Player 7 y posterior, el resultado es `true` si la longitud de la cadena es mayor que 0; el valor es `false` si la cadena está vacía.

Si `expression` es una cadena, el resultado es `true` si la longitud de la cadena es mayor que cero; el valor es `false` si la cadena está vacía.

- Si `expression` es `undefined` o `NaN` (no un número), el valor devuelto es `false`.
- Si `expression` es un clip de película o un objeto, el valor devuelto es `true`.

A diferencia del constructor de la clase Boolean, la función `Boolean()` no utiliza la palabra clave new. Además, el constructor de la clase Boolean inicializa un objeto Boolean como `false` si no se especifica ningún parámetro, mientras que la función `Boolean()` devuelve `undefined` si no se especifica ningún parámetro.

Disponibilidad: Flash Player 5; ActionScript 1.0

Parámetros

`expression``:Object` - Una expresión para convertir en un valor booleano.

Valor devuelto

`Boolean` - Valor booleano.

Ejemplo

```
trace(Boolean(-1)); // output: true
trace(Boolean(0)); // output: false
trace(Boolean(1)); // output: true


trace(Boolean(true)); // output: true
trace(Boolean(false)); // output: false


trace(Boolean("true")); // output: true
trace(Boolean("false")); // output: true

trace(Boolean("Craiggers")); // output: true
trace(Boolean("")); // output: false
```

Si los archivos se publican para Flash Player 6 y versiones anteriores, los resultados serán diferentes para tres de los ejemplos anteriores:

```
trace(Boolean("true")); // output: false
trace(Boolean("false")); // output: false
trace(Boolean("Craiggers")); // output: false
```

Este ejemplo muestra una diferencia significativa entre el uso de la función `Boolean()` y la clase Boolean. La función `Boolean()` crea un valor booleano, mientras que la clase Boolean crea un objeto booleano. Los valores booleanos se comparan por valor y los objetos booleanos se comparan por referencia.

```
// Variables representing Boolean values are compared by value
var a:Boolean = Boolean("a"); // a is true
var b:Boolean = Boolean(1); // b is true
trace(a==b); // true

// Variables representing Boolean objects are compared by reference
var a:Boolean = new Boolean("a"); // a is true
var b:Boolean = new Boolean(1); // b is true
trace(a == b); // false
```

Véase también

`Boolean`

Función call

```
call(frame)
```

Desfasada desde Flash Player 5. Esta acción está desfasada y en su lugar debe utilizarse la sentencia `function`.

Ejecuta el script en el fotograma llamado sin mover la cabeza lectora a ese fotograma. Las variables locales no existen después de ejecutar el script.

- Si no se declaran variables en un bloque (`{}`) pero la lista de acciones de ejecutó con una acción call(), las variables serán locales y caducarán al final de la lista actual.
- Si no se declaran variables en un bloque y la lista de acciones actual no se ejecutó con la acción call(), las variables se interpretarán como variables de la línea de tiempo.

Disponibilidad: Flash Player 4; ActionScript 1.0

Parámetros

`frame``:Object` - La etiqueta o el número de un fotograma en la línea de tiempo.

Véase también

`Sentencia function`, `call (método Function.call)`

Función chr

```
chr(number:Number) : String
```

Desfasada desde Flash Player 5. Esta función está desfasada y en su lugar debe utilizarse `String.fromCharCode()`.

Convierte números de código ASCII en caracteres.

Disponibilidad: Flash Player 4; ActionScript 1.0

Parámetros

`number``:Number` - Un número de código ASCII.

Valor devuelto

`String` - El valor de carácter del código ASCII especificado.

Ejemplo

El ejemplo siguiente convierte el número 65 en la letra A y la asigna a la variable `myVar`: `myVar = chr(65);`

Véase también

`fromCharCode (método String.fromCharCode)`

Función clearInterval

```
clearInterval(intervalID:Number) : Void
```

Detiene la llamada a setInterval().

Disponibilidad: Flash Player 6; ActionScript 1.0

Parámetros

`intervalID``:Number` - Un identificador numérico (entero) devuelto desde una llamada a `setInterval()`.

Ejemplo

El ejemplo siguiente establece primero y luego borra una llamada de intervalo:

```
function callback() {
 trace("interval called: "+getTimer()+" ms.");
}
```

```
var intervalID:Number = setInterval(callback, 1000);
```

Debe borrar el intervalo cuando termine de utilizar la función. Cree un botón llamado `clearInt_btn` y utilice el código ActionScript siguiente para borrar `setInterval()`:

```
clearInt_btn.onRelease = function(){
 clearInterval( intervalID );
 trace("cleared interval");
};
```

Véase también

Función setInterval

Función duplicateMovieClip

```
duplicateMovieClip(target:String, newname:String, depth:Number) : Void
duplicateMovieClip(target:MovieClip, newname:String, depth:Number) : Void
```

Crea una instancia de un clip de película durante la reproducción del archivo SWF. En los clips de película duplicados, la cabeza lectora siempre empieza en el fotograma 1, independientemente del lugar en el que se encuentre la cabeza lectora del clip de película original. Las variables del clip de película original no se copian en el clip de película duplicado. Utilice la función o el método `removeMovieClip()` para eliminar una instancia de clip de película creada mediante `duplicateMovieClip()`.

Disponibilidad: Flash Player 4; ActionScript 1.0

Parámetros

`target``:Object` - La ruta de destino del clip de película que se va a duplicar. Este parámetro puede ser una cadena (por ejemplo, "my_mc") o una referencia directa a la instancia de clip de película (por ejemplo, my_mc). Los parámetros que pueden aceptar más de un tipo de datos se engloban en el tipo `Object`.

`newname``:String` - Identificador exclusivo del clip de película duplicado.

`depth``:Number` - Nivel de profundidad exclusivo del clip de película duplicado. El nivel de profundidad es el orden de apilamiento de los clips de película duplicados. Este orden de apilamiento es similar al de las capas de la línea de tiempo; los clips de película con un nivel de profundidad inferior se ocultan bajo los clips con un orden de apilamiento superior. Debe asignar a cada clip de película duplicado un nivel de profundidad exclusivo para evitar la sustitución de archivos SWF en niveles de profundidad ocupados.

Ejemplo

En el ejemplo siguiente, se crea un nuevo clip de película con el nombre `img_mc`.. Se carga una imagen en el clip de película y, a continuación, se duplica el clip `img_mc`. El clip duplicado se llama `newImg_mc` y este nuevo clip pasa al escenario para que no se superponga al clip original, y se carga la misma imagen en el segundo clip.

```
this.createEmptyMovieClip("img_mc", this.getNextHighestDepth());
img_mc.loadMovie("http://www.helpexamples.com/flash/images/image1.jpg");
duplicateMovieClip(img_mc, "newImg_mc", this.getNextHighestDepth());
newImg_mc._x = 200;
newImg_mc.loadMovie("http://www.helpexamples.com/flash/images/image1.jpg");
```

Para eliminar el clip de película duplicado puede añadir este código para un botón llamado `myButton_btn`.

```
this.myButton_btn.onRelease = function(){
 removeMovieClip(newImg_mc);
};
```

Véase también

`Función removeMovieClip`, `duplicateMovieClip (método MovieClip.duplicateMovieClip)`, `removeMovieClip (método MovieClip.removeMovieClip)`

Función escape

```
escape(expression:String) : String
```

Convierte el parámetro en una cadena y la codifica con un formato URL codificado donde todos los caracteres no alfanuméricos se sustituyen por secuencias hexadecimales %. Si se utiliza en una cadena con codificación URL, el símbolo de porcentaje (%) introduce caracteres de escape y no equivale al operador de módulo (%).

Disponibilidad: Flash Player 5; ActionScript 1.0

Parámetros

`expression``:String` - Expresión para convertir en una cadena y codificar en un formato URL codificado.

Valor devuelto

`String` - Cadena con codificación URL.

Ejemplo

El siguiente código genera el resultado `someuser%40somedomain%2Ecom`:

```
var email:String = "someuser@somedomain.com";
trace(escape(email));
```

En este ejemplo, el símbolo de arroba (@) se ha sustituido por `%40` y el símbolo de punto (.), por `%2E`. Estas sustituciones resultan útiles cuando se intenta pasar información a un servidor remoto y los datos contienen caracteres especiales (por ejemplo, `&` o `?`), como se muestra en el código siguiente:

```
var redirectUrl:String = "http://
  www.somedomain.com?loggedin=true&username=Gus";
getURL("http://www.myothersite.com?returnurl="+ escape(redirectUrl));
```

Véase también

`Función unescape`

Función eval

```
eval(expression:Object) : Object
eval(expression:String) : Object
```

Accede a las variables, propiedades, objetos o clips de película por su nombre. Si la expresión es una variable o una propiedad, se devuelve el valor de la variable o propiedad. Si la expresión es un objeto o clip de película, se devuelve una referencia al objeto o clip de película. Si no se encuentra el elemento designado en la expresión, se devuelve undefined.

En Flash 4, se utilizaba `eval()` para simular matrices; en Flash 5 o posterior, debería utilizarse la clase Array para simular matrices.

En Flash 4, se puede utilizar además `eval()` para establecer y recuperar dinámicamente el valor de una variable o nombre de instancia. Sin embargo, esto también puede realizarse mediante el operador de acceso a la matriz (`[]`).

En Flash 5 o posterior, no se puede utilizar `eval()` para establecer y recuperar dinámicamente el valor de una variable o nombre de instancia porque no se puede utilizar `eval()` en el lado izquierdo de una ecuación. Por ejemplo, sustituya el código

```
eval ("var" + i) = "first";
```

por esto:

```
this["var"+i] = "first"
```

o esto:

```
set ("var" + i, "first");
```

Disponibilidad: Flash Player 5; ActionScript 1.0

Parámetros

expression:Object - El nombre de una variable, propiedad, objeto o clip de película que debe recuperarse. Este parámetro puede ser una cadena o una referencia directa a la instancia del objeto (el uso de las comillas (" ") es opcional).

Valor devuelto

Object - Un valor, una referencia a un objeto o clip de película, o undefined.

Ejemplo

El ejemplo siguiente utiliza eval() para establecer propiedades para clips de película con nombres dinámicos. Este código ActionScript define la propiedad _rotation para tres clips de película, denominados square1_mc, square2_mc y square3_mc.

```
for (var i = 1; i <= 3; i++) {
 setProperty(eval("square"+i+"_mc"), _rotation, 5);
}
```

También puede utilizar el código ActionScript siguiente:

```
for (var i = 1; i <= 3; i++) {
 this["square"+i+"_mc"]._rotation = -5;
}
```

Véase también

Array, Sentencia set variable

Función fscommand

```
fscommand(command:String, parameters:String) : Void
```

Permite que el archivo SWF se comunique con Flash Player o con el programa que aloja Flash Player, por ejemplo, un navegador Web. También es posible utilizar la función fscommand() para pasar mensajes a Macromedia Director o a Visual Basic (VB), Visual C++ y otros programas que puedan alojar controles ActiveX.

La función `fscommand()` permite que un archivo SWF se comunique con un script en una página Web. Sin embargo, el acceso de script está controlador por el parámetro `allowScriptAccess` de la página Web. (Esta opción se define en el código HTML que se incluye en el archivo SWF, por ejemplo, en la etiqueta `PARAM` para Internet Explorer o la etiqueta `EMBED` para Netscape.) Cuando `allowScriptAccess` se define como `"never"`, un archivo SWF no puede acceder a scripts de páginas Web. Con Flash Player 7 y versiones posteriores, cuando `allowScriptAccess` está definido como `"always"`, los archivos SWF pueden acceder siempre a scripts de páginas Web. Cuando `allowScriptAccess` está definido como `"sameDomain"`, sólo se admite la creación de scripts en archivos SWF que pertenecen al mismo dominio que la página Web; en versiones anteriores de Flash Player se admitía siempre. Si `allowScriptAccess` no está especificado como una página HTML, el atributo se define de forma predeterminada como `"sameDomain"` para archivos SWF de versión 8 y posterior, y como `"always"` para archivos SWF de versión 7 y anterior.

Sintaxis 1: Para utilizar `fscommand()` para enviar un mensaje a Flash Player, debe utilizar comandos y parámetros predefinidos. La tabla siguiente muestra los valores que puede especificar para los parámetros `command` y `parameters` de la función `fscommand()`. Estos valores controlan archivos SWF que se reproducen en Flash Player, incluidos los proyectores. (Un *proyector* es un archivo SWF que se guarda en un formato que puede ejecutarse como aplicación independiente, es decir, sin Flash Player.)

Comando	Parámetro	Propósito
`quit`	Ninguno	Cierra el proyector.
`fullscreen`	`true` o `false`	Si se especifica `true`, Flash Player se establece en el modo de pantalla completa. Si se especifica `false`, el reproductor vuelve a la vista de menú normal.
`allowscale`	`true` o `false`	Si se especifica `false`, el reproductor se establece de forma que el archivo SWF se dibuje siempre con su tamaño original y nunca se cambie su escala. Si se especifica `true`, se obliga al archivo SWF a cambiar su escala al 100% del reproductor.
`showmenu`	`true` o `false`	Si se especifica `true`, se activa el conjunto completo de elementos de menú contextual. Si se especifica `false`, se atenúan todos los elementos de menú contextual excepto Configuración y Acerca de Flash Player.

Comando	Parámetro	Propósito
exec	Ruta de acceso a la aplicación	Ejecuta una aplicación desde el proyector.
trapallkeys	true o false	Si se especifica true, se envían todos los eventos clave, incluidas las teclas de aceleración, al controlador onClipEvent(keyDown/keyUp) en Flash Player.

Disponibilidad:

- Ninguno de los comandos que se describen en la tabla están disponibles como reproductores Web.
- Todos los comandos están disponibles en aplicaciones autónomas, como proyectores.
- Sólo allowscale y exec están disponibles en reproductores de películas de prueba.

El comando exec puede contener únicamente los caracteres A-Z, a-z, 0-9, punto (.) y subrayado (_). El comando exec se ejecuta únicamente en el subdirectorio fscommand. Dicho de otro modo, si se utiliza el comando exec para llamar a una aplicación, ésta debe residir en un subdirectorio denominado fscommand. El comando exec sólo funciona desde un archivo de proyector de Flash.

Sintaxis 2: Para utilizar fscommand() a fin de enviar un mensaje a un lenguaje de creación de scripts como JavaScript en un navegador Web, puede pasar los dos parámetros que desee en los parámetros command y parameters. Estos parámetros pueden ser cadenas o expresiones y se utilizan en una función de JavaScript que controla o *captura* la función fscommand().

En un navegador Web, fscommand() llama a la función JavaScript moviename_DoFScommand, que reside en la página Web que contiene el archivo SWF. Para moviename, utilice el nombre del objeto de Flash que utilizó para el atributo NAME de la etiqueta EMBED o la propiedad ID de la etiqueta OBJECT. Si asigna al archivo SWF el nombre myMovie, se llama a la función de JavaScript myMovie_DoFScommand.

En la página Web que contiene el archivo SWF, defina el atributo `allowScriptAccess` para permitir o denegar al archivo SWF el acceso a la página Web. (Esta opción se define en el código HTML que se incluye en el archivo SWF, por ejemplo, en la etiqueta `PARAM` para Internet Explorer o la etiqueta `EMBED` para Netscape.) Cuando `allowScriptAccess` está definido como `"never"`, los scripts de salida siempre fallan. Cuando `allowScriptAccess` está definido como `"always"`, los scripts de salida siempre funcionan. Cuando se establece como `"sameDomain"`, sólo es posible acceder a los scripts de archivos SWF que están en el mismo dominio que la página Web. Si no se especifica `allowScriptAccess` en una página Web, se utilizará de forma predeterminada `"sameDomain"` para Flash Player 8, y `"always"` para las versiones anteriores de Flash Player.

Cuando utilice esta función, puede ser conveniente utilizar el modelo de seguridad de Flash Player. Para Flash Player 8, la función `fscommand()` no se admite si el archivo SWF que realiza la llamada está en la libre configuración local-con-sistema-de-archivos o local-con-red y la página HTML que lo contiene está en una libre configuración que no es de confianza. Para más información, consulte lo siguiente:

- Capítulo 17, "Aspectos básicos de la seguridad," en *Aprendizaje de ActionScript 2.0 en Flash*
- El documento técnico sobre la seguridad de Flash Player 8 en http://www.macromedia.com/go/fp8_security
- El documento técnico sobre interfaces API relativas a seguridad de Flash Player 8 en http://www.macromedia.com/go/fp8_security_apis

Sintaxis 3: la función `fscommand()` puede enviar mensajes a Macromedia Director. Lingo (el lenguaje de creación de scripts de Director) interpreta estos mensajes como cadenas, eventos o código Lingo ejecutable. Si el mensaje es una cadena o un evento, debe escribir el código Lingo para recibir el mensaje de la función `fscommand()` y llevar a cabo una acción en Director. Para más información, consulte el Centro de soporte de Director en www.macromedia.com/support/director.

Sintaxis 4: en VisualBasic, Visual C++, y otros programas que pueden alojar controles ActiveX, la función `fscommand()` envía un evento VB con dos cadenas que pueden gestionarse en el lenguaje de programación del entorno. Para obtener más información, utilice las palabras clave "Flash method" (método de Flash) para realizar búsquedas en el Centro de soporte Flash en www.macromedia.com/support/flash.

Nota: Si publica para Flash Player 8 o una versión posterior, la clase ExternalInterface ofrece mejor funcionalidad para comunicaciones entre JavaScript y ActionScript (Sintaxis 2) y entre ActionScript y VisualBasic, Visual C++, u otros programas que puedan incluir controles ActiveX (Sintaxis 4). Debe seguir usando `fscommand()` para enviar mensajes a Flash Player (Sintaxis 1) y Macromedia Director (Sintaxis 3).

Disponibilidad: Flash Player 3; ActionScript 1.0

Parámetros

command:String - Una cadena que se pasa a la aplicación host para cualquier uso o un comando que se pasa a Flash Player.

parameters:String - Una cadena que se pasa a la aplicación host para cualquier uso o un valor que se pasa a Flash Player.

Ejemplo

En el ejemplo siguiente, la función `fscommand()` establece que Flash Player ajuste la escala del archivo SWF al tamaño de pantalla completa cuando se suelte el botón `fullscreen_btn` o `unfullscreen_btn`:

```
this.fullscreen_btn.onRelease = function() {
 fscommand("fullscreen", true);
};

this.unfullscreen_btn.onRelease = function() {
 fscommand("fullscreen", false);
};
```

El ejemplo siguiente aplica la función `fscommand()` a un botón en Flash con el fin de abrir un cuadro de mensaje de JavaScript en una página HTML. El mensaje propiamente dicho se envía a JavaScript como parámetro `fscommand`.

Deberá añadir una función a la página Web que contiene el archivo SWF. Esta función, `myDocument _DoFSCommand()`, espera una llamada `fscommand()`. Cuando se activa `fscommand()` en Flash (por ejemplo, cuando un usuario hace clic en el botón), las cadenas `command` y `parameter` se pasan a la función `myDocument _DoFSCommand()`. Puede utilizar las cadenas pasadas en el código JavaScript o VBScript de la forma que desee. En este ejemplo, la función contiene una sentencia condicional `if` que comprueba si la cadena de comando es `"messagebox"`. En caso afirmativo, un cuadro de alerta de JavaScript muestra el contenido de la cadena `parameters` de la función `fscommand()`.

```
function myDocument_DoFSCommand(command, args) {
 if (command == "messagebox") {
 alert(args);
 }
}
```

En el documento de Flash, añada la función `fscommand()` a un botón:

```
fscommand("messagebox", "This is a message box called from within Flash.")
```

También puede utilizar expresiones para los parámetros de la función `fscommand()`, como se muestra en el ejemplo siguiente:

```
fscommand("messagebox", "Hello, " + name + ", welcome to our website!")
```

Para comprobar el archivo SWF, seleccione Archivo > Vista previa de publicación > HTML. Si publica el archivo SWF empleando Flash con la plantilla FSCommand (en el cuadro de diálogo Configuración de publicación, seleccione la etiqueta HTML), Flash insertará automáticamente la función `myDocument_DoFSCommand()`. Los atributos `NAME` e `ID` del archivo SWF serán el nombre del archivo. Por ejemplo, para el archivo myDocument.fla, los atributos se establecerían como `myDocument`.

Véase también

`ExternalInterface (flash.external.ExternalInterface)`

Función getProperty

```
getProperty(my_mc:Object, property:Object) : Object
```

Devuelve el valor de la propiedad especificada para el clip de película *`my_mc`*.

Disponibilidad: Flash Player 4; ActionScript 1.0

Parámetros

`my_mc``:String` - Nombre de instancia de un clip de película cuya propiedad se recupera.

`property` - Propiedad de un clip de película.

Valor devuelto

`Object` - Valor de la propiedad especificada.

Ejemplo

El ejemplo siguiente crea el clip de película nuevo `someClip_mc` y muestra el valor alfa (`_alpha`) del clip `someClip_mc` en el panel Salida:

```
this.createEmptyMovieClip("someClip_mc", 999);
trace("The alpha of "+getProperty(someClip_mc, _name)+" is:
  "+getProperty(someClip_mc, _alpha));
```

Función getTimer

```
getTimer() : Number
```

Devuelve el número de milisegundos transcurridos desde que se inició la reproducción del archivo SWF.

Disponibilidad: Flash Player 4; ActionScript 1.0

Valor devuelto

`Number` - Número de milisegundos transcurridos desde que se inició la reproducción del archivo SWF.

Ejemplo

En el ejemplo siguiente, las funciones `getTimer()` y `setInterval()` se utilizan para crear un temporizador sencillo:

```
this.createTextField("timer_txt", this.getNextHighestDepth(), 0, 0, 100,
  22);
function updateTimer():Void {
 timer_txt.text = getTimer();
}

var intervalID:Number = setInterval(updateTimer, 100);
```

Función getURL

```
getURL(url:String, [window:String, [method:String]]) : Void
```

Carga un documento de una URL específica en una ventana o pasa variables a otra aplicación en una URL definida. Para probar esta función, asegúrese de que el archivo que se va a cargar se encuentra en la ubicación especificada. Para utilizar una URL absoluta (por ejemplo, *`http://www.myserver.com`*), se necesita una conexión de red.

Disponibilidad: Flash Player 4; ActionScript 1.0

Parámetros

`url``:String` - URL del cual se obtiene el documento.

`window``:String` [opcional] - Especifica la ventana o marco HTML donde debería cargarse el documento. Puede introducir el nombre de una ventana específica o seleccionarlo de entre los siguientes nombres de destino reservados:

- `_self` especifica el fotograma actual en la ventana actual.
- `_blank` especifica una nueva ventana.
- `_parent` especifica el elemento principal del fotograma actual.
- `_top` especifica el fotograma de nivel superior de la ventana actual.

`method``:String` [opcional] - Un método `GET` o `POST` para enviar variables. Si no hay ninguna variable, omita este parámetro. El método `GET` añade las variables al final de la URL y se utiliza para números reducidos de variables. El método `POST` envía las variables en un encabezado HTTP independiente y se utiliza para enviar cadenas de variables largas.

Ejemplo

Este ejemplo carga una imagen en un clip de película. Cuando se hace clic en la imagen, se carga un nuevo URL en una nueva ventana del navegador.

```
var listenerObject:Object = new Object();
listenerObject.onLoadInit = function(target_mc:MovieClip) {
 target_mc.onRelease = function() {
 getURL("http://www.macromedia.com/software/flash/flashpro/", "_blank");
 };
};
var logo:MovieClipLoader = new MovieClipLoader();
logo.addListener(listenerObject);
logo.loadClip("http://www.helpexamples.com/flash/images/image1.jpg",
 this.createEmptyMovieClip("macromedia_mc", this.getNextHighestDepth()));
```

En el ejemplo siguiente, se utiliza `getURL()` para enviar un mensaje de correo electrónico:

```
myBtn_btn.onRelease = function(){
 getURL("mailto:you@somedomain.com");
};
```

En el código ActionScript siguiente, se utiliza JavaScript para abrir una ventana de alerta cuando el archivo SWF se incorpora en una ventana del navegador (recuerde que cuando se llama a JavaScript con getURL(), el parámetro url tiene un límite de 508 caracteres):

```
myBtn_btn.onRelease = function(){
 getURL("javascript:alert('you clicked me')");
};
```

También puede utilizar `GET` o `POST` para enviar variables. El ejemplo siguiente utiliza `GET` para añadir variables a una URL:

```
var firstName:String = "Gus";
var lastName:String = "Richardson";
var age:Number = 92;
myBtn_btn.onRelease = function() {
 getURL("http://www.macromedia.com", "_blank", "GET");
};
```

El código ActionScript siguiente utiliza `POST` para enviar variables en el encabezado HTTP. Compruebe los documentos en una ventana del navegador. En caso contrario, las variables se enviarán utilizando `GET`:

```
var firstName:String = "Gus";
var lastName:String = "Richardson";
var age:Number = 92;
getURL("http://www.macromedia.com", "_blank", "POST");
```

Véase también

Función loadVariables, send (método XML.send), sendAndLoad (método XML.sendAndLoad)

Función getVersion

```
getVersion() : String
```

Devuelve una cadena que contiene información de la versión de Flash Player y de la plataforma. La función `getVersion` devuelve información únicamente para Flash Player 5 o versiones posteriores de Flash Player.

Disponibilidad: Flash Player 5; ActionScript 1.0

Valor devuelto

`String` - Cadena que contiene información de la versión de Flash Player y de la plataforma.

Ejemplo

Los ejemplos siguientes realizan un seguimiento del número de versión de Flash Player que está reproduciendo el archivo SWF:

```
var flashVersion:String = getVersion();
trace(flashVersion); // output: WIN 8,0,1,0
trace($version); // output: WIN 8,0,1,0
trace(System.capabilities.version); // output: WIN 8,0,1,0
```

La función `getVersion` devuelve la cadena siguiente:

```
WIN 8,0,1,0
```

Esta cadena devuelta indica que la plataforma es Microsoft Windows y el número de versión de Flash Player es la versión superior 8, versión inferior 1 (8.1).

Véase también

`os (propiedad capabilities.os)`, `version (propiedad capabilities.version)`

Función gotoAndPlay

```
gotoAndPlay([scene:String], frame:Object) : Void
```

Envía la cabeza lectora al fotograma especificado en una escena y reproduce desde dicho fotograma. Si no se especifica ninguna escena, la cabeza lectora se desplaza al fotograma especificado de la escena actual. Puede utilizar el parámetro *`scene`* únicamente en la línea de tiempo de raíz, no en las líneas de tiempo de los clips de película u otros objetos del documento.

Disponibilidad: Flash Player 2; ActionScript 1.0

Parámetros

scene:String [opcional] - Una cadena que especifica el nombre de la escena a la que se envía la cabeza lectora.

frame:Object - Número que representa el número de fotograma o la cadena que representa la etiqueta del fotograma al que se envía la cabeza lectora.

Ejemplo

En el ejemplo siguiente un documento tiene dos escenas: sceneOne y sceneTwo. La escena uno contiene una etiqueta de fotograma en el fotograma 10 llamada newFrame y dos botones, myBtn_btn y myOtherBtn_btn. Este código ActionScript se inserta en el fotograma 1, escena 1 de la línea de tiempo principal.

```
stop();
myBtn_btn.onRelease = function(){
 gotoAndPlay("newFrame");
};

myOtherBtn_btn.onRelease = function(){
 gotoAndPlay("sceneTwo", 1);
};
```

Cuando el usuario hace clic en los botones, la cabeza lectora pasa a la ubicación especificada y sigue reproduciéndose.

Véase también

gotoAndPlay (método MovieClip.gotoAndPlay), Función nextFrame, Función play, Función prevFrame

Función gotoAndStop

```
gotoAndStop([scene:String], frame:Object) : Void
```

Envía la cabeza lectora al fotograma especificado en una escena y la detiene. Si no se ha especificado ninguna escena, la cabeza lectora se envía al fotograma de la escena actual. Puede utilizar el parámetro *scene* sólo en la línea de tiempo raíz, no en las líneas de tiempo de los clips de película u otros objetos del documento.

Disponibilidad: Flash Player 2; ActionScript 1.0

Parámetros

scene:String [opcional] - Una cadena que especifica el nombre de la escena a la que se envía la cabeza lectora.

frame:Object - Número que representa el número de fotograma o la cadena que representa la etiqueta del fotograma al que se envía la cabeza lectora.

Ejemplo

En el ejemplo siguiente un documento tiene dos escenas: sceneOne y sceneTwo. La escena uno contiene una etiqueta de fotograma en el fotograma 10 llamada newFrame y dos botones, myBtn_btn y myOtherBtn_btn. Este código ActionScript se inserta en el fotograma 1, escena 1 de la línea de tiempo principal:

```
stop();

myBtn_btn.onRelease = function(){
 gotoAndStop("newFrame");
};

myOtherBtn_btn.onRelease = function(){
 gotoAndStop("sceneTwo", 1);
};
```

Cuando el usuario hace clic en los botones, la cabeza lectora pasa a la ubicación especificada y se detiene.

Véase también

gotoAndStop (método MovieClip.gotoAndStop), Función stop, Función play, Función gotoAndPlay

Función ifFrameLoaded

```
ifFrameLoaded([scene:String], frame) {
statement(s);
}
```

Desfasada desde Flash Player 5. Esta función está desfasada. Macromedia recomienda que utilice la propiedad MovieClip._framesloaded.

Comprueba si el contenido de un fotograma específico está disponible localmente. Utilice ifFrameLoaded para comenzar a reproducir una animación sencilla mientras se descarga el resto del archivo SWF en un equipo local. La diferencia entre _framesloaded y ifFrameLoaded radica en que _framesloaded permite añadir sentencias else o if personalizadas.

Disponibilidad: Flash Player 4; ActionScript 1.0

Parámetros

scene:String [opcional] - Una cadena que especifica el nombre de la escena que se debe cargar.

frame:Object - Es necesario cargar el número de fotograma o la etiqueta de fotograma antes de ejecutar la sentencia siguiente.

Véase también

, addListener (método MovieClipLoader.addListener)

Función int

```
int(value:Number) : Number
```

Desfasada desde Flash Player 5. Esta función está desfasada y en su lugar debe utilizarse Math.round().

Convierte un número decimal en un valor de entero truncando el valor decimal. Esta función es equivalente a Math.floor() si el parámetro *value* es positivo y Math.ceil() si el parámetro *value* es negativo.

Disponibilidad: Flash Player 4; ActionScript 1.0

Parámetros

value:Number - Un número que se redondeará a un entero.

Valor devuelto

Number - El valor de entero truncado.

Véase también

round (método Math.round), floor (método Math.floor), ceil (método Math.ceil)

Función isFinite

```
isFinite(expression:Object) : Boolean
```

Evalúa *expression* y devuelve true si es un número finito o false si es infinito o infinito negativo. La presencia de infinito o infinito negativo indica un error matemático como, por ejemplo, la división por 0.

Disponibilidad: Flash Player 5; ActionScript 1.0

Parámetros

expression:Object - Valor booleano, variable u otra expresión que se va a evaluar.

Valor devuelto

Boolean - Valor booleano.

Ejemplo

El ejemplo siguiente muestra los valores devueltos para isFinite:

```
isFinite(56)
// returns true

isFinite(Number.POSITIVE_INFINITY)
//returns false
```

Función isNaN

```
isNaN(expression:Object) : Boolean
```

Evalúa el parámetro y devuelve true si el valor es NaN (no es un número). Esta función es útil para comprobar si una expresión matemática da como resultado un número.

Disponibilidad: Flash Player 5; ActionScript 1.0

Parámetros

expression:Object - Valor booleano, variable u otra expresión que se va a evaluar.

Valor devuelto

Boolean - Valor booleano.

Ejemplo

El código siguiente ilustra los valores devueltos para la función isNaN():

```
trace( isNaN("Tree") );
// returns true

trace( isNaN(56) );
// returns false

trace( isNaN(Number.POSITIVE_INFINITY) )
// returns false
```

El ejemplo siguiente muestra cómo puede utilizar `isNAN()` para comprobar si una expresión matemática contiene un error:

```
var dividend:Number;
var divisor:Number;
divisor = 1;
trace( isNaN(dividend/divisor) );
// output: true
// The output is true because the variable dividend is undefined.
// Do not use isNAN() to check for division by 0 because it will return
  false.
// A positive number divided by 0 equals Infinity
  (Number.POSITIVE_INFINITY).
// A negative number divided by 0 equals -Infinity
  (Number.NEGATIVE_INFINITY).
```

Véase también

`Constante NaN`, `NaN (propiedad Number.NaN)`

Función length

```
length(expression:String)length(variable)
```

Desfasada desde Flash Player 5. Esta función, junto con todas las funciones string, está desfasada. Macromedia recomienda que utilice los métodos de la clase String y la propiedad `String.length` para realizar las mismas operaciones.

Devuelve la longitud de la cadena o la variable especificada.

Disponibilidad: Flash Player 4; ActionScript 1.0

Parámetros

`expression``:String` - Una cadena.

`variable``:Object` - El nombre de una variable.

Valor devuelto

`Number` - La longitud de la cadena o la variable especificada.

Ejemplo

El ejemplo siguiente devuelve la longitud de la cadena "Hello": `length("Hello");` El resultado es 5.

Véase también

Operador de delimitador de cadena ("), String, length (propiedad String.length)

Función loadMovie

```
loadMovie(url:String, target:Object, [method:String]) : Void
loadMovie(url:String, target:String, [method:String]) : Void
```

Carga un archivo SWF, JPEG, GIF o PNG en un clip de película en Flash Player mientras se reproduce el archivo SWF original. En Flash Player 8 se ha añadido compatibilidad con archivos GIF no animados, archivos PNG y archivos JPEG progresivos. Si carga un archivo GIF animado, sólo se mostrará el primer fotograma.

Sugerencia: si desea controlar el progreso de la descarga, utilice `MovieClipLoader.loadClip()` en lugar de esta función.

La función `loadMovie()` permite mostrar varios archivos SWF simultáneamente y cambiar entre archivos SWF sin cargar otro documento HTML. Sin la función `loadMovie()`, Flash Player muestra un solo archivo SWF.

Si desea cargar un archivo SWF o JPEG en un nivel específico, utilice `loadMovieNum()` en lugar de `loadMovie()`.

Cuando se carga un archivo SWF en un clip de película de destino, se puede utilizar la ruta de destino de dicho clip de película para buscar el archivo SWF cargado. Un archivo SWF o una imagen que se carga en un destino hereda las propiedades de posición, rotación y escala del clip de película de destino. La esquina superior izquierda de la imagen o archivo SWF cargado se alinea con el punto de registro del clip de película de destino. Como alternativa, si el destino se encuentra en la línea de tiempo raíz, la esquina superior izquierda de la imagen o del archivo SWF se alinea con la esquina superior izquierda del escenario.

Utilice `unloadMovie()` para eliminar los archivos SWF que se cargaron con `loadMovie()`.

Cuando utilice esta función, puede ser conveniente usar el modelo de seguridad de Flash Player.

Para Flash Player 8:

- Es posible que no se pueda realizar la carga si el clip de película que realiza la llamada está en la libre configuración local con sistema de archivos y el clip de película cargado procede de la libre configuración de red.
- No se permite la carga si el archivo SWF que realiza la llamada está en la libre configuración de red y el clip de película que se va a cargar es local.

- Para acceder a la libre configuración de red desde la libre configuración local de confianza o local con acceso a la red se necesitan permisos del sitio Web a través de un archivo de política de varios dominios.
- Los clips de película de libre configuración local con sistema de archivos no pueden crear scripts para clips de película de libre configuración local con acceso a la red (tampoco se puede realizar la operación inversa).

Flash Player 7 y versiones anteriores:

- Los sitios Web pueden conceder acceso a varios dominios a un recurso mediante un archivo de política de varios dominios.
- La creación de scripts entre archivos SWF está limitada según el dominio de origen de los archivos SWF. Utilice el método `System.security.allowDomain()` para ajustar estas limitaciones.

Para más información, consulte lo siguiente:

- Capítulo 17, "Aspectos básicos de la seguridad," en *Aprendizaje de ActionScript 2.0 en Flash*
- El documento técnico sobre la seguridad de Flash Player 8 en http://www.macromedia.com/go/fp8_security
- El documento técnico sobre interfaces API relativas a seguridad de Flash Player 8 en http://www.macromedia.com/go/fp8_security_apis

Disponibilidad: Flash Player 3; ActionScript 1.0

Parámetros

url:`String` - URL absoluto o relativo del archivo SWF o JPEG que se va a cargar. Una ruta relativa debe ser relativa al archivo SWF en el nivel 0. Las URL absolutas deben incluir la referencia al protocolo, como http:// o file:///.

target:`Object` - Referencia a un objeto de clip de película o a una cadena que representa la ruta de acceso a un clip de película de destino. El clip de película de destino se sustituye por el archivo SWF o la imagen que se carga.

method:`String` [opcional] - Especifica un método HTTP para enviar variables. El parámetro debe ser la cadena `GET` o `POST`. Si no hay ninguna variable para enviar, omita este parámetro. El método `GET` añade las variables al final de la URL y se utiliza para números reducidos de variables. El método `POST` envía las variables en un encabezado HTTP independiente y se utiliza para cadenas de variables largas.

Ejemplo

Sintaxis 1: El ejemplo siguiente carga el archivo SWF circle.swf desde el mismo directorio y reemplaza un clip de película llamado `mySquare` que ya existe en el escenario:

```
loadMovie("circle.swf", mySquare);
// equivalent statement (Usage 1): loadMovie("circle.swf",
  _level0.mySquare);
// equivalent statement (Usage 2): loadMovie("circle.swf", "mySquare");
```

El ejemplo siguiente carga el archivo SWF circle.swf desde el mismo directorio, pero reemplaza el clip de película principal en lugar del clip `mySquare`:

```
loadMovie("circle.swf", this);
// Note that using "this" as a string for the target parameter will not work
// equivalent statement (Usage 2): loadMovie("circle.swf", "_level0");
```

La siguiente sentencia `loadMovie()` carga el archivo SWF sub.swf desde el mismo directorio en un clip de película nuevo llamado `logo_mc` que se crea empleando `createEmptyMovieClip()`:

```
this.createEmptyMovieClip("logo_mc", 999);
loadMovie("sub.swf", logo_mc);
```

Puede añadir el código siguiente para cargar una imagen JPEG llamada image1.jpg desde el mismo directorio que el archivo SWF que carga sub.swf. El JPEG se carga al hacer clic en un botón denominado `myBtn_btn`. Este código carga el JPEG en `logo_mc`. Por tanto, reemplazará a sub.swf por la imagen JPEG.

```
myBtn_btn.onRelease = function(){
 loadMovie("image1.jpg", logo_mc);
};
```

Sintaxis 2: El ejemplo siguiente carga el archivo SWF circle.swf desde el mismo directorio y reemplaza un clip de película llamado `mySquare` que ya existe en el escenario:

```
loadMovie("circle.swf", "mySquare");
```

Véase también

Propiedad _level, Función loadMovieNum, loadMovie (método MovieClip.loadMovie), loadClip (método MovieClipLoader.loadClip), Función unloadMovie

Función loadMovieNum

```
loadMovieNum(url:String, level:Number, [method:String]) : Void
```

Carga un archivo SWF, JPEG, GIF o PNG en un nivel mientras se reproduce el archivo SWF original. En Flash Player 8 se ha añadido compatibilidad con archivos GIF no animados, archivos PNG y archivos JPEG progresivos. Si carga un archivo GIF animado, sólo se mostrará el primer fotograma.

Sugerencia: si desea controlar el progreso de la descarga, utilice `MovieClipLoader.loadClip()` en lugar de esta función.

Normalmente, Flash Player muestra un solo archivo SWF y se cierra. La acción `loadMovieNum()` permite mostrar varios archivos SWF simultáneamente y cambiar entre archivos SWF sin cargar otro documento HTML.

Si desea especificar un destino en lugar de un nivel, utilice `loadMovie()` en lugar de `loadMovieNum()`.

Flash Player tiene un orden de apilamiento de niveles a partir del nivel 0. Estos niveles son como capas de acetato: son transparentes, excepto en los objetos de cada nivel. Cuando utilice `loadMovieNum()`, debe especificar un nivel en Flash Player donde se cargará el archivo SWF. Cuando se carga un archivo SWF en un nivel, puede utilizar la sintaxis `_level` *`N`* , donde *`N`* es el número de nivel, para buscar el archivo SWF.

Cuando cargue un archivo SWF, puede especificar cualquier número de nivel y puede cargar archivos SWF en un nivel que ya tenga un archivo SWF cargado. De esta forma, el nuevo archivo SWF sustituirá al archivo SWF existente. Si carga un archivo SWF en el nivel 0, se descargarán todos los niveles de Flash Player y se sustituirá el nivel 0 por el nuevo archivo. El archivo SWF en el nivel 0 establece la velocidad de fotogramas, el color de fondo y el tamaño de fotograma de todos los demás archivos SWF cargados.

La acción `loadMovieNum()` también permite cargar archivos JPEG en un archivo SWF durante su reproducción. En las imágenes y archivos SWF, la esquina superior izquierda de la imagen se alinea con la esquina superior izquierda del escenario cuando se carga el archivo. Además, en ambos casos, el archivo cargado hereda la rotación y la escala, y el contenido original se sobrescribe en el nivel especificado.

Nota: los archivos JPEG guardados en formato progresivo no son compatibles.

Utilice `unloadMovieNum()` para eliminar los archivos SWF o las imágenes que se cargaron con `loadMovieNum()`.

Cuando utilice este método, puede ser conveniente usar el modelo de seguridad de Flash Player.

Para Flash Player 8:

- Es posible que no se pueda realizar la carga si el clip de película que realiza la llamada está en la libre configuración local con sistema de archivos y el clip de película cargado procede de la libre configuración de red.
- No se permite la carga si el archivo SWF que realiza la llamada está en la libre configuración de red y el clip de película que se va a cargar es local.
- Para acceder a la libre configuración de red desde la libre configuración local de confianza o local con acceso a la red se necesitan permisos del sitio Web a través de un archivo de política de varios dominios.
- Los clips de película de libre configuración local con sistema de archivos no pueden crear scripts para clips de película de libre configuración local con acceso a la red (tampoco se puede realizar la operación inversa).

Flash Player 7 y versiones anteriores:

- Los sitios Web pueden conceder acceso a varios dominios a un recurso mediante un archivo de política de varios dominios.
- La creación de scripts entre archivos SWF está limitada según el dominio de origen de los archivos SWF. Utilice el método `System.security.allowDomain()` para ajustar estas limitaciones.

Para más información, consulte lo siguiente:

- Capítulo 17, "Aspectos básicos de la seguridad," en *Aprendizaje de ActionScript 2.0 en Flash*
- El documento técnico sobre la seguridad de Flash Player 8 en http://www.macromedia.com/go/fp8_security
- El documento técnico sobre interfaces API relativas a seguridad de Flash Player 8 en http://www.macromedia.com/go/fp8_security_apis

Disponibilidad: Flash Player 4; ActionScript 1.0

Parámetros

`url``:String` - URL absoluto o relativo del archivo SWF o JPEG que se va a cargar. Una ruta relativa debe ser relativa al archivo SWF en el nivel 0. Para utilizar Flash Player de forma independiente o para realizar pruebas en el modo de prueba de la aplicación de edición de Flash, todos los archivos SWF deben almacenarse en la misma carpeta y los nombres de archivo no pueden contener especificaciones de carpeta o unidad de disco.

`level``:Number` - Entero que especifica el nivel de Flash Player en el que se cargará el archivo SWF.

method:String [opcional] - Especifica un método HTTP para enviar variables. El parámetro debe ser la cadena GET o POST. Si no hay ninguna variable para enviar, omita este parámetro. El método GET añade las variables al final de la URL y se utiliza para números reducidos de variables. El método POST envía las variables en un encabezado HTTP independiente y se utiliza para cadenas de variables largas.

Ejemplo

El ejemplo siguiente carga la imagen JPEG tim.jpg en el nivel 2 de Flash Player:

```
loadMovieNum("http://www.helpexamples.com/flash/images/image1.jpg", 2);
```

Véase también

Función unloadMovieNum, Función loadMovie, loadClip (método MovieClipLoader.loadClip), Propiedad _level

Función loadVariables

```
loadVariables(url:String, target:Object, [method:String]) : Void
```

Lee datos de un archivo externo, como un archivo de texto o texto generado por ColdFusion, un script CGI, páginas Active Server (ASP), script PHP o Perl, y establece los valores de las variables en un clip de película de destino. Esta acción se puede utilizar además para actualizar las variables del archivo SWF activo con nuevos valores.

El texto y la URL especificados deben tener el formato MIME estándar *application/x-www-form-urlencoded* (formato estándar que se utiliza en los scripts CGI). Se puede especificar cualquier número de variables. Por ejemplo, la siguiente frase define varias variables:

```
company=Macromedia&address=600+Townsend&city=San+Francisco&zip=94103
```

En archivos SWF que se ejecuten en una versión anterior a Flash Player 7, *url* debe estar en el mismo superdominio que el archivo SWF que emite esta llamada. El superdominio se obtiene al eliminar el componente más a la izquierda de la URL de un archivo. Por ejemplo, un archivo SWF situado en www.someDomain.com puede cargar datos de una fuente situada en store.someDomain.com porque ambos archivos se encuentran en el mismo superdominio, denominado someDomain.com.

En los archivos SWF de cualquier versión que se ejecute en Flash Player 7 o posterior, el *url* debe pertenecer exactamente al mismo dominio que el archivo SWF que realiza esta llamada (véase "Funciones de seguridad de Flash Player" en *Utilización de ActionScript en Flash*). Por ejemplo, un archivo SWF en www.someDomain.com puede cargar datos únicamente desde orígenes que también se encuentren en www.someDomain.com. Si desea cargar datos de un dominio distinto, puede colocar un *archivo de política para distintos dominios* en el servidor que aloja el archivo SWF al que se obtiene acceso. Para más información, consulte "Carga de datos de varios dominios" en *Utilización de ActionScript en Flash*.

Si desea cargar variables en un nivel específico, utilice `loadVariablesNum()` en lugar de `loadVariables()`.

Disponibilidad: Flash Player 4; ActionScript 1.0

Parámetros

url:`String` - URL donde se ubican las variables. Si el archivo SWF que realiza esta llamada se ejecuta en un navegador Web, el valor *url* debe pertenecer al mismo dominio que el archivo SWF; para ver más detalles, consulte la sección Descripción.

target:`Object` - Ruta de destino de un clip de película que recibe las variables que se cargan.

method:`String` [opcional] - Especifica un método HTTP para enviar variables. El parámetro debe ser la cadena `GET` o `POST`. Si no hay ninguna variable para enviar, omita este parámetro. El método `GET` añade las variables al final de la URL y se utiliza para números reducidos de variables. El método `POST` envía las variables en un encabezado HTTP independiente y se utiliza para cadenas de variables largas.

Ejemplo

El ejemplo siguiente carga información desde un archivo de texto llamado params.txt en el clip de película `target_mc` que se crea utilizando `createEmptyMovieClip()`. La función `setInterval()` se utiliza para comprobar el progreso de la carga. El script busca en el archivo params.txt una variable denominada `done`.

```
this.createEmptyMovieClip("target_mc", this.getNextHighestDepth());
loadVariables("params.txt", target_mc);
function checkParamsLoaded() {
 if (target_mc.done == undefined) {
 trace("not yet.");
 } else {
 trace("finished loading. killing interval.");
 trace("-------------");
 for (i in target_mc) {
 trace(i+": "+target_mc[i]);
 }
```

```
 trace("-------------");
 clearInterval(param_interval);
 }
}
var param_interval:Number = setInterval(checkParamsLoaded, 100);
```

El archivo externo, `params.txt`, incluye el texto siguiente:

```
var1="hello"&var2="goodbye"&done="done"
```

Véase también

Función loadVariablesNum, Función loadMovie, Función loadMovieNum, Función getURL, loadMovie (método MovieClip.loadMovie), loadVariables (método MovieClip.loadVariables), load (método LoadVars.load)

Función loadVariablesNum

```
loadVariablesNum(url:String, level:Number, [method:String]) : Void
```

Lee datos de un archivo externo, como un archivo de texto o texto generado por ColdFusion, un script CGI, páginas Active Server (ASP), script PHP o Perl, y establece los valores de las variables en un nivel de Flash Player. Esta función se puede utilizar además para actualizar las variables del archivo SWF activo con nuevos valores.

El texto y la URL especificada deben tener el formato MIME estándar *`application/x-www-form-urlencoded`* (formato estándar que se utiliza en los scripts CGI). Se puede especificar cualquier número de variables. Por ejemplo, la siguiente frase define varias variables:

```
company=Macromedia&address=601+Townsend&city=San+Francisco&zip=94103
```

En archivos SWF que se ejecuten en una versión del reproductor anterior a Flash Player 7, *`url`* debe estar en el mismo superdominio que el archivo SWF que emite esta llamada. El superdominio se obtiene al eliminar el componente más a la izquierda de la URL de un archivo. Por ejemplo, un archivo SWF en www.someDomain.com puede cargar datos desde un origen en store.someDomain.com porque ambos archivos pertenecen al mismo superdominio (someDomain.com).

En los archivos SWF de cualquier versión que se ejecute en Flash Player 7 o posterior, el *url* debe pertenecer exactamente al mismo dominio que el archivo SWF que realiza esta llamada (véase "Funciones de seguridad de Flash Player" en *Utilización de ActionScript en Flash*). Por ejemplo, un archivo SWF situado en www.someDomain.com sólo puede cargar datos de fuentes que también se encuentren en www.someDomain.com. Si desea cargar datos de un dominio diferente, puede colocar un *archivo de política para distintos dominios* en el servidor en el que se aloja el archivo SWF. Para más información, consulte "Carga de datos de varios dominios" en *Utilización de ActionScript en Flash*.

Si desea cargar variables en un nivel específico, utilice `loadVariables()` en lugar de `loadVariablesNum()`.

Disponibilidad: Flash Player 4; ActionScript 1.0

Parámetros

url:`String` - URL donde se ubican las variables. Si el archivo SWF que realiza esta llamada se ejecuta en un navegador Web, el valor *url* debe pertenecer al mismo dominio que el archivo SWF; para ver más detalles, consulte la sección Descripción.

level:`Number` - Entero que especifica el nivel de Flash Player donde se van a recibir las variables.

method:`String` [opcional] - Especifica un método HTTP para enviar variables. El parámetro debe ser la cadena `GET` o `POST`. Si no hay ninguna variable para enviar, omita este parámetro. El método `GET` añade las variables al final de la URL y se utiliza para números reducidos de variables. El método `POST` envía las variables en un encabezado HTTP independiente y se utiliza para cadenas de variables largas.

Ejemplo

El ejemplo siguiente carga información desde un archivo de texto llamado params.txt en la línea de tiempo principal del SWF en el nivel 2 de Flash Player. Los nombres de las variables de los campos de texto deben coincidir con los de las variables del archivo params.txt. La función `setInterval()` sirve para comprobar el progreso de los datos que se están cargando en el SWF. El script busca en el archivo params.txt una variable denominada `done`.

```
loadVariablesNum("params.txt", 2);
function checkParamsLoaded() {
 if (_level2.done == undefined) {
 trace("not yet.");
 } else {
 trace("finished loading. killing interval.");
 trace("-------------");
 for (i in _level2) {
 trace(i+": "+_level2[i]);
```

```
 }
 trace("-------------");
 clearInterval(param_interval);
 }
}
var param_interval:Number = setInterval(checkParamsLoaded, 100);

// Params.txt includes the following text
var1="hello"&var2="goodbye"&done="done"
```

Véase también

Función getURL, Función loadMovie, Función loadMovieNum, Función loadVariables, loadMovie (método MovieClip.loadMovie), loadVariables (método MovieClip.loadVariables), load (método LoadVars.load)

Función mbchr

```
mbchr(number:Number)
```

Desfasada desde Flash Player 5. Esta función está desfasada y en su lugar debe utilizarse el método `String.fromCharCode()`.

Convierte un número de código ASCII en un carácter multibyte.

Disponibilidad: Flash Player 4; ActionScript 1.0

Parámetros

number:Number - El número que se va a convertir en un carácter multibyte.

Véase también

fromCharCode (método String.fromCharCode)

Función mblength

```
mblength(string:String) : Number
```

Desfasada desde Flash Player 5. Esta función está desfasada y en su lugar deben utilizarse los métodos y propiedades de la clase String.

Devuelve la longitud de la cadena de caracteres multibyte.

Disponibilidad: Flash Player 4; ActionScript 1.0

Parámetros

string:String - La cadena que se va a medir.

Valor devuelto

Number - La longitud de la cadena de caracteres multibyte.

Véase también

String, length (propiedad String.length)

Función mbord

```
mbord(character:String) : Number
```

Desfasada desde Flash Player 5. Esta función está desfasada y en su lugar debe utilizarse String.charCodeAt().

Convierte el carácter especificado en un número multibyte.

Disponibilidad: Flash Player 4; ActionScript 1.0

Parámetros

character:String - *character* El carácter que se va a convertir en un número multibyte.

Valor devuelto

Number - El carácter convertido.

Véase también

charCodeAt (método String.charCodeAt)

Función mbsubstring

```
mbsubstring(value:String, index:Number, count:Number) : String
```

Desfasada desde Flash Player 5. Esta función está desfasada y en su lugar debe utilizarse String.substr().

Extrae una cadena de caracteres multibyte nueva de la cadena de caracteres multibyte.

Disponibilidad: Flash Player 4; ActionScript 1.0

Parámetros

value:String - La cadena multibyte de la que se va a extraer una nueva cadena multibyte.

index:Number - El número del primer carácter que se va a extraer.

count:Number - El número de caracteres que se va a incluir en la cadena extraída, excluido el carácter de índice.

Valor devuelto

String - La cadena extraída de la cadena de caracteres multibyte.

Véase también

substr (método String.substr)

Función MMExecute

```
MMExecute("Flash JavaScript API command;":String) : String
```

Permite emitir comandos de la API JavaScript de Flash (JSAPI) desde ActionScript. En Flash MX 2004, la función MMExecute sólo puede llamarse desde una película que se utilice como panel de Flash (archivo almacenado en el directorio WindowSWF), desde un cuadro de diálogo XMLtoUI o desde la interfaz de usuario personalizada de un componente. Los comandos JSAPI no tienen ningún efecto en el reproductor, en el modo de prueba de película ni fuera del entorno de edición.

La JSAPI de Flash proporciona varios objetos, métodos y propiedades para duplicar o emular comandos que un usuario puede introducir en el entorno de edición. El uso de JSAPI permite escribir scripts que amplían Flash de diversas formas: añadir comandos a los menús, manipular objetos del escenario, repetir secuencias de comandos, etc.

En general, un usuario ejecuta un script JSAPI mediante la selección de Comandos > Ejecutar comando. Sin embargo, puede utilizar esta función en un script de ActionScript para llamar directamente a un comando JSAPI. Si utiliza MMExecute() en un script en el fotograma 1 del archivo, el comando se ejecuta cuando se carga el archivo SWF.

Para más información sobre la JSAPI, consulte www.macromedia.com/go/jsapi_info_en.

Disponibilidad: Flash Player 7; ActionScript 1.0

Parámetros

command:String - Cualquier comando que pueda utilizar en un archivo Flash JavaScript (JSFL).

Valor devuelto

`String` - Una representación de cadena del resultado, si lo hay, enviada por la sentencia de JavaScript.

Ejemplo

El comando siguiente presentará el número de elementos de la biblioteca del documento actual en la ventana de seguimiento. Deberá ejecutar este ejemplo como un panel Flash porque los archivos Flash no pueden llamar a `MMExecute` si se ejecutan en Probar película o en el navegador.

- Inserte el código siguiente en el fotograma 1 de la línea de tiempo principal de un documento vacío de Flash:

```
var numLibItems = MMExecute("fl.getDocumentDOM().library.items.length");

var message = numLibItems + " items in library";

MMExecute('fl.trace("' + message + '");');
```

- Guarde el archivo FLA en el directorio WindowSWF que se encuentra dentro del directorio Configuration y, a continuación, seleccione Archivo > Publicar (o guárdelo en otra ubicación y publique el archivo SWF directamente en ese directorio o desplace el archivo SWF a ese directorio).
- Cierre y reinicie la aplicación (deberá dar este paso la primera vez que añada el archivo al directorio WindowSWF).

Ahora puede seleccionar el archivo desde la parte inferior del menú Ventana > Otros paneles.

La función trace de ActionScript no funciona desde un panel de Flash; este ejemplo utiliza la versión `fl.trace` de JavaScript para obtener los resultados. Quizá resulte más sencillo copiar los resultados de `MMExecute` en un campo de texto que forme parte del archivo del panel de Flash.

Función nextFrame

```
nextFrame() : Void
```

Envía la cabeza lectora al siguiente fotograma.

Disponibilidad: Flash Player 2; ActionScript 1.0

Ejemplo

En el ejemplo siguiente, cuando el usuario pulsa la tecla de flecha derecha o abajo, la cabeza lectora pasa al siguiente fotograma y se detiene. Si el usuario pulsa la tecla izquierda o arriba, la cabeza lectora pasa al fotograma anterior y se detiene. El detector se inicializa para esperar a que se pulse la tecla de flecha y la variable `init` se utiliza para evitar que el detector se redefina si la cabeza lectora vuelve al fotograma 1.

```
stop();

if (init == undefined) {
 someListener = new Object();
 someListener.onKeyDown = function() {
 if (Key.isDown(Key.LEFT) || Key.isDown(Key.UP)) {
 _level0.prevFrame();
 } else if (Key.isDown(Key.RIGHT) || Key.isDown(Key.DOWN)) {
 _level0.nextFrame();
 }
 };
 Key.addListener(someListener);
 init = 1;
}
```

Véase también

Función prevFrame

Función nextScene

```
nextScene() : Void
```

Envía la cabeza lectora al fotograma 1 de la siguiente escena.

Disponibilidad: Flash Player 2; ActionScript 1.0

Ejemplo

En el ejemplo siguiente, cuando un usuario hace clic en el botón que se crea en tiempo de ejecución, la cabeza lectora se envía al fotograma 1 de la escena siguiente. Cree dos escenas e introduzca el código ActionScript siguiente en el fotograma 1 de la escena 1.

```
stop();

if (init == undefined) {
 this.createEmptyMovieClip("nextscene_mc", this.getNextHighestDepth());
 nextscene_mc.createTextField("nextscene_txt", this.getNextHighestDepth(),
  200, 0, 100, 22);
 nextscene_mc.nextscene_txt.autoSize = true;
```

```
 nextscene_mc.nextscene_txt.border = true;
 nextscene_mc.nextscene_txt.text = "Next Scene";
 this.createEmptyMovieClip("prevscene_mc", this.getNextHighestDepth());
 prevscene_mc.createTextField("prevscene_txt", this.getNextHighestDepth(),
  00, 0, 100, 22);
 prevscene_mc.prevscene_txt.autoSize = true;
 prevscene_mc.prevscene_txt.border = true;
 prevscene_mc.prevscene_txt.text = "Prev Scene";
 nextscene_mc.onRelease = function() {
 nextScene();
 };

 prevscene_mc.onRelease = function() {
 prevScene();
 };

 init = true;
}
```

Asegúrese de insertar una acción `stop()` en el fotograma 1 de la escena 2.

Véase también

`Función prevScene`

Función Number

```
Number(expression) : Number
```

Convierte el parámetro *`expression`* en un número y devuelve un valor como se describe en la siguiente lista:

- Si *`expression`* es un número, el valor devuelto es *`expression`* .
- Si *`expression`* es un valor booleano, el valor devuelto es 1 si *`expression`* es `true`, 0 si *`expression`* es `false`.
- Si *`expression`* es una cadena, la función intenta analizar *`expression`* como un número decimal con un exponente final opcional (es decir, 1.57505e-3).
- Si *`expression`* es un `NaN`, el valor devuelto es `NaN`
- Si *`expression`* es `undefined`, el valor devuelto es : - En archivos publicados para Flash Player 6 o una versión anterior, el resultado es 0. - En archivos publicados para Flash Player 7 o una versión posterior, el resultado es `NaN`.

Disponibilidad: Flash Player 4; ActionScript 1.0

Parámetros

expression:Object - Expresión para convertir en un número. Los números o las cadenas que comienzan por 0x se interpretan como valores hexadecimales. Los números o las cadenas que comienzan por 0 se interpretan como valores octales.

Valor devuelto

Number - Un número o Number (no es un número).

Ejemplo

En el ejemplo siguiente se crea un campo de texto en el escenario en tiempo de ejecución:

```
this.createTextField("counter_txt", this.getNextHighestDepth(), 0, 0, 100,
  22);
counter_txt.autoSize = true;
counter_txt.text = 0;
function incrementInterval():Void {
 var counter:Number = counter_txt.text;
 // Without the Number() function, Flash would concatenate the value instead
 // of adding values. You could also use "counter_txt.text++;"
 counter_txt.text = Number(counter) + 1;
}
var intervalID:Number = setInterval(incrementInterval, 1000);
```

Véase también

Constante NaN, Number, Función parseInt, Función parseFloat

Función Object

```
Object([value:Object]) : Object
```

Crea un nuevo objeto vacío o convierte en un objeto el número, cadena o valor booleano especificado. Este comando equivale a crear un objeto mediante el constructor de Object (véase "Constructor de la clase Object").

Disponibilidad: Flash Player 5; ActionScript 1.0

Parámetros

value:Object [opcional] - Un número, cadena o valor booleano.

Valor devuelto

Object - Un objeto.

Ejemplo

En el ejemplo siguiente se crea un nuevo objeto vacío y, a continuación, el objeto se llena de valores:

```
var company:Object = new Object();
company.name = "Macromedia, Inc.";
company.address = "600 Townsend Street";
company.city = "San Francisco";
company.state = "CA";
company.postal = "94103";
for (var i in company) {
 trace("company."+i+" = "+company[i]);
}
```

Véase también

`Object`

on handler

`on(`*mouseEvent:Object*`) {`
// your statements here
`}`

Especifica el evento de ratón o la pulsación de tecla que activa una acción.

Disponibilidad: Flash Player 2; ActionScript 1.0

Parámetros

`mouseEvent``:Object` - Un *`mouseEvent`* es un desencadenante denominado *evento*. Cuando se produce el evento, se ejecutan las sentencias que aparecen a continuación entre llaves ({ }). Puede especificarse cualquiera de los siguientes valores para el parámetro *`mouseEvent`* :

- `press` Se presiona el botón del ratón cuando el puntero se encuentra sobre el botón.
- `release` Se suelta el botón del ratón cuando el puntero se encuentra sobre el botón.
- `releaseOutside` Cuando el puntero se encuentra sobre el botón, se presiona el botón del ratón y se desplaza fuera del área del botón justo antes de soltarlo. Los eventos `press` y `dragOut` siempre preceden al evento `releaseOutside`.
- `rollOut` El puntero se desplaza fuera el área del botón.
- `rollOver` El puntero del ratón se desplaza sobre el botón.
- `dragOut` Cuando el puntero se encuentra sobre el botón, se presiona el botón del ratón y se desplaza fuera del área del botón.

- `dragOver` Cuando el puntero se encuentra sobre el botón, se presiona el botón del ratón, a continuación se desplaza fuera del botón y vuelve a desplazarse sobre él.
- `keyPress "< key >"` Se presiona la tecla especificada del teclado. En la parte *`key`* del parámetro, especifique una constante de tecla, tal y como se muestra en la sugerencia de código del panel Acciones. Puede utilizar este parámetro para interceptar una pulsación de tecla, es decir, para sustituir cualquier comportamiento incorporado por la tecla especificada. El botón puede encontrarse en cualquier lugar de la aplicación, ya sea dentro o fuera del escenario. Una limitación de esta técnica es que no se puede aplicar el controlador `on()` en tiempo de ejecución, sino que debe hacerse durante la edición. Asegúrese de que selecciona Control > Deshabilitar métodos abreviados de teclado, ya que, de lo contrario, algunas teclas con comportamiento incorporado no se sustituirán cuando pruebe la aplicación empleando Control > Probar película.

Para ver una lista de las constantes de teclas, consulte la clase Key.

Ejemplo

En el script siguiente, la función `startDrag()` se ejecuta cuando se pulsa el ratón y el script condicional se ejecuta cuando se suelta el botón del ratón y se coloca el objeto:

```
on (press) {
 startDrag(this);
}
on (release) {
 trace("X:"+this._x);
 trace("Y:"+this._y);
 stopDrag();
}
```

Véase también

`Controlador onClipEvent`, `Key`

Controlador onClipEvent

```
onClipEvent(movieEvent:Object) {
// your statements here
}
```

Activa acciones definidas para una determinada instancia de un clip de película.

Disponibilidad: Flash Player 5; ActionScript 1.0

Parámetros

movieEvent:Object - El *movieEvent* es un desencadenante denominado ***evento*** . Cuando se produce el evento, se ejecutan las sentencias que aparecen a continuación entre llaves ({}). Puede especificarse cualquiera de los siguientes valores para el parámetro *movieEvent* :

- load La acción se inicia tan pronto como se crea una instancia del clip de película y aparece en la línea de tiempo.
- unload La acción se inicia en el primer fotograma después de eliminar el clip de película de la línea de tiempo. Las acciones asociadas con el evento de clip de película Unload se procesan antes de añadir ninguna acción al fotograma afectado.
- enterFrame La acción se activa continuamente a la velocidad de fotogramas del clip de película. Las acciones asociadas con el evento de clip enterFrame se procesan antes de añadir ninguna acción a los fotogramas afectados.
- mouseMove La acción se inicia cada vez que se mueve el ratón. Utilice las propiedades _xmouse e _ymouse para determinar la posición actual del ratón.
- mouseDown La acción se inicia cuando se presiona el botón izquierdo del ratón.
- mouseUp La acción se inicia cuando se suelta el botón izquierdo del ratón.
- keyDown La acción se inicia cuando se presiona una tecla. Utilice Key.getCode() para recuperar información sobre la última tecla presionada.
- keyUp La acción se inicia cuando se suelta una tecla. Utilice el método Key.getCode() para recuperar información sobre la última tecla presionada.
- data La acción se inicia cuando se reciben datos en una acción loadVariables() o loadMovie(). Si se especifica con una acción loadVariables(), el evento data sólo se produce una vez, concretamente cuando se carga la última variable. Si se especifica con una acción loadMovie(), el evento data se produce repetidas veces, cuando se recupera cada una de las porciones de datos.

Ejemplo

El ejemplo siguiente utiliza onClipEvent() con el evento de película keyDown y está diseñado para asociarse a un clip de película o a un botón. El evento de película keyDown suele emplearse con uno o varios métodos o propiedades del objeto Key. El script siguiente utiliza Key.getCode() para averiguar qué tecla ha pulsado el usuario; si la tecla pulsada coincide con la propiedad Key.RIGHT, la cabeza lectora se enviará al siguiente fotograma; si la tecla pulsada coincide con la propiedad Key.LEFT, la cabeza lectora se enviará al fotograma anterior.

```
onClipEvent (keyDown) {
 if (Key.getCode() == Key.RIGHT) {
 this._parent.nextFrame();
 } else if (Key.getCode() == Key.LEFT) {
```

```
 this._parent.prevFrame();
 }
}
```

El ejemplo siguiente utiliza `onClipEvent()` con los eventos de película `load` y `mouseMove`. Las propiedades `xmouse` e `ymouse` realizan un seguimiento de la posición del ratón cada vez que éste se mueve. Dicha posición aparece en el campo de texto que se crea en tiempo de ejecución.

```
onClipEvent (load) {
 this.createTextField("coords_txt", this.getNextHighestDepth(), 0, 0, 100,
  22);
 coords_txt.autoSize = true;
 coords_txt.selectable = false;
}
onClipEvent (mouseMove) {
 coords_txt.text = "X:"+_root._xmouse+",Y:"+_root._ymouse;
}
```

Véase también

`Key`, `_xmouse (propiedad MovieClip._xmouse)`, `_ymouse (propiedad MovieClip._ymouse)`, `on handler`, `Función updateAfterEvent`

Función ord

```
ord(character:String) : Number
```

Desfasada desde Flash Player 5. Esta función está desfasada y en su lugar deben utilizarse los métodos y propiedades de la clase String.

Convierte caracteres en números de código ASCII.

Disponibilidad: Flash Player 4; ActionScript 1.0

Parámetros

`character``:String` - El carácter que se va a convertir en un número del código ASCII.

Valor devuelto

`Number` - El número del código ASCII del carácter especificado.

Véase también

`String`, `charCodeAt (método String.charCodeAt)`

Función parseFloat

```
parseFloat(string:String) : Number
```

Convierte una cadena en un número de coma flotante. La función lee, o *analiza*, y devuelve los números en una cadena hasta que se alcanza un carácter que no forma parte del número inicial. Si la cadena no empieza con un número que pueda analizarse, `parseFloat()` devuelve `NaN`. Se ignoran los espacios en blanco que preceden a los enteros válidos, pues se trata de caracteres no numéricos finales.

Disponibilidad: Flash Player 5; ActionScript 1.0

Parámetros

`string``:String` - Cadena que se va a leer y convertir en un número de coma flotante.

Valor devuelto

`Number` - Un número o `Number` (no es un número).

Ejemplo

Los ejemplos siguientes utilizan la función `parseFloat()` para evaluar distintos tipos de números:

```
trace(parseFloat("-2")); // output: -2
trace(parseFloat("2.5")); // output: 2.5
trace(parseFloat(" 2.5")); // output: 2.5
trace(parseFloat("3.5e6")); // output: 3500000
trace(parseFloat("foobar")); // output: NaN
trace(parseFloat("3.75math")); // output: 3.75
trace(parseFloat("0garbage")); // output: 0
```

Véase también

`Constante NaN`, `Función parseInt`

Función parseInt

```
parseInt(expression:String, [radix:Number]) : Number
```

Convierte una cadena en un entero. Si la cadena especificada en los parámetros no puede convertirse en un número, la función devuelve `NaN`. Las cadenas que empiezan con 0x se interpretan como números hexadecimales. Los enteros que empiezan con 0 o que especifican una base de 8 se interpretan como números octales. Se ignoran los espacios en blanco que preceden a los enteros válidos, pues se trata de caracteres no numéricos finales.

Disponibilidad: Flash Player 5; ActionScript 1.0

Parámetros

expression:String - Cadena que se va a convertir en un entero.

radix:Number [opcional] - Entero que representa la base (radix) del número que se va a analizar. Los valores válidos van de 2 a 36.

Valor devuelto

Number - Un número o Number (no es un número).

Ejemplo

Los ejemplos de esta sección utilizan la función parseInt() para evaluar distintos tipos de números.

El ejemplo siguiente devuelve 3:

```
parseInt("3.5")
```

El ejemplo siguiente devuelve NaN:

```
parseInt("bar")
```

El ejemplo siguiente devuelve 4:

```
parseInt("4foo")
```

El ejemplo siguiente muestra una conversión hexadecimal que devuelve 1016:

```
parseInt("0x3F8")
```

El ejemplo siguiente muestra una conversión hexadecimal que utiliza el parámetro opcional *radix* y devuelve 1000:

```
parseInt("3E8", 16)
```

El ejemplo siguiente muestra una conversión binaria y devuelve 10, que es la representación decimal del 1010 binario:

```
parseInt("1010", 2)
```

Los ejemplo siguientes muestran análisis de números octales y devuelven 511, que es la representación decimal del 777 octal:

```
parseInt("0777")
parseInt("777", 8)
```

Véase también

, Función parseFloat

Función play

```
play() : Void
```

Mueve la cabeza lectora hacia delante en la línea de tiempo.

Disponibilidad: Flash Player 2; ActionScript 1.0

Ejemplo

En el ejemplo siguiente hay dos instancias de clip de película en el escenario con los nombres `stop_mc` y `play_mc`. El código ActionScript detiene la reproducción del archivo SWF cuando se hace clic en la instancia de clip de película `stop_mc`. La reproducción se reanuda cuando se hace clic en la instancia `play_mc`.

```
this.stop_mc.onRelease = function() {
 stop();
};
this.play_mc.onRelease = function() {
 play();
};
trace("frame 1");
```

Véase también

`Función gotoAndPlay`, `gotoAndPlay (método MovieClip.gotoAndPlay)`

Función prevFrame

```
prevFrame() : Void
```

Envía la cabeza lectora al fotograma anterior. Si el fotograma actual es el fotograma 1, la cabeza lectora no se mueve.

Disponibilidad: Flash Player 2; ActionScript 1.0

Ejemplo

Cuando el usuario hace clic en un botón llamado `myBtn_btn` y se inserta el código ActionScript siguiente en un fotograma de la línea de tiempo para ese botón, la cabeza lectora se envía al fotograma anterior:

```
stop();
this.myBtn_btn.onRelease = function(){
 prevFrame();
};
```

Véase también

Función nextFrame, prevFrame (método MovieClip.prevFrame)

Función prevScene

```
prevScene() : Void
```

Envía la cabeza lectora al fotograma 1 de la escena anterior.

Disponibilidad: Flash Player 2; ActionScript 1.0

Véase también

Función nextScene

Función print

```
print(target:Object, boundingBox:String) : Void
```

Imprime el clip de película de destino (`target`) según los límites especificados en el parámetro (`bmovie`, `bmax` o `bframe`). Si desea imprimir determinados fotogramas del clip de película de destino, añada una etiqueta de fotograma `#p` a dichos fotogramas. Aunque `print()` proporciona impresiones de mayor calidad que `printAsBitmap()`, no puede utilizarse para imprimir clips de película que utilicen transparencias alfa o efectos de color especiales.

Si utiliza `bmovie` para el parámetro `boundingBox` pero no asigna una etiqueta `#b` a un fotograma, el área de impresión se determina por el tamaño del escenario del clip de película cargado. (El clip de película cargado no hereda el tamaño del escenario del clip de película principal.)

Todos los elementos imprimibles de un clip de película deben cargarse completamente para poder iniciar la impresión.

La función de impresión de Flash Player admite las impresoras que son PostScript y las que no lo son. Las impresoras que no son PostScript convierten los vectores en mapas de bits.

Disponibilidad: Flash Player 4; ActionScript 1.0

Parámetros

`target``:Object` - Nombre de instancia de un clip de película que se va a imprimir. De forma predeterminada, es posible imprimir todos los fotogramas de la instancia de destino. Si desea imprimir determinados fotogramas del clip de película, asigne una etiqueta de fotograma `#p` a dichos fotogramas.

boundingBox:String - Modificador que establece el área de impresión del clip de película. Ponga el parámetro entre comillas (" o ') y especifique uno de los siguientes valores:

- bmovie Designa el recuadro de delimitación de un fotograma específico de un clip de película como área de impresión de todos los fotogramas imprimibles del clip de película. Asigne una etiqueta de fotograma #b al fotograma cuyo recuadro de delimitación desea utilizar como área de impresión.
- bmax Designa como área de impresión una composición de todos los recuadros de delimitación de todos los fotogramas imprimibles. Especifique bmax cuando los fotogramas imprimibles del clip de película tengan distintos tamaños.
- bframe Indica que el recuadro de delimitación de cada fotograma imprimible debe utilizarse como área de impresión para dicho fotograma, lo que supone un cambio en el área de impresión de cada fotograma y un redimensionamiento de los objetos para ajustarlos al área de impresión. Utilice bframe si tiene objetos de distintos tamaños en cada fotograma y desea que cada uno de ellos llene la página impresa.

Ejemplo

El ejemplo siguiente imprime todos los fotogramas imprimibles de holder_mc con un área de impresión definida por el recuadro de delimitación de cada fotograma:

```
this.createEmptyMovieClip("holder_mc", 999);
holder_mc.loadMovie("http://www.helpexamples.com/flash/images/image1.jpg");

this.myBtn_btn.onRelease = function() {
 print(this._parent.holder_mc, "bframe");
};
```

En el código ActionScript anterior puede reemplazar bframe por bmovie de modo que el área de impresión se defina mediante el recuadro de delimitación de un fotograma con la etiqueta de fotograma #b asociada.

Véase también

Función printAsBitmap, Función printAsBitmapNum, PrintJob, Función printNum

Función printAsBitmap

```
printAsBitmap(target:Object, boundingBox:String) : Void
```

Imprime el clip de película de destino como un mapa de bits (`target`) según los límites especificados en el parámetro (`bmovie`, `bmax` o `bframe`). Utilice `printAsBitmap()` para imprimir clips de película que contienen fotogramas con objetos que utilizan efectos de color o transparencias. La acción `printAsBitmap()` imprime con la mayor resolución posible de la impresora con el fin de mantener la calidad y la definición en la medida de lo posible.

Si el clip de película no contiene transparencias alfa o efectos de color, Macromedia recomienda utilizar `print()` para obtener resultados de mejor calidad.

Si utiliza `bmovie` para el parámetro `boundingBox` pero no asigna una etiqueta `#b` a un fotograma, el área de impresión se determina por el tamaño del escenario del clip de película cargado. (El clip de película cargado no hereda el tamaño del escenario del clip de película principal.)

Todos los elementos imprimibles de un clip de película deben cargarse completamente para poder iniciar la impresión.

La función de impresión de Flash Player admite las impresoras que son PostScript y las que no lo son. Las impresoras que no son PostScript convierten los vectores en mapas de bits.

Disponibilidad: Flash Player 4; ActionScript 1.0

Parámetros

`target``:Object` - Nombre de instancia del clip de película que se va a imprimir. De forma predeterminada, se imprimen todos los fotogramas del clip de película. Si desea imprimir determinados fotogramas del clip de película, asocie una etiqueta de fotograma `#p` a dichos fotogramas.

`boundingBox``:String` - Modificador que establece el área de impresión del clip de película. Ponga el parámetro entre comillas (" o ') y especifique uno de los siguientes valores:

- `bmovie` Designa el recuadro de delimitación de un fotograma específico de un clip de película como área de impresión de todos los fotogramas imprimibles del clip de película. Asigne una etiqueta de fotograma `#b` al fotograma cuyo recuadro de delimitación desea utilizar como área de impresión.
- `bmax` Designa como área de impresión una composición de todos los recuadros de delimitación de todos los fotogramas imprimibles. Especifique el parámetro `bmax` cuando los fotogramas imprimibles del clip de película tengan distintos tamaños.

- `bframe` Indica que el recuadro de delimitación de cada fotograma imprimible debe utilizarse como área de impresión de dicho fotograma. Esto supone un cambio en el área de impresión de cada fotograma y un redimensionamiento de los objetos para ajustarlos al área de impresión. Utilice `bframe` si tiene objetos de distintos tamaños en cada fotograma y desea que cada uno de ellos llene la página impresa.

Ejemplo

El ejemplo siguiente imprime todos los fotogramas imprimibles de `holder_mc` con un área de impresión definida por el recuadro de delimitación del fotograma:

```
this.createEmptyMovieClip("holder_mc", 999);
holder_mc.loadMovie("http://www.helpexamples.com/flash/images/image1.jpg");

this.myBtn_btn.onRelease = function() {
 printAsBitmap(this._parent.holder_mc, "bframe");
};
```

Véase también

`Función print`, `Función printAsBitmapNum`, `Función printNum`, `PrintJob`

Función printAsBitmapNum

```
printAsBitmapNum(level:Number, boundingBox:String) : Void
```

Imprime un nivel en Flash Player como un mapa de bits según los límites especificados en el parámetro (`bmovie`, `bmax` o `bframe`). Utilice `printAsBitmapNum()` para imprimir clips de película que contienen fotogramas con objetos que utilizan efectos de color o transparencias. La acción `printAsBitmapNum()` imprime con la mayor resolución posible de la impresora con el fin de mantener la mayor calidad y definición posibles. Para calcular el tamaño del archivo imprimible de un fotograma diseñado para imprimirse como un mapa de bits, multiplique la anchura de píxeles por la altura de píxeles y por la resolución de la impresora.

Si el clip de película no contiene transparencias alfa o efectos de color, utilice `printNum()` para obtener resultados de mejor calidad.

Si utiliza `bmovie` para el parámetro `boundingBox` pero no asigna una etiqueta `#b` a un fotograma, el área de impresión se determina por el tamaño del escenario del clip de película cargado. (El clip de película cargado no hereda el tamaño del escenario de la película principal.)

Todos los elementos imprimibles de un clip de película deben cargarse completamente para poder iniciar la impresión.

La función de impresión de Flash Player admite las impresoras que son PostScript y las que no lo son. Las impresoras que no son PostScript convierten los vectores en mapas de bits.

Disponibilidad: Flash Player 5; ActionScript 1.0

Parámetros

level:Number - Nivel en Flash Player que se va a imprimir. De forma predeterminada, se imprimen todos los fotogramas del nivel. Si desea imprimir determinados fotogramas del nivel, asigne una etiqueta de fotograma #p a dichos fotogramas.

boundingBox:String - Modificador que establece el área de impresión del clip de película. Ponga el parámetro entre comillas (" o ') y especifique uno de los siguientes valores:

- bmovie Designa el recuadro de delimitación de un fotograma específico de un clip de película como área de impresión de todos los fotogramas imprimibles del clip de película. Asigne una etiqueta de fotograma #b al fotograma cuyo recuadro de delimitación desea utilizar como área de impresión.
- bmax Designa como área de impresión una composición de todos los recuadros de delimitación de todos los fotogramas imprimibles. Especifique el parámetro bmax cuando los fotogramas imprimibles del clip de película tengan distintos tamaños.
- bframe Indica que el recuadro de delimitación de cada fotograma imprimible debe utilizarse como área de impresión de dicho fotograma. Esto supone un cambio en el área de impresión de cada fotograma y un redimensionamiento de los objetos para ajustarlos al área de impresión. Utilice bframe si tiene objetos de distintos tamaños en cada fotograma y desea que cada uno de ellos llene la página impresa.

Ejemplo

El ejemplo siguiente imprime el contenido del escenario cuando el usuario hace clic en el botón myBtn_btn. El área de impresión está definida por el recuadro de delimitación del fotograma.

```
myBtn_btn.onRelease = function(){
 printAsBitmapNum(0, "bframe")
};
```

Véase también

Función print, Función printAsBitmap, PrintJob, Función printNum

Función printNum

```
printNum(level:Number, boundingBox:String) : Void
```

Imprime el nivel en Flash Player según los límites especificados en el parámetro `boundingBox` (`bmovie`, `bmax` o `bframe`). Si desea imprimir determinados fotogramas del clip de película de destino, añada una etiqueta de fotograma `#p` a dichos fotogramas. Aunque al utilizar `printNum()` se obtienen impresiones de mayor calidad que al utilizar `printAsBitmapNum()`, no puede emplear `printNum()` para imprimir películas con transparencias alfa o efectos de color especiales.

Si utiliza `bmovie` para el parámetro `boundingBox` pero no asigna una etiqueta `#b` a un fotograma, el área de impresión se determina por el tamaño del escenario del clip de película cargado. (El clip de película cargado no hereda el tamaño del escenario de la película principal.)

Todos los elementos imprimibles de un clip de película deben cargarse completamente para poder iniciar la impresión.

La función de impresión de Flash Player admite las impresoras que son PostScript y las que no lo son. Las impresoras que no son PostScript convierten los vectores en mapas de bits.

Disponibilidad: Flash Player 5; ActionScript 1.0

Parámetros

`level``:Number` - Nivel en Flash Player que se va a imprimir. De forma predeterminada, se imprimen todos los fotogramas del nivel. Si desea imprimir determinados fotogramas del nivel, asigne una etiqueta de fotograma `#p` a dichos fotogramas.

`boundingBox``:String` - Modificador que establece el área de impresión del clip de película. Ponga el parámetro entre comillas (" o ') y especifique uno de los siguientes valores:

- `bmovie` Designa el recuadro de delimitación de un fotograma específico de un clip de película como área de impresión de todos los fotogramas imprimibles del clip de película. Asigne una etiqueta de fotograma `#b` al fotograma cuyo recuadro de delimitación desea utilizar como área de impresión.
- `bmax` Designa como área de impresión una composición de todos los recuadros de delimitación de todos los fotogramas imprimibles. Especifique el parámetro `bmax` cuando los fotogramas imprimibles del clip de película tengan distintos tamaños.
- `bframe` Indica que el recuadro de delimitación de cada fotograma imprimible debe utilizarse como área de impresión de dicho fotograma. Esto supone un cambio en el área de impresión de cada fotograma y un redimensionamiento de los objetos para ajustarlos al área de impresión. Utilice `bframe` si tiene objetos de distintos tamaños en cada fotograma y desea que cada uno de ellos llene la página impresa.

Véase también

Función print, Función printAsBitmap, Función printAsBitmapNum, PrintJob

Función random

```
random(value:Number) : Number
```

Desfasada desde Flash Player 5. Esta función está desfasada y en su lugar debe utilizarse `Math.random()`.

Devuelve un entero aleatorio entre 0 y uno menos que el entero especificado en el parámetro *value* .

Disponibilidad: Flash Player 4; ActionScript 1.0

Parámetros

value:Number - Un entero.

Valor devuelto

Number - Un entero aleatorio.

Ejemplo

El uso siguiente de `random()` devuelve un valor de 0, 1, 2, 3 o 4:`random(5);`

Véase también

random (método Math.random)

Función removeMovieClip

```
removeMovieClip(target:Object)
```

Elimina el clip de película especificado.

Disponibilidad: Flash Player 4; ActionScript 1.0

Parámetros

target:Object - La ruta de destino de una instancia de clip de película creada con `duplicateMovieClip()` o el nombre de instancia de un clip de película creada con `MovieClip.attachMovie()`, `MovieClip.duplicateMovieClip()` o `MovieClip.createEmptyMovieClip()`.

Ejemplo

El ejemplo siguiente crea un clip de película nuevo llamado `myClip_mc` y lo duplica. El segundo clip de película se llama `newClip_mc`. Las imágenes se cargan en ambos clips de película. Cuando se hace clic en un botón, `button_mc`, el clip de película duplicado se elimina del escenario.

```
this.createEmptyMovieClip("myClip_mc", this.getNextHighestDepth());
myClip_mc.loadMovie("http://www.helpexamples.com/flash/images/image1.jpg");
duplicateMovieClip(this.myClip_mc, "newClip_mc",
  this.getNextHighestDepth());
newClip_mc.loadMovie("http://www.helpexamples.com/flash/images/
  image1.jpg");
newClip_mc._x = 200;
this.button_mc.onRelease = function() {
 removeMovieClip(this._parent.newClip_mc);
};
```

Véase también

Función duplicateMovieClip, duplicateMovieClip (método MovieClip.duplicateMovieClip), attachMovie (método MovieClip.attachMovie), removeMovieClip (método MovieClip.removeMovieClip), createEmptyMovieClip (método MovieClip.createEmptyMovieClip)

Función setInterval

`setInterval(`*functionReference:Function*, *interval:Number*, [*param1:Object*, *param2*, ..., *paramN*]) *: Number*
`setInterval(`*objectReference:Object, methodName:String*, *interval:Number*, [*param1:Object*, *param2*, ..., *paramN*]) *: Number*

Llama a una función, un método o un objeto en intervalos periódicos durante la reproducción de un archivo SWF. Puede utilizar `setInterval()` para ejecutar cualquier función de forma repetitiva.

Utilice las siguientes sugerencias cuando trabaje con `setInterval()`:

- Identifique el ámbito de la función que se va a llamar.
- Identifique el ámbito donde se ha definido el ID de intervalo (el valor devuelto por `setInterval()`).
- Elimine los intervalos definidos anteriormente, antes de iniciar otros nuevos.

Estas sugerencias se describen más detalladamente en los párrafos que siguen.

Identifique el ámbito de la función que se va a llamar. Para identificar el ámbito de la función que se está llamando, pase el objeto en el que se puede ejecutar el método `setInterval()` (el objeto de ámbito) como el primer parámetro y el nombre de método que desea ejecutar como el segundo parámetro (como se muestra en la segunda firma). De esta forma se asegura que se ejecute el método adecuado desde el ámbito de la referencia de objeto pasada. Cuando el método se ejecuta de esta forma, puede utilizar variables de miembros en el objeto utilizando la palabra clave `this`.

Identifique el ámbito en el que se definió el identificador de intervalo. Para identificar el ámbito donde se definió el identificador de intervalo (`intervalId`), puede asignarlo a una variable de miembro en el objeto de ámbito que pase a `setInterval()`. De esta forma, la función llamada puede localizar el identificador de intervalo en `this.intervalId`.

Elimine intervalos definidos anteriormente. Para eliminar intervalos definidos anteriormente antes de iniciar otros nuevos, normalmente debe llamar a `clearInterval()` ***antes de*** llamar a `setInterval()`. De esta forma asegura que no va a sobrescribir o destruir de ninguna otra forma la variable `intervalId`, que es la única referencia al intervalo de ejecución actual. Para llamar a `clearInterval()` antes de llamar a `setInterval()`, el script de inicio y el script que se va a ejecutar deben tener acceso a `intervalId`, como se muestra en Ejemplos.

Nota: Llame siempre a `clearInterval()` si desea que el script deje de reproducirse indefinidamente.

Disponibilidad: Flash Player 6; ActionScript 1.0

Parámetros

`functionReference``:Function` - Una referencia a la función que se va a llamar.

`interval``:Number` - Tiempo en milisegundos entre llamadas la función `functionReference` o `methodName` pasada.

Si `interval` es menor que la velocidad de fotogramas del archivo SWF (por ejemplo, 10 fotogramas por segundo [fps] equivale a 100 milisegundos), la función de intervalo se llama lo más cerca posible en tiempo del valor de `interval`. La ejecución de scripts largos que consumen mucha memoria durante un intervalo provoca retrasos. Si la función que se va a llamar inicia un cambio en elementos visuales, debe utilizar la función `updateAfterEvent()` para que la pantalla se actualice con la frecuencia necesaria. Si `interval` es superior a la velocidad de fotogramas del archivo SWF, la función de intervalo sólo se llama cuando ha transcurrido `interval` *y* la cabeza lectora ha entrado en el siguiente fotograma; de esta forma se minimiza el impacto cada vez que se actualiza la pantalla.

param:Object [opcional] - Parámetros pasados a la función que se envió a functionReference o methodName. Cuando hay varios parámetros, deben ir separados por comas: *param1* , *param2* , ... , *paramN*

objectReference:Object - Un objeto que contiene el método especificado por methodName.

methodName:String - Un método que existe en el ámbito del objeto especificado por objectReference.

Valor devuelto

Number - Un entero que identifica el intervalo (el ID de intervalo), que puede pasar a clearInterval() para cancelar el intervalo.

Ejemplo

Ejemplo 1: El ejemplo siguiente muestra un mensaje a un intervalo de 20 milisegundos, hasta 10 veces, y a continuación elimina el intervalo. El ámbito de objeto, this, se pasa como el primer parámetro y el nombre de método, executeCallback, como el segundo. De esta forma, executeCallback() se ejecuta desde el mismo ámbito que el script que realiza la llamada.

```
var intervalId:Number;
var count:Number = 0;
var maxCount:Number = 10;
var duration:Number = 20;

function executeCallback():Void {
 trace("executeCallback intervalId: " + intervalId + " count: " + count);
 if(count >= maxCount) {
 clearInterval(intervalId);
 }
 count++;
}

intervalId = setInterval(this, "executeCallback", duration);
```

Ejemplo 2: El siguiente ejemplo es similar al primero, con la excepción de que llama a clearInterval() antes que setInterval(). De esta forma se evitan bucles no deseados y es especialmente importante en sistemas basados en eventos en los que el script de inicio puede ejecutarse varias veces antes de que se haya eliminado ningún intervalo.

```
var intervalId:Number;
var count:Number = 0;
var maxCount:Number = 10;
```

```
var duration:Number = 20;

function executeCallback():Void {
 trace("executeCallback intervalId: " + intervalId + " count: " + count);
 if(count >= maxCount) {
 clearInterval(intervalId);
 }
 count++;
}

function beginInterval():Void {
 if(intervalId != null) {
 trace("clearInterval");
 clearInterval(intervalId);
 }
 intervalId = setInterval(this, "executeCallback", duration);
}

beginInterval();
beginInterval();
beginInterval();
```

Ejemplo 3: El ejemplo siguiente muestra cómo pasar un argumento personalizado a la función que se va a llamar.

```
var intervalId:Number;
var count:Number = 0;
var maxCount:Number = 10;
var duration:Number = 20;
var colors:Array = new Array("red",
 "blue",
 "yellow",
 "purple",
 "green",
 "orange",
 "salmon",
 "pink",
 "lilac",
 "powder blue",
 "mint");

function executeCallback(param:String) {
 trace("executeCallback intervalId: " + intervalId + " count: " + count + "
  param: " + param);
 clearInterval(intervalId);
 if(count < maxCount) {
 count++;
 intervalId = setInterval(this, "executeCallback", duration,
  colors[count]);
```

```
 }
}

if(intervalId != null) {
 clearInterval(intervalId);
}

intervalId = setInterval(this, "executeCallback", duration, colors[count]);
```

Ejemplo 4: El ejemplo siguiente muestra cómo utilizar `setInterval()` correctamente desde una clase personalizada de ActionScript 2.0. Observe que como ocurre en ejemplos anteriores, `this` se pasa a la función `setInterval()` para que el método llamado se ejecute en el ámbito correcto.

```
class CustomClass {
 private var intervalId:Number;
 private var count:Number = 0;
 private var maxCount:Number = 10;
 private var duration:Number = 20;

 public function CustomClass():Void {
 beginInterval();
 }

 private function beginInterval():Void {
 if(intervalId != null) {
 trace("clearInterval");
 clearInterval(intervalId);
 }
 intervalId = setInterval(this, "executeCallback", duration);
 }

 public function executeCallback():Void {
 trace("executeCallback intervalId: " + intervalId + " count: " + count);
 if(count >= maxCount) {
 clearInterval(intervalId);
 }
 count++;
 }
}
```

En un nuevo documento, cree una nueva instancia de la nueva clase:

```
var custom:CustomClass = new CustomClass();
```

Véase también

Función clearInterval, Función updateAfterEvent, Sentencia class

Función setProperty

```
setProperty(target:Object, property:Object, expression:Object) : Void
```

Cambia un valor de propiedad de un clip de película durante la reproducción del clip de película.

Disponibilidad: Flash Player 4; ActionScript 1.0

Parámetros

target:Object - Ruta al nombre de instancia del clip de película cuya propiedad va a establecerse.

property:Object - Propiedad que va a establecerse.

expression:Object - El nuevo valor literal de la propiedad o una ecuación que da como resultado el nuevo valor de la propiedad.

Ejemplo

El código ActionScript siguiente crea un clip de película nuevo y carga una imagen en él. Se establecen las coordenadas _x e _y para el clip utilizando setProperty(). Cuando se hace clic en el botón llamado right_btn, la coordenada _x de un clip de película llamado params_mc se incrementa en 20 píxeles.

```
this.createEmptyMovieClip("params_mc", 999);
params_mc.loadMovie("http://www.helpexamples.com/flash/images/image1.jpg");
setProperty(this.params_mc, _y, 20);
setProperty(this.params_mc, _x, 20);
this.right_btn.onRelease = function() {
 setProperty(params_mc, _x, getProperty(params_mc, _x)+20);
};
```

Véase también

Función getProperty

Función showRedrawRegions

```
showRedrawRegions(enable:Boolean, [color:Number]) : Void
```

Ofrece la posibilidad al reproductor del depurador de resaltar las zonas de la pantalla que se van a redibujar (es decir, zonas no definitivas que se están actualizando). También se puede utilizar el resaltado con la opción de menú Mostrar regiones de redibujo.

Disponibilidad: Flash Player 8; ActionScript 1.0

Parámetros

enable:Boolean - Especifica si se van a activar (`true`) o desactivar (`false`) las regiones de redibujo. Cuando se define como `true`, se muestran los rectángulos de redibujo. Cuando se define como `false`, se eliminan los rectángulos de redibujo.

color:Number [opcional] - El color para dibujar. El valor predeterminado es rojo: 0xFF0000.

Ejemplo

En el ejemplo siguiente muestra la función `showRedrawRegions`.

```
var w:Number = 100;
var h:Number = 100;

var shape1:MovieClip = createShape("shape1");
shape1.onEnterFrame = function():Void {
 this._x += 5;
 this._y += 5;
}

var shape2:MovieClip = createShape("shape2");
shape2.onEnterFrame = function():Void {
 this._y += 5;
}

_global.showRedrawRegions(true);

function createShape(name:String):MovieClip {
 var mc:MovieClip = this.createEmptyMovieClip(name,
  this.getNextHighestDepth());
 mc.beginFill(0xFFCC00);
 mc.moveTo(200, 200);
 mc.curveTo(300, 200, 300, 100);
 mc.curveTo(300, 0, 200, 0);
 mc.curveTo(100, 0, 100, 100);
 mc.curveTo(100, 200, 200, 200);
 mc.endFill();
 return mc;
}
```

Función startDrag

`startDrag(target:Object, [lock:Boolean, left:Number, top:Number, right:Number, bottom:Number]) : Void`

Permite arrastrar el clip de película *target* durante la reproducción de la película. Sólo un clip de película puede arrastrarse al mismo tiempo. Tras ejecutar la operación `startDrag()`, todavía es posible arrastrar el clip de película hasta que se detenga de forma explícita mediante `stopDrag()` o hasta que se llame a una acción `startDrag()`) de otro clip de película.

Disponibilidad: Flash Player 4; ActionScript 1.0

Parámetros

target:Object - Ruta de destino del clip de película que se va a arrastrar.

`true` [opcional] - Valor booleano que especifica si el clip de película arrastrable está bloqueado en el centro de la posición del ratón (`false`) o en el punto donde el usuario hizo clic por primera vez en el clip de película (*lock*:Boolean).

left,*top*,*right*,*bottom*:Number [opcional] - Valores relativos a las coordenadas del elemento principal del clip de película, que especifican un rectángulo limitado para el clip de película.

Ejemplo

El ejemplo siguiente crea un clip de película, `pic_mc`, en tiempo de ejecución que los usuarios pueden arrastrar a cualquier ubicación asociando las acciones `startDrag()` y `stopDrag()` al clip de película. Se carga una imagen en `pic_mc` empleando la clase MovieClipLoader.

```
var pic_mcl:MovieClipLoader = new MovieClipLoader();
pic_mcl.loadClip("http://www.helpexamples.com/flash/images/image1.jpg",
 this.createEmptyMovieClip("pic_mc", this.getNextHighestDepth()));
var listenerObject:Object = new Object();
listenerObject.onLoadInit = function(target_mc) {
 target_mc.onPress = function() {
 startDrag(this);
 };
 target_mc.onRelease = function() {
 stopDrag();
 };
};
pic_mcl.addListener(listenerObject);
```

Véase también

Función stopDrag, _droptarget (propiedad MovieClip._droptarget), startDrag (método MovieClip.startDrag)

Función stop

```
stop() : Void
```

Detiene el archivo SWF que se está reproduciendo. El uso más común de esta acción es controlar clips de película mediante botones.

Disponibilidad: Flash Player 2; ActionScript 1.0

Véase también

Función gotoAndStop, gotoAndStop (método MovieClip.gotoAndStop)

Función stopAllSounds

```
stopAllSounds() : Void
```

Detiene todos los sonidos que se están reproduciendo en un archivo SWF sin detener la cabeza lectora. Se reanudará la reproducción de los sonidos que deben transmitirse mientras la cabeza lectora se mueve sobre los fotogramas donde se encuentran.

Disponibilidad: Flash Player 3; ActionScript 1.0

Ejemplo

El código siguiente crea un archivo de texto en el que aparece información ID3 sobre la canción. Se crea una instancia nueva del objeto Sound y se carga el MP3 en el archivo SWF. La información ID3 se extrae del archivo de sonido. Cuando el usuario hace clic en `stop_mc`,, se hace una pausa en el sonido. Cuando el usuario hace clic en `play_mc`,, la canción se reanuda desde la posición en pausa.

```
this.createTextField("songinfo_txt", this.getNextHighestDepth, 0, 0,
  Stage.width, 22);
var bg_sound:Sound = new Sound();
bg_sound.loadSound("yourSong.mp3", true);
bg_sound.onID3 = function() {
 songinfo_txt.text = "(" + this.id3.artist + ") " + this.id3.album + " - " +
  this.id3.track + " - "
 + this.id3.songname;
 for (prop in this.id3) {
 trace(prop+" = "+this.id3[prop]);
 }
 trace("ID3 loaded.");
};
this.play_mc.onRelease = function() {
 /* get the current offset. if you stop all sounds and click the play
  button, the MP3 continues from
 where it was stopped, instead of restarting from the beginning. */
```

```
 var numSecondsOffset:Number = (bg_sound.position/1000);
 bg_sound.start(numSecondsOffset);
};
this.stop_mc.onRelease = function() {
 stopAllSounds();
};
```

Véase también

Sound

Función stopDrag

```
stopDrag() : Void
```

Detiene la operación de arrastre actual.

Disponibilidad: Flash Player 4; ActionScript 1.0

Ejemplo

El código siguiente, situado en la línea de tiempo principal, detiene la acción de arrastre en la instancia de clip de película `my_mc` cuando el usuario suelta el botón del ratón:

```
my_mc.onPress = function () {
 startDrag(this);
}

my_mc.onRelease = function() {
 stopDrag();
}
```

Véase también

Función startDrag, _droptarget (propiedad MovieClip._droptarget), startDrag (método MovieClip.startDrag), stopDrag (método MovieClip.stopDrag)

Función String

```
String(expression:Object) : String
```

Devuelve una representación de cadena del parámetro especificado, tal y como se describe en la siguiente lista:

- Si *expression* es un número, la cadena devuelta es una representación de texto del número.
- Si *expression* es una cadena, la cadena devuelta es *expression* .

- Si *expression* es un objeto, el valor devuelto es una cadena que representa el objeto generado al llamar a la propiedad de cadena del objeto o al llamar a `Object.toString()` si no existe la propiedad.
- Si *expression* es un valor booleano, el valor devuelto es `"true"` o `"false"`.
- Si *expression* es un clip de película, el valor devuelto es la ruta de destino del clip de película en notación con barras (/).

Si *expression* es `undefined`, los valores devueltos son los siguientes:

- En los archivos publicados para Flash Player 6 y anterior, el resultado es una cadena vacía ("").
- En los archivos publicados para Flash Player 7 o posterior, el resultado es `undefined`.

Nota: ActionScript 2.0 no admite la notación con barras.

Disponibilidad: Flash Player 4; ActionScript 1.0

Parámetros

expression:`Object` - Expresión para convertir en una cadena.

Valor devuelto

`String` - Una cadena.

Ejemplo

En el ejemplo siguiente, utilice ActionScript para convertir expresiones especificadas en una cadena:

```
var string1:String = String("3");
var string2:String = String("9");
trace(string1+string2); // output: 39
```

Dado que ambos parámetros son cadenas, los valores se concatenan en lugar de añadirse.

Véase también

`toString (método Number.toString)`, `toString (método Object.toString)`, `String`, `Operador de delimitador de cadena (")`

Función substring

```
substring(string:String, index:Number, count:Number) : String
```

Desfasada desde Flash Player 5. Esta función está desfasada y en su lugar debe utilizarse `String.substr()`.

Extrae parte de una cadena. Esta función se basa en uno, mientras que los métodos del objeto String se basan en cero.

Disponibilidad: Flash Player 4; ActionScript 1.0

Parámetros

`string``:String` - La cadena de la que se va a extraer la nueva cadena.

`index``:Number` - El número del primer carácter que se va a extraer.

`count``:Number` - El número de caracteres que se va a incluir en la cadena extraída, excluido el carácter de índice.

Valor devuelto

`String` - La subcadena extraída.

Véase también

`substr (método String.substr)`

Función targetPath

```
targetpath(targetObject:Object) : String
```

Devuelve una cadena que contiene la ruta de destino de MovieClip, Button, TextField o Videoobject. Para devolver la ruta de destino se utiliza la notación con puntos (.). Para recuperar la ruta de destino en notación con barras (/), utilice la propiedad `_target`.

Disponibilidad: Flash Player 5; ActionScript 1.0

Parámetros

`targetObject``:Object` - Referencia (por ejemplo, `_root` o `_parent`) al objeto para el que se recupera la ruta de destino. Puede ser un objeto MovieClip, Button o TextField.

Valor devuelto

`String` - Cadena que contiene la ruta de destino del objeto especificado.

Ejemplo

El ejemplo siguiente realiza un seguimiento de la ruta de destino de un clip de película en cuanto se carga:

```
this.createEmptyMovieClip("myClip_mc", this.getNextHighestDepth());
trace(targetPath(myClip_mc)); // _level0.myClip_mc
```

Véase también

Función eval

Función tellTarget

```
tellTarget(target:String) {
statement(s);
}
```

Desfasada desde Flash Player 5. Macromedia recomienda el uso de notación con puntos (.) y la sentencia `with`.

Aplica las instrucciones especificadas en el parámetro *`statements`* a la línea de tiempo especificada en el parámetro *`target`*. La acción `tellTarget` resulta útil para los controles de navegación. Asigne `tellTarget` a botones que detengan o inicien clips de película en otros lugares del escenario. También puede hacer que los clips de película pasen a un determinado fotograma de ese clip. Por ejemplo, puede asignar `tellTarget` a botones que detienen o inician clips de película en el escenario o que hacen que clips de película pasen a un determinado fotograma.

En Flash 5 o versiones posteriores, puede utilizar la notación de puntos (.) en lugar de la acción `tellTarget`. Puede utilizar la acción `with` para emitir múltiples acciones en la misma línea de tiempo. Puede utilizar la acción `with` para referirse a cualquier objeto, mientras que la acción `tellTarget` sólo puede referirse a clips de película.

Disponibilidad: Flash Player 3; ActionScript 1.0

Parámetros

`target``:String` - Una cadena que especifica la ruta de destino de la línea de tiempo que se va a controlar.

`statement(s)` - Instrucciones que se ejecutarán si la condición es `true`.

Ejemplo

Esta sentencia tellTarget controla la instancia del clip de película ball en la línea de tiempo principal. El fotograma 1 de la instancia ball está en blanco y tiene una acción stop() para que no sea visible en el escenario. Cuando se hace clic en el botón con la acción siguiente, tellTarget indica a la cabeza lectora en ball que pase al fotograma 2, donde comienza la animación:

```
on(release) {
tellTarget("_parent.ball") {
gotoAndPlay(2);
}
}
```

El ejemplo siguiente utiliza la notación de puntos (.) para conseguir los mismos resultados:

```
on(release) {
_parent.ball.gotoAndPlay(2);
}
```

Si necesita emitir múltiples comandos a la instancia ball, puede emplear la acción with, como se muestra en la sentencia siguiente:

```
on(release) {
with(_parent.ball) {
gotoAndPlay(2);
_alpha = 15;
_xscale = 50;
_yscale = 50;
}
}
```

Véase también

`Sentencia with`

Función toggleHighQuality

```
toggleHighQuality()
```

Desfasada desde Flash Player 5. Esta función está desfasada y en su lugar debe utilizarse `_quality`.

Activa y desactiva el suavizado en Flash Player. El suavizado alisa los bordes de los objetos y ralentiza la reproducción del SWF. Esta acción afecta a todos los archivos SWF de Flash Player.

Disponibilidad: Flash Player 2; ActionScript 1.0

Ejemplo

El código siguiente puede aplicarse a un botón que, cuando se hace clic en él, activa o desactiva el suavizado:

```
on(release) {
toggleHighQuality();
}
```

Véase también

, Propiedad _quality

Función trace

```
trace(expression:Object)
```

Puede usar el Reproductor de depuración de Flash para capturar la salida de la función `trace()` y mostrar el resultado.

Utilice esta sentencia para registrar las notas de programación o para mostrar mensajes en el panel Salida mientras prueba un archivo SWF. Utilice el parámetro *expression* para comprobar si existe una condición o para mostrar valores en el panel Salida. La sentencia `trace()` es similar a la función `alert` de JavaScript.

Puede utilizar el comando Omitir acciones de trazado del cuadro de diálogo Configuración de publicación para eliminar acciones `trace()` del archivo SWF exportado.

Disponibilidad: Flash Player 4; ActionScript 1.0

Parámetros

expression:Object - Expresión que se va a evaluar. Cuando se abre un archivo SWF en la herramienta de edición Flash (mediante el comando Probar película), el valor del parámetro *expression* se muestra en el panel Salida.

Ejemplo

El ejemplo siguiente utiliza una sentencia `trace()` para mostrar en el panel Salida los métodos y propiedades del campo de texto creado dinámicamente que se llama `error_txt`:

```
this.createTextField("error_txt", this.getNextHighestDepth(), 0, 0, 100,
  22);
for (var i in error_txt) {
trace("error_txt."+i+" = "+error_txt[i]);
}
/* output:
error_txt.styleSheet = undefined
```

```
error_txt.mouseWheelEnabled = true
error_txt.condenseWhite = false
...
error_txt.maxscroll = 1
error_txt.scroll = 1
*/
```

Función unescape

```
unescape(string:String) : String
```

Evalúa el parámetro *x* como una cadena, descodifica la cadena con formato URL codificado (convirtiendo todas las secuencias hexadecimales en caracteres ASCII) y devuelve la cadena.

Disponibilidad: Flash Player 5; ActionScript 1.0

Parámetros

string:String - Cadena con secuencias hexadecimales que se van a interpretar literalmente.

Valor devuelto

String - Cadena descodificada de un parámetro URL codificado.

Ejemplo

El ejemplo siguiente muestra el proceso de conversión de escape en unescape:

```
var email:String = "user@somedomain.com";
trace(email);
var escapedEmail:String = escape(email);
trace(escapedEmail);
var unescapedEmail:String = unescape(escapedEmail);
trace(unescapedEmail);
```

El resultado siguiente se muestra en el panel Salida.

```
user@somedomain.com
user%40somedomain%2Ecom
user@somedomain.com
```

Función unloadMovie

```
unloadMovie(target:MovieClip) : Void
unloadMovie(target:String) : Void
```

Elimina un clip de película que se cargó mediante `loadMovie()` desde Flash Player. Para descargar un clip de película cargado mediante `loadMovieNum()`, use `unloadMovieNum()` en lugar de `unloadMovie()`.

Disponibilidad: Flash Player 3; ActionScript 1.0

Parámetros

`target``:Object` - Ruta de destino de un clip de película. Este parámetro puede ser una cadena (por ejemplo, "my_mc") o una referencia directa a la instancia de clip de película (por ejemplo, my_mc). Los parámetros que pueden aceptar más de un tipo de datos se engloban en el tipo `Object`.

Ejemplo

El ejemplo siguiente crea un clip de película nuevo llamado `pic_mc` y carga una imagen en él. Se carga utilizando la clase MovieClipLoader. Cuando se hace clic en la imagen, el clip de película se descarga del archivo SWF:

```
var pic_mcl:MovieClipLoader = new MovieClipLoader();
pic_mcl.loadClip("http://www.helpexamples.com/flash/images/image1.jpg",
 this.createEmptyMovieClip("pic_mc", this.getNextHighestDepth()));
var listenerObject:Object = new Object();
listenerObject.onLoadInit = function(target_mc) {
 target_mc.onRelease = function() {
 unloadMovie(pic_mc);
 /* or you could use the following, which refers to the movie clip
  referenced by 'target_mc'. */
 //unloadMovie(this);
 };
};
pic_mcl.addListener(listenerObject);
```

Véase también

loadMovie (método MovieClip.loadMovie), unloadClip (método MovieClipLoader.unloadClip)

Función unloadMovieNum

```
unloadMovieNum(level:Number) : Void
```

Elimina un archivo SWF o una imagen que se cargó mediante loadMovieNum() desde Flash Player. Para descargar un SWF o imagen cargados con `MovieClip.loadMovie()`, use `unloadMovie()` en lugar de `unloadMovieNum()`.

Disponibilidad: Flash Player 3; ActionScript 1.0

Parámetros

`level``:Number` - Nivel (`_level N`) de una película cargada.

Ejemplo

El ejemplo siguiente carga una imagen en un archivo SWF. Cuando se hace clic en `unload_btn` se elimina el contenido cargado.

```
loadMovieNum("yourimage.jpg", 1);
unload_btn.onRelease = function() {
 unloadMovieNum(1);
}
```

Véase también

`Función loadMovieNum`, `Función unloadMovie`, `loadMovie (método MovieClip.loadMovie)`

Función updateAfterEvent

```
updateAfterEvent() : Void
```

Actualiza la visualización (independientemente de los fotogramas por segundo establecidos para la película) cuando se llama en un controlador `onClipEvent()` o como parte de una función o método que se pasa a setInterval(). Flash ignora las llamadas a `updateAfterEvent` que no se incluyen en un controlador `onClipEvent()` o no son parte de una función o método que se pasa a setInterval(). Sólo funciona con determinados controladores de Mouse y MovieClip: los controladores mouseDown, mouseUp, mouseMove, keyDown y keyUp de la clase Mouse; los controladores onMouseMove, onMouseDown, onMouseUp, onKeyDown y onKeyUp de la clase MovieClip. No funciona con la clase Key.

Disponibilidad: Flash Player 5; ActionScript 1.0

Ejemplo

El ejemplo siguiente muestra cómo crear un cursor personalizado llamado `cursor_mc`. Se utiliza ActionScript para reemplazar el cursor del ratón por `cursor_mc`. A continuación se emplea `updateAfterEvent()` para actualizar continuamente el escenario de modo que el movimiento del cursor parezca fluido.

```
Mouse.hide();
cursor_mc.onMouseMove = function() {
 this._x = this._parent._xmouse;
 this._y = this._parent._ymouse;
 updateAfterEvent();
};
```

Véase también

`Controlador onClipEvent`, `Función setInterval`

Propiedades globales

Las propiedades globales están disponibles en cada script y son visibles en todas las líneas de tiempo y ámbitos del documento. Por ejemplo, las propiedades globales permiten acceder a las líneas de tiempo de otros clips de película cargados, tanto relativos (`_parent`) como absolutos (`_root`). También permiten restringir (`this`) o ampliar (`super`) el ámbito. Además, puede utilizar las propiedades globales para ajustar la configuración en tiempo de ejecución como la accesibilidad del lector de pantalla, la calidad de la reproducción y el tamaño de búfer de sonido.

Resumen de propiedades globales

Modificadores	Propiedad	Descripción
	`_accProps`	Permite controlar las opciones de accesibilidad del lector de pantalla para archivos SWF, clips de película, botones, campos de texto dinámico y campos de texto de entrada en tiempo de ejecución.
	`_focusrect`	Propiedad (global); especifica si aparece un rectángulo amarillo alrededor del botón o clip de película que se selecciona con el teclado.
	`_global`	Referencia al objeto global que aloja las clases principales de ActionScript, como String, Object, Math y Array.

Modificadores	Propiedad	Descripción
	`_highquality`	*Desfasada* desde Flash Player 5. Esta propiedad está desfasada y en su lugar debe utilizarse `_quality`. Especifica el nivel de visualización suavizada que se aplica al archivo SWF actual.
	`_level`	Referencia a la línea de tiempo raíz de `_level N`.
	`maxscroll`	*Desfasada* desde Flash Player 5. Esta propiedad está desfasada y en su lugar debe utilizarse `TextField.maxscroll`. Indica el número de la línea superior de texto visible en un campo de texto cuando la línea inferior del campo también está visible.
	`_parent`	Especifica o devuelve una referencia al clip de película u objeto que contiene el clip de película u objeto actual.
	`_quality`	Establece o recupera la calidad de representación que se utiliza para un clip de película.
	`_root`	Especifica o devuelve una referencia a la línea de tiempo del clip de película raíz.
	`scroll`	*Desfasada*desde Flash Player 5. Esta propiedad está desfasada y en su lugar debe utilizarse `TextField.scroll`. Controla la visualización de información en un campo de texto asociado a una variable.
	`_soundbuftime`	Establece el número de segundos para que se almacene en el búfer el flujo de sonido.
	`this`	Hace referencia a una instancia de clip de película u objeto.

Propiedad _accProps

`_accProps.`*propertyName*
instanceName._accProps.*propertyName*

Permite controlar las opciones de accesibilidad del lector de pantalla para archivos SWF, clips de película, botones, campos de texto dinámico y campos de texto de entrada en tiempo de ejecución. Estas propiedades sustituyen la configuración correspondiente disponible en el panel Accesibilidad durante la edición. Para que los cambios realizados en estas propiedades surtan efecto, debe llamar a Accessibility.updateProperties().

Para más información sobre el panel Accesibilidad, consulte "Panel Accesibilidad de Flash" en *Utilización de Flash*.

Para determinar si el reproductor se está ejecutando en un entorno que admite elementos de accesibilidad, utilice el método System.capabilities.hasAccessibility().

La tabla siguiente muestra el nombre y el tipo de datos de cada propiedad `_accProps`, su configuración equivalente en el panel Accesibilidad y los tipos de objetos a los que puede aplicarse la propiedad. El término *lógica inversa* significa que la configuración de la propiedad es inversa a la configuración correspondiente del panel Accesibilidad. Por ejemplo, establecer la propiedad `silent` en `true` equivale a desactivar la opción Permitir acceso a la película o Hacer que el objeto sea accesible.

Propiedad	Tipo de datos	Equivalente en el panel Accesibilidad	Se aplica a
`silent`	Boolean	Permitir acceso a la película/Hacer que el objeto sea accesible *(lógica inversa)*	Archivos SWF completos Clips de película Botones Texto dinámico Texto de entrada
`forceSimple`	Boolean	Hacer que los objetos secundarios sean accesibles *(lógica inversa)*	Archivos SWF completos Clips de película
`name`	String	Nombre	Archivos SWF completos Clips de película Botones Texto de entrada
`description`	String	Descripción	Archivos SWF completos Clips de película Botones Texto dinámico Texto de entrada
`shortcut`	String	Métodos abreviados	Clips de película Botones Texto de entrada

En el campo Método abreviado, utilice nombres con el formato Control+A. Cuando se añade un método abreviado del teclado al panel Accesibilidad, no se crea un método abreviado del teclado sino que simplemente se avisa a los lectores de pantalla de la existencia de un método abreviado. Para más información sobre la asignación de un método abreviado del teclado a un objeto accesible, consulte `Key.addListener()`.

Para especificar la configuración correspondiente al valor de Índice de fichas en el panel Accesibilidad, utilice las propiedades Button.tabIndex, MovieClip.tabIndex o TextField.tabIndex.

No es posible especificar el valor de Etiquetado automático en tiempo de ejecución.

Para hacer referencia al objeto `_accProps` que representa todo el documento de Flash, omita el parámetro `instanceName`. El valor de `_accProps` debe ser un objeto. Esto significa que si no existe ya un objeto `_accProps`, debe crearlo, tal y como se muestra en el siguiente ejemplo, para poder asignar valores a las propiedades del objeto `_accProps`:

```
if ( _accProps == undefined )
{
 _accProps = new Object();
}
_accProps.name = "My SWF file";
```

Cuando `_accProps` se utiliza sin el parámetro *`instanceName`* , los cambios realizados en las propiedades `_accProps` se aplican a todo el archivo SWF. Por ejemplo, el siguiente código establece la propiedad de `name` de accesibilidad de todo el documento SWF en la cadena `"Pet Store"` y, a continuación, llama a `Accessibility.updateProperties()` para provocar el cambio:

```
_accProps.name = "Pet Store";
Accessibility.updateProperties();
```

Por el contrario, el siguiente código establece la propiedad de `name` de un clip de película con el nombre de instancia `price_mc` en la cadena `"Price"`:

```
price_mc._accProps.name = "Price";
Accessibility.updateProperties();
```

Si especifica varias propiedades de accesibilidad, realice todo los cambios que pueda antes de llamar a `Accessibility.updateProperties()`, en lugar de realizar la llamada después de cada sentencia de propiedad, tal y como se muestra en el siguiente ejemplo:

```
_accProps.name = "Pet Store";

animal_mc._accProps.name = "Animal";
animal_mc._accProps.description = "Cat, dog, fish, etc.";

price_mc._accProps.name = "Price";
price_mc._accProps.description = "Cost of a single item";

Accessibility.updateProperties();
```

Si no especifica una propiedad de accesibilidad para un documento u objeto, se implementan los valores establecidos en el panel Accesibilidad.

Una vez especificada una propiedad de accesibilidad, no es posible recuperar el valor establecido en el panel Accesibilidad. Sin embargo, puede establecer el valor predeterminado de la propiedad (`false` para valores booleanos; cadenas vacías para valores de cadena) eliminando la propiedad del objeto `_accProps`, tal y como se muestra en el siguiente ejemplo:

```
 my_mc._accProps.silent = true; // set a property
// other code here
delete my_mc._accProps.silent; // revert to default value
```

El valor de _accProps debe ser un objeto. Esto significa que si no existe ya un objeto `_accProps`, debe crearlo para poder asignar indicaciones a las propiedades del objeto `_accProps`.

```
if (_accProps == undefined)
{
 _accProps = new Object();
}
_accProps.name = "My movie";
```

Disponibilidad: Flash Player 6,0,65,0; ActionScript 1.0

Parámetros

`propertyName``:Boolean or String` - Nombre de la propiedad de accesibilidad (véase la siguiente descripción de nombres válidos). *`instanceName`*

`instanceName``:String` - Nombre de instancia asignado a una instancia de un clip de película, botón, campo de texto dinámico o campo de texto de entrada. Para hacer referencia al objeto `_accProps` que representa todo el documento de Flash, omita el parámetro *`instanceName`* .

Ejemplo

Si cambia una imagen y desea actualizar su descripción de accesibilidad, puede utilizar el siguiendo código ActionScript:

```
my_mc.gotoAndStop(2);

if (my_mc._accProps == undefined ) {
 my_mc._accProps = new Object();
}

my_mc._accProps.name = "Photo of Mount Rushmore";
Accessibility.updateProperties();
```

Véase también

`isActive (Método Accessibility.isActive)`, `updateProperties (método Accessibility.updateProperties)`, `hasAccessibility (propiedad capabilities.hasAccessibility)`

Propiedad _focusrect

```
_focusrect = Boolean;
```

Especifica si aparece un rectángulo amarillo alrededor del botón o clip de película que se selecciona con el teclado. Si `_focusrect` se establece en su valor predeterminado o en `true`, aparece un rectángulo amarillo alrededor del botón o clip de película actualmente seleccionado mientras el usuario presiona la tecla de tabulación para desplazarse por los objetos de un archivo SWF. Especifique `false` si no desea mostrar el rectángulo amarillo. Es una propiedad global que puede sustituirse en determinadas instancias.

Si la propiedad global `_focusrect` se establece como `false`, el comportamiento predeterminado para todos los botones y clips de película es que la navegación mediante el teclado se limita a la tecla Tabulador. Todas las demás teclas, incluida Intro y las teclas de flecha, quedan anuladas. Para restablecer el desplazamiento completo con el teclado, es preciso configurar `_focusrect` con el valor `true`. Para restaurar las funciones completas del teclado para un botón o clip de película específico, puede sustituir esta propiedad global empleando `Button._focusrect` o `MovieClip._focusrect`.

Nota: Si utiliza un componente, FocusManager sustituye el control de selección de Flash Player, incluido el uso de esta propiedad global.

Disponibilidad: Flash Player 4; ActionScript 1.0

Ejemplo

El ejemplo siguiente muestra cómo ocultar el rectángulo amarillo alrededor de cualquier instancia en un archivo SWF cuando está seleccionada en una ventana del navegador. Cree algunos botones o clips de película y añada el código ActionScript siguiente al fotograma 1 de la línea de tiempo:

```
_focusrect = false;
```

Cambie la configuración de publicación a Flash Player 6 y compruebe el archivo SWF en una ventana del navegador seleccionando Archivo> Vista previa de publicación > HTML. Seleccione el SWF haciendo clic en él en la ventana del navegador y utilice la tecla Tabulador para seleccionar cada instancia. Si pulsa Intro o la barra espaciadora cuando `_focusrect` desactivado, no se invocará el controlador de eventos `onRelease` como ocurre cuando `_focusrect` está activado o es `true`.

Véase también

`_focusrect (propiedad Button._focusrect)`, `_focusrect (propiedad MovieClip._focusrect)`

Propiedad _global

```
_global.identifier
```

Referencia al objeto global que aloja las clases principales de ActionScript, como String, Object, Math y Array. Por ejemplo, puede crear una biblioteca que se expone como un objeto de ActionScript global, similar al objeto Math o Date. A diferencia de las funciones y variables declaradas localmente o en la línea de tiempo, las funciones y variables globales son visibles en todas las líneas de tiempo y ámbitos del archivo SWF, siempre y cuando no queden ocultas por identificadores con los mismos nombres en ámbitos interiores.

Nota: Al establecer el valor de una variable global, debe utilizarse el nombre completo de la variable, por ejemplo, _global.variableName. De lo contrario, se creará una variable local con el mismo nombre, que ocultará la variable global que se intenta establecer.

*Valor devuelto*Una referencia al objeto global que aloja las clases principales de ActionScript, como String, Object, Math y Array.

Disponibilidad: Flash Player 6; ActionScript 1.0

Ejemplo

El ejemplo siguiente crea una función de nivel superior, `factorial()`, , que está disponible para todas las líneas de tiempo y ámbitos en un archivo SWF:

```
_global.factorial = function(n:Number) {
 if(n <= 1) {
 return 1;
 }
 else {
 return n * factorial(n - 1);
 }
}

trace(factorial(1)); // 1
trace(factorial(2)); // 2
trace(factorial(3)); // 6
trace(factorial(4)); // 24
```

El ejemplo siguiente muestra cómo la no utilización del nombre de variable completo cuando se establece el valor de una variable global produce resultados imprevistos:

```
_global.myVar = "globalVariable";
trace(_global.myVar); // globalVariable
trace(myVar); // globalVariable

myVar = "localVariable";
trace(_global.myVar); // globalVariable
trace(myVar); // localVariable
```

Véase también

`Sentencia var`, `Sentencia set variable`

Propiedad _highquality

`_highquality`

Desfasada desde Flash Player 5. Esta propiedad está desfasada y en su lugar debe utilizarse `_quality`.

Especifica el nivel de visualización suavizada que se aplica al archivo SWF actual. Especifique 2 (mejor calidad) para aplicar alta calidad con el suavizado de mapa de bits siempre activado. Especifique 1 (alta calidad) para aplicar la visualización suavizada, que suaviza los mapas de bits si el archivo SWF no contiene animación. Especifique 0 (baja calidad) para evitar el suavizado.

Disponibilidad: Flash Player 4; ActionScript 1.0

Ejemplo

El siguiente ActionScript se sitúa en la línea de tiempo principal y establece la propiedad de calidad global para aplicar siempre suavizado de mapas de bits en archivos no animados.

```
_highquality = 1;
```

Véase también

`Propiedad _quality`

Propiedad _level

`_level`*`N`*

Referencia a la línea de tiempo raíz de `_level` *`N`* . Debe utilizar `loadMovieNum()` para cargar archivos SWF en Flash Player antes de utilizar la propiedad `_level` para buscarlos. También puede utilizar _level `_level` *`N`* para buscar un archivo SWF cargado en el nivel asignado por *`N`* .

El archivo SWF inicial cargado en la instancia de Flash Player se carga automáticamente en `_level0`. El archivo SWF que se encuentra en `_level0` establece la velocidad de fotogramas, el color de fondo y el tamaño de fotograma de todos los archivos SWF que se cargan posteriormente. A continuación, los archivos SWF se apilan en niveles de numeraciones superiores, por encima del archivo SWF de `_level0`.

Debe asignar un nivel a cada archivo SWF que cargue en Flash Player mediante `loadMovieNum()`. Puede asignar niveles en cualquier orden. Si asigna un nivel que ya contiene un archivo SWF (incluido `_level0`), se descargará el archivo SWF de dicho nivel y será sustituido por el nuevo archivo SWF.

Disponibilidad: Flash Player 4; ActionScript 1.0

Ejemplo

El ejemplo siguiente detiene la cabeza lectora en la línea de tiempo principal del archivo SWF sub.swf que se carga en `_level9`. El archivo sub.swf contiene animación y se encuentra en el mismo directorio que el documento que contiene el siguiente ActionScript:

```
loadMovieNum("sub.swf", 9);
myBtn_btn.onRelease = function() {
 _level9.stop();
};
```

Puede reemplazar `_level9.stop()` en el ejemplo anterior por el código siguiente:

```
_level9.gotoAndStop(5);
```

Esta acción envía la cabeza lectora en la línea de tiempo principal del archivo SWF en `_level9` al fotograma 5 en lugar de detener la cabeza lectora.

Véase también

`Función loadMovie`, `swapDepths (método MovieClip.swapDepths)`

Propiedad maxscroll

`variable_name``.maxscroll`

Desfasada desde Flash Player 5. Esta propiedad está desfasada y en su lugar debe utilizarse `TextField.maxscroll`.

Indica el número de la línea superior de texto visible en un campo de texto cuando la línea inferior del campo también está visible. La propiedad `maxscroll` funciona con la propiedad `scroll` para controlar cómo aparece la información en un campo de texto. Esta propiedad puede recuperarse, pero no modificarse.

Disponibilidad: Flash Player 4; ActionScript 1.0

Véase también

`maxscroll (propiedad TextField.maxscroll)`, `scroll (propiedad TextField.scroll)`

Propiedad _parent

```
_parent.property
_parent._parent.property
```

Especifica o devuelve una referencia al clip de película u objeto que contiene el clip de película u objeto actual. El objeto actual es el que contiene el código ActionScript que hace referencia a `_parent`. Utilice `_parent` para especificar una ruta de acceso relativa a los clips de película u objetos que se encuentran por encima del clip de película u objeto actual.

Disponibilidad: Flash Player 5; ActionScript 1.0

Ejemplo

En el ejemplo siguiente hay dos instancias de clip de película en el escenario con el nombre `square_mc`. En ese clip de película hay otro clip de película con el nombre de instancia `circle_mc`. El siguiente ActionScript permite modificar la instancia principal `circle_mc` (que es `square_mc`) cuando se hace clic en el círculo. Cuando se trabaja con direccionamiento relativo (empleando `_parent` en lugar de `_root`), puede resultar más fácil utilizar el botón Insertar ruta de destino en el panel Acciones en primer lugar.

```
this.square_mc.circle_mc.onRelease = function() {
 this._parent._alpha -= 5;
};
```

Véase también

Propiedad _root, Función targetPath

Propiedad _quality

```
_quality:String
```

Establece o recupera la calidad de representación que se utiliza para un clip de película. Las fuentes de dispositivo siempre se muestran dentadas y, por consiguiente, no se ven afectadas por la propiedad `_quality`.

La propiedad `_quality` puede configurarse con los valores que se describen en la tabla siguiente.

Valor	Descripción	Suavizado de gráficos	Suavizado de mapa de bits
`"LOW"`	Calidad de representación baja.	Los gráficos no se suavizan.	Los mapas de bits no se suavizan.
`"MEDIUM"`	Calidad de representación media. Esta ajuste resulta adecuado para películas que no contengan texto.	Los gráficos se suavizan empleando una cuadrícula de 2 x 2 píxeles.	Flash Player 8: Los mapas de bits se suavizan en función del parámetro `smoothing` utilizado en las llamadas a `MovieClip.attachBitmap()` y `MovieClip.beginBitmapFill()`. Flash Player 6 y 7: Los mapas de bits no se suavizan.
`"HIGH"`	Calidad de representación alta. Esta es la calidad de representación predeterminada de Flash.	Los gráficos se suavizan empleando una cuadrícula de 4 x 4 píxeles.	Flash Player 8: Los mapas de bits se suavizan en función del parámetro `smoothing` utilizado en las llamadas a `MovieClip.attachBitmap()` y `MovieClip.beginBitmapFill()`. Flash Player 6 y 7: Los mapas de bits se suavizan si el clip de película es estático.
`"BEST"`	Calidad de representación muy alta.	Los gráficos se suavizan empleando una cuadrícula de 4 x 4 píxeles.	Flash Player 8: Los mapas de bits se suavizan en función del parámetro `smoothing` utilizado en las llamadas a `MovieClip.attachBitmap()` y `MovieClip.beginBitmapFill()`. Cuando se define `smoothing` como "Best", el resultado se representa con una calidad muy alta cuando el clip de película se reduce, utilizando un algoritmo de valores medios. Esto puede ralentizar la representación, pero puede permitir, por ejemplo, miniaturas de alta calidad de imágenes de gran tamaño. Flash Player 6 y 7: Los mapas de bits se suavizan siempre.

Disponibilidad: Flash Player 5; ActionScript 1.0

Ejemplo

El ejemplo siguiente establece la calidad de representación como `LOW`:

```
_quality = "LOW";
```

Propiedad _root

`_root.`*`movieClip`*
`_root.`*`action`*
`_root.`*`property`*

Especifica o devuelve una referencia a la línea de tiempo del clip de película raíz. Si un clip de película tiene varios niveles, la línea de tiempo del clip de película raíz está en el nivel que contiene el script que se está ejecutando. Por ejemplo, si un script del nivel 1 evalúa `_root`, se devuelve `_level1`.

Especificar `_root` equivale a utilizar la notación con barras (`/`) desfasada para especificar una ruta absoluta en el nivel actual.

Nota: Si un clip de película que contiene _root se carga en otro clip de película, _root hace referencia a la línea de tiempo del clip de película que se carga y no a la línea de tiempo que contiene _root. Si desea asegurarse de que _root haga referencia a la línea de tiempo del clip de película cargado incluso cuando se cargue en otro clip de película, utilice MovieClip._lockroot.

Disponibilidad: Flash Player 5; ActionScript 1.0

Parámetros

`movieClip``:String` - Nombre de instancia del clip de película.

`action``:String` - Acción o método.

`property``:String` - Propiedad del objeto MovieClip.

Ejemplo

El ejemplo siguiente detiene la línea de tiempo del nivel que contiene el script que se está ejecutando:

```
_root.stop();
```

El ejemplo siguiente realiza un seguimiento de las variables e instancias en el ámbito de `_root`:

```
for (prop in _root) {
 trace("_root."+prop+" = "+_root[prop]);
}
```

Véase también

`_lockroot (propiedad MovieClip._lockroot)`, `Propiedad _parent`, `Función targetPath`

Propiedad scroll

`textFieldVariableName``.scroll = x`

*Desfasada*desde Flash Player 5. Esta propiedad está desfasada y en su lugar debe utilizarse `TextField.scroll`.

Controla la visualización de información en un campo de texto asociado a una variable. La propiedad `scroll` define dónde comienza el campo de texto a mostrar contenido; después de establecerla, Flash Player la actualiza a medida que el usuario se desplaza por el campo de texto. La propiedad `scroll` es útil para dirigir a los usuarios a un párrafo específico en un pasaje largo, o para crear campos de texto con desplazamiento. Esta propiedad puede recuperarse y modificarse.

Disponibilidad: Flash Player 4; ActionScript 1.0

Ejemplo

El código siguiente se adjunta a un botón Arriba que desplaza el campo de texto llamado myText:

```
on (release) {
myText.scroll = myText.scroll + 1;
}
```

Véase también

`maxscroll (propiedad TextField.maxscroll)`, `scroll (propiedad TextField.scroll)`

Propiedad _soundbuftime

`_soundbuftime`*:Number = integer*

Establece el número de segundos para que se almacene en el búfer el flujo de sonido. El valor predeterminado es 5 segundos.

Disponibilidad: Flash Player 4; ActionScript 1.0

Parámetros

integer: Number - Número de segundos transcurridos antes de que se inicie la reproducción sin interrupción del archivo SWF.

Ejemplo

El ejemplo siguiente reproduce un archivo MP3 en flujo y almacena el sonido en el búfer antes de reproducirlo para el usuario. Se crean dos campos de texto en tiempo de ejecución para contener un temporizador e información de depuración. La propiedad `_soundbuftime` se configura para almacenar el MP3 en búfer durante 10 segundos. Se crea una nueva instancia de objeto Sound para el MP3.

```
// create text fields to hold debug information.
this.createTextField("counter_txt", this.getNextHighestDepth(), 0, 0, 100,
  22);
this.createTextField("debug_txt", this.getNextHighestDepth(), 0, 20, 100,
  22);
// set the sound buffer to 10 seconds.
_soundbuftime = 10;
// create the new sound object instance.
var bg_sound:Sound = new Sound();
// load the MP3 sound file and set streaming to true.
bg_sound.loadSound("yourSound.mp3", true);
// function is triggered when the song finishes loading.
bg_sound.onLoad = function() {
 debug_txt.text = "sound loaded";
};
debug_txt.text = "sound init";
function updateCounter() {
 counter_txt.text++;
}
counter_txt.text = 0;
setInterval(updateCounter, 1000);
```

Propiedad this

```
this
```

Hace referencia a una instancia de clip de película u objeto. Cuando se ejecuta un script, `this` hace referencia a la instancia del clip de película que contiene el script. Cuando se llama a un método, `this` contiene una referencia al objeto que contiene el método al que se llama.

Dentro de un controlador de eventos `on()` asociado a un botón, `this` hace referencia a la línea de tiempo que contiene el botón. Dentro de un controlador de eventos `onClipEvent()` asociado a un clip de película, `this` hace referencia a la línea de tiempo del propio clip de película.

Dado que `this` se evalúa en el contexto del script que lo contiene, no puede utilizar `this` en un script para hacer referencia a una variable definida en un archivo de clase.

Disponibilidad: Flash Player 5; ActionScript 1.0

Ejemplo

Cree un archivo ActionsScript llamado ApplyThis.as e introduzca el siguiente código:

```
class ApplyThis {
 var str:String = "Defined in ApplyThis.as";
 function conctStr(x:String):String {
 return x+x;
 }
 function addStr():String {
 return str;
 }
}
```

A continuación, añada el siguiente código a un archivo FLA o a un archivo ActionScript distinto

```
var obj:ApplyThis = new ApplyThis();
var abj:ApplyThis = new ApplyThis();
abj.str = "defined in FLA or AS";
trace(obj.addStr.call(abj, null)); //output: defined in FLA or AS
trace(obj.addStr.call(this, null)); //output: undefined
trace(obj.addStr.call(obj, null)); //output: Defined in applyThis.as
```

Del mismo modo, para llamar a una función definida en una clase dinámica, debe utilizar `this` para invocar la función en el ámbito adecuado:

```
// incorrect version of Simple.as
/*
dynamic class Simple {
 function callfunc() {
 trace(func());
 }
}
*/
// correct version of Simple.as
dynamic class simple {
 function callfunc() {
 trace(this.func());
 }
}
```

Dentro del archivo FLA o del archivo ActionScript distinto, añada el siguiente código:

```
var obj:Simple = new Simple();
obj.num = 0;
obj.func = function() {
 return true;
```

```
};
obj.callfunc();
// output: true
```

El código anterior funciona cuando utiliza `this` en el método callfunc(). Sin embargo, aparecerá un error de sintaxis si ha utilizado la versión incorrecta de Simple.as, que se convirtió en comentario en el ejemplo anterior.

En el ejemplo siguiente, la palabra clave `this` hace referencia al objeto Circle:

```
function Circle(radius:Number):Void {
 this.radius = radius;
 this.area = Math.PI*Math.pow(radius, 2);
}
var myCircle = new Circle(4);
trace(myCircle.area);
```

En la sentencia siguiente asignada a un fotograma dentro de un clip de película, la palabra clave `this` hace referencia al clip de película actual.

```
// sets the alpha property of the current movie clip to 20
this._alpha = 20;
```

En la sentencia siguiente dentro de un controlador de MovieClip.onPress, la palabra clave `this` hace referencia al clip de película actual:

```
this.square_mc.onPress = function() {
 startDrag(this);
};
this.square_mc.onRelease = function() {
 stopDrag();
};
```

Véase también

`on handler`, `Controlador onClipEvent`

Operadores

Los operadores simbólicos son caracteres que especifican cómo combinar, comparar o modificar los valores de una expresión.

Resumen de operadores

Operador	Descripción
`+ (addition)`	Añade expresiones numéricas o concatena (combina) cadenas.
`+= (addition assignment)`	Asigna a *`expression1`* el valor de *`expression1 + expression2`* .

Operador	Descripción
`[] (array access)`	Inicializa una nueva matriz o matriz multidimensional con los elementos especificados (*a0* y así sucesivamente) o accede a elementos de una matriz.
`= (assignment)`	Asigna el valor de *expression2* (el parámetro a la derecha) a la variable, elemento de matriz o propiedad de *expression1* .
`& (bitwise AND)`	Convierte *expression1* y *expression2* en enteros de 32 bits sin signo y ejecuta una operación booleana AND en cada bit de los parámetros integer.
`&= (bitwise AND assignment)`	Asigna a *expression1* el valor de *expression1* & *expression2* .
`<< (bitwise left shift)`	Convierte *expression1* y *expression2* en enteros de 32 bits y desplaza todos los bits de *expression1* a la izquierda el número de posiciones especificado por el entero resultante de la conversión de *expression2* .
`<<= (bitwise left shift and assignment)`	Este operador ejecuta una operación de desplazamiento a la izquierda en modo bit (<<=) y almacena el contenido como un resultado en *expression1* .
`~ (bitwise NOT)`	También conocido como operador complementario de uno u operador complementario en modo bit.
`| (bitwise OR)`	Convierte *expression1* y *expression2* en enteros de 32 bits sin signo y devuelve un 1 en cada posición de bit donde los correspondientes bits tanto de *expression1* como de *expression2* sean 1.
`|= (bitwise OR assignment)`	Asigna a *expression1* el valor de *expression1* | *expression2* .
`>> (bitwise right shift)`	Convierte *expression1* y *expression2* en enteros de 32 bits y desplaza todos los bits de *expression1* a la derecha el número de posiciones especificado por el entero resultante de la conversión de *expression2* .
`>>= (bitwise right shift and assignment)`	Este operador ejecuta una operación de desplazamiento a la derecha en modo bit y almacena el contenido como un resultado en *expression1* .
`>>> (bitwise unsigned right shift)`	Igual que el operador de desplazamiento a la derecha en modo bit (>>), salvo que no conserva el signo de la *expression* original porque los bits a la izquierda se completan siempre con un 0. Los números de coma flotante se convierten en enteros al descartarse los dígitos después de la coma decimal.

Operador	Descripción
`>>>= (bitwise unsigned right shift and assignment)`	Ejecuta una operación de desplazamiento a la derecha en modo bit sin signo y almacena el contenido como un resultado en *expression1*.
`^ (bitwise XOR)`	Convierte *expression1* y *expression2* en enteros de 32 bits sin signo y devuelve un 1 en cada posición de bit donde los correspondientes bits de *expression1* o de *expression2* sean 1 (no ambos).
`^= (bitwise XOR assignment)`	Asigna a *expression1* el valor de *expression1* `^` *expression2*.
`/*..*/ (block comment delimiter)`	Indica una o varias líneas de comentarios de script.
`, (comma)`	Evalúa primero *expression1*, a continuación, *expression2* y así sucesivamente.
`add (concatenation (strings))`	*Desfasado* desde Flash Player 5. Macromedia recomienda que utilice el operador de suma (+) cuando cree contenido para Flash Player 5 o versiones posteriores. Este operador no se admite en Flash Player 8 ni en versiones posteriores. Concatena dos o más cadenas.
`?: (conditional)`	Indica a Flash que evalúe *expression1* y si el valor de *expression1* es `true`, el operador devuelve el valor de *expression2*; en caso contrario devuelve el valor de *expression3*.
`-- (decrement)`	Operador unario de decremento previo y decremento posterior que resta 1 de *expression*.
`/ (division)`	Divide *expression1* por *expression2*.
`/= (division assignment)`	Asigna a *expression1* el valor de *expression1* `/` *expression2*.
`. (dot)`	Se utiliza para navegar por las jerarquías de clips de película y acceder a variables, propiedades o clips de película anidados (secundarios).
`== (equality)`	Comprueba la igualdad de dos expresiones.
`eq (equality (strings))`	*Desfasado* desde Flash Player 5. Este operador está desfasado y en su lugar debe utilizarse `== (equality)`. Devuelve `true` si la representación de cadena de *expression1* es igual a la representación de cadena de *expression2*; en caso contrario devuelve `false`.
`> (greater than)`	Compara dos expresiones y determina si *expression1* es mayor que *expression2*; si lo es, el operador devuelve `true`.

Operador	Descripción
`gt (greater than (strings))`	*Desfasado* desde Flash Player 5. Este operador etá desfasado y en su lugar debe utilizarse > (mayor que). Compara la representación de cadena de *expression1* con la representación de cadena de *expression2* y devuelve `true` si *expression1* es mayor que *expression2* ; en caso contrario devuelve `false`.
`>= (greater than or equal to)`	Compara dos expresiones y determina si *expression1* es mayor o igual que *expression2* (`true`) o *expression1* es menor que *expression2* (`false`).
`ge (greater than or equal to (strings))`	*Desfasado* desde Flash Player 5. Este operador está desfasado y en su lugar debe utilizarse >= (mayor o igual que). Devuelve `true` si *expression1* es mayor o igual que *expression2* ; de lo contrario devuelve `false` .
`++ (increment)`	Operador unario de incremento previo e incremento posterior que añade 1 a *expression* .
`!= (inequality)`	Prueba el contrario exacto del operador de igualdad (==).
`<> (inequality)`	*Desfasado* desde Flash Player 5. Este operador está desfasado. Macromedia recomienda que utilice el operador `!= (inequality)`. Prueba el contrario exacto del operador de igualdad (==).
`instanceof`	Comprueba si `object` es una instancia de `classConstructor` o una subclase de `classConstructor`.
`< (less than)`	Compara dos expresiones y determina si *expression1* es menor que *expression2* ; si lo es, el operador devuelve `true`.
`lt (less than (strings))`	*Desfasado* desde Flash Player 5. Este operador está desfasado y en su lugar debe utilizarse < (menor que). Devuelve true si expression1 es menor que expression2, y false en caso contrario.
`<= (less than or equal to)`	Compara dos expresiones y determina si *expression1* es menor o igual que *expression2* ; si lo es, el operador devuelve `true`.
`le (less than or equal to (strings))`	*Desfasado* desde Flash Player 5. Este operador está desfasado y en su lugar debe utilizarse <= (menor o igual que). Devuelve `true` si *expression1* es menor o igual que *expression2* ; de lo contrario devuelve `false` .
`// (line comment delimiter)`	Indica el principio de un comentario de script.
`&& (logical AND)`	Ejecuta una operación booleana en los valores de ambas expresiones.

Operador	Descripción
`and (logical AND)`	*Desfasado* desde Flash Player 5. Macromedia recomienda el uso del operador lógico AND (`&&`). Ejecuta una operación de AND lógica (`&&`) en Flash Player 4.
`! (logical NOT)`	Invierte el valor booleano de una variable o expresión.
`not (logical NOT)`	*Desfasado* desde Flash Player 5. Este operador está desfasado y en su lugar debe utilizarse `! (logical NOT)`. Ejecuta una operación NOT (!) lógica en Flash Player 4.
`|| (logical OR)`	Evalúa *`expression1`* (la expresión en la parte izquierda del operador) y devuelve `true` si la expresión da como resultado `true`.
`or (logical OR)`	*Desfasado* desde Flash Player 5. Este operador está desfasado y en su lugar debe utilizarse `|| (logical OR)`. Evalúa *`condition1`* y *`condition2`* , y si alguna es `true`, toda la expresión es `true`.
`% (modulo)`	Calcula el resto de *`expression1`* dividido entre *`expression2`* .
`%= (modulo assignment)`	Asigna a *`expression1`* el valor de *`expression1 % expression2`* .
`* (multiplication)`	Multiplica dos expresiones numéricas.
`*= (multiplication assignment)`	Asigna a *`expression1`* el valor de *`expression1 * expression2`* .
`new`	Crea un objeto nuevo, inicialmente anónimo, y llama a la función identificada por el parámetro `constructor`.
`ne (not equal (strings))`	*Desfasado* desde Flash Player 5. Este operador está desfasado y en su lugar debe utilizarse `!= (inequality)`. Devuelve `true` si *`expression1`* no igual que *`expression2`* ; de lo contrario devuelve `false` .
`{} (object initializer)`	Crea un nuevo objeto y lo inicializa con los pares de propiedades *`name`* y *`value`* .
`() (parentheses)`	Ejecuta una operación de agrupación en uno o varios parámetros, lleva a cabo una evaluación secuencial de las expresiones o rodea uno o varios parámetros y los pasa como parámetros a una función fuera del paréntesis.
`=== (strict equality)`	Comprueba la igualdad de dos expresiones; el operador de igualdad estricta (===) hace lo mismo que el operador de igualdad (==) con la diferencia de que los tipos de datos no se convierten.
`!== (strict inequality)`	Prueba el contrario exacto del operador de igualdad estricta (===).

Operador	Descripción
`" (string delimiter)`	Si se utilizan antes y después de caracteres, las comillas (") indican que los caracteres tienen un valor literal y se consideran una *cadena* y no una variable ni un valor numérico ni otro elemento de ActionScript.
`- (subtraction)`	Se emplea para negar o restar.
`-= (subtraction assignment)`	Asigna a *`expression1`* el valor de *`expression1 - expression2`* .
`: (type)`	Se utiliza en la técnica "strict data typing"; este operador especifica el tipo de variable, el tipo de devolución de función o el tipo de parámetro de función.
`typeof`	El operador `typeof` evalúa `expression` y devuelve una cadena que especifica si la expresión es `String`, `MovieClip`, `Object`, `Function`, `Number`, o valor `Boolean`.
`void`	El operador `void` evalúa una expresión y, a continuación, descarta su valor, devolviendo `undefined`.

Operador suma (+)

`expression1 + expression2`

Añade expresiones numéricas o concatena (combina) cadenas. Si una expresión es una cadena, todas las demás expresiones se convierten en cadenas y se concatenan. Si ambas expresiones son enteros, la suma es un entero; si al menos una de las expresiones es un número de coma flotante, la suma es un número de coma flotante.

Disponibilidad: Flash Player 4; ActionScript 1.0

Operandos

`expression1` - Un número o cadena.

`expression2 : Number` - Un número o cadena.

Valor devuelto

`Object` - Una cadena, un entero o un número de coma flotante.

Ejemplo

Sintaxis 1: El ejemplo siguiente concatena dos cadenas y muestra el resultado en el panel Salida.

```
var name:String = "Cola";
var instrument:String = "Drums";
```

```
trace(name + " plays " + instrument); // output: Cola plays Drums
```

Sintaxis 2: Esta sentencia añade los enteros 2 y 3 y muestra el entero resultante, 5, en el panel Salida:

```
trace(2 + 3); // output: 5
```

Esta sentencia añade los números de coma flotante 2,5 y 3,25 y muestra el número de coma flotante resultante, 5,75, en el panel Salida:

```
trace(2.5 + 3.25); // output: 5.75
```

Sintaxis 3: Las variables asociadas con los campos de texto dinámico y de entrada tienen un tipo de datos String. En el ejemplo siguiente, la variable `deposit` es un campo de texto de entrada en el escenario. Cuando un usuario introduce una cantidad de depósito, el script intenta añadir `deposit` a `oldBalance`. Sin embargo, dado que `deposit` es un tipo de datos String, el script concatena (se combina para formar una cadena) los valores variables en lugar de sumarlos.

```
var oldBalance:Number = 1345.23;
var currentBalance = deposit_txt.text + oldBalance;
trace(currentBalance);
```

Por ejemplo, si un usuario introduce 475 en el campo de texto de depósito, la sentencia trace() envía el valor 4751345.23 al panel Salida. Para corregir esto, utilice la función Number() para convertir la cadena en un número, como se indica a continuación:

```
var oldBalance:Number = 1345.23;
var currentBalance:Number = Number(deposit_txt.text) + oldBalance;
trace(currentBalance);
```

El ejemplo siguiente muestra cómo no se calculan las sumas numéricas a la derecha de una expresión de cadena:

```
var a:String = 3 + 10 + "asdf";
trace(a); // 13asdf
var b:String = "asdf" + 3 + 10;
trace(b); // asdf310
```

Operador asignación de suma (+=)

`expression1 += expression2`

Asigna a *`expression1`* el valor de *`expression1 + expression2`*. Por ejemplo, las dos sentencias siguientes tienen el mismo resultado:

```
x += y;
x = x + y;
```

Este operador realiza además una concatenación de cadenas. Todas las reglas del operador de suma (+) se aplican al operador de asignación de suma `(+=)`.

Disponibilidad: Flash Player 4; ActionScript 1.0

Operandos

`expression1 : Number` - Un número o cadena.

`expression2 : Number` - Un número o cadena.

Valor devuelto

`Number` - El resultado de la suma.

Ejemplo

Sintaxis 1: Este ejemplo utiliza el operador += con una expresión de cadena y envía "My name is Gilbert" al panel Salida.

```
var x1:String = "My name is ";
x1 += "Gilbert";
trace(x1); // output: My name is Gilbert
```

Sintaxis 2: El ejemplo siguiente muestra un uso numérico del operador de asignación de suma (+=):

```
var x:Number = 5;
var y:Number = 10;
x += y;
trace(x); // output: 15
```

Véase también

`Operador suma (+)`

Operador acceso a matriz ([])

myArray = [*a0, a1,...aN*]
myArray[*i*] = *value*
myObject [*propertyName*]

Inicializa una nueva matriz o matriz multidimensional con los elementos especificados (`a0` y así sucesivamente) o accede a elementos de una matriz. El operador de acceso a matriz permite establecer dinámicamente y recuperar nombres de instancia, variable y objeto. También permite acceder a propiedades de objeto.

Sintaxis 1: Una matriz es un objeto cuyas propiedades se denominan *elementos*, que se identifican mediante un número denominado *índice*. Cuando se crea una matriz, se rodean los elementos con el operador de acceso a matriz ([]) (o *corchetes*). Una matriz puede contener elementos de diversos tipos. Por ejemplo, la siguiente matriz, denominada `employee`, tiene tres elementos; el primero es un número y los dos siguientes son cadenas (entre comillas):

```
var employee:Array = [15, "Barbara", "Jay"];
```

Puede anidar los corchetes para simular matrices multidimensionales. Puede anidar matrices hasta un máximo de 256 niveles de profundidad. El siguiente código crea una matriz denominada `ticTacToe`, con tres elementos; cada elemento es a su vez una matriz con tres elementos:

```
var ticTacToe:Array = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]; // Select Debug > 
  List Variables in test mode
// to see a list of the array elements.
```

Sintaxis 2: Ponga entre corchetes ([]) el índice de cada elemento para acceder a él directamente; puede añadir un nuevo elemento a una matriz o puede cambiar o recuperar el valor de un elemento existente. El primer índice de una matriz es siempre 0, tal y como se muestra en el siguiente ejemplo:

```
var my_array:Array = new Array();
my_array[0] = 15;
my_array[1] = "Hello";
my_array[2] = true;
```

Puede utilizar corchetes ([]) para añadir un cuarto elemento, tal y como se muestra en el siguiente ejemplo:

```
my_array[3] = "George";
```

Puede utilizar corchetes ([]) para acceder a un elemento de una matriz multidimensional. El primer conjunto de corchetes identifica el elemento en la matriz original y el segundo conjunto identifica el elemento en la matriz anidada. Las siguientes líneas de código envían el número 6 al panel Salida.

```
var ticTacToe:Array = [[1, 2, 3], [4, 5, 6], [7, 8, 9]];
trace(ticTacToe[1][2]);// output: 6
```

Sintaxis 3: Puede utilizar el operador de acceso a matriz ([]) en lugar de la función `eval()` para establecer dinámicamente y recuperar los valores de los nombres de clip de película o cualquier propiedad de un objeto. La siguiente línea de código envía el número 6 al panel Salida.

```
name["mc" + i] = "left_corner";
```

Disponibilidad: Flash Player 4; ActionScript 1.0

Operandos

`myArray : Object` - *`myArray`* Nombre de una matriz.

`a0, a1,...aN : Object` - *`a0,a1,...aN`* Elementos de una matriz; cualquier tipo nativo o instancia de objeto, incluidas las matrices anidadas.

`i : Number` - *`i`* Índice de número entero mayor o igual que 0.

`myObject : Object` - *`myObject`* Nombre de un objeto.

`propertyName : String` - *`propertyName`* Cadena que denomina una propiedad del objeto.

Valor devuelto

`Object` -

Sintaxis 1: Una referencia a una matriz.

Sintaxis 2: Un valor de la matriz, ya sea un tipo nativo o una instancia de objeto (incluida una instancia de Array).

Sintaxis 3: Una propiedad del objeto, ya sea un tipo nativo o una instancia de objeto (incluida una instancia de Array).

Ejemplo

El ejemplo siguiente muestra dos formas de crear un nuevo objeto Array vacío; la primera línea utiliza corchetes ([]):

```
var my_array:Array = [];
var my_array:Array = new Array();
```

El ejemplo siguiente crea una matriz llamada `employee_array` y utiliza la sentencia trace() para enviar los elementos al panel Salida. En la cuarta línea, se cambia un elemento de la matriz, y la quinta línea envía la matriz recién modificada al panel Salida:

```
var employee_array = ["Barbara", "George", "Mary"];
trace(employee_array); // output: Barbara,George,Mary
employee_array[2] = "Sam";
trace(employee_array); // output: Barbara,George,Sam
```

En el ejemplo siguiente, se evalúa la expresión entre corchetes (`"piece" + i`) y el resultado se utiliza como nombre de la variable que se va a recuperar del clip de película `my_mc`: En este ejemplo, la variable `i` debe encontrarse en la misma línea de tiempo que el botón. Si la variable `i` es igual a 5, por ejemplo, el valor de la variable `piece5` en el clip de película `my_mc` se mostrará en el panel Salida:

```
myBtn_btn.onRelease = function() {
 x = my_mc["piece"+i];
 trace(x);
};
```

En el ejemplo siguiente, se evalúa la expresión entre corchetes y el resultado se utiliza como nombre de la variable que se va a recuperar del clip de película `name_mc`:

```
name_mc["A" + i];
```

Si conoce la sintaxis de barras de Flash 4 ActionScript, puede emplear la función `eval()` para conseguir el mismo resultado:

```
eval("name_mc.A" & i);
```

Puede utilizar el siguiente ActionScript para reproducir indefinidamente todos los objetos del ámbito `_root`, algo que resulta útil para depurar:

```
for (i in _root) {
 trace(i+": "+_root[i]);
}
```

También puede utilizar el operador de acceso a una matriz ([]) del lado izquierdo de una sentencia de asignación para establecer dinámicamente nombres de instancia, variable y objeto:

```
employee_array[2] = "Sam";
```

Véase también

`Array`, `Object`, `Función eval`

Operador asignación (=)

`expression1 = expression2`

Asigna el valor de *`expression2`* (el parámetro a la derecha) a la variable, elemento de matriz o propiedad de *`expression1`* . La asignación puede ser por valor o por referencia. La asignación por valor copia el valor actual de *`expression2`* y lo almacena en *`expression1`* . La asignación por valor se utiliza cuando se asigna un número o un literal de cadena a una variable. La asignación por referencia almacena una referencia a *`expression2`* en *`expression1`* . La asignación por referencia suele utilizarse con operador `new`. El uso del operador `new` crea un objeto en la memoria y se asigna a la variable una referencia a dicha ubicación en la memoria.

Disponibilidad: Flash Player 4; ActionScript 1.0

Operandos

`expression1 : Object` - Una variable, un elemento de una matriz o una propiedad de un objeto.

`expression2 : Object` - Un valor de cualquier tipo.

Valor devuelto

`Object` - El valor asignado, *`expression2`* .

Ejemplo

El ejemplo siguiente utiliza asignación por valor para asignar el valor de 5 a la variable `x`.

```
var x:Number = 5;
```

El ejemplo siguiente utiliza asignación por valor para asignar el valor "hello" a la variable x:

```
var x:String;
x = " hello ";
```

El ejemplo siguiente utiliza asignación por referencia para crear la variable moonsOfJupiter, que contiene una referencia a un objeto Array recién creado. A continuación se utiliza asignación por valor para copiar el valor "Callisto" al primer elemento de la matriz a la que hace referencia la variable moonsOfJupiter:

```
var moonsOfJupiter:Array = new Array();
moonsOfJupiter[0] = "Callisto";
```

El ejemplo siguiente utiliza asignación por referencia para crea un objeto nuevo y asignar una referencia a ese objeto a la variable mercury. A continuación se utiliza asignación por valor para asignar el valor de 3030 a la propiedad diameter del objeto mercury:

```
var mercury:Object = new Object(); mercury.diameter = 3030; // in miles
trace (mercury.diameter); // output: 3030
```

El ejemplo siguiente se basa en el ejemplo anterior creando una variable llamada merkur ("mercury" en alemán) y asignándole el valor de mercury. De este modo se crean dos variables que hacen referencia al mismo objeto en la memoria, lo que significa que puede utilizar cualquiera de ellas para acceder a las propiedades del objeto. Seguidamente podemos cambiar la propiedad diameter para utilizar kilómetros en lugar de millas:

```
var merkur:Object = mercury;
merkur.diameter = 4878; // in kilometers
trace (mercury.diameter); // output: 4878
```

Véase también

Operador de igualdad (==)

Operador & AND en modo bit

expression1 & *expression2*

Convierte *expression1* y *expression2* en enteros de 32 bits sin signo y ejecuta una operación booleana AND en cada bit de los parámetros integer. Los números de coma flotante se convierten en enteros al descartarse los dígitos después de la coma decimal. El resultado es un nuevo entero de 32 bits.

Los enteros positivos se convierten en un valor hexadecimal sin signo con un valor máximo de 4294967295 o 0xFFFFFFFF; se descartan los dígitos más significativos de los valores mayores que el máximo cuando se convierten, de forma que el valor siga siendo de 32 bits. Los números negativos se convierten en un valor hexadecimal sin signo a través de la notación complementaria del dos, siendo el mínimo -2147483648 o 0x800000000; los números menores que el mínimo se convierten en el complemento del dos con una mayor precisión y también se descartan sus dígitos más significativos.

El valor devuelto se interpreta como un número complementario del dos con signo, de forma que el valor devuelto sea un entero del intervalo comprendido entre -2147483648 y 2147483647.

Disponibilidad: Flash Player 5; ActionScript 1.0

Operandos

`expression1 : Number` - Un número.

`expression2 : Number` - Un número.

Valor devuelto

`Number` - El resultado de la operación en modo bit.

Ejemplo

El ejemplo siguiente compara la representación de bits de los números y devuelve 1 sólo si los dos bits en la misma posición son 1. En este ActionScript, añade 13 (binario 1101) y 11 (binario1011) y devuelve 1 sólo en la posición en la que los números tienen un 1.

```
var insert:Number = 13;
var update:Number = 11;
trace(insert & update); // output : 9 (or 1001 binary)
```

En los números 13 y 11 el resultado es 9 porque sólo la primera y la última posición de ambos números tienen el número 1.

Los ejemplos siguientes muestran el comportamiento de la conversión del valor devuelto:

```
trace(0xFFFFFFFF); // 4294967295
trace(0xFFFFFFFF & 0xFFFFFFFF); // -1
trace(0xFFFFFFFF & -1); // -1
trace(4294967295 & -1); // -1
trace(4294967295 & 4294967295); // -1
```

Véase también

Operador &asignación de AND en modo bit (=), Operador XOR en modo bit (^), Operador asignación de XOR en modo bit (^=), Operador OR en modo bit (|), Operador asignación de OR en modo bit (|=), Operador NOT en modo bit (~)

Operador &asignación de AND en modo bit (=)

expression1 &= *expression2*

Asigna a *expression1* el valor de *expression1* & *expression2* . Por ejemplo, las dos expresiones siguientes son equivalentes:

```
x &= y;
x = x & y;
```

Disponibilidad: Flash Player 5; ActionScript 1.0

Operandos

`expression1 : Number` - Un número.

`expression2 : Number` - Un número.

Valor devuelto

`Number` - El valor de *expression1* & *expression2* .

Ejemplo

El ejemplo siguiente asigna el valor de `9` a `x`:

```
var x:Number = 15;
var y:Number = 9;
trace(x &= y); // output: 9
```

Véase también

Operador & AND en modo bit, Operador XOR en modo bit (^), Operador asignación de XOR en modo bit (^=), Operador OR en modo bit (|), Operador asignación de OR en modo bit (|=), Operador NOT en modo bit (~)

Operador desplazamiento a la izquierda en modo bit (<<)

expression1 << expression2

Convierte `expression1` y `expression2` a valores enteros de 32 bits; puede llamarlos V1 y V2. Desplaza todos los bits del valor de V1 a la izquierda V2 posiciones. Descarta los bits desplazados fuera del extremo izquierdo de V1 mediante esta operación e inserta ceros en las posiciones de bit vacías de la derecha. Desplazar un valor una posición a la izquierda equivale a multiplicarlo por 2.

Los números de coma flotante se convierten en enteros al descartarse los dígitos después de la coma decimal. Los enteros positivos se convierten en un valor hexadecimal sin signo con un valor máximo de 4294967295 o 0xFFFFFFFF; se descartan los dígitos más significativos de los valores mayores que el máximo cuando se convierten, de forma que el valor siga siendo de 32 bits. Los números negativos se convierten en un valor hexadecimal sin signo a través de la notación complementaria del dos, siendo el mínimo -2147483648 o 0x800000000; los números menores que el mínimo se convierten en el complemento del dos con una mayor precisión y también se descartan sus dígitos más significativos.

El valor devuelto se interpreta como un número complementario del dos con signo, de forma que el valor devuelto será un entero del intervalo comprendido entre -2147483648 y 2147483647.

Disponibilidad: Flash Player 5; ActionScript 1.0

Operandos

`expression1 : Number` - Número o expresión que se va a desplazar a la izquierda.

`expression2 : Number` - Número o expresión que se convierte en un entero de 0 a 31.

Valor devuelto

`Number` - El resultado de la operación en modo bit.

Ejemplo

En el ejemplo siguiente, el entero 1 se desplaza 10 bits a la izquierda: `x = 1 << 10` El resultado de esta operación es `x = 1024`. Eso se debe a que 1 decimal es igual a 1 binario, 1 binario desplazado 10 bits a la izquierda es 10000000000 binario, y 10000000000 binario es 1024 decimal. En el ejemplo siguiente, el entero 7 se desplaza 8 bits a la izquierda: `x = 7 << 8` El resultado de esta operación es `x = 1792`. Eso se debe a que 7 decimal es igual a 111 binario, 111 binario desplazado 8 bits a la izquierda es 11100000000 binario, y 11100000000 binario es 1792 decimal. Si observa el ejemplo siguiente, comprobará que los bits se han desplazado dos espacios a la izquierda:

```
// 2 binary == 0010
// 8 binary == 1000
trace(2 << 2); // output: 8
```

Véase también

Operador asignación y desplazamiento a la derecha en modo bit (>>=), Operador desplazamiento a la derecha en modo bit (>>), Operador asignación y desplazamiento a la izquierda en modo bit (<<=), Operador desplazamiento a la derecha en modo bit sin signo (>>>), Operador asignación y desplazamiento a la derecha en modo bit sin signo (>>>=)

Operador asignación y desplazamiento a la izquierda en modo bit (<<=)

expression1 `<<=` *expression2*

Este operador ejecuta una operación de desplazamiento a la izquierda en modo bit (<<=) y almacena el contenido como un resultado en *expression1* . Las dos expresiones siguientes son equivalentes:

```
A <<= B;
A = (A << B)
```

Disponibilidad: Flash Player 5; ActionScript 1.0

Operandos

`expression1 : Number` - Número o expresión que se va a desplazar a la izquierda.

`expression2 : Number` - Número o expresión que se convierte en un entero de 0 a 31.

Valor devuelto

`Number` - El resultado de la operación en modo bit.

Ejemplo

En el ejemplo siguiente, utilice el operador de asignación y desplazamiento a la izquierda en modo bit (<<=) para desplazar todos los bits un espacio a la izquierda:

```
var x:Number = 4;
// shift all bits one slot to the left.
x <<= 1;
trace(x); // output: 8
// 4 decimal = 0100 binary
// 8 decimal = 1000 binary
```

Véase también

Operador desplazamiento a la izquierda en modo bit (<<), Operador asignación y desplazamiento a la derecha en modo bit (>>=), Operador desplazamiento a la derecha en modo bit (>>)

Operador NOT en modo bit (~)

~expression

También conocido como operador complementario de uno u operador complementario en modo bit. Convierte *expression* en un entero de 32 bits sin signo y, a continuación, aplica un complemento de uno en modo bit. Es decir, que cada bit que sea 0 se establece como 1 en el resultado y cada bit que sea 1 se establece como 0 en el resultado. El resultado es un entero de 32 bits sin signo.

Por ejemplo, el valor hexadecimal 0x7777 se representa como este número binario: 0111011101110111

La negación en modo bit de dicho valor hexadecimal, ~0x7777, se corresponde con este número binario: 1000100010001000

El valor hexadecimal correspondiente es 0x8888. Por lo tanto, ~0x7777 es 0x8888.

El uso más común de los operadores en modo bit es la representación de *bits indicadores* (valores booleanos empaquetados en 1 bit cada uno).

Los números de coma flotante se convierten en enteros al descartarse los dígitos después de la coma decimal. Los enteros positivos se convierten en un valor hexadecimal sin signo con un valor máximo de 4294967295 o 0xFFFFFFFF; se descartan los dígitos más significativos de los valores mayores que el máximo cuando se convierten, de forma que el valor siga siendo de 32 bits. Los números negativos se convierten en un valor hexadecimal sin signo a través de la notación complementaria del dos, siendo el mínimo -2147483648 o 0x800000000; los números menores que el mínimo se convierten en el complemento del dos con una mayor precisión y también se descartan sus dígitos más significativos.

El valor devuelto se interpreta como un número complementario del dos con signo, de forma que el valor devuelto sea un entero del intervalo comprendido entre -2147483648 y 2147483647.

Disponibilidad: Flash Player 5; ActionScript 1.0

Operandos

`expression : Number` - Un número.

Valor devuelto

`Number` - El resultado de la operación en modo bit.

Ejemplo

El ejemplo siguiente demuestra un uso del operador NOT en modo bit (-) con bits indicadores:

```
var ReadOnlyFlag:Number = 0x0001; // defines bit 0 as the read-only flag
var flags:Number = 0;
trace(flags);
/* To set the read-only flag in the flags variable,
 the following code uses the bitwise OR:
*/
flags |= ReadOnlyFlag;
trace(flags);
/* To clear the read-only flag in the flags variable,
 first construct a mask by using bitwise NOT on ReadOnlyFlag.
 In the mask, every bit is a 1 except for the read-only flag.
 Then, use bitwise AND with the mask to clear the read-only flag.
 The following code constructs the mask and performs the bitwise AND:
*/
flags &= ~ReadOnlyFlag;
trace(flags);
// output: 0 1 0
```

Véase también

Operador & AND en modo bit, Operador &asignación de AND en modo bit (=), Operador XOR en modo bit (^), Operador asignación de XOR en modo bit (^=), Operador OR en modo bit (|), Operador asignación de OR en modo bit (|=)

Operador OR en modo bit (|)

expression1 | *expression2*

Convierte *expression1* y *expression2* en enteros de 32 bits sin signo y devuelve 1 en cada posición de bit donde los correspondientes bits, tanto de *expression1* como *expression2* son 1. Los números de coma flotante se convierten en enteros al descartarse los dígitos después de la coma decimal. El resultado es un nuevo entero de 32 bits.

Los enteros positivos se convierten en un valor hexadecimal sin signo con un valor máximo de 4294967295 o 0xFFFFFFFF; se descartan los dígitos más significativos de los valores mayores que el máximo cuando se convierten, de forma que el valor siga siendo de 32 bits. Los números negativos se convierten en un valor hexadecimal sin signo a través de la notación complementaria del dos, siendo el mínimo -2147483648 o 0x800000000; los números menores que el mínimo se convierten en el complemento del dos con una mayor precisión y también se descartan sus dígitos más significativos.

El valor devuelto se interpreta como un número complementario del dos con signo, de forma que el valor devuelto será un entero del intervalo comprendido entre -2147483648 y 2147483647.

Disponibilidad: Flash Player 5; ActionScript 1.0

Operandos

`expression1 : Number` - Un número.

`expression2 : Number` - Un número.

Valor devuelto

`Number` - El resultado de la operación en modo bit.

Ejemplo

A continuación se ofrece un ejemplo de una operación OR (|) en modo bit:

```
// 15 decimal = 1111 binary
var x:Number = 15;
// 9 decimal = 1001 binary
var y:Number = 9;
// 1111 | 1001 = 1111
trace(x | y); // returns 15 decimal (1111 binary)
```

No se debe confundir | (OR en modo bit) con || (OR lógico).

Véase también

Operador & AND en modo bit, Operador &asignación de AND en modo bit (=), Operador XOR en modo bit (^), Operador asignación de XOR en modo bit (^=), Operador asignación de OR en modo bit (|=), Operador NOT en modo bit (~)

Operador asignación de OR en modo bit (|=)

expression1 |= *expression2*

Asigna a *expression1* el valor de *expression1* | *expression2* . Por ejemplo, las dos sentencias siguientes son equivalentes:

```
x |= y;
x = x | y;
```

Disponibilidad: Flash Player 5; ActionScript 1.0

Operandos

`expression1 : Number` - Un número o variable.

`expression2 : Number` - Un número o variable.

Valor devuelto

`Number` - El resultado de la operación en modo bit.

Ejemplo

El ejemplo siguiente utiliza el operador de asignación OR en modo bit (|=):

```
// 15 decimal = 1111 binary
var x:Number = 15;
// 9 decimal = 1001 binary
var y:Number = 9;
// 1111 |= 1001 = 1111
trace(x |= y); // returns 15 decimal (1111 binary)
```

Véase también

Operador & AND en modo bit, Operador &asignación de AND en modo bit (=), Operador XOR en modo bit (^), Operador asignación de XOR en modo bit (^=), Operador OR en modo bit (|), Operador asignación de OR en modo bit (|=), Operador NOT en modo bit (~)

Operador desplazamiento a la derecha en modo bit (»)

expression1 >> *expression2*

Convierte *expression1* y *expression2* en enteros de 32 bits y desplaza todos los bits de *expression1* a la derecha el número de posiciones especificado por el entero resultante de la conversión de *expression2* . Los bits que se desplazan fuera del extremo derecho se descartan. Para conservar el signo del original *expression* , los bits a la izquierda se completan con un 0 si el bit más importante (el situado en el extremo izquierdo) de *expression1* es 0, y con 1 si el bit más importante es 1. Desplazar un valor una posición a la derecha equivale a dividir por 2 y descartar el resto.

Los números de coma flotante se convierten en enteros al descartarse los dígitos después de la coma decimal. Los enteros positivos se convierten en un valor hexadecimal sin signo con un valor máximo de 4294967295 o 0xFFFFFFFF; se descartan los dígitos más significativos de los valores mayores que el máximo cuando se convierten, de forma que el valor siga siendo de 32 bits. Los números negativos se convierten en un valor hexadecimal sin signo a través de la notación complementaria del dos, siendo el mínimo -2147483648 o 0x800000000; los números menores que el mínimo se convierten en el complemento del dos con una mayor precisión y también se descartan sus dígitos más significativos.

El valor devuelto se interpreta como un número complementario del dos con signo, de forma que el valor devuelto será un entero del intervalo comprendido entre -2147483648 y 2147483647.

Disponibilidad: Flash Player 5; ActionScript 1.0

Operandos

`expression1 : Number` - Número o expresión que se va a desplazar a la derecha.

`expression2 : Number` - Número o expresión que se convierte en un entero de 0 a 31.

Valor devuelto

`Number` - El resultado de la operación en modo bit.

Ejemplo

El ejemplo siguiente convierte 65535 en un entero de 32 bits y lo desplaza 8 bits a la derecha:

```
var x:Number = 65535 >> 8;
trace(x); // outputs 255
```

El ejemplo siguiente muestra el resultado del ejemplo anterior:

```
var x:Number = 255;
```

Eso se debe a que 65535 decimal es igual a 1111111111111111 binario (dieciséis unos), 1111111111111111 binario desplazado 8 bits a la derecha es 11111111 binario, y 11111111 binario es 255 decimal. El bit más significativo es 0 porque los enteros son de 32 bits, por lo que el bit de relleno es 0.

El ejemplo siguiente convierte -1 en un entero de 32 bits y lo desplaza 1 bit a la derecha:

```
var x:Number = -1 >> 1;
trace(x); // outputs -1
```

El ejemplo siguiente muestra el resultado del ejemplo anterior:

```
var x:Number = -1;
```

Esto se debe a que -1 decimal es igual a 11111111111111111111111111111111 binario (treinta y dos unos), desplazando un bit a la derecha hace que se descarte el bit menos significativo (el situado más a la derecha) y que se rellene con 1 el bit más significativo. El resultado es 11111111111111111111111111111111 binario (treinta y dos unos), lo que representa el entero de 32 bits -1.

Véase también

`Operador asignación y desplazamiento a la derecha en modo bit (>>=)`

Operador asignación y desplazamiento a la derecha en modo bit (>>=)

`expression1 >>= expression2`

Este operador ejecuta una operación de desplazamiento a la derecha en modo bit y almacena el contenido como un resultado en *`expression1`* .

Las dos sentencias siguientes son equivalentes:

```
A >>= B;
A = (A >> B);
```

Disponibilidad: Flash Player 5; ActionScript 1.0

Operandos

`expression1 : Number` - Número o expresión que se va a desplazar a la derecha.

`expression2 : Number` - Número o expresión que se convierte en un entero de 0 a 31.

Valor devuelto

`Number` - El resultado de la operación en modo bit.

Ejemplo

El siguiente código comentado utiliza el operador de asignación y desplazamiento a la derecha en modo bit (>>=).

```
function convertToBinary(numberToConvert:Number):String {
 var result:String = "";
 for (var i = 0; i<32; i++) {
 // Extract least significant bit using bitwise AND
 var lsb:Number = numberToConvert & 1;
 // Add this bit to the result
 string result = (lsb ? "1" : "0")+result;
 // Shift numberToConvert right by one bit, to see next bit
 numberToConvert >>= 1;
 }
 return result;
}
trace(convertToBinary(479));
// Returns the string 00000000000000000000000111011111
// This string is the binary representation of the decimal
// number 479
```

Véase también

Operador desplazamiento a la derecha en modo bit (>>)

Operador desplazamiento a la derecha en modo bit sin signo (>>>)

expression1 >>> *expression2*

Es igual al operador de desplazamiento a la derecha en modo bit (>>), con la diferencia de que no conserva el signo de la *expression* original porque los bits a la izquierda se rellenan siempre con un 0.

Los números de coma flotante se convierten en enteros al descartarse los dígitos después de la coma decimal. Los enteros positivos se convierten en un valor hexadecimal sin signo con un valor máximo de 4294967295 o 0xFFFFFFFF; se descartan los dígitos más significativos de los valores mayores que el máximo cuando se convierten, de forma que el valor siga siendo de 32 bits. Los números negativos se convierten en un valor hexadecimal sin signo a través de la notación complementaria del dos, siendo el mínimo -2147483648 o 0x800000000; los números menores que el mínimo se convierten en el complemento del dos con una mayor precisión y también se descartan sus dígitos más significativos.

Disponibilidad: Flash Player 5; ActionScript 1.0

Operandos

`expression1 : Number` - Número o expresión que se va a desplazar a la derecha.

`expression2 : Number` - Número o expresión que se convierte en un entero entre 0 y 31.

Valor devuelto

`Number` - El resultado de la operación en modo bit.

Ejemplo

El ejemplo siguiente convierte -1 en un entero de 32 bits y lo desplaza 1 bit a la derecha:

```
var x:Number = -1 >>> 1;
trace(x); // output: 2147483647
```

Esto se debe a que -1 decimal es 11111111111111111111111111111111 binario (treinta y dos unos) y cuando se desplaza a la derecha (sin signo) 1 bit, se descarta el bit menos significativo (situado más a la derecha) y se rellena el bit más significativo (situado más a la izquierda) con un 0. El resultado es 01111111111111111111111111111111 binario, lo que representa el entero de 32 bits 2147483647.

Véase también

`Operador asignación y desplazamiento a la derecha en modo bit (>>=)`

Operador asignación y desplazamiento a la derecha en modo bit sin signo (>>>=)

`expression1 >>>= expression2`

Ejecuta una operación de desplazamiento a la derecha en modo bit sin signo y almacena el contenido como un resultado en *`expression1`* . Las dos sentencias siguientes son equivalentes:

```
A >>>= B;
A = (A >>> B);
```

Disponibilidad: Flash Player 5; ActionScript 1.0

Operandos

`expression1 : Number` - Número o expresión que se va a desplazar a la derecha.

`expression2 : Number` - Número o expresión que se convierte en un entero de 0 a 31.

Valor devuelto

`Number` - El resultado de la operación en modo bit.

Ejemplo

Véase también

`Operador desplazamiento a la derecha en modo bit sin signo (>>>)`, `Operador asignación y desplazamiento a la derecha en modo bit (>>=)`

Operador XOR en modo bit (^)

`expression1 ^ expression2`

Convierte *`expression1`* y *`expression2`* en enteros de 32 bits sin signo y devuelve 1 en cada posición de bit donde los correspondientes bits de *`expression1`* o de *`expression2`* (pero no ambos) son 1. Los números de coma flotante se convierten en enteros al descartarse los dígitos después de la coma decimal. El resultado es un nuevo entero de 32 bits.

Los enteros positivos se convierten en un valor hexadecimal sin signo con un valor máximo de 4294967295 o 0xFFFFFFFF; se descartan los dígitos más significativos de los valores mayores que el máximo cuando se convierten, de forma que el valor siga siendo de 32 bits. Los números negativos se convierten en un valor hexadecimal sin signo a través de la notación complementaria del dos, siendo el mínimo -2147483648 o 0x800000000; los números menores que el mínimo se convierten en el complemento del dos con una mayor precisión y también se descartan sus dígitos más significativos.

El valor devuelto se interpreta como un número complementario del dos con signo, de forma que el valor devuelto será un entero del intervalo comprendido entre -2147483648 y 2147483647.

Disponibilidad: Flash Player 5; ActionScript 1.0

Operandos

`expression1 : Number` - Un número.

`expression2 : Number` - Un número.

Valor devuelto

`Number` - El resultado de la operación en modo bit.

Ejemplo

El ejemplo siguiente utiliza el operador XOR en modo bit en los decimales 15 y 9, y asigna el resultado a la variable `x`:

```
// 15 decimal = 1111 binary
// 9 decimal = 1001 binary
```

```
var x:Number = 15 ^ 9;
trace(x);
// 1111 ^ 1001 = 0110
// returns 6 decimal (0110 binary)
```

Véase también

Operador & AND en modo bit, Operador &asignación de AND en modo bit (=), Operador asignación de XOR en modo bit (^=), Operador OR en modo bit (|), Operador asignación de OR en modo bit (|=), Operador NOT en modo bit (~)

Operador asignación de XOR en modo bit (^=)

expression1 ^= *expression2*

Asigna a *expression1* el valor de *expression1* ^ *expression2* . Por ejemplo, las dos sentencias siguientes son equivalentes:

```
x ^= y;
x = x ^ y;
```

Disponibilidad: Flash Player 5; ActionScript 1.0

Operandos

`expression1 : Number` - Enteros y variables.

`expression2 : Number` - Enteros y variables.

Valor devuelto

`Number` - El resultado de la operación en modo bit.

Ejemplo

El ejemplo siguiente muestra una operación de asignación de XOR en modo bit (^=):

```
// 15 decimal = 1111 binary
var x:Number = 15;
// 9 decimal = 1001 binary
var y:Number = 9;
trace(x ^= y); // returns 6 decimal (0110 binary)
```

Véase también

Operador & AND en modo bit, Operador &asignación de AND en modo bit (=), Operador XOR en modo bit (^), Operador OR en modo bit (|), Operador asignación de OR en modo bit (|=), Operador NOT en modo bit (~)

Operador delimitador de comentario en bloque (/*..*/)

```
/* comment */
/* comment
comment */
```

Indica una o varias líneas de comentarios de script. Los caracteres que aparecen entre la etiqueta de apertura de comentario (`/*`) y la etiqueta de cierre de comentario (`*/`) se interpretan como un comentario y el interpretador de ActionScript los omite. Utilice el delimitador de comentario // para identificar los comentarios de una sola línea. Utilice el delimitador de comentario /* para identificar los comentarios en varias líneas sucesivas. Si no se inserta la etiqueta de cierre (`*/`) cuando se utiliza esta forma de delimitador de comentario, aparece un mensaje de error. También aparece un mensaje de error si se intenta anidar comentarios. Después de utilizar una etiqueta de apertura de comentario (/*), la primera etiqueta de cierre de comentario (`*/`) finalizará el comentario, independientemente del número de etiquetas de apertura (/*) que haya entre ambas.

Disponibilidad: Flash Player 5; ActionScript 1.0

Operandos

`comment` - Cualquier carácter.

Ejemplo

El script siguiente utiliza delimitadores de comentarios al principio:

```
/* records the X and Y positions of
the ball and bat movie clips */
var ballX:Number = ball_mc._x;
var ballY:Number = ball_mc._y;
var batX:Number = bat_mc._x;
var batY:Number = bat_mc._y;
```

Este intento de anidar comentarios producirá un mensaje de error:

```
/* this is an attempt to nest comments.
/* But the first closing tag will be paired
with the first opening tag */
and this text will not be interpreted as a comment */
```

Véase también

Operador delimitador de comentario de línea (//)

Operador coma (,)

`(expression1 , expression2 [, expressionN... ])`

Evalúa primero *expression1* , a continuación, *expression2* y así sucesivamente. Este operador se utiliza principalmente con la sentencia de bucle `for` y a menudo con el operador de paréntesis ().

Disponibilidad: Flash Player 4; ActionScript 1.0

Operandos

`expression1 : Number` - Expresión que se va a evaluar.

`expression2 : Number` - Expresión que se va a evaluar.

`expressionN : Number` - Cualquier número de expresiones adicionales que se van a evaluar.

Valor devuelto

`Object` - El valor de *expression1* , a continuación, *expression2* y así sucesivamente.

Ejemplo

El ejemplo siguiente utiliza el operador de coma (,) en un bucle `for`:

```
for (i = 0, j = 0; i < 3 && j < 3; i++, j+=2) {
 trace("i = " + i + ", j = " + j);
}
// Output:
// i = 0, j = 0
// i = 1, j = 2
```

El ejemplo siguiente utiliza el operador de coma (,) sin el operador de paréntesis () e ilustra que el operador de coma devuelve únicamente el valor de la primera expresión sin el operador de paréntesis ():

```
var v:Number = 0;
v = 4, 5, 6;
trace(v); // output: 4
```

El ejemplo siguiente utiliza el operador de coma (,) con el operador de paréntesis () e ilustra que el operador de coma devuelve el valor de la última expresión cuando se utiliza con el operador de paréntesis ():

```
var v:Number = 0;
v = (4, 5, 6);
trace(v); // output: 6
```

El ejemplo siguiente utiliza el operador de coma (,) sin el operador de paréntesis () e ilustra que el operador de coma evalúa secuencialmente todas las expresiones pero devuelve el valor de la primera expresión. La segunda expresión, `z++`, se evalúa y `z` se incrementa en uno.

```
var v:Number = 0;
var z:Number = 0;
v = v + 4 , z++, v + 6;
trace(v); // output: 4
trace(z); // output: 1
```

El ejemplo siguiente es idéntico al ejemplo anterior, salvo por la adición del operador de paréntesis () e ilustra una vez más que, cuando se utiliza con el operador de paréntesis (), el operador de coma (,) devuelve el valor de la última expresión de la serie:

```
var v:Number = 0;
var z:Number = 0;
v = (v + 4, z++, v + 6);
trace(v); // output: 6
trace(z); // output: 1
```

Véase también

`Operador paréntesis (())`

Operador suma de concatenación (cadenas)

`string1` `add` *`string2`*

Desfasado desde Flash Player 5. Macromedia recomienda que utilice el operador de suma (+) cuando cree contenido para Flash Player 5 o versiones posteriores. Este operador no se admite en Flash Player 8 ni en versiones posteriores.

Concatena dos o más cadenas. El operador de suma (+) sustituye al operador `&` de Flash 4; los archivos de Flash Player 4 que utilizan el operador `&` se convierten automáticamente para utilizar el operador de suma (+) para la concatenación de cadenas cuando pasan al entorno de edición de Flash 5 o versiones posteriores. Utilice el operador de suma (+) para concatenar cadenas si está creando contenido para Flash Player 4 o versiones anteriores de Player.

Disponibilidad: Flash Player 4; ActionScript 1.0

Operandos

`string1 : String` - Una cadena.

`string2 : String` - Una cadena.

Valor devuelto

`String` - La cadena concatenada.

Véase también

Operador suma (+)

Operador condicional (?:)

expression1 ? *expression2* : *expression3*

Indica a Flash que evalúe *expression1* y si el valor de *expression1* es `true`, el operador devuelve el valor de *expression2* ; en caso contrario devuelve el valor de *expression3* .

Disponibilidad: Flash Player 4; ActionScript 1.0

Operandos

`expression1 : Object` - Expresión que da como resultado un valor booleano; normalmente una expresión de comparación, como `x < 5`.

`expression2 : Object` - Valores de cualquier tipo.

`expression3 : Object` - Valores de cualquier tipo.

Valor devuelto

`Object` - El valor de *expression2* o *expression3* .

Ejemplo

La sentencia siguiente asigna el valor de la `x` a la variable `z` porque `expression1` da como resultado `true`:

```
var x:Number = 5;
var y:Number = 10;
var z = (x < 6) ? x: y;
trace (z); // returns 5
```

El ejemplo siguiente muestra una sentencia condicional escrita en forma abreviada:

```
var timecode:String = (new Date().getHours() < 11) ? "AM" : "PM";
trace(timecode);
```

La misma sentencia condicional también puede escribirse en forma no abreviada, como se muestra en el ejemplo siguiente:

```
if (new Date().getHours() < 11) {
 var timecode:String = "AM";
} else {
 var timecode:String = "PM";
} trace(timecode);
```

Operador decremento (--)

--expression
expression--

Operador unario de decremento previo y decremento posterior que resta 1 de *expression*. La *expression* puede ser una variable, un elemento de una matriz o una propiedad de un objeto. La forma de decremento previo del operador (*--expression*) resta 1 de *expression* y devuelve el resultado. La forma de decremento posterior del operador (*expression--*) resta 1 de *expression* y devuelve el valor inicial de *expression* (el valor antes de la resta).

Disponibilidad: Flash Player 4; ActionScript 1.0

Operandos

`expression : Number` - Número o variable que da como resultado un número.

Valor devuelto

`Number` - El resultado del valor decrementado.

Ejemplo

La forma de decremento previo del operador decrementa `x` hasta 2 (`x - 1 = 2`) y devuelve el resultado como `y`:

```
var x:Number = 3;
var y:Number = --x; //y is equal to 2
```

La forma de decremento posterior del operador decrementa `x` hasta 2 (`x - 1 = 2`) y devuelve el valor original de `x` como el resultado de `y`:

```
var x:Number = 3;
var y:Number = x--; //y is equal to 3
```

El ejemplo siguiente reproduce indefinidamente de 10 a 1, y cada repetición del bucle decrementa la variable de contador `i` en 1.

```
for (var i = 10; i>0; i--) {
 trace(i);
}
```

Operador división (/)

expression1 / expression2

Divide *expression1* por *expression2* . El resultado de la operación de división es un número de coma flotante de doble precisión.

Disponibilidad: Flash Player 4; ActionScript 1.0

Operandos

`expression : Number` - Número o variable que da como resultado un número.

Valor devuelto

`Number` - El resultado de coma flotante de la operación.

Ejemplo

La sentencia siguiente divide el ancho y el alto actuales del escenario y, a continuación, muestra el resultado en el panel Salida.

```
trace(Stage.width/2);
trace(Stage.height/2);
```

Con un ancho y un alto de escenario predeterminados de 550 x 400, el resultado es 275 y 150.

Véase también

Operador módulo (%)

Operador asignación de división (/=)

expression1 /= expression2

Asigna a *expression1* el valor de *expression1 / expression2*. Por ejemplo, las dos sentencias siguientes son equivalentes:

```
x /= y;
x = x / y;
```

Disponibilidad: Flash Player 4; ActionScript 1.0

Operandos

`expression1 : Number` - Número o variable que da como resultado un número.

`expression2 : Number` - Número o variable que da como resultado un número.

Valor devuelto

`Number` - Un número.

Ejemplo

El código siguiente ilustra el uso del operador asignación de división (`/=`) con variables y números:

```
var x:Number = 10;
var y:Number = 2;
x /= y; trace(x); // output: 5
```

Véase también

`Operador división (/)`

. Operador punto

object.property_or_methodinstancename.variable
instancename.childinstanceinstancename.childinstance.variable

Se utiliza para navegar por las jerarquías de clips de película y acceder a variables, propiedades o clips de película anidados (secundarios). El operador de punto se utiliza también para probar o establecer las propiedades de un objeto o clase de nivel superior, para ejecutar un método de un objeto o clase de nivel superior o para crear una estructura de datos.

Disponibilidad: Flash Player 4; ActionScript 1.0

Operandos

`object : Object` - Una instancia de una clase. El objeto puede ser una instancia de cualquiera de las clases de ActionScript incorporadas o una clase personalizada. Este parámetro se ubica siempre a la izquierda del operador de punto (.).

`property_or_method` - Nombre de una propiedad o método asociado con un objeto. Todos los métodos y propiedades válidos para las clases incorporadas se enumeran en las tablas de resumen de método y propiedades de dicha clase. Este parámetro se ubica siempre a la derecha del operador de punto (.).

`instancename : MovieClip` - Nombre de instancia de un clip de película.`variable` - El nombre de instancia a la izquierda del operador de punto (`.`) también puede representar una variable en la línea de tiempo del clip de película.

`childinstance : MovieClip` - Nombre de instancia de un clip de película que es un elemento secundario o anidado de otro clip de película.

Valor devuelto

`Object` - El método, propiedad o clip de película que aparece a la derecha del punto.

Ejemplo

El ejemplo siguiente identifica el valor actual de la variable `hairColor` en el clip de película `person_mc`:

```
person_mc.hairColor
```

El entorno de edición de Flash 4 no admitía la sintaxis con punto, pero los archivos de Flash MX 2004 publicados para Player 4 pueden utilizar el operador de punto. El ejemplo anterior equivale a la siguiente sintaxis de Flash 4 (desfasada):

```
/person_mc:hairColor
```

El ejemplo siguiente crea un clip de película nuevo en el ámbito `_root`. A continuación se crea un campo de texto dentro del clip de película llamado `container_mc`. La propiedad `autoSize` del campo de texto se establece como `true` y a continuación se rellena con la fecha actual.

```
this.createEmptyMovieClip("container_mc", this.getNextHighestDepth());
this.container_mc.createTextField("date_txt", this.getNextHighestDepth(),
  0, 0, 100, 22);
this.container_mc.date_txt.autoSize = true;
this.container_mc.date_txt.text = new Date();
```

El operador de punto (.) se utiliza para referirse a instancia dentro del archivo SWF y cuando es necesario establecer propiedades y valores para esas instancias.

Operador de igualdad (==)

`expression1 == expression2`

Comprueba la igualdad de dos expresiones. El resultado es `true` si las expresiones son iguales.

La definición de igual depende del tipo de datos del parámetro:

- Los valores numéricos y booleanos se comparan por su valor y se consideran iguales si tienen el mismo valor.
- Las expresiones de cadena son iguales si tienen el mismo número de caracteres y éstos son idénticos.
- Las variables que representan objetos, matrices y funciones se comparan por su referencia. Dos de estas variables son iguales si hacen referencia al mismo objeto, matriz o función. Dos matrices independientes nunca se consideran iguales, aunque tengan el mismo número de elementos.

Si se comparan por su valor y *expression1* y *expression2* tienen tipos de datos distintos, ActionScript intentará convertir el tipo de datos de *expression2* para que coincida con el de *expression1*.

Disponibilidad: Flash Player 5; ActionScript 1.0

Operandos

`expression1 : Object` - Un número, cadena, valor booleano, variable, objeto, matriz o función.

`expression2 : Object` - Un número, cadena, valor booleano, variable, objeto, matriz o función.

Valor devuelto

`Boolean` - El resultado booleano de la comparación.

Ejemplo

El ejemplo siguiente utiliza el operador de igualdad (`==`) con una sentencia `if`:

```
var a:String = "David", b:String = "David";
if (a == b) {
 trace("David is David");
}
```

Los ejemplos siguientes muestran los resultados de operaciones que comparan tipos mixtos:

```
var x:Number = 5;
var y:String = "5";
trace(x == y); // output: true
var x:String = "5";
var y:String = "66";
trace(x == y); // output: false
var x:String = "chris";
var y:String = "steve";
trace(x == y); // output: false
```

Los ejemplos siguientes muestran comparación por referencia. El primer ejemplo compara dos matrices con una longitud y elementos idénticos. El operador de igualdad devolverá false para estas dos matrices. Si bien las matrices parecen iguales, la comparación por referencia requiere que ambas hagan referencia a la misma matriz. El segundo ejemplo crea la variable thirdArray, que señala a la misma matriz que la variable firstArray. El operador de igualdad devolverá true para estas dos matrices porque las dos variables hacen referencia a la misma matriz.

```
var firstArray:Array = new Array("one", "two", "three");
var secondArray:Array = new Array("one", "two", "three");
trace(firstArray == secondArray);
// will output false
```

```
// Arrays are only considered equal
// if the variables refer to the same array.
var thirdArray:Array = firstArray;
trace(firstArray == thirdArray); // will output true
```

Véase también

`! Operador NOT lógico`, `Operador de desigualdad (!=)`, `Operador de desigualdad estricta (!==)`, `Operador AND lógico (&&)`, `Operador OR lógico (||)`, `Operador de igualdad estricta (===)`

Operador de igualdad (eq) (cadenas)

`expression1` `eq` *`expression2`*

Desfasado desde Flash Player 5. Este operador está desfasado y en su lugar debe utilizarse `==` `(equality)`.

Compara la igualdad de dos expresiones y devuelve un valor de `true` si la representación de cadena de *`expression1`* es igual a la representación de cadena *`expression2`*, y `false` en caso contrario.

Disponibilidad: Flash Player 4; ActionScript 1.0

Operandos

`expression1 : Object` - Números, cadenas o variables.

`expression2 : Object` - Números, cadenas o variables.

Valor devuelto

`Boolean` - El resultado de la comparación.

Véase también

`Operador de igualdad (==)`

Operador mayor que (>)

expression1 > *expression2*

Compara dos expresiones y determina si *expression1* es mayor que *expression2* ; si lo es, el operador devuelve `true`. Si *expression1* es menor o igual que *expression2*, el operador devuelve `false`. Las expresiones de cadena se evalúan por orden alfabético; todas las letras mayúsculas preceden a las minúsculas.

Disponibilidad: Flash Player 4; ActionScript 1.0

Operandos

`expression1 : Object` - Un número o cadena.

`expression2 : Object` - Un número o cadena.

Valor devuelto

`Boolean` - El resultado booleano de la comparación.

Ejemplo

En el ejemplo siguiente, el operador de mayor que (>) se utiliza para determinar si el valor del campo de texto `score_txt` es mayor que 90:

```
if (score_txt.text>90) {
 trace("Congratulations, you win!");
} else {
 trace("sorry, try again");
}
```

Operador mayor que (gt) (cadenas)

expression1 gt *expression2*

Desfasado desde Flash Player 5. Este operador etá desfasado y en su lugar debe utilizarse > (mayor que).

Compara la representación de cadena de *expression1* con la representación de cadena de *expression2* y devuelve `true` si *expression1* es mayor que *expression2* ; en caso contrario devuelve `false`.

Disponibilidad: Flash Player 4; ActionScript 1.0

Operandos

`expression1 : Object` - Números, cadenas o variables.

`expression2 : Object` - Números, cadenas o variables.

Valor devuelto

`Boolean` - El resultado booleano de la comparación.

Véase también

`Operador mayor que (>)`

Operador mayor o igual que (>=)

`expression1 >= expression2`

Compara dos expresiones y determina si *`expression1`* es mayor o igual que *`expression2`* (`true`) o *`expression1`* es menor que *`expression2`* (`false`).

Disponibilidad: Flash Player 4; ActionScript 1.0

Operandos

`expression1 : Object` - Una cadena, un entero o un número de coma flotante.

`expression2 : Object` - Una cadena, un entero o un número de coma flotante.

Valor devuelto

`Boolean` - El resultado booleano de la comparación.

Ejemplo

En el ejemplo siguiente, el operador de mayor o igual que (>=) se emplea para determinar si la hora actual es mayor o igual que 12:

```
if (new Date().getHours() >= 12) {
 trace("good afternoon");
} else {
 trace("good morning");
}
```

Operador mayor o igual que (ge) (cadenas)

expression1 `ge` *expression2*

Desfasado desde Flash Player 5. Este operador está desfasado y en su lugar debe utilizarse >= (mayor o igual que).

Compara la representación de cadena de *expression1* con la representación de cadena de *expression2* y devuelve `true` si *expression1* es mayor o igual que *expression2* ; en caso contrario devuelve `false`.

Disponibilidad: Flash Player 4; ActionScript 1.0

Operandos

`expression1 : Object` - Números, cadenas o variables.

`expression2 : Object` - Números, cadenas o variables.

Valor devuelto

`Boolean` - El resultado de la comparación.

Véase también

`Operador mayor o igual que (>=)`

Operador incremento (++)

++expression
expression++

Operador unario de incremento previo e incremento posterior que añade 1 a *expression* . La *expression* puede ser una variable, un elemento de una matriz o una propiedad de un objeto. La forma de incremento previo del operador (*++expression*) suma 1 de *expression* y devuelve el resultado. La forma de incremento posterior del operador (*expression++*) suma 1 de *expression* y devuelve el valor inicial *expression* (el valor antes de la suma).

```
The pre-increment form of the operator increments x to 2 (x + 1 = 2) and
returns the result as y:

var x:Number = 1;
var y:Number = ++x;
trace("x:"+x); //traces x:2
trace("y:"+y); //traces y:2
```

La forma de incremento posterior del operador aumenta `x` hasta 2 (`x + 1 = 2`) y devuelve el valor original de `x` como el resultado de `y`:

```
var x:Number = 1;
var y:Number = x++;
trace("x:"+x); //traces x:2
trace("y:"+y); //traces y:1
```

Disponibilidad: Flash Player 4; ActionScript 1.0

Operandos

`expression : Number` - Número o variable que da como resultado un número.

Valor devuelto

`Number` - El resultado del incremento.

Ejemplo

El ejemplo siguiente utiliza ++ como operador de incremento posterior para hacer que un bucle `while` se ejecute cinco veces:

```
var i:Number = 0;
while (i++ < 5) {
 trace("this is execution " + i);
}
/* output:
 this is execution 1
 this is execution 2
 this is execution 3
 this is execution 4
 this is execution 5
*/
```

El ejemplo siguiente utiliza ++ como operador de incremento previo:

```
var a:Array = new Array();
var i:Number = 0;
while (i < 10) {
 a.push(++i);
}
trace(a.toString()); //traces: 1,2,3,4,5,6,7,8,9,10
```

Este ejemplo también utiliza ++ como operador de incremento previo.

```
var a:Array = [];
for (var i = 1; i <= 10; ++i) {
 a.push(i);
}
trace(a.toString()); //traces: 1,2,3,4,5,6,7,8,9,10
```

Este script muestra el siguiente resultado en el panel Salida: `1,2,3,4,5,6,7,8,9,10` El ejemplo siguiente utiliza ++ como operador de incremento posterior en un bucle `while` :

```
// using a while loop
var a:Array = new Array();
var i:Number = 0;
while (i < 10) {
 a.push(i++);
}
trace(a.toString()); //traces 0,1,2,3,4,5,6,7,8,9
```

El ejemplo siguiente utiliza ++ como operador de incremento posterior en un bucle `for` :

```
// using a for loop
var a:Array = new Array();
for (var i = 0; i < 10; i++) {
 a.push(i);
}
trace(a.toString()); //traces 0,1,2,3,4,5,6,7,8,9
```

Este script muestra el siguiente resultado en el panel Salida:

```
0,1,2,3,4,5,6,7,8,9
```

Operador de desigualdad (!=)

expression1 != *expression2*

Prueba el contrario exacto del operador de igualdad (==). Si *expression1* es igual que *expression2* , el resultado es `false`. Como ocurre con el operador de igualdad (==), la definición de igual depende de los tipos de datos que se comparan, tal y como se muestra en la siguiente lista:

- Los números, cadenas y valores booleanos se comparan por su valor.
- Los objetos, matrices y funciones se comparan por su referencia.
- Una variable se compara por su valor o referencia, en función de su tipo.

La comparación por su valor significa que dos expresiones tienen el mismo valor. Por ejemplo, la expresión (2 + 3) es igual que la expresión (1 + 4) si se comparan sus valores.

La comparación por su referencia significa que dos expresiones son únicamente iguales si ambas hacen referencia al mismo objeto, matriz o función. No se comparan los valores dentro del objeto, matriz o función.

Si se comparan por su valor y *expression1* y *expression2* tienen tipos de datos distintos, ActionScript intentará convertir el tipo de datos de *expression2* para que coincida con el de *expression1*.

Disponibilidad: Flash Player 5; ActionScript 1.0

Operandos

`expression1 : Object` - Un número, cadena, valor booleano, variable, objeto, matriz o función.

`expression2 : Object` - Un número, cadena, valor booleano, variable, objeto, matriz o función.

Valor devuelto

`Boolean` - El resultado booleano de la comparación.

Ejemplo

El ejemplo siguiente ilustra el resultado del operador de desigualdad (`!=`):

```
trace(5 != 8); // returns true
trace(5 != 5) //returns false
```

El ejemplo siguiente ilustra el uso del operador de desigualdad (`!=`) en una sentencia `if`:

```
var a:String = "David";
var b:String = "Fool";
if (a != b) {
 trace("David is not a fool");
}
```

El ejemplo siguiente ilustra la comparación por referencia con dos funciones:

```
var a:Function = function() { trace("foo"); };
var b:Function = function() { trace("foo"); };
a(); // foo
b(); // foo
trace(a != b); // true
a = b;
a(); // foo
b(); // foo
trace(a != b); // false
// trace statement output: foo foo true foo foo false
```

El ejemplo siguiente ilustra la comparación por referencia con dos matrices:

```
var a:Array = [ 1, 2, 3 ];
var b:Array = [ 1, 2, 3 ];
trace(a); // 1, 2, 3
trace(b); // 1, 2, 3
trace(a!=b); // true
a = b;
trace(a); // 1, 2, 3
trace(b); // 1, 2, 3
trace(a != b); // false
// trace statement output: 1,2,3 1,2,3 true 1,2,3 1,2,3 false
```

Véase también

`! Operador NOT lógico`, `Operador de desigualdad estricta (!==)`, `Operador AND lógico (&&)`, `Operador OR lógico (||)`, `Operador de igualdad (==)`, `Operador de igualdad estricta (===)`

Operador de desigualdad (<>)

`expression1 <> expression2`

Desfasado desde Flash Player 5. Este operador está desfasado. Macromedia recomienda que utilice el operador `!= (inequality)`.

Prueba el contrario exacto del operador de igualdad (==). Si *`expression1`* es igual que *`expression2`*, el resultado es `false`. Como ocurre con el operador de igualdad (==), la definición de igual depende de los tipos de datos que se comparan:

- Los números, cadenas y valores booleanos se comparan por su valor.
- Los objetos, matrices y funciones se comparan por su referencia.
- Las variables se comparan por su valor o por referencia, en función de su tipo.

Disponibilidad: Flash Player 2; ActionScript 1.0

Operandos

`expression1 : Object` - Un número, cadena, valor booleano, variable, objeto, matriz o función.

`expression2 : Object` - Un número, cadena, valor booleano, variable, objeto, matriz o función.

Valor devuelto

`Boolean` - El resultado booleano de la comparación.

Véase también

`Operador de desigualdad (!=)`

Operador instanceof

object instanceof *classConstructor*

Comprueba si `object` es una instancia de `classConstructor` o una subclase de `classConstructor`. El operador `instanceof` no convierte tipos de datos simples en objetos envolventes. Por ejemplo, el código siguiente devuelve `true`:

```
new String("Hello") instanceof String;
```

Mientras que el código siguiente devuelve false:

```
"Hello" instanceof String;
```

Disponibilidad: Flash Player 6; ActionScript 1.0

Operandos

`object : Object` - Un objeto ActionScript.

`classConstructor : Function` - Una referencia a una función constructora de ActionScript, como `String` o `Date`.

Valor devuelto

`Boolean` - Si `object` es una instancia o una subclase de `classConstructor`, `instanceof` devuelve `true`; de lo contrario devuelve `false`. Además, `_global instanceof Object` devuelve `false`.

Véase también

`Operador typeof`

Operador menor que (<)

expression1 < *expression2*

Compara dos expresiones y determina si *expression1* es menor que *expression2*; si lo es, el operador devuelve `true`. Si *expression1* es mayor o igual que *expression2* , el operador devuelve `false`. Las expresiones de cadena se evalúan por orden alfabético; todas las letras mayúsculas preceden a las minúsculas.

Disponibilidad: Flash Player 4; ActionScript 1.0

Operandos

`expression1 : Number` - Un número o cadena.

`expression2 : Number` - Un número o cadena.

Valor devuelto

`Boolean` - El resultado booleano de la comparación.

Ejemplo

Los ejemplos siguientes muestran resultados de `true` y `false` para comparaciones numéricas y de cadena:

```
trace(3 < 10); // true
trace(10 < 3); // false
trace("Allen" < "Jack"); // true
trace("Jack" < "Allen"); //false
trace("11" < "3"); // true
trace("11" < 3); // false (numeric comparison)
trace("C" < "abc"); // true
trace("A" < "a"); // true
```

Operador menor que (lt) (cadenas)

expression1 `lt` *expression2*

Desfasado desde Flash Player 5. Este operador está desfasado y en su lugar debe utilizarse `<` (menor que).

Compara *expression1* con *expression2* y devuelve `true`si *expression1* es menor que *expression2*; de lo contrario devuelve `false` .

Disponibilidad: Flash Player 4; ActionScript 1.0

Operandos

`expression1 : Object` - Números, cadenas o variables.

`expression2 : Object` - Números, cadenas o variables.

Valor devuelto

`Boolean` - El resultado de la comparación.

Véase también

`Operador menor que (<)`

Operador menor o igual que (<=)

expression1 <= *expression2*

Compara dos expresiones y determina si *expression1* es menor o igual que *expression2* ; si lo es, el operador devuelve `true`. Si *expression1* es mayor que *expression2* , el operador devuelve `false`. Las expresiones de cadena se evalúan por orden alfabético; todas las letras mayúsculas preceden a las minúsculas.

Disponibilidad: Flash Player 4; ActionScript 1.0

Operandos

`expression1 : Object` - Un número o cadena.

`expression2 : Object` - Un número o cadena.

Valor devuelto

`Boolean` - El resultado booleano de la comparación.

Ejemplo

Los ejemplos siguientes muestran resultados de `true` y `false` para comparaciones numéricas y de cadena:

```
trace(5 <= 10); // true
trace(2 <= 2); // true
trace(10 <= 3); // false
trace("Allen" <= "Jack"); // true
trace("Jack" <= "Allen"); // false
trace("11" <= "3"); // true
trace("11" <= 3); // false (numeric comparison)
trace("C" <= "abc"); // true
trace("A" <= a); // true
```

Operador menor o igual que (le) (cadenas)

expression1 le *expression2*

Desfasado desde Flash Player 5. Este operador está desfasado y en su lugar debe utilizarse <= (menor o igual que).

Compara *expression1* con *expression2* y devuelve un valor de `true` si*expression1* es menor o igual que *expression2*; de lo contrario devuelve `false`.

Disponibilidad: Flash Player 4; ActionScript 1.0

Operandos

`expression1 : Object` - Números, cadenas o variables.

`expression2 : Object` - Números, cadenas o variables.

Valor devuelto

`Boolean` - El resultado de la comparación.

Véase también

`Operador menor o igual que (<=)`

Operador delimitador de comentario de línea (//)

```
// comment
```

Indica el principio de un comentario de script. Los caracteres que aparecen entre el delimitador de comentario (`//`) y el carácter de final de línea se interpretan como un comentario y el interpretador de ActionScript los omite.

Disponibilidad: Flash Player 1.0; ActionScript 1.0

Operandos

`comment` - Cualquier carácter.

Ejemplo

El script siguiente utiliza delimitadores de comentario para identificar la primera, tercera, quinta y séptima líneas como comentarios:

```
// record the X position of the ball movie clip
var ballX:Number = ball_mc._x;
// record the Y position of the ball movie clip
var ballY:Number = ball_mc._y;
// record the X position of the bat movie clip
var batX:Number = bat_mc._x;
// record the Y position of the ball movie clip
var batY:Number = bat_mc._y;
```

Véase también

`Operador delimitador de comentario en bloque (/*..*/ )`

Operador AND lógico (&&)

expression1 && *expression2*

Ejecuta una operación booleana en los valores de ambas expresiones. Si *expression1* y *expression2* son ambos `true`, devuelve `true`; de lo contrario devuelve `false`.

Expresión	Resultado
`true&&true`	true
`true&&false`	false
`false&&false`	false
`false&&true`	false

Disponibilidad: Flash Player 4; ActionScript 1.0

Operandos

`expression1 : Number` - Un valor booleano o una expresión que se convierta en un valor booleano.

`expression2 : Number` - Un valor booleano o una expresión que se convierta en un valor booleano.

Valor devuelto

`Boolean` - Un resultado booleano de la operación lógica.

Ejemplo

El ejemplo siguiente utiliza el operador AND lógico (`&&`) para realizar una prueba para determinar si un jugador ha ganado la partida. La variable `turns` y la variable `score` se actualizan cuando un jugador juega o gana puntos durante el juego. El script muestra "You Win the Game!" en el panel Salida cuando la puntuación del jugador llega a 75 o más en tres jugadas o menos.

```
var turns:Number = 2;
var score:Number = 77;
if ((turns <= 3) && (score >= 75)) {
 trace("You Win the Game!");
} else {
 trace("Try Again!");
}
// output: You Win the Game!
```

Véase también

`! Operador NOT lógico`, `Operador de desigualdad (!=)`, `Operador de desigualdad estricta (!==)`, `Operador OR lógico (||)`, `Operador de igualdad (==)`, `Operador de igualdad estricta (===)`

Operador AND lógico (and)

`condition1 and condition2`

Desfasado desde Flash Player 5. Macromedia recomienda el uso del operador lógico AND (`&&`).

Realiza una operación de AND lógica (`&&`) en Flash Player 4. Si las dos expresiones dan como resultado `true`, toda la expresión será `true`.

Disponibilidad: Flash Player 4; ActionScript 1.0

Operandos

`condition1 : Boolean` - *condition1,condition2* Condiciones o expresiones que dan como resultado `true` o `false`.

`condition2 : Boolean` - *condition1,condition2* Condiciones o expresiones que dan como resultado `true` o `false`.

Valor devuelto

`Boolean` - Un resultado booleano de la operación lógica.

Véase también

`Operador AND lógico (&&)`

! Operador NOT lógico

`! expression`

Invierte el valor booleano de una variable o expresión. Si *expression* es una variable con el valor absoluto o convertido `true`, el valor de `! expression` es `false`. Si la expresión `x && y` da como resultado `false`, la expresión `!(x && y)` da como resultado `true`.

Las siguientes expresiones ilustran el resultado del uso del operador NOT lógico (!):

`! true` devuelve `false` `! false` devuelve `true`

Disponibilidad: Flash Player 4; ActionScript 1.0

Operandos

`expression : Boolean` - Expresión o variable que da como resultado un valor booleano.

Valor devuelto

`Boolean` - El resultado booleano de la operación lógica.

Ejemplo

En el ejemplo siguiente, la variable `happy` se establece como `false`. La condición `if` da como resultado la condición `!happy` y, si la condición es `true`, la sentencia trace() envía una cadena al panel Salida.

```
var happy:Boolean = false;
if (!happy) {
 trace("don't worry, be happy"); //traces don't worry, be happy
}
```

La sentencia realiza seguimiento porque `!false` es igual a `true`.

Véase también

`Operador de desigualdad (!=)`, `Operador de desigualdad estricta (!==)`, `Operador AND lógico (&&)`, `Operador OR lógico (||)`, `Operador de igualdad (==)`, `Operador de igualdad estricta (===)`

Operador NOT lógico (not)

`not` *`expression`*

Desfasado desde Flash Player 5. Este operador está desfasado y en su lugar debe utilizarse `!` `(logical NOT)`.

Ejecuta una operación NOT (!) lógica en Flash Player 4.

Disponibilidad: Flash Player 4; ActionScript 1.0

Operandos

`expression : Object` - Una variable u otra expresión que se convierte en un valor booleano.

Valor devuelto

`Boolean` - El resultado de la operación lógica.

Véase también

`! Operador NOT lógico`

Operador OR lógico (||)

expression1 || *expression2*

Evalúa *expression1* (la expresión en la parte izquierda del operador) y devuelve `true` si la expresión da como resultado `true`. Si *expression1* da como resultado `false`, se evalúa *expression2* (la expresión en la parte derecha del operador. Si *expression2* da como resultado `false`, el resultado final es `false`; de lo contrario es `true`.

Si utiliza una llamada de función como *expression2*, esa llamada no ejecutará la función si *expression1* da como resultado true.

El resultado es `true` si una o ambas expresiones dan como resultado `true`; el resultado sólo será `false` si ambas expresiones dan como resultado `false`. Puede utilizar el operador OR lógico con cualquier número de operandos; si alguno de los operandos da como resultado `true`, el resultado es `true`.

Disponibilidad: Flash Player 4; ActionScript 1.0

Operandos

`expression1 : Number` - Un valor booleano o una expresión que se convierta en un valor booleano.

`expression2 : Number` - Un valor booleano o una expresión que se convierta en un valor booleano.

Valor devuelto

`Boolean` - El resultado de la operación lógica.

Ejemplo

El ejemplo siguiente utiliza el operador lógico OR (`||`) en una sentencia `if`. La segunda expresión da como resultado `true`, , por lo que el resultado final es `true`:

```
var x:Number = 10;
var y:Number = 250;
var start:Boolean = false;
if ((x > 25) || (y > 200) || (start)) {
 trace("the logical OR test passed"); // output: the logical OR test passed
}
```

El mensaje que ha pasado la prueba de OR lógico aparece porque una de las condiciones de la sentencia `if` es true (`y>200`). Si bien las otras dos expresiones dan como resultado `false`, siempre que una condición dé como resultado `true`, se ejecutará el bloque `if`.

El ejemplo siguiente demuestra cómo el uso de una llamada de función como *expression2* puede producir resultados imprevistos. Si la expresión de la izquierda del operador da como resultado `true`, ese resultado se devolverá sin evaluar la expresión de la derecha (no se llamará a la función `fx2()`).

```
function fx1():Boolean {
 trace("fx1 called");
 return true;
}
function fx2():Boolean {
 trace("fx2 called");
 return true;
}
if (fx1() || fx2()) {
 trace("IF statement entered");
}
```

Se envía la siguiente información al panel Salida: fx1 called IF statement entered

Véase también

`! Operador NOT lógico`, `Operador de desigualdad (!=)`, `Operador de desigualdad estricta (!==)`, `Operador AND lógico (&&)`, `Operador de igualdad (==)`, `Operador de igualdad estricta (===)`

Operador OR lógico (or)

condition1 `or` *condition2*

Desfasado desde Flash Player 5. Este operador está desfasado y en su lugar debe utilizarse `||` `(logical OR)`.

Evalúa *condition1* y *condition2* , y si alguna es `true`, toda la expresión es `true`.

Disponibilidad: Flash Player 4; ActionScript 1.0

Operandos

`condition1 : Boolean` - Expresión que da como resultado `true` o `false`.

`condition2 : Boolean` - Expresión que da como resultado `true` o `false`.

Valor devuelto

`Boolean` - El resultado de la operación lógica.

Véase también

`Operador OR lógico (||)`, `Operador OR en modo bit (|)`

Operador módulo (%)

expression1 % expression2

Calcula el resto de *expression1* dividido entre *expression2*. Si uno de los parámetros *expression* es no numérico, el operador de módulo (%) intenta convertirlos en números. *expression* puede ser un número o una cadena que se convierte en un valor numérico.

El signo del resultado de la operación de módulo coincide con el signo del dividendo (el primer número). Por ejemplo, `-4 % 3` y `-4 % -3` dan ambos como resultado `-1`.

Disponibilidad: Flash Player 4; ActionScript 1.0

Operandos

`expression1 : Number` - Número o expresión que da como resultado un número.

`expression2 : Number` - Número o expresión que da como resultado un número.

Valor devuelto

`Number` - El resultado de la operación aritmética.

Ejemplo

El siguiente ejemplo numérico utiliza el operador de módulo (%):

```
trace(12%5); // traces 2
trace(4.3%2.1); // traces 0.0999999999999996
trace(4%4); // traces 0
```

El primer trace devuelve 2, en lugar de 12/5 o 2,4, porque el operador de módulo (%) sólo devuelve el resto. El segundo trace devuelve 0,0999999999999996 en lugar del previsto 0,1 debido a las limitaciones en la precisión de coma flotante del cálculo binario.

Véase también

`Operador división (/)`, `round (método Math.round)`

Operador de asignación de módulo (%=)

expression1 %= expression2

Asigna a *expression1* el valor de *expression1 % expression2* . Las dos sentencias siguientes son equivalentes:

```
x %= y;
x = x % y;
```

Disponibilidad: Flash Player 4; ActionScript 1.0

Operandos

`expression1 : Number` - Número o expresión que da como resultado un número.

`expression2 : Number` - Número o expresión que da como resultado un número.

Valor devuelto

`Number` - El resultado de la operación aritmética.

Ejemplo

El ejemplo siguiente asigna el valor de `4` a la variable `x`:

```
var x:Number = 14;
var y:Number = 5;
trace(x = y); // output: 4
```

Véase también

`Operador módulo (%)`

Operador de multiplicación (*)

`expression1 * expression2`

Multiplica dos expresiones numéricas. Si ambas expresiones son enteros, el producto es un entero. Si una o ambas expresiones son números de coma flotante, el producto es un número de coma flotante.

Disponibilidad: Flash Player 4; ActionScript 1.0

Operandos

`expression1 : Number` - Número o expresión que da como resultado un número.

`expression2 : Number` - Número o expresión que da como resultado un número.

Valor devuelto

`Number` - Un entero o un número de coma flotante.

Ejemplo

Sintaxis 1: La sentencia siguiente multiplica los enteros 2 y 3:

```
trace(2*3); // output: 6
```

El resultado, 6, es un entero. Sintaxis 2: Esta sentencia multiplica los números de coma flotante 2,0 y 3,1416:

```
trace(2.0 * 3.1416); // output: 6.2832
```

El resultado, 6,2832, es un número de coma flotante.

Operador de asignación de multiplicación (*=)

*expression1 *= expression2*

Asigna a *expression1* el valor de *expression1 * expression2*. Por ejemplo, las dos expresiones siguientes son equivalentes:

```
x *= y;
x = x * y
```

Disponibilidad: Flash Player 4; ActionScript 1.0

Operandos

`expression1 : Number` - Número o expresión que da como resultado un número.

`expression2 : Number` - Número o expresión que da como resultado un número.

Valor devuelto

`Number` - El valor de *expression1 * expression2*. Si una expresión no puede convertirse en un valor numérico, devuelve `NaN` (no es un número).

Ejemplo

Sintaxis 1: El ejemplo siguiente asigna el valor 50 a la variable `x`:

```
var x:Number = 5;
var y:Number = 10;
trace(x *= y); // output: 50
```

Sintaxis 2: La segunda y tercera líneas del ejemplo siguiente calculan las expresiones de la parte derecha del signo igual y asignan los resultados a `x` e `y`:

```
var i:Number = 5;
var x:Number = 4 - 6;
var y:Number = i + 2;
trace(x *= y); // output: -14
```

Véase también

Operador de multiplicación (*)

Operador new

```
new constructor()
```

Crea un objeto nuevo, inicialmente anónimo, y llama a la función identificada por el parámetro `constructor`. El operador `new` pasa a la función parámetros opcionales entre paréntesis, así como el objeto recién creado, al que se hace referencia utilizando la palabra clave `this`. A continuación, la función `constructor` puede utilizar `this` para establecer las variables del objeto.

Disponibilidad: Flash Player 5; ActionScript 1.0

Operandos

`constructor : Object` - Una función seguida de parámetros opcionales entre paréntesis. La función suele ser el nombre del tipo de objeto (por ejemplo, `Array`, `Number` o `Object`) que se va a construir.

Ejemplo

El ejemplo siguiente crea la función `Book()` y, a continuación, utiliza el operador `new` para crear los objetos `book1` y `book2`.

```
function Book(name, price){
 this.name = name;
 this.price = price;
}

book1 = new Book("Confederacy of Dunces", 19.95);
book2 = new Book("The Floating Opera", 10.95);
```

El ejemplo siguiente utiliza el operador `new` para crear un objeto `Array` con 18 elementos:

```
golfCourse_array = new Array(18);
```

Véase también

Operador acceso a matriz ([]), Operador de inicializador de objeto ({})

Operador distinto (ne) (cadenas)

expression1 ne *expression2*

Desfasado desde Flash Player 5. Este operador está desfasado y en su lugar debe utilizarse != (inequality).

Compara *expression1* con *expression2* y devuelve truesi *expression1* no es igual que *expression2* ; de lo contrario devuelve false .

Disponibilidad: Flash Player 4; ActionScript 1.0

Operandos

expression1 : Object - Números, cadenas o variables.

expression2 : Object - Números, cadenas o variables.

Valor devuelto

Boolean - Devuelve true si expression1 es distinto de expression2, y false en caso contrario.

Véase también

Operador de desigualdad (!=)

Operador de inicializador de objeto ({})

object = { *name1 : value1 , name2 : value2 ,... nameN : valueN* }
{*expression1; [...expressionN]*}

Crea un nuevo objeto y lo inicializa con los pares de propiedades *name* y *value* . Utilizar este operador equivale a utilizar la nueva sintaxis de new Object y llenar los pares de propiedades con el operador de asignación. El prototipo del objeto recién creado se denomina de forma genérica el objeto Object.

Este operador se utiliza además para marcar bloques de código contiguo asociado con sentencias de control de flujo (for, while, if, else, switch) y funciones.

Disponibilidad: Flash Player 5; ActionScript 1.0

Operandos

object : Object - El objeto que se va a crear.. *name1,2,...N* Los nombres de las propiedades. *value1,2,...N* Los valores correspondientes para cada propiedad *name*.

Valor devuelto

`Object` -

Sintaxis 1: Un objeto Object.

Sintaxis 2: Ninguno, excepto cuando una función tiene una sentencia `return` explícita, en cuyo caso el tipo devuelto se especifica en la implementación de la función.

Ejemplo

La primera línea del código siguiente crea un objeto vacío utilizando el operador de inicializador de objeto ({}); la segunda línea crea un objeto nuevo empleando una función constructora:

```
var object:Object = {};
var object:Object = new Object();
```

El ejemplo siguiente crea una `account` de objeto e inicializa las propiedades `name`, `address`, `city`, `state`, `zip` y `balance` con los valores correspondientes:

```
var account:Object = {name:"Macromedia, Inc.", address:"600 Townsend
  Street", city:"San Francisco", state:"California", zip:"94103",
  balance:"1000"};
for (i in account) {
 trace("account." + i + " = " + account[i]);
}
```

El ejemplo siguiente muestra cómo pueden anidarse los inicializadores de objeto y matriz entre sí:

```
var person:Object = {name:"Gina Vechio", children:["Ruby", "Chickie",
  "Puppa"]};
```

El ejemplo siguiente utiliza la información del ejemplo anterior y produce el mismo resultado utilizando las funciones constructoras:

```
var person:Object = new Object();
person.name = "Gina Vechio";
person.children = new Array();
person.children[0] = "Ruby";
person.children[1] = "Chickie";
person.children[2] = "Puppa";
```

El ejemplo ActionScript anterior también se puede escribir con el formato siguiente:

```
var person:Object = new Object();
person.name = "Gina Vechio";
person.children = new Array("Ruby", "Chickie", "Puppa");
```

Véase también

`Object`

Operador paréntesis (())

`(`*expression1 [, expression2]*`)`
`(` *expression1, expression2* `)`
function `(` *parameter1,..., parameterN* `)`

Ejecuta una operación de agrupación en uno o varios parámetros, lleva a cabo una evaluación secuencial de las expresiones o rodea uno o varios parámetros y los pasa como parámetros a una función fuera del paréntesis.

Sintaxis 1: Controla el orden de ejecución de los operadores en la expresión. Los paréntesis sustituyen el orden de precedencia normal y pueden hacer que las expresiones entre paréntesis se evalúen primero. Cuando se anidan los paréntesis, el contenido de los paréntesis más interiores se evalúa antes que el contenido de los más exteriores.

Sintaxis 2: Da como resultado una serie de expresiones, separadas por comas, en una secuencia y devuelve el resultado de la expresión final.

Sintaxis 3: Rodea uno o varios parámetros y los pasa como parámetros a la función que está fuera del paréntesis.

Disponibilidad: Flash Player 4; ActionScript 1.0

Operandos

`expression1 : Object` - Números, cadenas, variables o texto.

`expression2 : Object` - Números, cadenas, variables o texto.

`function : Function` - Función que se ejecutará sobre el contenido del paréntesis.

`parameter1...parameterN : Object` - Una serie de parámetros que se ejecutará antes de que se pasen los resultados como parámetros a la función que está fuera del paréntesis.

Ejemplo

Sintaxis 1: Las sentencias siguientes muestran el uso de los paréntesis para controlar el orden de ejecución de las expresiones (el valor de cada expresión aparece en el panel Salida):

```
trace((2 + 3)*(4 + 5)); // Output: 45
trace((2 + 3) * (4 + 5)); // Output: 45trace(2 + (3 * (4 + 5))); // //
  writes 29
trace(2 + (3 * (4 + 5))); // Output: 29trace(2+(3*4)+5); // writes 19
trace(2 + (3 * 4) + 5); // Output: 19
```

Sintaxis 2: El ejemplo siguiente evalúa la función `foo()` y, a continuación, la función `bar()`, y devuelve el resultado de la expresión `a + b`:

```
var a:Number = 1;
var b:Number = 2;
function foo() { a += b; }
```

```
function bar() { b *= 10; }
trace((foo(), bar(), a + b)); // outputs 23
```

Sintaxis 3: El ejemplo siguiente muestra el uso de paréntesis con funciones:

```
var today:Date = new Date();
trace(today.getFullYear()); // traces current year
function traceParameter(param):Void { trace(param); }
traceParameter(2 * 2); //traces 4
```

Véase también

Sentencia with

Operador de igualdad estricta (===)

expression1 === expression2

Comprueba la igualdad de dos expresiones; el operador de igualdad estricta (===) hace lo mismo que el operador de igualdad (==) con la diferencia de que los tipos de datos no se convierten. El resultado es `true` si ambas expresiones, incluidos sus tipos de datos, son iguales.

La definición de igual depende del tipo de datos del parámetro:

- Los valores numéricos y booleanos se comparan por su valor y se consideran iguales si tienen el mismo valor.
- Las expresiones de cadena son iguales si tienen el mismo número de caracteres y éstos son idénticos.
- Las variables que representan objetos, matrices y funciones se comparan por su referencia. Dos de estas variables son iguales si hacen referencia al mismo objeto, matriz o función. Dos matrices independientes nunca se consideran iguales, aunque tengan el mismo número de elementos.

Disponibilidad: Flash Player 6; ActionScript 1.0

Operandos

`expression1 : Object` - Un número, cadena, valor booleano, variable, objeto, matriz o función.

`expression2 : Object` - Un número, cadena, valor booleano, variable, objeto, matriz o función.

Valor devuelto

`Boolean` - El resultado booleano de la comparación.

Ejemplo

Los comentarios del código siguiente muestran el valor devuelto de operaciones que utilizan los operadores de igualdad y de igualdad estricta:

```
// Both return true because no conversion is done
var string1:String = "5";
var string2:String = "5";
trace(string1 == string2); // true
trace(string1 === string2); // true
// Automatic data typing in this example converts 5 to "5"
var string1:String = "5";
var num:Number = 5;
trace(string1 == num); // true
trace(string1 === num); // false
// Automatic data typing in this example converts true to "1"
var string1:String = "1";
var bool1:Boolean = true;
trace(string1 == bool1); // true
trace(string1 === bool1); // false
// Automatic data typing in this example converts false to "0"
var string1:String = "0";
var bool2:Boolean = false;
trace(string1 == bool2); // true
trace(string1 === bool2); // false
```

Los ejemplos siguientes muestran cómo el operador de igualdad estricta trata las variables que son referencias de forma distinta a las variables que contienen valores literales. Esta es una razón para utilizar en todo momento literales de cadena y evitar el uso del operador `new` con la clase String.

```
// Create a string variable using a literal value
var str:String = "asdf";
// Create a variable that is a reference
var stringRef:String = new String("asdf");
// The equality operator does not distinguish among literals, variables,
// and references
trace(stringRef == "asdf"); // true
trace(stringRef == str); // true
trace("asdf" == str); // true
// The strict equality operator considers variables that are references
// distinct from literals and variables
trace(stringRef === "asdf"); // false
trace(stringRef === str); // false
```

Véase también

`!` Operador NOT lógico, Operador de desigualdad (`!=`), Operador de desigualdad estricta (`!==`), Operador AND lógico (`&&`), Operador OR lógico (`||`), Operador de igualdad (`==`)

Operador de desigualdad estricta (!==)

expression1 `!==` *expression2*

Prueba el contrario exacto del operador de igualdad estricta (`===`). El operador de desigualdad estricta realiza lo mismo que el operador de desigualdad, con la diferencia de que no se convierten los tipos de datos.

Si *expression1* es igual que *expression2* y sus tipos de datos son iguales, el resultado es `false`. Como ocurre con el operador de igualdad estricta (===), la definición de igual depende de los tipos de datos que se comparan, tal y como se muestra en la siguiente lista:

- Los números, cadenas y valores booleanos se comparan por su valor.
- Los objetos, matrices y funciones se comparan por su referencia.
- Una variable se compara por su valor o referencia, en función de su tipo.

Disponibilidad: Flash Player 6; ActionScript 1.0

Operandos

`expression1 : Object` - Un número, cadena, valor booleano, variable, objeto, matriz o función.

`expression2 : Object` - Un número, cadena, valor booleano, variable, objeto, matriz o función.

Valor devuelto

`Boolean` - El resultado booleano de la comparación.

Ejemplo

Los comentarios del código siguiente muestran el valor devuelto de operaciones que utilizan los operadores de igualdad (==), de igualdad estricta (===) y de desigualdad estricta (!==):

```
var s1:String = "5";
var s2:String = "5";
var s3:String = "Hello";
var n:Number = 5;
var b:Boolean = true;
trace(s1 == s2); // true
trace(s1 == s3); // false
trace(s1 == n); // true
trace(s1 == b); // false
trace(s1 === s2); // true
trace(s1 === s3); // false
trace(s1 === n); // false
trace(s1 === b); // false
trace(s1 !== s2); // false
```

```
trace(s1 !== s3); // true
trace(s1 !== n); // true
trace(s1 !== b); // true
```

Véase también

`! Operador NOT lógico`, `Operador de desigualdad (!=)`, `Operador AND lógico (&&)`, `Operador OR lógico (||)`, `Operador de igualdad (==)`, `Operador de igualdad estricta (===)`

Operador de delimitador de cadena (")

`"text"`

Si se utilizan antes y después de caracteres, las comillas (") indican que los caracteres tienen un valor literal y se consideran una *cadena* y no una variable ni un valor numérico ni otro elemento de ActionScript.

Disponibilidad: Flash Player 4; ActionScript 1.0

Operandos

`text : String` - Una secuencia de cero o más caracteres.

Ejemplo

El ejemplo siguiente utiliza comillas (") para indicar que el valor de la variable *yourGuess* es la cadena literal `"Prince Edward Island"` y no el nombre de una variable. El valor de `province` es una variable, no un literal; para determinar el valor de `province` es necesario localizar el valor de *yourGuess*.

```
var yourGuess:String = "Prince Edward Island";
submit_btn.onRelease = function() { trace(yourGuess); };
// displays Prince Edward Island
```

Véase también

`String`, `Función String`

Operador de resta (-)

(Negation) -*expression*
(Subtraction) *expression1 - expression2*

Se emplea para negar o restar.

Sintaxis 1: Si se utiliza para negar, invierte el signo de la *expression* numérica. Sintaxis 2: Si se utiliza para restar, ejecuta una resta aritmética en dos expresiones numéricas, restando *expression2* a *expression1* . Si ambas expresiones son enteros, la diferencia es un entero. Si una o ambas expresiones son números de coma flotante, la diferencia es un número de coma flotante.

Disponibilidad: Flash Player 4; ActionScript 1.0

Operandos

expression1 : Number - Número o expresión que da como resultado un número.

expression2 : Number - Número o expresión que da como resultado un número.

Valor devuelto

Number - Un entero o un número de coma flotante.

Ejemplo

Sintaxis 1: La sentencia siguiente invierte el signo de la expresión 2 + 3:

```
trace(-(2+3)); // output: -5
```

Sintaxis 2: La sentencia siguiente resta el entero 2 del entero 5:

```
trace(5-2); // output: 3
```

El resultado, 3, es un entero. Sintaxis 3: La sentencia siguiente resta el número de coma flotante 1,5 del número de coma flotante 3,25:

```
trace(3.25-1.5); // output: 1.75
```

El resultado, 1,75, es un número de coma flotante.

Operador de asignación de resta (-=)

expression1 -= expression2

Asigna a *expression1* el valor de *expression1 - expression2*. Por ejemplo, las dos sentencias siguientes son equivalentes: `x -= y ;x = x - y;`

Las expresiones de cadenas deben convertirse a números; en caso contrario, el resultado es NaN (que no es número).

Disponibilidad: Flash Player 4; ActionScript 1.0

Operandos

`expression1 : Number` - Número o expresión que da como resultado un número.

`expression2 : Number` - Número o expresión que da como resultado un número.

Valor devuelto

`Number` - El resultado de la operación aritmética.

Ejemplo

El ejemplo siguiente utiliza el operador de asignación de resta (`-=`) para restar 10 de 5 y asigna el resultado a la variable `x`:

```
var x:Number = 5;
var y:Number = 10;
x -= y; trace(x); // output: -5
```

El ejemplo siguiente muestra cómo se convierten las cadenas en números:

```
var x:String = "5";
var y:String = "10";
x -= y; trace(x); // output: -5
```

Véase también

`Operador de resta (-)`

: Operador de tipo (:)

[modifiers] var variableName : *type*
function functionName () : *type { ... }*
function functionName (*parameter1*:*type , ... , parameterN*:*type*) *[:type]{ ... }*

Se utiliza en la técnica "strict data typing"; este operador especifica el tipo de variable, el tipo de devolución de función o el tipo de parámetro de función. Cuando se utiliza en una asignación o declaración de variable, este operador especifica el tipo de la variable; cuando se utiliza en una declaración o definición de función, este operador especifica el tipo de devolución de la función; cuando se utiliza con un parámetro de función en una definición de función, este operador especifica el tipo de variable que se espera para dicho parámetro.

Los tipos son una función sólo de compilación. Todos los tipos se comprueban durante la compilación y se generan errores cuando hay una discordancia. Pueden producirse discordancias durante las operaciones de asignación, llamadas a funciones y eliminación de referencias a miembros de clase con el operador de punto (`.`). Para evitar errores de discordancia de tipo, utilice "strict data typing".

Los tipos que pueden utilizarse son todos los tipos de objetos nativos, clases e interfaces definidas por el usuario, Function y Void. Los tipos nativos reconocidos son Boolean, Number y String. También se admiten como tipos nativos todas las clases incorporadas.

Disponibilidad: Flash Player 6; ActionScript 1.0

Operandos

`variableName : Object` - Identificador de una variable. *`type`* - Tipo de datos nativo, nombre de clase definido o nombre de interfaz. *`functionName`* - Un identificador de una función. *`parameter`* - Un identificador de un parámetro de función.

Ejemplo

Sintaxis 1: El ejemplo siguiente declara una variable pública llamada `userName` cuyo tipo es String y le asigna una cadena vacía:

```
var userName:String = "";
```

Sintaxis 2: El ejemplo siguiente muestra cómo especificar el tipo de parámetro de una función definiendo una función llamada `randomInt()` que toma un parámetro llamado `integer` de tipo Number:

```
function randomInt(integer:Number):Number {
 return Math.round(Math.random()*integer);
}
trace(randomInt(8));
```

Sintaxis 3: El ejemplo siguiente define una función llamada `squareRoot()` que toma un parámetro llamado `val` de tipo Number y devuelve la raíz cuadrada de `val`, también de tipo Number:

```
function squareRoot(val:Number):Number {
 return Math.sqrt(val);
}
trace(squareRoot(121));
```

Véase también

`Sentencia var`, `Sentencia function`

Operador typeof

`typeof(`*`expression`*`)`

Evalúa la `expression` y devuelve una cadena que especifica si la expresión es `String`, `MovieClip`, `Object`, `Function`, `Number`, o un valor `Boolean`

Disponibilidad: Flash Player 5; ActionScript 1.0

Operandos

`expression : Object` - Una cadena, clip de película, botón, objeto o función.

Valor devuelto

`String` - Una representación de `String` del tipo de `expression`. La tabla siguiente muestra los resultados del operador `typeof` en cada tipo de `expression`.

Tipo de expresión	Resultado
Cadena	`string`
Clip de película	`movieclip`
Botón	`object`
Campo de texto	`object`
Número	`number`
Booleano	`boolean`
Objeto	`object`
Función	`function`

Véase también

`Operador instanceof`

Operador void

`void` *`expression`*

El operador `void` evalúa una expresión y, a continuación, descarta su valor, devolviendo `undefined`. El operador `void` se emplea a menudo en comparaciones utilizando el operador == para comprobar valores no definidos.

Disponibilidad: Flash Player 5; ActionScript 1.0

Operandos

`expression : Object` - Expresión que se va a evaluar.

Sentencias

Las sentencias son elementos del lenguaje que realizan o especifican una acción. Por ejemplo, la sentencia return devuelve un resultado como valor de la función en la que se ejecuta. La sentencia if evalúa una condición para determinar la siguiente acción que debe realizarse. La sentencia switch crea una estructura ramificada para sentencias de ActionScript.

Resumen de sentencias

Sentencia	Descripción
`break`	Aparece en un bucle (`for`, `for..in`, `do..while` o`while`) o en un bloque de sentencias asociadas con un determinado caso de una sentencia `switch`.
`case`	Define una condición de la sentencia `switch`.
`class`	Define una clase personalizada que permite crear instancias de objetos que comparten métodos y propiedades definidas por el usuario.
`continue`	Salta por encima de todas las sentencias restantes en el bucle más interior e inicia la siguiente repetición del bucle como si se hubiera pasado el control hasta el final del bucle de la forma habitual.
`default`	Define el caso predeterminado de una sentencia `switch`.
`delete`	Destruye la referencia de objeto especificada por el parámetro *`reference`* y devuelve `true` si se elimina la referencia correctamente; de lo contrario, devuelve `false`.
`do..while`	Similar al bucle `while`, con la diferencia de que las sentencias se ejecutan una vez antes de la evaluación inicial de la condición.
`dynamic`	Especifica que el objeto basado en la clase especificada puede añadir propiedades dinámicas y acceder a ellas en tiempo de ejecución.
`else`	Especifica las sentencias que se ejecutarán si la condición en la sentencia `if` devuelve `false`.
`else if`	Evalúa una condición y especifica las sentencias que se ejecutarán si la condición en la sentencia `if` inicial devuelve `false`.
`extends`	Define una clase que es una subclase de otra clase; la última es la superclase.
`for`	Evalúa la expresión *`init`* (inicializar) una vez y, a continuación, inicia una secuencia de reproducción indefinida.
`for..in`	Repite las propiedades de un objeto o los elementos de una matriz y ejecuta *`statement`* para cada propiedad o elemento.

Sentencia	Descripción
`function`	Consta de un conjunto de sentencias que se definen para ejecutar una determinada tarea.
`get`	Permite la *obtención* implícita de propiedades asociadas con objetos basados en las clases definidas en archivos de clase externos.
`if`	Evalúa una condición para determinar la siguiente acción en un archivo SWF.
`implements`	Especifica que una clase debe definir todos los métodos declarados en la interfaz o interfaces que se están implementando.
`import`	Permite acceder a las clases sin especificar sus nombres completos.
`interface`	Define una interfaz.
`intrinsic`	Permite verificar en tiempo de compilación los tipos de clases definidas anteriormente.
`private`	Especifica que una variable o función esté únicamente disponible para la clase que la declara o define o para las subclases de dicha clase.
`public`	Especifica que una variable o función esté disponible para cualquier origen de llamada.
`return`	Especifica el valor devuelto por una función.
`set`	Permite el establecimiento implícito de propiedades asociadas con objetos basados en las clases definidas en archivos de clase externos.
`set variable`	Asigna un valor a una variable.
`static`	Especifica que una variable o función se cree únicamente una sola vez por cada clase, en lugar de crearse en cada objeto basado en dicha clase.
`super`	Invoca la versión de superclase de un método o constructor.
`switch`	Crea una estructura ramificada para sentencias de ActionScript.
`throw`	Genera o *emite* un error que puede controlarse o *capturarse* mediante un bloque de código `catch{}`.
`try..catch..finally`	Incluya un bloque de código en el que puede producirse un error y, a continuación, responda al error.
`var`	Se utiliza para declarar variables locales o de la línea de tiempo.
`while`	Evalúa una condición y, si ésta da como resultado `true`, ejecuta una sentencia o serie de sentencias antes de volver a ejecutar el bucle para evaluar la condición nuevamente.
`with`	Permite especificar un objeto (como un clip de película) con el parámetro *`object`* y evaluar expresiones y acciones en dicho objeto con el parámetro *`statement(s)`*.

Sentencia break

```
break
```

Aparece en un bucle (`for`, `for..in`, `do..while` o`while`) o en un bloque de sentencias asociadas con un determinado caso de una sentencia `switch`. Cuando se utiliza en un bucle, la sentencia `break` ordena a Flash que omita el resto del cuerpo del bucle, detenga la acción de bucle y ejecute la sentencia a continuación de la sentencia de bucle. Cuando se utiliza en una sentencia `switch`, la sentencia `break` ordena a Flash que omita el resto de sentencias en ese bloque de sentencias `case` y que salte a la primera sentencia que vaya a continuación del bloque `switch`.

En bucles anidados, la sentencia `break` sólo omite el resto del bucle inmediato, no toda la serie de bucles anidados. Para salir de toda una serie de bucles anidados, debe usar `try..catch..finally`.

Disponibilidad: Flash Player 4; ActionScript 1.0

Ejemplo

El ejemplo siguiente utiliza la sentencia `break` para salir de un bucle que, de otro modo, sería infinito:

```
var i:Number = 0;
while (true) {
 trace(i);
 if (i >= 10) {
 break; // this will terminate/exit the loop
 }
 i++;
}
```

que obtiene el resultado siguiente:

```
0
1
2
3
4
5
6
7
8
9
10
```

Véase también

`Sentencia for`

Sentencia case

```
case expression : statement(s)
```

Define una condición de la sentencia `switch`. Si el parámetro *`expression`* es igual que el parámetro *`expression`* de la sentencia `switch` que utiliza igual estricta (===), Flash Player ejecutará sentencias en el parámetro *`statement(s)`* hasta que encuentre una sentencia `break` o el final de la sentencia `switch`.

Si se utiliza la sentencia `case` fuera de una sentencia `switch`, se produce un error y el script no se compila.

Nota: Debería finalizar siempre el parámetro *`statement(s)`* con una sentencia break. Si omite `break statement` del parámetro *`statement(s)`*, continúa ejecutándose con la siguiente sentencia `case` en la lugar de salir de la sentencia `switch`.

Disponibilidad: Flash Player 4; ActionScript 1.0

Parámetros

`expression``:String` - Cualquier expresión.

Ejemplo

El ejemplo siguiente define condiciones para la sentencia switch `thisMonth`. Si `thisMonth` es igual que la expresión de la sentencia case, se ejecutará la sentencia.

```
var thisMonth:Number = new Date().getMonth();
switch (thisMonth) {
 case 0 :
 trace("January");
 break;
 case 1 :
 trace("February");
 break;
 case 5 :
 case 6 :
 case 7 :
 trace("Some summer month");
 break;
 case 8 :
 trace("September");
 break;
 default :
 trace("some other month");
}
```

Véase también

`Sentencia break`

Sentencia class

```
[dynamic] class className [ extends superClass ] [ implements interfaceName[,
  interfaceName... ] ] {
 // class definition here
}
```

Define una clase personalizada que permite crear instancias de objetos que comparten métodos y propiedades definidas por el usuario. Por ejemplo, si está desarrollando un sistema de seguimiento de facturas, podría crear una clase invoice (factura) que defina todos los métodos y propiedades que debería tener cada factura. Utilizaría entonces el nuevo comando `new invoice()` para crear objetos invoice.

El nombre de la clase debe coincidir con el nombre del archivo externo que contiene la clase. El nombre del archivo externo debe ser el nombre de la clase con la extensión de archivo .as. Por ejemplo, si se denomina a una clase Student, el archivo que defina la clase debe denominarse Student.as.

Si una clase está contenida en un paquete, la declaración de clase debe utilizar el nombre completo con el formato base.sub1.sub2.MyClass. Además, el archivo AS de la clase debe almacenarse en la ruta de una estructura de directorios que refleje la estructura del paquete, como base/sub1/sub2/MyClass.as. Si la definición de la clase tiene el formato "class MyClass", se encuentra en el paquete predeterminado y el archivo MyClass.as debería situarse en el nivel superior de algún directorio de la ruta.

Por este motivo, es aconsejable planificar una estructura de directorios antes de empezar a crear clases. De otro modo, si decide mover archivos de clases después de crearlos, deberá modificar las sentencias de declaración de clases para reflejar su nueva ubicación.

No es posible anidar definiciones de clase, es decir, no se pueden definir clases adicionales en una definición de clase.

Para indicar que los objetos pueden añadir propiedades dinámicas y acceder a ellas en tiempo de ejecución, incluya la palabra clave `dynamic` antes de la sentencia class. Para declarar que una clase implemente una interfaz, utilice la palabra clave `implements`. Para crear subclases de una clase, utilice la palabra clave `extends`. (Una clase sólo puede ampliar una clase pero puede implementar varias interfaces.) Puede utilizar las palabras clave `implements` y `extends` en una sola sentencia. En el siguiente ejemplo se muestran diversos usos habituales de las palabras claves `implements` y `extends`:

```
class C implements Interface_i, Interface_j // OK
class C extends Class_d implements Interface_i, Interface_j // OK
class C extends Class_d, Class_e // not OK
```

Disponibilidad: Flash Player 6; ActionScript 2.0

Parámetros

className:String - Nombre completo de la clase.

Ejemplo

En el siguiente ejemplo se crea una clase llamada Plant. El constructor Plant utiliza dos parámetros.

```
// Filename Plant.as
class Plant {
 // Define property names and types
 var leafType:String;
 var bloomSeason:String;
 // Following line is constructor
 // because it has the same name as the class
 function Plant(param_leafType:String, param_bloomSeason:String) {
 // Assign passed values to properties when new Plant object is created
 this.leafType = param_leafType;
 this.bloomSeason = param_bloomSeason;
 }
 // Create methods to return property values, because best practice
 // recommends against directly referencing a property of a class
 function getLeafType():String {
 return leafType;
 }
 function getBloomSeason():String {
 return bloomSeason;
 }
}
```

En un archivo de script externo o en el panel Acciones, utilice el operador `new` para crear un objeto Plant.

```
var pineTree:Plant = new Plant("Evergreen", "N/A");
// Confirm parameters were passed correctly
trace(pineTree.getLeafType());
trace(pineTree.getBloomSeason());
```

En el siguiente ejemplo se crea una clase llamada ImageLoader. El constructor `ImageLoader` utiliza tres parámetros.

```
// Filename ImageLoader.as
class ImageLoader extends MovieClip {
 function ImageLoader(image:String, target_mc:MovieClip, init:Object) {
 var listenerObject:Object = new Object();
 listenerObject.onLoadInit = function(target) {
 for (var i in init) {
 target[i] = init[i];
 }
 };
 var JPEG_mcl:MovieClipLoader = new MovieClipLoader();
```

```
 JPEG_mcl.addListener(listenerObject);
 JPEG_mcl.loadClip(image, target_mc);
 }
}
```

En un archivo de script externo o en el panel Acciones, utilice el operador `new` para crear un objeto ImageLoader.

```
var jakob_mc:MovieClip = this.createEmptyMovieClip("jakob_mc",
  this.getNextHighestDepth());
var jakob:ImageLoader = new ImageLoader("http://www.helpexamples.com/flash/
  images/image1.jpg", jakob_mc, {_x:10, _y:10, _alpha:70, _rotation:-5});
```

Véase también

`Sentencia dynamic`

Sentencia continue

```
continue
```

Salta por encima de todas las sentencias restantes en el bucle más interior e inicia la siguiente repetición del bucle como si se hubiera pasado el control hasta el final del bucle de la forma habitual. No tiene ningún efecto fuera de un bucle.

Disponibilidad: Flash Player 4; ActionScript 1.0

Ejemplo

En el siguiente bucle `while`, `continue` hace que el intérprete de Flash omita el resto del cuerpo del bucle y salte a la parte superior del bucle, donde se comprueba la condición:

```
trace("example 1");
var i:Number = 0;
while (i < 10) {
 if (i % 3 == 0) {
 i++;
 continue;
 }
 trace(i);
 i++;
}
```

En el siguiente bucle `do..while`, `continue` hace que el intérprete de Flash omita el resto del cuerpo del bucle y salte a la parte inferior del bucle, donde se comprueba la condición:

```
trace("example 2");
var i:Number = 0;
do {
 if (i % 3 == 0) {
 i++;
```

```
 continue;
 }
 trace(i);
 i++;
}
while (i < 10);
```

En un bucle `for`, `continue` hace que el intérprete de Flash omita el resto del cuerpo del bucle. En el ejemplo siguiente, si el módulo i 3 es igual a 0, se omitirá la sentencia `trace(i)`:

```
trace("example 3");
for (var i = 0; i < 10; i++) {
 if (i % 3 == 0) {
 continue;
 }
 trace(i);
}
```

En el siguiente bucle `for..in`, `continue` hace que el intérprete de Flash omita el resto del cuerpo del bucle y salte otra vez a la parte superior del bucle, donde se procesa el siguiente valor de la numeración:

```
for (i in _root) {
 if (i == "$version") {
 continue;
 }
 trace(i);
}
```

Véase también

Sentencia default

```
default: statements
```

Define el caso predeterminado de una sentencia `switch`. Las sentencias se ejecutan si el parámetro *`expression`* de la sentencia `switch` no equivale (mediante la operación de igualdad estricta [===]) a ninguno de los parámetros *`expression`* que siguen a las palabras claves `case` de una determinada sentencia `switch`.

No es necesario que la sentencia `switch` tenga una sentencia case `default`. Una sentencia case `default` no tiene que ser necesariamente la última de la lista. Si se utiliza una sentencia `default` fuera de una sentencia `switch`, se produce un error y el script no se compila.

Disponibilidad: Flash Player 6; ActionScript 1.0

Parámetros

`statements``:String` - Cualquier sentencia.

Ejemplo

En el ejemplo siguiente, la expresión A no es igual a las expresiones B o D, por lo que se ejecuta la sentencia que sigue a la palabra clave `default` y la sentencia `trace()` se envía al panel Salida.

```
var dayOfWeek:Number = new Date().getDay();
switch (dayOfWeek) {
 case 1 :
 trace("Monday");
 break;
 case 2 :
 trace("Tuesday");
 break;
 case 3 :
 trace("Wednesday");
 break;
 case 4 :
 trace("Thursday");
 break;
 case 5 :
 trace("Friday");
 break;
 default :
 trace("Weekend");
}
```

Véase también

`Sentencia switch`

Sentencia delete

```
delete reference
```

Destruye la referencia de objeto especificada por el parámetro *`reference`* y devuelve `true` si se elimina la referencia correctamente; de lo contrario, devuelve `false`. Este operador es útil para liberar la memoria que utilizan los scripts. Puede utilizar el operador `delete` para eliminar referencias a objetos. Tras eliminar todas las referencias a un objeto, Flash Player elimina el objeto y libera la memoria utilizada por dicho objeto.

Aunque `delete` es un operador, suele utilizarse como una sentencia, tal y como se muestra en el siguiente ejemplo:

```
delete x;
```

El operador `delete` puede no ejecutarse correctamente y devolver `false` si el parámetro *`reference`* no existe o no puede eliminarse. No pueden eliminarse los objetos y propiedades predefinidos, ni las variables declaradas en una función con la sentencia `var`. No se puede utilizar el operador `delete` para eliminar clips de película.

Disponibilidad: Flash Player 5; ActionScript 1.0

Valor devuelto

`Boolean` - Valor booleano.

Parámetros

`reference``:Object` - Nombre de la variable o del objeto que se va a eliminar.

Ejemplo

Sintaxis 1: El ejemplo siguiente crea un objeto, lo utiliza y lo elimina cuando ya no es necesario:

```
var account:Object = new Object();
account.name = "Jon";
account.balance = 10000;
trace(account.name); //output: Jon
delete account;
trace(account.name); //output: undefined
```

Sintaxis 2: El ejemplo siguiente elimina una propiedad de un objeto:

```
// create the new object "account"
var account:Object = new Object();
// assign property name to the account
account.name = "Jon";
// delete the property
delete account.name;
```

Sintaxis 3: El ejemplo siguiente elimina una propiedad de un objeto:

```
var my_array:Array = new Array();
my_array[0] = "abc"; // my_array.length == 1
my_array[1] = "def"; // my_array.length == 2
my_array[2] = "ghi"; // my_array.length == 3
// my_array[2] is deleted, but Array.length is not changed
delete my_array[2];
trace(my_array.length); // output: 3
trace(my_array); // output: abc,def,undefined
```

Sintaxis 4: El ejemplo siguiente muestra el comportamiento de `delete` en referencias de objeto:

```
var ref1:Object = new Object();
ref1.name = "Jody";
// copy the reference variable into a new variable
```

```
// and delete ref1
ref2 = ref1;
delete ref1;
trace("ref1.name "+ref1.name); //output: ref1.name undefined
trace("ref2.name "+ref2.name); //output: ref2.name Jody
```

Si no se hubiera copiado `ref1` en `ref2`, el objeto se habría eliminado cuando se eliminó `ref1` porque no habría referencias a él. Si elimina `ref2`, no hay referencias al objeto, que se destruirá y la memoria que ocupaba quedará disponible.

Véase también

`Sentencia var`

Sentencia do..while

`do { statement(s) } while (condition)`

Similar al bucle `while`, con la diferencia de que las sentencias se ejecutan una vez antes de la evaluación inicial de la condición. Las sentencias a continuación sólo se ejecutan si la condición da como resultado `true`.

Un bucle `do..while` garantiza que el código dentro del bucle se ejecutará al menos una vez. Aunque esto también puede realizarse con un bucle `while` colocando una copia de las sentencias que se van a ejecutar antes de que se inicie el bucle `while`, muchos programadores creen que es más fácil leer los bucles `do..while`.

Si la condición da siempre como resultado `true`, el bucle `do..while` es infinito. Si se introduce un bucle infinito, habrá problemas con Flash Player y finalmente se obtendrá un mensaje de advertencia o se bloqueará el reproductor. Siempre que sea posible, utilice un bucle `for` si sabe el número de veces que desea ejecutar el bucle. Aunque es fácil leer y depurar los bucles `for`, no siempre pueden sustituir a los bucles `do..while`.

Disponibilidad: Flash Player 4; ActionScript 1.0

Parámetros

`condition``:Boolean` - La condición que se va a evaluar. Las *`statement(s)`* dentro del bloque de código `do` se ejecutan siempre que el parámetro *`condition`* de como resultado `true` .

Ejemplo

El ejemplo siguiente utiliza un bucle `do..while` para evaluar si una condición es `true`, y realiza un seguimiento de `myVar` hasta `myVar` que es mayor que 5. Cuando `myVar` es mayor que 5, el bucle termina.

```
var myVar:Number = 0;
```

```
do {
 trace(myVar);
 myVar++;
}
while (myVar < 5);
/* output:
0
1
2
3
4
*/
```

Véase también

Sentencia break

Sentencia dynamic

```
dynamic class className [ extends superClass ] [ implements interfaceName[,
  interfaceName... ] ] {
 // class definition here
}
```

Especifica que el objeto basado en la clase especificada puede añadir propiedades dinámicas y acceder a ellas en tiempo de ejecución.

La verificación de tipos en clases dinámicas es menos estricto que en clases no dinámicas porque los miembros a los que se accede dentro de la definición de clase y en instancias de la clase no se comparan con los definidos en el ámbito de la clase. Sin embargo, pueden verificarse los tipos return y parameter de las funciones de miembros de clase. Este comportamiento es especialmente útil cuando se trabaja con objetos MovieClip, porque hay varias formas de añadir propiedades y objetos a un clip de película dinámicamente, por ejemplo, mediante `MovieClip.createEmptyMovieClip()` y `MovieClip.createTextField()`.

Las subclases de clases dinámicas son también dinámicas.

Asegúrese de especificar el tipo cuando declare un objeto, como se muestra a continuación:

```
var x:MyClass = new MyClass();
```

Si *no* especifica el tipo cuando declara un objeto (como se muestra a continuación), el objeto se considera dinámico:

```
var x = new MyClass();
```

Disponibilidad: Flash Player 6; ActionScript 2.0

Ejemplo

En el ejemplo siguiente, la clase `Person2` aún no se ha marcado como dinámica, por lo que al llamar a una función no declarada en ella se genera un error en tiempo de compilación:

```
class Person2 {
 var name:String;
 var age:Number;
 function Person2(param_name:String, param_age:Number) {
 trace ("anything");
 this.name = param_name;
 this.age = param_age;
 }
}
```

En un archivo FLA o AS que se encuentre en el mismo directorio, añada el código ActionScript siguiente al fotograma 1 de la línea de tiempo:

```
// Before dynamic is added
var craig:Person2 = new Person2("Craiggers", 32);
for (i in craig) {
 trace("craig." + i + " = " + craig[i]);
}
/* output:
craig.age = 32
craig.name = Craiggers */
```

Si añade una función no declarada, `dance`, se generará un error, como se muestra en el ejemplo siguiente:

```
trace("");
craig.dance = true;
for (i in craig) {
 trace("craig." + i + " = " + craig[i]);
}
/* output: **Error** Scene=Scene 1, layer=Layer 1, frame=1:Line 14: There is
   no property with the name 'dance'. craig.dance = true; Total ActionScript
   Errors: 1 Reported Errors: 1 */
```

Añada la palabra clave `dynamic` a la clase Person2, de modo que la primera línea aparezca de este modo:

```
dynamic class Person2 {
```

Pruebe el código de nuevo. Obtendrá el resultado siguiente:

```
craig.dance = true craig.age = 32 craig.name = Craiggers
```

Véase también

`Sentencia class`

Sentencia else

```
if (condition){
statement(s);
} else {
statement(s);
}
```

Especifica las sentencias que se ejecutarán si la condición en la sentencia `if` devuelve `false`. Las llaves (`{}`) que se utilizan para incluir el bloque de sentencias que se ejecutarán con la sentencia `else` no son necesarias si sólo se va a ejecutar una sentencia.

Disponibilidad: Flash Player 4; ActionScript 1.0

Parámetros

condition`:Boolean` - Expresión que da como resultado `true` o `false`.

Ejemplo

En el ejemplo siguiente se utiliza la condición `else` para comprobar si la variable `age_txt` es mayor o menor que 18:

```
if (age_txt.text>=18) {
 trace("welcome, user");
}
else {
 trace("sorry, junior");
 userObject.minor = true;
 userObject.accessAllowed = false;
}
```

En el ejemplo siguiente las llaves (`{}`) no son necesarias porque sólo una sentencia sigue a la sentencia `else`:

```
if (age_txt.text>18) { trace("welcome, user"); } else trace("sorry,
  junior");
```

Véase también

`Sentencia if`

Sentencia else if

```
if(condition) {
statement(s);
} else if(condition) {
statement(s);
}
```

Evalúa una condición y especifica las sentencias que se ejecutarán si la condición en la sentencia `if` inicial devuelve `false`. Si la condición `else if` devuelve `true`, el intérprete de Flash ejecuta las sentencias que hay entre llaves (`{}`), a continuación de la condición. Si la condición `else if` es `false`, Flash omite las sentencias entre llaves y ejecuta las sentencias que hay a continuación.

Utilice la sentencia `else if` para crear una lógica ramificada en los scripts. Si hay varias ramas, debería pensar en la posibilidad de utilizar una sentencia `switch`.

Disponibilidad: Flash Player 4; ActionScript 1.0

Parámetros

condition:`Boolean` - Expresión que da como resultado `true` o `false`.

Ejemplo

El ejemplo siguiente utiliza sentencias `else if` para comparar `score_txt` con un valor especificado:

```
if (score_txt.text>90) {
 trace("A");
}
else if (score_txt.text>75) {
 trace("B");
}
else if (score_txt.text>60) {
 trace("C");
}
else {
 trace("F");
}
```

Véase también

`Sentencia if`

Sentencia extends

`class` *className* extends *otherClassName* `{}`
`interface` *interfaceName* extends *otherInterfaceName* `{}`

Define una clase que es una subclase de otra clase; la última es la superclase. La subclase hereda todos los métodos, propiedades, funciones, etc. definidos en la superclase.

Las interfaces también pueden ampliarse con la palabra clave `extends`. Una interfaz que amplía otra interfaz incluye todas las declaraciones de métodos de la interfaz original.

Disponibilidad: Flash Player 6; ActionScript 2.0

Parámetros

`className``:String` - Nombre de la clase que se define.

Ejemplo

En el ejemplo siguiente, la clase Car amplía la clase Vehicle de modo que se hereden todos sus métodos, propiedades y funciones. Si el script crea una instancia de un objeto Car, podrán utilizarse los métodos de la clase Car y de la clase Vehicle.

El ejemplo siguiente muestra el contenido de un archivo llamado Vehicle.as, que define la clase Vehicle:

```
class Vehicle {
 var numDoors:Number;
 var color:String;
 function Vehicle(param_numDoors:Number, param_color:String) {
 this.numDoors = param_numDoors;
 this.color = param_color;
 }
 function start():Void {
 trace("[Vehicle] start");
 }
 function stop():Void {
 trace("[Vehicle] stop");
 }
 function reverse():Void {
 trace("[Vehicle] reverse");
 }
}
```

El ejemplo siguiente muestra un segundo archivo AS, llamado Car.as, en el mismo directorio. Esta clase amplía la clase Vehicle, modificándola de tres formas. En primer lugar, la clase Car añade una variable `fullSizeSpare` para realizar un seguimiento de si el objeto car tiene una rueda de repuesto a tamaño completo. En segundo lugar, añade un nuevo método específico de coches, `activateCarAlarm()`, que activa la alarma antirrobo del coche. En tercer lugar, anula la función `stop()`para añadir el hecho de que la clase Car utiliza un sistema de frenado ABS para detenerse.

```
class Car extends Vehicle {
 var fullSizeSpare:Boolean;
 function Car(param_numDoors:Number, param_color:String,
  param_fullSizeSpare:Boolean) {
 this.numDoors = param_numDoors;
 this.color = param_color;
 this.fullSizeSpare = param_fullSizeSpare;
 }
 function activateCarAlarm():Void {
 trace("[Car] activateCarAlarm");
 }
 function stop():Void {
 trace("[Car] stop with anti-lock brakes");
 }
}
```

El ejemplo siguiente crea una instancia de un objeto Car, llama a un método definido en la clase Vehicle (`start()`), luego llama al método anulado por la clase Car (`stop()`) y, por último, llama a un método de la clase Car (`activateCarAlarm()`):

```
var myNewCar:Car = new Car(2, "Red", true);
myNewCar.start(); // output: [Vehicle] start
myNewCar.stop(); // output: [Car] stop with anti-lock brakes
myNewCar.activateCarAlarm(); // output: [Car] activateCarAlarm
```

También se puede escribir una subclase de la clase Vehicle utilizando la palabra clave `super`, que puede utilizar la subclase para acceder a propiedades y métodos de la superclase. El ejemplo siguiente muestra un tercer archivo AS, llamado Truck.as, también en el mismo directorio. La clase Truck utiliza la palabra clave `super` en el constructor y también en la función `reverse()` anulada.

```
class Truck extends Vehicle {
 var numWheels:Number;
 function Truck(param_numDoors:Number, param_color:String,
  param_numWheels:Number) {
 super(param_numDoors, param_color);
 this.numWheels = param_numWheels;
 }
 function reverse():Void {
 beep();
 super.reverse();
```

```
 }
 function beep():Void {
 trace("[Truck] make beeping sound");
 }
}
```

El ejemplo siguiente crea una instancia del objeto Truck, llama a un método anulado por la clase Truck (`reverse()`) y, a continuación, llama a un método definido en la clase Vehicle (`stop()`):

```
var myTruck:Truck = new Truck(2, "White", 18);
myTruck.reverse(); // output: [Truck] make beeping sound [Vehicle] reverse
myTruck.stop(); // output: [Vehicle] stop
```

Véase también

`Sentencia class`

Sentencia for

`for(`*init*; *condition*; *next*`) {`
statement(s);
`}`

Evalúa la expresión *`init`* (inicializar) una vez y, a continuación, inicia una secuencia de reproducción indefinida. La secuencia de reproducción indefinida se inicia evaluando la expresión *`condition`* . Si la expresión *`condition`* da como resultado true, se ejecuta *`statement`* y se evalúa la expresión *`next`* . La secuencia de reproducción indefinida se inicia nuevamente con la evaluación de la expresión *`condition`* .

Las llaves (`{}`) que se utilizan para incluir el bloque de sentencias que se ejecutarán con la sentencia `for` no son necesarias si sólo se va a ejecutar una sentencia.

Disponibilidad: Flash Player 5; ActionScript 1.0

Parámetros

`init` - Expresión que se va a evaluar antes de iniciar la secuencia de reproducción indefinida; normalmente es una expresión de asignación. En este parámetro también se permite la sentencia `var`.

Ejemplo

El ejemplo siguiente utiliza `for` para añadir los elementos a una matriz:

```
var my_array:Array = new Array();
for (var i:Number = 0; i < 10; i++) {
 my_array[i] = (i + 5) * 10;
}
trace(my_array); // output: 50,60,70,80,90,100,110,120,130,140
```

El ejemplo siguiente utiliza `for` para realizar la misma acción repetidamente. En el código, el bucle `for` añade los números de 1 a 100.

```
var sum:Number = 0;
for (var i:Number = 1; i <= 100; i++) {
 sum += i;
}
trace(sum); // output: 5050
```

El ejemplo siguiente muestra que las llaves ({}) no son necesarias si sólo se va a ejecutar una sentencia:

```
var sum:Number = 0;
for (var i:Number = 1; i <= 100; i++)
 sum += i;
trace(sum); // output: 5050
```

Véase también

`Operador incremento (++)`

Sentencia for..in

`for (`*variableIterant* in *object*`) {`
statement(s);
`}`

Repite las propiedades de un objeto o los elementos de una matriz y ejecuta *`statement`* para cada propiedad o elemento. Los métodos de un objeto no se enumeran mediante la acción `for..in`.

Algunas propiedades no pueden enumerarse mediante la acción `for..in`. Por ejemplo, las propiedades de clip de película como `_x` e `_y` no se enumeran. En archivos de clase externos, los miembros estáticos no pueden enumerarse, a diferencia de los miembros de instancias.

La sentencia `for..in` repite las propiedades de los objetos de la cadena prototipo del objeto repetido. Primero se enumeran las propiedades del objeto, luego las propiedades de su prototipo inmediato y seguidamente las propiedades del prototipo del prototipo, y así sucesivamente. La sentencia `for..in` no enumera el mismo nombre de propiedad dos veces. Si el objeto `child` tiene un prototipo `parent` y ambos contienen la propiedad `prop`, la sentencia `for..in` a la que se llama en `child` enumera la propiedad `prop` correspondiente a `child` pero ignora la de `parent`.

Las llaves (`{}`) que se utilizan para incluir el bloque de sentencias que se ejecutarán con la sentencia `for..in` no son necesarias si sólo se va a ejecutar una sentencia.

Si escribe un bucle `for..in` en un archivo de clase (un archivo AS externo), los miembros de instancias no estarán disponibles para el bucle, pero sí los miembros estáticos. Sin embargo, si escribe un bucle `for..in` en un archivo FLA para una instancia de la clase, los miembros de instancias estarán disponibles, pero no los miembros estáticos.

Disponibilidad: Flash Player 5; ActionScript 1.0

Parámetros

variableIterant`:String` - Nombre de una variable que actuará como repetidor, haciendo referencia a cada propiedad de un objeto o elemento de una matriz.

Ejemplo

En el siguiente ejemplo se utiliza `for..in` para repetir las propiedades de un objeto:

```
var myObject:Object = {firstName:"Tara", age:27, city:"San Francisco"};
for (var prop in myObject) {
 trace("myObject."+prop+" = "+myObject[prop]);
}
//output
myObject.firstName = Tara
myObject.age = 27
myObject.city = San Francisco
```

En el siguiente ejemplo se utiliza `for..in` para repetir los elementos de una matriz:

```
var myArray:Array = new Array("one", "two", "three");
for (var index in myArray)
 trace("myArray["+index+"] = " + myArray[index]);
// output:
myArray[2] = three
myArray[1] = two
myArray[0] = one
```

El ejemplo siguiente utiliza el operador `typeof` con `for..in` para repetir un determinado tipo de elemento secundario:

```
for (var name in this) {
 if (typeof (this[name]) == "movieclip") {
 trace("I have a movie clip child named "+name);
 }
}
```

Nota: Si tiene varios clips de película, el resultado consiste en los nombres de instancia de esos clips.

El ejemplo siguiente enumera los elementos secundarios de un clip de película y envía cada uno al fotograma 2 en sus líneas de tiempo respectivas. El clip de película `RadioButtonGroup` es un elemento principal con tres secundarios: `_RedRadioButton_`, `_GreenRadioButton_`, y `_BlueRadioButton_`.

```
for (var name in RadioButtonGroup) { RadioButtonGroup[name].gotoAndStop(2);
  }
```

Sentencia function

```
Usage 1: (Declares a named function.)
function functionname([parameter0, parameter1,...parameterN]){
statement(s)
}
Usage 2: (Declares an anonymous function and returns a reference to it.)
function ([parameter0, parameter1,...parameterN]){
statement(s)
}
```

Consta de un conjunto de sentencias que se definen para ejecutar una determinada tarea. Puede definir una función en una ubicación e invocarla o *llamarla* desde distintos scripts de un archivo SWF. Cuando defina una función, puede además especificar parámetros para dicha función. Los parámetros son marcadores de posición de los valores sobre los que opera la función. Puede pasar distintos parámetros a una función cada vez que la llame para poder reutilizar una misma función en situaciones diferentes.

Utilice la sentencia `return` en sentencias (*`statement(s)`*) de una función para que ésta genere o *devuelva* un valor.

Puede utilizar esta sentencia para definir una función (`function`) con los valores especificados de *`functionname`*, *`parameters`* y *`statement(s)`*. Cuando un script llama a una función, se ejecutan las sentencias de la definición de la función. Se permite la creación de referencias hacia delante; en un mismo script, una función puede declararse después de llamarse. Una definición de función sustituye cualquier definición anterior de la misma función. Puede utilizar esta sintaxis donde se permita una sentencia.

También puede utilizar esta sentencia para crear una función anónima y devolverle una referencia. Esta sintaxis se utiliza en expresiones y es especialmente útil para instalar métodos en objetos.

Para añadir funcionalidad, puede utilizar el objeto `arguments` en la definición de la función. El objeto `arguments` suele utilizarse para crear una función que acepte un número variable de parámetros y para crear una función anónima recursiva.

Disponibilidad: Flash Player 5; ActionScript 1.0

Valor devuelto

`String` - Sintaxis 1: El formulario de declaración no devuelve nada. Sintaxis 2: Una referencia a la función anónima.

Parámetros

`functionname``:String` - Nombre de la función declarada.

Ejemplo

El ejemplo siguiente define la función `sqr`, que acepta un parámetro y devuelve `Math.pow(x, 2)` del parámetro:

```
function sqr(x:Number) {
 return Math.pow(x, 2);
}
var y:Number = sqr(3);
trace(y); // output: 9
```

Si se define y utiliza la función en el mismo script, la definición de función puede aparecer después de usar la función:

```
var y:Number = sqr(3);
trace(y); // output: 9
function sqr(x:Number) {
 return Math.pow(x, 2);
}
```

La función siguiente crea un objeto LoadVars y carga params.txt en el archivo SWF. Cuando el archivo se carga correctamente, se realiza un seguimiento de variables cargadas (`variables loaded`):

```
var myLV:LoadVars = new LoadVars();
myLV.load("params.txt");
myLV.onLoad = function(success:Boolean) {
 trace("variables loaded");
}
```

Sentencia get

```
function get property () {
 // your statements here
}
```

Permite la *obtención* implícita de propiedades asociadas con objetos basados en las clases definidas en archivos de clase externos. El uso de métodos get implícitos permite acceder a propiedades de los objetos sin necesidad de acceder directamente a la propiedad. Los métodos get/set implícitos son la abreviatura sintáctica del método `Object.addProperty()` en ActionScript 1.0.

Disponibilidad: Flash Player 6; ActionScript 2.0

Parámetros

property:String - Palabra que se utiliza para hacer referencia a la propiedad a la que accede get; este valor debe coincidir con el valor utilizado en el comando set correspondiente.

Ejemplo

En el ejemplo siguiente se define una clase Team. La clase Team incluye métodos get/set que permiten recuperar y establecer propiedades dentro de la clase:

```
class Team {
 var teamName:String;
 var teamCode:String;
 var teamPlayers:Array = new Array();
 function Team(param_name:String, param_code:String) {
 this.teamName = param_name;
 this.teamCode = param_code;
 }
 function get name():String {
 return this.teamName;
 }
 function set name(param_name:String):Void {
 this.teamName = param_name;
 }
}
```

Introduzca el código ActionScript siguiente en un fotograma de la línea de tiempo:

```
var giants:Team = new Team("San Fran", "SFO");
trace(giants.name);
giants.name = "San Francisco";
trace(giants.name);
/* output:
San Fran San Francisco */
```

Cuando realiza un seguimiento de giants.name, utilice el método get para devolver el valor de la propiedad.

Véase también

addProperty (método Object.addProperty)

Sentencia if

```
if(condition) {
statement(s);
}
```

Evalúa una condición para determinar la siguiente acción en un archivo SWF. Si la condición es `true`, Flash ejecuta las sentencias que hay entre llaves (`{}`), a continuación de la condición. Si la condición es `false`, Flash omite las sentencias entre llaves y ejecuta las sentencias que hay a continuación. Utilice la sentencia `if` junto con las sentencias `else` y `else if` para crear una lógica ramificada en los scripts.

Las llaves (`{}`) que se utilizan para incluir el bloque de sentencias que se ejecutarán con la sentencia `if` no son necesarias si sólo se va a ejecutar una sentencia.

Disponibilidad: Flash Player 4; ActionScript 1.0

Parámetros

`condition``:Boolean` - Expresión que da como resultado `true` o `false`.

Ejemplo

En el ejemplo siguiente, la condición dentro de los paréntesis evalúa la variable `name` para comprobar si tiene el valor literal `"Erica"`. Si lo tiene, se ejecutará la función `play()` dentro de las llaves.

```
if(name == "Erica"){
 play();
}
```

El ejemplo siguiente utiliza una sentencia `if` para evaluar cuánto tarda un usuario en hacer clic en la instancia `submit_btn` en un archivo SWF. Si un usuario hace clic en el botón más de 10 segundos después de que se reproduzca el archivo SWF, la condición dará como resultado `true` y aparecerá el mensaje dentro de las llaves (`{}`) en un campo de texto que se crea en tiempo de ejecución (utilizando `createTextField()`). Si el usuario hace clic en el botón menos de 10 segundos de que se reproduzca el archivo SWF, la condición dará como resultado `false` y aparecerá otro mensaje.

```
this.createTextField("message_txt", this.getNextHighestDepth, 0, 0, 100,
  22);
message_txt.autoSize = true;
var startTime:Number = getTimer();
this.submit_btn.onRelease = function() {
 var difference:Number = (getTimer() - startTime) / 1000;
 if (difference > 10) {
 this._parent.message_txt.text = "Not very speedy, you took "+difference+"
  seconds.";
 }
```

```
 else {
 this._parent.message_txt.text = "Very good, you hit the button in
  "+difference+" seconds.";
 }
};
```

Véase también

Sentencia else

Sentencia implements

myClass implements *interface01* [, *interface02* , ...]

Especifica que una clase debe definir todos los métodos declarados en la interfaz o interfaces que se están implementando.

Disponibilidad: Flash Player 6; ActionScript 2.0

Ejemplo

Véase `interface`.

Véase también

Sentencia class

Sentencia import

`import` *`className`*
`import` *`packageName`*`.*`

Permite acceder a las clases sin especificar sus nombres completos. Por ejemplo, si desea utilizar una clase personalizada macr.util.users.UserClass en un script, debe referirse a ella por su nombre completo o importarla; si la importa, puede referirse a ella por el nombre de clase:

```
// before importing
var myUser:macr.util.users.UserClass = new macr.util.users.UserClass();
// after importing
import macr.util.users.UserClass;
var myUser:UserClass = new UserClass();
```

Si hay varios archivos de clase en el paquete (*`working_directory`*`/macr/utils/users`) a los que desea acceder, puede importarlos todos en una sola sentencia, tal y como se muestra en el siguiente ejemplo:

```
import macr.util.users.*;
```

Debe emitir la sentencia `import` antes de intentar acceder a la clase importada sin especificar su nombre completo.

Si importa una clase pero no la utiliza en el script, la clase no se exporta como parte del archivo SWF. Eso significa que puede importar grandes paquetes sin preocuparse del tamaño del archivo SWF; el código de bytes asociado con una clase se incluye en un archivo SWF únicamente si dicha clase se utiliza realmente.

La sentencia `import` sólo se aplica al script actual (fotograma u objeto) en el que se llama. Por ejemplo, supongamos que importa todas las clases del paquete macr.util en el fotograma 1 de un documento de Flash. En dicho fotograma, puede hacer referencia a las clases del paquete por sus nombres simples:

```
// On Frame 1 of a FLA:
import macr.util.*;
var myFoo:foo = new foo();
```

Sin embargo, en otro script del fotograma necesitaría hacer referencia a las clases del paquete por sus nombres completos (`var myFoo:foo = new macr.util.foo();`) o añadir una sentencia `import` al otro fotograma que importa las clases en dicho paquete.

Disponibilidad: Flash Player 6; ActionScript 2.0

Parámetros

`className``:String` - Nombre completo de una clase definida en un archivo de clase externo.

Ejemplo

Sentencia interface

`interface` *InterfaceName* [extends *InterfaceName* `] {}`

Define una interfaz. Una interfaz es similar a una clase, pero presenta estas diferencias importantes:

- Las interfaces sólo contienen declaraciones de métodos, no su implementación. Es decir, cada clase que implementa una interfaz debe proporcionar una implementación para cada método declarado en la interfaz.
- Sólo se permiten miembros públicos en una definición de interfaz; no se permiten miembros de instancias ni de clases.
- En las definiciones de interfaces no se permiten las sentencias `get` ni `set`.

Disponibilidad: Flash Player 6; ActionScript 2.0

Ejemplo

El ejemplo siguiente muestra varias formas de definir e implementar interfaces:

```
(in top-level package .as files Ia, B, C, Ib, D, Ic, E)
// filename Ia.as
interface Ia {
 function k():Number; // method declaration only
 function n(x:Number):Number; // without implementation
}
// filename B.as
class B implements Ia {
 function k():Number {
 return 25;
 }
 function n(x:Number):Number {
 return x + 5;
 }
} // external script or Actions panel // script file
var mvar:B = new B();
trace(mvar.k()); // 25
trace(mvar.n(7)); // 12
// filename c.as
class C implements Ia {
 function k():Number {
 return 25;
 }
} // error: class must implement all interface methods
// filename Ib.as
interface Ib {
 function o():Void;
}
class D implements Ia, Ib {
 function k():Number {
 return 15;
 }
 function n(x:Number):Number {
 return x * x;
 }
 function o():Void {
 trace("o");
 }
} // external script or Actions panel // script file
mvar = new D();
trace(mvar.k()); // 15
trace(mvar.n(7)); // 49
trace(mvar.o()); // "o"
interface Ic extends Ia {
 function p():Void;
}
class E implements Ib, Ic {
```

```
 function k():Number {
 return 25;
 }
 function n(x:Number):Number {
 return x + 5;
 }
 function o():Void {
 trace("o");
 }
 function p():Void {
 trace("p");
 }
}
```

Véase también

`Sentencia class`

Sentencia intrinsic

```
intrinsic class className [extends superClass] [implements interfaceName [,
  interfaceName...] ] {
 //class definition here
}
```

Permite verificar en tiempo de compilación los tipos de clases definidas anteriormente. Flash utiliza declaraciones de clase intrinsic para permitir la comprobación de tipos de clases incorporadas en tiempo de compilación, como `Array`, `Object` y `String`. Esta palabra clave indica al compilador que no se requiere una implementación de función y que no se debe generar ningún código de bytes para ella.

La palabra clave `intrinsic` también puede emplearse con declaraciones de variable y función. Flash utiliza esta palabra clave para permitir la comprobación de tipos en tiempo de compilación para funciones y propiedades globales.

La palabra clave `intrinsic` se ha creado específicamente para permitir la comprobación de tipos en tiempo de compilación para clases y objetos incorporados, así como variables y funciones globales. Esta palabra clave no se ha diseñado par uso general, aunque puede tener cierto valor para desarrolladores que desean permitir la comprobación de tipos en tiempo de compilación con clases definidas anteriormente, especialmente si las clases se definen empleando ActionScript 1.0.

Esta palabra clave sólo se admite cuando se utiliza en archivos de script externos y no en scripts escritos en el panel Acciones.

Disponibilidad: Flash Player 6; ActionScript 2.0

Ejemplo

El ejemplo siguiente muestra cómo permitir la comprobación de archivos en tiempo de compilación para una clase ActionScript 1.0 definida anteriormente. El código generará un error de tiempo de compilación porque la llamada `myCircle.setRadius()` envía un valor de cadena (`String`) como parámetro en lugar de un valor de número (`Number`). Puede evitar el error cambiando el parámetro por un valor de número (`Number`) (por ejemplo, cambiando `"10"` por `10`).

```
// The following code must be placed in a file named Circle.as
// that resides within your classpath:
intrinsic class Circle {
 var radius:Number;
 function Circle(radius:Number);
 function getArea():Number;
 function getDiameter():Number;
 function setRadius(param_radius:Number):Number;
}

// This ActionScript 1.0 class definition may be placed in your FLA file.
// Circle class is defined using ActionScript 1.0
function Circle(radius) {
 this.radius = radius;
 this.getArea = function(){
 return Math.PI*this.radius*this.radius;
 };
 this.getDiameter = function() {
 return 2*this.radius;
 };
 this.setRadius = function(param_radius) {
 this.radius = param_radius;
 }
}

// ActionScript 2.0 code that uses the Circle class
var myCircle:Circle = new Circle(5);
trace(myCircle.getArea());
trace(myCircle.getDiameter());
myCircle.setRadius("10");
trace(myCircle.radius);
trace(myCircle.getArea());
trace(myCircle.getDiameter());
```

Véase también

Sentencia class

Sentencia private

```
class someClassName{
 private var name;
 private function name() {
 // your statements here
 }
}
```

Especifica que una variable o función esté únicamente disponible para la clase que la declara o define o para las subclases de dicha clase. Palabra clave; una variable o función está disponible para cualquier origen de llamada. Utilice esta palabra clave si desea restringir el acceso a una variable o función. Esta palabra clave está pensada como ayuda al desarrollo de software con objeto de facilitar prácticas de codificación adecuadas, como el encapsulado, y no como mecanismo de seguridad para ocultar o proteger datos confidenciales. No impide necesariamente el acceso a una variable en tiempo de ejecución.

Puede utilizar esta palabra clave únicamente en definiciones de clases, no en definiciones de interfaces.

Disponibilidad: Flash Player 6; ActionScript 2.0

Parámetros

name:String - Nombre de la variable o función que desea especificar como private (privada).

Ejemplo

El ejemplo siguiente ilustra la manera de restringir el acceso a variables o funciones mediante la palabra clave `private`. Cree un archivo AS nuevo llamado Alpha.as.

```
class Alpha {
 private var privateProperty = "visible only within class and subclasses";
 public var publicProperty = "visible everywhere";
}
```

En el mismo directorio que Alpha.as, crea un nuevo archivo AS llamado Beta.as que contiene el siguiente código:

```
class Beta extends Alpha {
 function Beta() {
 trace("privateProperty is " + privateProperty);
 }
}
```

Como demuestra el siguiente código , el constructor de la clase Beta puede acceder a la propiedad `privateProperty` que se hereda de la clase Alpha:

```
var myBeta:Beta = new Beta(); // Output: privateProperty is visible only
  within class and subclasses
```

Intenta acceder a la variable `privateProperty` desde fuera de la clase Alpha o una clase que hereda de la clase Alpha genera un error. El código siguiente, que reside fuera de todas clases, provoca un error:

```
trace(myBeta.privateProperty); // Error
```

Véase también

Sentencia public

Sentencia public

```
class someClassName{
 public var name;
 public function name() {
 // your statements here
 }
}
```

Especifica que una variable o función esté disponible para cualquier origen de llamada. Dado que las variables y funciones son públicas de forma predeterminada, esta palabra clave se utiliza principalmente por motivos de estilo. Por ejemplo, quizá desee utilizarla para mantener la coherencia de un bloque de código que contenga además variables privadas o estáticas.

Disponibilidad: Flash Player 6; ActionScript 2.0

Parámetros

name:String - Nombre de la variable o función que desea especificar como public (pública).

Ejemplo

El ejemplo siguiente muestra cómo se pueden utilizar variables públicas en un archivo de clase. Cree un archivo de clase nuevo llamado User.as e introduzca el código siguiente:

```
class User {
 public var age:Number;
 public var name:String;
}
```

A continuación, cree un archivo FLA o AS nuevo en el mismo directorio e introduzca el código ActionScript siguiente en el fotograma 1 de la línea de tiempo:

```
import User;
var jimmy:User = new User();
jimmy.age = 27;
jimmy.name = "jimmy";
```

Si cambia una de las variables públicas en la clase User por una variable privada, se generará un error al intentar acceder a la propiedad.

Véase también

Sentencia private

Sentencia return

return[*expression*]

Especifica el valor devuelto por una función. La sentencia return evalúa *expression* y devuelve el resultado como valor de la función en la que se ejecuta. La sentencia return devuelve inmediatamente la ejecución a la función que realiza la llamada. Si la sentencia return se utiliza sola, devuelve undefined..

No es posible devolver múltiples valores. Si intenta hacerlo, sólo se devolverá el último valor. En el siguiente ejemplo se devuelve c:

```
return a, b, c ;
```

Si es necesario devolver múltiples valores, quizá desee utilizar una matriz u objeto en su lugar.

Disponibilidad: Flash Player 5; ActionScript 1.0

Valor devuelto

String - El parámetro *expression* evaluado, si se proporciona.

Parámetros

expression - Cadena, número, valor booleano, matriz u objeto que se va a evaluar y devolver como un valor de la función. Este parámetro es opcional.

Ejemplo

El ejemplo siguiente utiliza la sentencia return dentro del cuerpo de la función sum() para devolver el valor sumado de los tres parámetros. La siguiente línea de código llama a sum() y asigna el valor devuelto a la variable newValue.

```
function sum(a:Number, b:Number, c:Number):Number {
 return (a + b + c);
}
var newValue:Number = sum(4, 32, 78);
trace(newValue); // output: 114
```

Véase también

Sentencia function

Sentencia set

```
function set property(varName) {
 // your statements here
}
```

Permite el establecimiento implícito de propiedades asociadas con objetos basados en las clases definidas en archivos de clase externos. El uso de métodos set implícitos permite modificar el valor de la propiedad de un objeto sin acceder a la propiedad directamente. Los métodos get/set implícitos son la abreviatura sintáctica del método `Object.addProperty()` en ActionScript 1.0.

Disponibilidad: Flash Player 6; ActionScript 2.0

Parámetros

property: `String` - Palabra que hace referencia a la propiedad a la que accede `set`; este valor debe coincidir con el valor utilizado en el comando `get` correspondiente.

Ejemplo

El ejemplo siguiente crea una clase Login que demuestra cómo se puede utilizar la palabra clave `set` para establecer variables privadas:

```
class Login {
 private var loginUserName:String;
 private var loginPassword:String;
 public function Login(param_username:String, param_password:String) {
 this.loginUserName = param_username;
 this.loginPassword = param_password;
 }
 public function get username():String {
 return this.loginUserName;
 }
 public function set username(param_username:String):Void {
 this.loginUserName = param_username;
 }
 public function set password(param_password:String):Void {
 this.loginPassword = param_password;
 }
}
```

En un archivo FLA o AS que se encuentre en el mismo directorio que Login.as, introduzca el código ActionScript siguiente en el fotograma 1 de la línea de tiempo:

```
var gus:Login = new Login("Gus", "Smith");
trace(gus.username); // output: Gus
gus.username = "Rupert";
trace(gus.username); // output: Rupert
```

En el ejemplo siguiente, la función `get` se ejecuta cuando se realiza un seguimiento del valor. La función `set` sólo se activa cuando se le pasa un valor, como se muestra en la línea:

```
gus.username = "Rupert";
```

Véase también

`Sentencia get`

Sentencia set variable

`set("variableString",expression)`

Asigna un valor a una variable. Una *variable* es un contenedor que almacena datos. El contenedor es siempre el mismo, pero el contenido puede cambiar. La modificación del valor de una variable a medida que se reproduce el archivo SWF permite registrar y guardar información sobre las acciones del usuario, registrar valores que se modifican conforme se reproduce el archivo SWF o comprobar si una determinada condición es `true` o `false`.

Las variables pueden contener cualquier tipo de datos (por ejemplo, String, Number, Boolean, Object o MovieClip). La línea de tiempo de cada archivo SWF y clip de película tiene su propio conjunto de variables y cada variable tiene su propio valor independiente de las variables de otras líneas de tiempo.

En una sentencia set `set` se permite el uso de "strict data typing". Si se utiliza esta sentencia para establecer una variable en un valor cuyo tipos de datos sea distinto del tipo de datos asociado con la variable en un archivo de clase, no se genera ningún error de compilador.

Una distinción sutil pero importante que hay que tener en cuenta es que el parámetro *`variableString`* es una cadena, no un nombre de variable. Si pasa a `set()` un nombre de variable que ya existe como primer parámetro, sin poner el nombre entre comillas (""), la variable se evalúa antes de que se le asigne el valor de *`expression`*. Por ejemplo, si crea una variable de cadena denominada `myVariable` y le asigna el valor "Tuesday" pero se olvida de ponerlo entre comillas, creará accidentalmente una nueva variable denominada `Tuesday` que contenga el valor que intentó asignar a `myVariable`:

```
var myVariable:String = "Tuesday";
set (myVariable, "Saturday");
trace(myVariable); // outputs Tuesday
trace(Tuesday); // outputs Saturday
```

Para evitarlo, utilice las comillas (""):

```
set ("myVariable", "Saturday");
trace(myVariable); //outputs Saturday
```

Disponibilidad: Flash Player 4; ActionScript 1.0

Parámetros

variableString:String - Cadena que denomina una variable que contiene el valor del parámetro *expression* .

Ejemplo

En el ejemplo siguiente se asigna un valor a una variable. Se asigna el valor de "Jakob" a la variable name.

```
set("name", "Jakob");
trace(name);
```

El código siguiente se reproduce tres veces y crea tres nuevas variables, llamadas caption0, caption1 y caption2:

```
for (var i = 0; i < 3; i++) {
 set("caption" + i, "this is caption " + i);
}
trace(caption0);
trace(caption1);
trace(caption2);
```

Véase también

Sentencia var

Sentencia static

```
class someClassName{
 static var name;
 static function name() {
 // your statements here
 }
}
```

Especifica que una variable o función se cree únicamente una sola vez por cada clase, en lugar de crearse en cada objeto basado en dicha clase.

Puede acceder a un miembro de clase estático sin crear una instancia de la clase mediante la sintaxis someClassName.name. Si crea una instancia de la clase, puede acceder además a un miembro estático a través de la instancia, pero únicamente a través de una función no estática que tenga acceso al miembro estático.

Puede utilizar esta palabra clave únicamente en definiciones de clases, no en definiciones de interfaces.

Disponibilidad: Flash Player 6; ActionScript 2.0

Parámetros

name:String - Nombre de la variable o función que desea especificar como static (estática).

Ejemplo

El ejemplo siguiente demuestra cómo se puede utilizar la palabra clave `static` para crear un contador que realiza un seguimiento del número de instancias de la clase que se han creado. Dado que la variable `numInstances` es estática, se creará sólo una vez para toda la clase, no para cada instancia individual. Cree un archivo nuevo llamado User.as e introduzca el código siguiente:

```
class Users {
 private static var numInstances:Number = 0;
 function Users() {
 numInstances++;
 }
 static function get instances():Number {
 return numInstances;
 }
}
```

Cree un documento FLA o AS en el mismo directorio e introduzca el código ActionScript siguiente en el fotograma 1 de la línea de tiempo:

```
trace(Users.instances);
var user1:Users = new Users();
trace(Users.instances);
var user2:Users = new Users();
trace(Users.instances);
```

Véase también

Sentencia private

Sentencia super

`super.`*method*`([`*arg1, ..., argN*`])`
`super([`*arg1, ..., argN*`])`

El primer estilo de sintaxis puede utilizarse en el cuerpo de un método de objeto para invocar la versión de superclase de un método y puede pasar opcionalmente parámetros `(arg1 ... argN)` al método de superclase. Esto resulta útil para crear métodos de subclase que añaden un comportamiento adicional a los métodos de superclase a la vez que invocan los métodos de superclase para que ejecuten su comportamiento original.

El segundo estilo de sintaxis puede utilizarse en el cuerpo de una función constructora para invocar la versión de superclase de la función constructora y, opcionalmente, puede pasarle parámetros. Esto resulta útil para crear una subclase que ejecute una inicialización adicional y que además invoque el constructor de superclase para que inicialice la superclase.

Disponibilidad: Flash Player 6; ActionScript 1.0

Valor devuelto

En ambos casos se invoca una función. La función puede devolver cualquier valor.

Parámetros

method:Function - Método que se invocará en la superclase.

argN - Parámetros opcionales que se pasan a la versión de superclase del método (sintaxis 1) o a la función constructora de la superclase (sintaxis 2).

Sentencia switch

```
switch (expression){
caseClause:
 [defaultClause:]
}
```

Crea una estructura ramificada para sentencias de ActionScript. Al igual que la sentencia `if`, la sentencia `switch` prueba una condición y ejecuta sentencias si la condición devuelve un valor `true`. Todas las sentencias switch deberían incluir un caso predeterminado. El caso predeterminado debería incluir una sentencia break para evitar un error de paso al siguiente caso si se añade posteriormente otro caso. Cuando se pasa al siguiente caso, no se incluye una sentencia break.

Disponibilidad: Flash Player 4; ActionScript 1.0

Parámetros

expression - Cualquier expresión.

Ejemplo

En el ejemplo siguiente, si el parámetro `String.fromCharCode(Key.getAscii())` da como resultado `A`, se ejecuta la sentencia `trace()` que sigue a `case "A"`; si el parámetro da como resultado `a`, se ejecuta la sentencia `trace()` que sigue a `case "a"` y así sucesivamente. Si ninguna expresión `case` coincide con el parámetro `String.fromCharCode(Key.getAscii())`, se ejecuta la sentencia `trace()` que sigue a la palabra clave `default`.

```
var listenerObj:Object = new Object();
listenerObj.onKeyDown = function() {
 switch (String.fromCharCode(Key.getAscii())) {
 case "A" :
 trace("you pressed A");
 break;
 case "a" :
 trace("you pressed a");
 break;
 case "E" :
 case "e" :
 trace("you pressed E or e");
 break;
 case "I" :
 case "i" :
 trace("you pressed I or i");
 break;
 default :
 trace("you pressed some other key");
 break;
 }
};
Key.addListener(listenerObj);
```

Véase también

`Operador de igualdad estricta (===)`

Sentencia throw

`throw` *`expression`*

Genera o *emite* un error que puede controlarse o *capturarse* mediante un bloque de código `catch{}`. Si un bloque `catch` no captura una excepción, la representación de cadena del valor emitido se envía al panel Salida.

Normalmente, se emiten instancias de la clase Error o de sus subclases (véase el apartado Ejemplo).

Disponibilidad: Flash Player 7; ActionScript 1.0

Parámetros

expression:Object - Expresión u objeto de ActionScript.

Ejemplo

En este ejemplo, una función llamada `checkEmail()` comprueba si la cadena que se le pasa es una dirección de correo electrónico con el formato adecuado. Si la cadena no contiene un símbolo @, la función emite un error.

```
function checkEmail(email:String) {
 if (email.indexOf("@") == -1) {
 throw new Error("Invalid email address");
 }
}
checkEmail("someuser_theirdomain.com");
```

El código siguiente llama a continuación a la función `checkEmail()` dentro de un bloque de código `try`. Si `email_txt` no contiene una dirección de correo electrónico válida, el mensaje de error aparece en un campo de texto `error_txt`.

```
try {
 checkEmail("Joe Smith");
}
catch (e) {
 error_txt.text = e.toString();
}
```

En el ejemplo siguiente se emite una subclase de la clase Error. La función `checkEmail()` se modifica para emitir una instancia de esa subclase.

```
// Define Error subclass InvalidEmailError // In InvalidEmailError.as:
  class InvalidEmailAddress extends Error { var message = "Invalid email
  address."; }
```

En un archivo FLA o AS, introduzca el código ActionScript siguiente en el fotograma 1 de la línea de tiempo:

```
import InvalidEmailAddress;
function checkEmail(email:String) {
 if (email.indexOf("@") == -1) {
 throw new InvalidEmailAddress();
 }
}
try {
 checkEmail("Joe Smith");
}
catch (e) {
 this.createTextField("error_txt", this.getNextHighestDepth(), 0, 0, 100,
  22);
 error_txt.autoSize = true;
 error_txt.text = e.toString();
}
```

Véase también

`Error`

Sentencia try..catch..finally

```
try {
// ... try block ...
} finally {
// ... finally block ...
}
try {
// ... try block ...
} catch(error [:ErrorType1]) {
// ... catch block ...
} [catch(error[:ErrorTypeN]) {
// ... catch block ...
}] [finally {
// ... finally block ...
}]
```

Incluye un bloque de código en el que puede producirse un error y, a continuación, responde al error. Si alguna parte del código del bloque `try` genera un error (mediante la sentencia `throw`), el control pasa al bloque `catch`, si existe uno, y a continuación al bloque de código `finally`, si existe uno. El bloque `finally` se ejecuta siempre, independientemente de que se haya emitido o no un error. Si el código del bloque `try` no emite un error (es decir, si el bloque `try` se completa con normalidad), continúa ejecutándose el código en el bloque `finally`. El bloque `finally` se ejecuta aunque el bloque `try` finalice con una sentencia `return`.

Un bloque `try` debe ir seguido de un bloque `catch`, un bloque `finally`, o ambos. Un único bloque `try` puede tener varios bloques `catch` pero sólo un bloque `finally` bloque. Puede anidar los bloques `try` tantos niveles como desee.

El parámetro *`error`* especificado en un controlador `catch` debe ser un identificador sencillo, como `e`, `theException` o `x`. También puede escribirse la variable en un controlador `catch`. Cuando se utilizan con varios bloques `catch`, los errores clasificados por tipos permiten capturar varios tipos de errores emitidos desde un solo bloque `try`.

Si la excepción emitida es un objeto, el tipo coincidirá si el objeto emitido es una subclase del tipo especificado. Si se emite un error de un tipo específico, se ejecuta el bloque `catch` que controla el error correspondiente. Si se emite una excepción que no es del tipo especificado, el bloque `catch` no se ejecuta y la excepción se envía automáticamente fuera del bloque `try` a un controlador `catch` coincidente.

Si se emite un error en una función y ésta no incluye un controlador `catch`, el intérprete de ActionScript abandona esta función y cualquier otra que realice llamadas hasta que encuentra un bloque `catch`. Durante este proceso, se realizan llamadas a controladores `finally` en todos los niveles.

Disponibilidad: Flash Player 7; ActionScript 1.0

Parámetros

error:`Object` - Expresión emitida desde una sentencia `throw`, normalmente una instancia de la clase Error o una de sus subclases.

Ejemplo

El ejemplo siguiente muestra cómo crear una sentencia `try..finally`. Dado que la ejecución del código del bloque `finally` está garantizada, suele emplearse para realizar la limpieza necesaria después de que se ejecuta un bloque `try`. En el ejemplo siguiente, `setInterval()` llama a una función cada 1.000 milisegundos (1 segundo). Si se produce un error, se emitirá dicho error y será recogido por el bloque `catch`. El bloque finally siempre se ejecuta, tanto si se produce un error como si no. Dado que se utiliza `setInterval()`, debe situarse `clearInterval()` en el bloque `finally` para garantizar que el intervalo se borra de la memoria.

```
myFunction = function () {
 trace("this is myFunction");
};
try {
 myInterval = setInterval(this, "myFunction", 1000);
 throw new Error("my error");
}
catch (myError:Error) {
 trace("error caught: "+myError);
}
finally {
 clearInterval(myInterval);
 trace("error is cleared");
}
```

En el ejemplo siguiente, el bloque `finally` se utiliza para eliminar un objeto ActionScript, independientemente de que se haya producido un error. Cree un archivo AS nuevo llamado Account.as:

```
class Account {
 var balance:Number = 1000;
 function getAccountInfo():Number {
 return (Math.round(Math.random() * 10) % 2);
 }
}
```

Cree un documento AS o FLA en el mismo directorio que Account.as e introduzca el código ActionScript siguiente en el fotograma 1 de la línea de tiempo:

```
import Account;
var account:Account = new Account();
try {
 var returnVal = account.getAccountInfo();
 if (returnVal != 0) {
 throw new Error("Error getting account information.");
 }
}
finally {
 if (account != null) {
 delete account;
 }
}
```

El ejemplo siguiente demuestra una sentencia `try..catch`. Se ejecutará el código que se encuentra dentro del bloque `try`. Si un código del bloque `try` emite una excepción, el control pasará al bloque `catch`, que mostrará el mensaje de error en un campo de texto utilizando el método `Error.toString()`.

Cree un documento FLA en el mismo directorio que Account.as e introduzca el código ActionScript siguiente en el fotograma 1 de la línea de tiempo:

```
import Account;
var account:Account = new Account();
try {
 var returnVal = account.getAccountInfo();
 if (returnVal != 0) {
 throw new Error("Error getting account information.");
 }
 trace("success");
}
catch (e) {
 this.createTextField("status_txt", this.getNextHighestDepth(), 0, 0, 100,
  22);
 status_txt.autoSize = true;
 status_txt.text = e.toString();
}
```

El ejemplo siguiente muestra un bloque de código `try` con múltiples bloques de código `catch` clasificados. Según el tipo de error que se haya producido, el bloque de código `try` emite un tipo de objeto distinto. En este caso, `myRecordSet` es una instancia de una clase (hipotética) llamada RecordSet cuyo método `sortRows()` puede emitir dos tipos de errores, RecordSetException y MalformedRecord.

En el ejemplo siguiente, los objetos RecordSetException y MalformedRecord son subclases de la clase Error. Cada uno se define en su propio archivo de clase AS.

```
// In RecordSetException.as:
class RecordSetException extends Error {
 var message = "Record set exception occurred.";
}
// In MalformedRecord.as:
class MalformedRecord extends Error {
 var message = "Malformed record exception occurred.";
}
```

En el método `sortRows()` de la clase RecordSet se emite uno de los objetos de error definidos anteriormente, según el tipo de excepción que se haya producido. El ejemplo siguiente muestra qué aspecto puede tener este código:

```
class RecordSet {
 function sortRows() {
 var returnVal:Number = randomNum();
 if (returnVal == 1) {
 throw new RecordSetException();
 }
 else if (returnVal == 2) {
 throw new MalformedRecord();
 }
 }
 function randomNum():Number {
 return Math.round(Math.random() * 10) % 3;
 }
}
```

Por último, en otro archivo AS o script FLA, el código siguiente invoca el método `sortRows()` en una instancia de la clase RecordSet. Define bloques `catch` para cada tipo de error que emite `sortRows()`.

```
import RecordSet;
var myRecordSet:RecordSet = new RecordSet();
try {
 myRecordSet.sortRows();
 trace("everything is fine");
}
catch (e:RecordSetException) {
 trace(e.toString());
}
catch (e:MalformedRecord) {
 trace(e.toString());
}
```

Véase también

`Error`

Sentencia var

```
var variableName [= value1][...,variableNameN[=valueN]]
```

Se utiliza para declarar variables locales. Si declara variables en una función, las variables son locales. Se definen para la función y caducan al finalizar la llamada a la función. En concreto, una variable definida mediante `var` es local en el bloque de código que la contiene. Los bloques de código se delimitan mediante llaves ({}).

Si declara variables fuera de una función, las variables estarán disponibles en toda la línea de tiempo que contiene la sentencia.

No puede declarar como variable local una variable incluida en el ámbito de otro objeto.

```
my_array.length = 25; // ok
var my_array.length = 25; // syntax error
```

Cuando utilice `var`, puede aplicar "strict data typing" a la variable.

Puede declarar múltiples variables en una sentencia, separando las declaraciones con comas (si bien esta sintaxis puede reducir la claridad del código):

```
var first:String = "Bart", middle:String = "J.", last:String = "Bartleby";
```

Nota: Debe utilizar además `var` cuando declare propiedades en definiciones de clase en scripts externos. Los archivos de clase también son compatibles con los ámbitos de variables públicas, privadas y estáticas.

Disponibilidad: Flash Player 5; ActionScript 1.0

Parámetros

`variableName``:String` - Un identificador.

Ejemplo

El código ActionScript siguiente crea una matriz nueva de nombres de producto. `Array.push` añade un elemento al final de la matriz.. Si desea utilizar "strict data typing", es fundamental que utilice la palabra clave `var`. Sin `var` delante de `product_array`, obtendrá errores cuando intente utilizar "strict data typing".

```
var product_array:Array = new Array("MX 2004", "Studio", "Dreamweaver",
  "Flash", "ColdFusion", "Contribute", "Breeze");
product_array.push("Flex");
trace(product_array);
// output: MX
  2004,Studio,Dreamweaver,Flash,ColdFusion,Contribute,Breeze,Flex
```

Sentencia while

```
while(condition) {
statement(s);
}
```

Evalúa una condición y, si ésta da como resultado `true`, ejecuta una sentencia o serie de sentencias antes de volver a ejecutar el bucle para evaluar la condición nuevamente. Después de que la condición dé como resultado `false`, se omite la sentencia o serie de sentencias y finaliza el bucle.

La sentencia `while` ejecuta la siguiente serie de pasos. Cada repetición de pasos del 1 al 4 se denomina *repetición* del bucle. Al principio de cada repetición se vuelve a probar la condición *`condition`* , tal y como se muestra en los siguientes pasos:

- Se evalúa la expresión *`condition`*.
- Si *`condition`* da como resultado `true` o un valor que se convierte en el valor booleano `true`, por ejemplo, un número distinto de cero, vaya al paso 3. De lo contrario, se completa la sentencia `while` y se reanuda la ejecución en la sentencia que hay a continuación del bucle `while`.
- Ejecute el bloque de sentencias *`statement(s)`*.
- Vaya al paso 1.

La reproducción indefinida suele utilizarse para ejecutar una acción mientras la variable de contador sea inferior al valor especificado. Al final de cada bucle se incrementa el contador hasta que se alcanza el valor especificado. En dicho punto, *`condition`* ya no es `true` y finaliza el bucle.

Las llaves (`{}`) que se utilizan para incluir el bloque de sentencias que se ejecutarán con la sentencia `while` no son necesarias si sólo se va a ejecutar una sentencia.

Disponibilidad: Flash Player 4; ActionScript 1.0

Parámetros

`condition``:Boolean` - Expresión que da como resultado `true` o `false`.

Ejemplo

En el ejemplo siguiente, la sentencia `while` se utiliza para comprobar una expresión. Cuando el valor de `i` es menor que 20, se realizará un seguimiento de dicho valor (`i`). Cuando la condición deja de ser `true`, se bucle finaliza.

```
var i:Number = 0;
while (i < 20) {
 trace(i);
 i += 3;
}
```

El resultado siguiente se muestra en el panel Salida.

```
0
3
6
9
12
15
18
```

Véase también

`Sentencia continue`

Sentencia with

```
with (object:Object) {
statement(s);
}
```

Permite especificar un objeto (como un clip de película) con el parámetro *`object`* y evaluar expresiones y acciones en dicho objeto con el parámetro *`statement(s)`*. De esta forma, evitará tener que escribir una y otra vez el nombre del objeto o la ruta de acceso al objeto.

El parámetro *`object`* se convierte en el contexto en el que se leen las propiedades, variables y funciones del parámetro *`statement(s)`*. Por ejemplo, si *`object`* es `my_array`, y dos de las propiedades especificadas son `length` y `concat`, dichas propiedades se leen automáticamente como `my_array.length` y `my_array.concat`. En otro ejemplo, si *`object`* es `state.california`, se llama a todas las acciones o sentencias dentro de la sentencia `with` incluidas desde dentro de la instancia `california`.

Para buscar el valor de un identificador en el parámetro *`statement(s)`*, ActionScript empieza al principio de la cadena de ámbito especificada por *`object`* y busca el identificador en cada nivel de la cadena de ámbito, en un orden específico.

La cadena de ámbito que utiliza la sentencia `with` para dirigirse a los identificadores empieza con el primer elemento de la siguiente lista y continúa hasta el último elemento:

- El objeto especificado en el parámetro *`object`* en la sentencia `with` más interior.
- El objeto especificado en el parámetro *`object`* en la sentencia `with` más exterior.
- El objeto Activation (un objeto temporal que se crea automáticamente cuando se llama a una función que contiene las variables locales a las que se llama en la función).
- El clip de película que contiene el script que se está ejecutando.
- El objeto Global (objetos incorporados como Math y String).

Para establecer una variable dentro de una sentencia `with`, debe haber declarado la variable fuera de la sentencia `with` o debe introducir la ruta de acceso completa a la línea de tiempo en la que desea incluir la variable. Si establece una variable en una sentencia `with` sin declararla, la sentencia `with` buscará el valor según la cadena de ámbito. Si la variable no existe ya, se establecerá el nuevo valor en la línea de tiempo desde la que se ha llamado a la sentencia `with`.

En lugar de utilizar `with()`, puede utilizar rutas directas. Si cree que es incómodo escribir las largas rutas de acceso, puede crear una variable local y almacenar la ruta en la variable, que a continuación puede reutilizar en el código, tal y como ocurre en el siguiente código ActionScript:

```
var shortcut = this._parent._parent.name_txt; shortcut.text = "Hank";
  shortcut.autoSize = true;
```

Disponibilidad: Flash Player 5; ActionScript 1.0

Parámetros

`object``:Object` - Una instancia de un objeto o clip de película de ActionScript.

Ejemplo

El ejemplo siguiente establece las propiedades `_x` e `_y` de la instancia `someOther_mc` y, a continuación, indica a `someOther_mc` que pase al fotograma 3 y se detenga.

```
with (someOther_mc) {
 _x = 50;
 _y = 100;
 gotoAndStop(3);
}
```

El fragmento de código siguiente muestra cómo escribir el código anterior sin utilizar una sentencia `with`.

```
someOther_mc._x = 50;
someOther_mc._y = 100;
someOther_mc.gotoAndStop(3);
```

La sentencia `with` resulta útil para acceder a múltiples elementos de una lista de cadena de ámbito simultáneamente. En el ejemplo siguiente, el objeto `Math` incorporado se sitúa delante de la cadena de ámbito. El establecimiento de `Math` como objeto predeterminado resuelve los identificadores `cos`, `sin` y `PI` en `Math.cos`, `Math.sin` y `Math.PI`, respectivamente. Los identificadores `a`, `x`, `y` y `r` no son métodos ni propiedades del objeto `Math`, pero dado que existen en el ámbito de activación de objetos de la función `polar()`, se resuelven en las variables locales correspondientes.

```
function polar(r:Number):Void {
 var a:Number, x:Number, y:Number;
 with (Math) {
 a = PI * pow(r, 2);
 x = r * cos(PI);
 y = r * sin(PI / 2);
 }
 trace("area = " + a);
 trace("x = " + x);
 trace("y = " + y);
} polar(3);
```

El resultado siguiente se muestra en el panel Salida.

```
area = 28.2743338823081
x = -3
y = 3
```

CAPÍTULO 2

Clases de ActionScript

2

La documentación de clases de ActionScript incluye sintaxis, información de uso y ejemplos de código de métodos, propiedades y controladores y detectores de eventos que pertenecen a una clase específica en ActionScript (frente a las funciones y propiedades globales). Las clases están ordenadas alfabéticamente e incluyen las nuevas clases de Flash Player 8 que se encuentran en los paquetes flash.*. Si tiene dudas sobre la clase a la que pertenece un método o propiedad, puede buscarla en el índice.

Accessibility

```
Object
  |
  +-Accessibility
```

```
public class Accessibility
extends Object
```

La clase Accessibility gestiona la comunicación con lectores de pantalla. Los lectores de pantalla son un tipo de tecnología auxiliar para usuarios invidentes que ofrece una versión audio del contenido de la pantalla. Los métodos de la clase Accessibility son estáticos, lo que significa que no tiene que crear una instancia de la clase para utilizar sus métodos.

Para obtener y establecer las propiedades de un objeto concreto, como, por ejemplo, un botón, un clip de película o un campo de texto, utilice la propiedad `_accProps`. Para determinar si el reproductor se está ejecutando en un entorno que admite elementos de accesibilidad, utilice `System.capabilities.hasAccessibility`.

Disponibilidad: ActionScript 1.0; Flash Player 6

Véase también

`hasAccessibility (propiedad capabilities.hasAccessibility)`, `Propiedad _accProps`

Resumen de propiedades

Propiedades heredadas de la clase Object

```
constructor (propiedad Object.constructor), __proto__ (propiedad
Object.__proto__), prototype (propiedad Object.prototype), __resolve
(propiedad Object.__resolve)
```

Resumen de métodos

Modificadores	Firma	Descripción
`static`	`isActive() : Boolean`	Indica si una ayuda de accesibilidad está activa actualmente y el reproductor se está comunicando con ella.
`static`	`updateProperties() : Void`	Hace que surtan efecto todos los cambios realizados en `_accProps` (propiedades de accesibilidad).

Métodos heredados de la clase Object

```
addProperty (método Object.addProperty), hasOwnProperty (método
Object.hasOwnProperty), isPropertyEnumerable (método
Object.isPropertyEnumerable), isPrototypeOf (método Object.isPrototypeOf),
registerClass (método Object.registerClass), toString (método
Object.toString), unwatch (método Object.unwatch), valueOf (método
Object.valueOf), watch (método Object.watch)
```

isActive (Método Accessibility.isActive)

```
public static isActive() : Boolean
```

Indica si una ayuda de accesibilidad está activa actualmente y el reproductor se está comunicando con ella. Utilice este método cuando desee que la aplicación se comporte de forma diferente en presencia de un lector de pantalla u otra ayuda de accesibilidad.

Nota: Si llama a este método en un período de uno o dos segundos desde la primera aparición de la ventana de Flash en la que se está reproduciendo el documento, es posible que se devuelva el valor `false` aunque haya activo un cliente Microsoft Active Accessibility (MSAA). Esto se debe a que el mecanismo de comunicación entre Flash y los clientes MSAA es asíncrono. Puede evitar esta limitación asegurándose de que se produce un retardo de entre uno y dos segundos tras la carga del documento antes de llamar a este método.

Disponibilidad: ActionScript 1.0; Flash Player 6

Valor devuelto

`Boolean` - Valor booleano: `true` si Flash Player está comunicando con una ayuda de accesibilidad (normalmente un lector de pantalla); `false` en cualquier otro caso.

Ejemplo

El ejemplo siguiente comprueba si está activa actualmente una ayuda de accesibilidad:

```
if (Accessibility.isActive()) {
  trace ("An accessibility aid is currently active");
} else {
  trace ("There is currently no active accessibility aid");
}
```

Véase también

`updateProperties (método Accessibility.updateProperties)`, `Propiedad _accProps`, `hasAccessibility (propiedad capabilities.hasAccessibility)`

updateProperties (método Accessibility.updateProperties)

```
public static updateProperties() : Void
```

Hace que todos los cambios realizados en t`_accProps` (propiedades de accesibilidad) surtan efecto. Para más información sobre la configuración de propiedades de accesibilidad, consulte `_accProps`.

Si modifica las propiedades de accesibilidad para múltiples objetos, sólo será necesaria una llamada a `Accessibility.updateProperties()`; si se producen múltiples llamadas, podría producirse una reducción en el rendimiento y podrían originarse resultados ininteligibles en el lector de pantalla.

Disponibilidad: ActionScript 1.0; Flash Player 6,0,65,0

Ejemplo

Si cambia una imagen y desea actualizar su descripción de accesibilidad, puede utilizar el siguiendo código ActionScript:

```
my_mc.gotoAndStop(2);

if (my_mc._accProps == undefined ) {
  my_mc._accProps = new Object();
}
my_mc._accProps.name = "Photo of Mount Rushmore";
Accessibility.updateProperties();
```

Véase también

isActive (Método Accessibility.isActive), Propiedad _accProps, hasAccessibility (propiedad capabilities.hasAccessibility)

arguments

```
Object
  |
  +-arguments
```

public class *arguments*
extends Object

Un objeto `arguments` se utiliza para almacenar y acceder a argumentos de función. Desde dentro del cuerpo de función se puede acceder a él con la variable `arguments` local.

Los argumentos se almacenan como elementos de matriz, al primero se accede como `arguments[0]`, al segundo como `arguments[1]`, etc. La propiedad `arguments.length` indica el número de argumentos que se pasan a la función. Observe que puede haber diferencia entre el número de argumentos que se pasan y los que declara la función.

Disponibilidad: ActionScript 1.0; Flash Player 5

Véase también

Function

Resumen de propiedades

Modificadores	Propiedad	Descripción
	`callee:Object`	Una referencia a la función que se está ejecutando.
	`caller:Object`	Una referencia a la función que llamó a la función que se está ejecutando o `null` si no se llamó desde otra función.
	`length:Number`	El número de argumentos que se pasan a la función.

Propiedades heredadas de la clase Object

constructor (propiedad Object.constructor), __proto__ (propiedad Object.__proto__), prototype (propiedad Object.prototype), __resolve (propiedad Object.__resolve)

Resumen de métodos

Métodos heredados de la clase Object

addProperty (método Object.addProperty), hasOwnProperty (método Object.hasOwnProperty), isPropertyEnumerable (método Object.isPropertyEnumerable), isPrototypeOf (método Object.isPrototypeOf), registerClass (método Object.registerClass), toString (método Object.toString), unwatch (método Object.unwatch), valueOf (método Object.valueOf), watch (método Object.watch)

callee (propiedad arguments.callee)

```
public callee : Object
```

Una referencia a la función que se está ejecutando.

Disponibilidad: ActionScript 1.0; Flash Player 5

Véase también

caller (propiedad arguments.caller)

caller (propiedad arguments.caller)

```
public caller : Object
```

Una referencia a la función que llamó a la función que se está ejecutando o `null` si no se llamó desde otra función.

Disponibilidad: ActionScript 1.0; Flash Player 6

Véase también

callee (propiedad arguments.callee)

length (propiedad arguments.length)

```
public length : Number
```

El número de argumentos que se pasan a la función. Puede ser más o menos el que declara la función.

Disponibilidad: ActionScript 1.0; Flash Player 5

Array

```
Object
  |
  +-Array
```

```
public dynamic class Array
extends Object
```

La clase Array le permite obtener acceso a matrices indexadas y manipularlas. Una matriz indexada es un objeto cuyas propiedades se identifican mediante un número que representa su posición en la matriz. Este número se conoce como *índice*. Todas las matrices indexadas están basadas en cero, lo que significa que el primer elemento de la matriz es [0], el segundo elemento, [1], y así sucesivamente. Para crear un objeto Array, deberá utilizar el constructor new `Array()`. Para obtener acceso a elementos de una matriz, utilice el operador de acceso a matriz (`[]`).

Puede almacenar una amplia variedad de tipos de datos en un elemento de matriz, incluidos números, cadenas, objetos e incluso otras matrices. Puede crear una matriz *multidimensional* creando una matriz indexada y asignando a cada uno de sus elementos una matriz indexada distinta. Una matriz de este tipo se considera multidimensional porque puede emplearse para representar datos en una tabla.

La asignación de matriz se hace por referencia en lugar de por valor: cuando se asigna una variable de matriz a otra variable de matriz, ambas hacen referencia a la misma matriz:

```
var oneArray:Array = new Array("a", "b", "c");
var twoArray:Array = oneArray; // Both array variables refer to the same
  array.
twoArray[0] = "z";
trace(oneArray); // Output: z,b,c.
```

No se debe utilizar la clase Array para crear *matriz asociativas*, que son estructuras de datos distintas que contienen elementos con nombre en lugar de elementos numerados. Deberá emplear el objeto Object para crear matrices asociativas (también llamadas *hashes*). Si bien ActionScript permite crear matrices asociativas empleando la clase Array, no es posible utilizar ninguno de los métodos o las propiedades de la clase Array. En el fondo, una matriz asociativa es una instancia de la clase Object y cada par clave-valor se representa mediante una propiedad y su valor. Otra razón para declarar una matriz asociativa como tipo Object es que permite utilizar un literal de objeto para llenar la matriz asociativa (pero sólo en el momento en que se declara). El ejemplo siguiente crea una matriz asociativa utilizando un literal de objeto, accede a los elementos empleando el operador de punto y el operador de acceso a matriz y, a continuación, añade un nuevo par clave-valor creando una propiedad nueva:

```
var myAssocArray:Object = {fname:"John", lname:"Public"};
trace(myAssocArray.fname); // Output: John
```

```
trace(myAssocArray["lname"]); // Output: Public
myAssocArray.initial = "Q";
trace(myAssocArray.initial); // Output: Q
```

Disponibilidad: ActionScript 1.0; Flash Player 5

Ejemplo

En el ejemplo siguiente, my_array contiene cuatro meses del año:

```
var my_array:Array = new Array();
my_array[0] = "January";
my_array[1] = "February";
my_array[2] = "March";
my_array[3] = "April";
```

Resumen de propiedades

Modificadores	Propiedad	Descripción
static	CASEINSENSITIVE:Number	En los métodos de ordenación, esta constante especifica una ordenación sin distinción entre mayúsculas y minúsculas.
static	DESCENDING:Number	En los métodos de ordenación, esta constante especifica una ordenación descendente.
	length:Number	Entero no negativo que especifica el número de elementos de la matriz.
static	NUMERIC:Number	En los métodos de ordenación, esta constante especifica una ordenación numérica (en lugar de la basada en cadenas)
static	RETURNINDEXEDARRAY:Number	Especifica que una ordenación devuelve una matriz indexada como resultado de llamar al método `sort()` o `sortOn()`.
static	UNIQUESORT:Number	En los métodos de ordenación, esta constante especifica requisito de ordenación exclusiva.

Propiedades heredadas de la clase Object

constructor (propiedad Object.constructor), __proto__ (propiedad Object.__proto__), prototype (propiedad Object.prototype), __resolve (propiedad Object.__resolve)

Resumen de constructores

Firma	Descripción
`Array([value:Object])`	Permite crear una matriz.

Resumen de métodos

Modificadores	Firma	Descripción
	`concat([value:Object]) : Array`	Concatena los elementos especificados en los parámetros con los elementos de una matriz y crea una matriz nueva.
	`join([delimiter:String]) : String`	Convierte los elementos de una matriz en cadenas, inserta el separador especificado entre los elementos, los concatena y devuelve la cadena resultante.
	`pop() : Object`	Elimina el último elemento de una matriz y devuelve el valor de dicho elemento.
	`push(value:Object) : Number`	Añade uno o varios elementos al final de una matriz y devuelve la nueva longitud de la matriz.
	`reverse() : Void`	Invierte la colocación de la matriz.
	`shift() : Object`	Elimina el primer elemento de una matriz y devuelve dicho elemento.
	`slice([startIndex:Number], [endIndex:Number]) : Array`	Devuelve una matriz nueva que consta de un rango de elementos de la matriz original sin modificar la matriz original.
	`sort([compareFunction:Object], [options:Number]) : Array`	Ordena los elementos de una matriz.
	`sortOn(fieldName:Object, [options:Object]) : Array`	Ordena los elementos de una matriz por uno o varios campos de la matriz.
	`splice(startIndex:Number, [deleteCount:Number], [value:Object]) : Array`	Añade y elimina elementos de una matriz.

Modificadores	Firma	Descripción
	`toString() : String`	Devuelve un valor de cadena que representa los elementos del objeto Array especificado.
	`unshift(value:Object) : Number`	Añade uno o varios elementos al comienzo de una matriz y devuelve la nueva longitud de la matriz.

Métodos heredados de la clase Object

`addProperty (método Object.addProperty), hasOwnProperty (método Object.hasOwnProperty), isPropertyEnumerable (método Object.isPropertyEnumerable), isPrototypeOf (método Object.isPrototypeOf), registerClass (método Object.registerClass), toString (método Object.toString), unwatch (método Object.unwatch), valueOf (método Object.valueOf), watch (método Object.watch)`

Contructor Array

`public Array([value:Object])`

Permite crear una matriz. Puede utilizar el constructor para crear diferentes tipos de matrices: una matriz vacía, una matriz con una longitud específica pero cuyos elementos tengan valores no definidos o una matriz cuyos elementos tengan valores específicos.

Sintaxis 1: Si no especifica ningún parámetro, se creará una matriz con una longitud de 0.

Sintaxis 2: Si especifica sólo una longitud, se creará una matriz con el número de elementos `length`. El valor de cada elemento se establece como `undefined`.

Sintaxis 3: Si utiliza los parámetros `element` para especificar los valores, se creará una matriz con valores específicos.

Disponibilidad: ActionScript 1.0; Flash Player 5

Parámetros

`value``:Object` [opcional] - Uno de los siguientes valores:

- Un entero que especifica el número de elementos de la matriz.
- Una lista de dos o más valores arbitrarios. Los valores pueden ser de tipo Boolean, Number, String, Object o Array. El primer elemento de una matriz siempre tiene un índice o una posición de 0.

Nota: Si sólo se pasa un parámetro numérico al constructor Array, se asumirá que es `length` y se convertirá a un entero empleando la función `Integer()`.

Ejemplo

Sintaxis 1: El ejemplo siguiente crea un objeto `Array` nuevo con una longitud inicial de 0:

```
var my_array:Array = new Array();
trace(my_array.length); // Traces 0.
```

Sintaxis 2: El ejemplo siguiente crea un objeto Array nuevo con una longitud inicial de 4:

```
var my_array:Array = new Array(4);
trace(my_array.length); // Returns 4.
trace(my_array[0]); // Returns undefined.
if (my_array[0] == undefined) { // No quotation marks around undefined.
  trace("undefined is a special value, not a string");
} // Traces: undefined is a special value, not a string.
```

Sintaxis 3: El ejemplo siguiente crea el objeto Array nuevo `go_gos_array` con una longitud inicial de 5:

```
var go_gos_array:Array = new Array("Belinda", "Gina", "Kathy", "Charlotte",
  "Jane");
trace(go_gos_array.length); // Returns 5.
trace(go_gos_array.join(", ")); // Displays elements.
```

Los elementos iniciales de la matriz `go_gos_array` están identificados, como se muestra en el ejemplo siguiente:

```
go_gos_array[0] = "Belinda";
go_gos_array[1] = "Gina";
go_gos_array[2] = "Kathy";
go_gos_array[3] = "Charlotte";
go_gos_array[4] = "Jane";
```

El código siguiente añade un sexto elemento a la matriz `go_gos_array` y cambia el segundo elemento:

```
go_gos_array[5] = "Donna";
go_gos_array[1] = "Nina"
trace(go_gos_array.join(" + "));
// Returns Belinda + Nina + Kathy + Charlotte + Jane + Donna.
```

Véase también

Operador acceso a matriz ([]), length (propiedad Array.length)

CASEINSENSITIVE (propiedad Array.CASEINSENSITIVE)

```
public static CASEINSENSITIVE : Number
```

En los métodos de ordenación, esta constante especifica una ordenación sin distinción entre mayúsculas y minúsculas. Puede utilizar esta constante para el parámetro `options` en el método `sort()` o `sortOn()`.

El valor de esta constante es 1.

Disponibilidad: ActionScript 1.0; Flash Player 7

Véase también

`sort (método Array.sort)`, `sortOn (método Array.sortOn)`

concat (método Array.concat)

```
public concat([value:Object]) : Array
```

Concatena los elementos especificados en los parámetros con los elementos de una matriz y crea una matriz nueva. Si los parámetros de `value` especifican una matriz, se concatenarán los elementos de dicha matriz en lugar de la propia matriz. La matriz `my_array` permanece sin cambios.

Disponibilidad: ActionScript 1.0; Flash Player 5

Parámetros

`value``:Object` [opcional] - Números, elementos o cadenas que se van a concatenar en una matriz nueva. Si no se pasa ningún valor, se creará un duplicado de `my_array`.

Valor devuelto

`Array` - Una matriz que contiene los elementos de esta matriz seguidos de los elementos de los parámetros.

Ejemplo

El código siguiente concatena dos matrices:

```
var alpha_array:Array = new Array("a","b","c");
var numeric_array:Array = new Array(1,2,3);
var alphaNumeric_array:Array =alpha_array.concat(numeric_array);
trace(alphaNumeric_array);
// Creates array [a,b,c,1,2,3].
```

El código siguiente concatena tres matrices:

```
var num1_array:Array = [1,3,5];
var num2_array:Array = [2,4,6];
var num3_array:Array = [7,8,9];
var nums_array:Array=num1_array.concat(num2_array,num3_array)
trace(nums_array);
// Creates array [1,3,5,2,4,6,7,8,9].
```

Las matrices anidadas no se alisan de la misma forma que las matrices normales. Los elementos de una matriz anidada no se dividen en elementos separados en la matriz `x_array`, como se muestra en el ejemplo siguiente:

```
var a_array:Array = new Array ("a","b","c");

// 2 and 3 are elements in a nested array.
var n_array:Array = new Array(1, [2, 3], 4);

var x_array:Array = a_array.concat(n_array);
trace(x_array[0]); // a
trace(x_array[1]); // b
trace(x_array[2]); // c
trace(x_array[3]); // 1
trace(x_array[4]); // 2, 3
trace(x_array[5]); // 4
```

DESCENDING (propiedad Array.DESCENDING)

```
public static DESCENDING : Number
```

En los métodos de ordenación, esta constante especifica una ordenación descendente. Puede utilizar esta constante para el parámetro `options` en el método `sort()` o `sortOn()`.

El valor de esta constante es 2.

Disponibilidad: ActionScript 1.0; Flash Player 7

Véase también

`sort (método Array.sort)`, `sortOn (método Array.sortOn)`

join (método Array.join)

```
public join([delimiter:String]) : String
```

Convierte los elementos de una matriz en cadenas, inserta el separador especificado entre los elementos, los concatena y devuelve la cadena resultante. Una matriz anidada siempre va separada por una coma (,), no por el separador pasado al método `join()`.

Disponibilidad: ActionScript 1.0; Flash Player 5

Parámetros

delimiter:String [opcional] - Un carácter o una cadena que separa los elementos de matriz en la cadena devuelta. Si omite este parámetro, se utilizará una coma (,) como separador predeterminado.

Valor devuelto

String - Una cadena.

Ejemplo

El ejemplo siguiente crea una matriz con tres elementos: Earth, Moon y Sun. A continuación une la matriz tres veces: la primera, utilizando el separador predeterminado (una coma [,] y un espacio), luego empleando un guión (-) y, por último, con el signo más (+).

```
var a_array:Array = new Array("Earth","Moon","Sun")
trace(a_array.join());
// Displays Earth,Moon,Sun.
trace(a_array.join(" - "));
// Displays Earth - Moon - Sun.
trace(a_array.join(" + "));
// Displays Earth + Moon + Sun.
```

El ejemplo siguiente crea una matriz anidada que contiene dos matrices. La primera matriz tiene tres elementos: Europa, Io y Callisto. La segunda matriz tiene dos elementos: Titan y Rhea. Une la matriz con el signo más (+), pero los elementos dentro de cada matriz anidada permanecen separados por comas (,).

```
var a_nested_array:Array = new Array(["Europa", "Io", "Callisto"],
  ["Titan", "Rhea"]);
trace(a_nested_array.join(" + "));
// Returns Europa,Io,Callisto + Titan,Rhea.
```

Véase también

split (método String.split)

length (propiedad Array.length)

```
public length : Number
```

Entero no negativo que especifica el número de elementos de la matriz. Esta propiedad se actualiza automáticamente cuando se añaden nuevos elementos a la matriz. Cuando se asigna un valor a un elemento de matriz (por ejemplo, `my_array[index] = value`), si `index` es un número e `index+1` es mayor que la propiedad `length`, esta propiedad `length` se actualizará a `index+1`.

Nota: Si asigna un valor a la propiedad `length` inferior a la longitud existente, la matriz se truncará.

Disponibilidad: ActionScript 1.0; Flash Player 5

Ejemplo

El código siguiente explica cómo se actualiza la propiedad `length`. La longitud inicial es 0 y luego se actualiza a 1, 2 y 10. Si asigna una valor a la propiedad `length` inferior a la longitud existente, la matriz se truncará:

```
var my_array:Array = new Array();
trace(my_array.length); // initial length is 0
my_array[0] = "a";
trace(my_array.length); // my_array.length is updated to 1
my_array[1] = "b";
trace(my_array.length); // my_array.length is updated to 2
my_array[9] = "c";
trace(my_array.length); // my_array.length is updated to 10
trace(my_array);
// displays:
//
  a,b,undefined,undefined,undefined,undefined,undefined,undefined,undefine
  d,c

// if the length property is now set to 5, the array will be truncated
my_array.length = 5;
trace(my_array.length); // my_array.length is updated to 5
trace(my_array); // outputs: a,b,undefined,undefined,undefined
```

NUMERIC (propiedad Array.NUMERIC)

```
public static NUMERIC : Number
```

En los métodos de ordenación, esta constante especifica una ordenación numérica (en lugar de la basada en cadenas) Si se incluye en el parámetro `options` hace que los métodos `sort()` y `sortOn()` ordenen los números como valores numéricos, no como cadenas de caracteres numéricos. Sin la constante NUMERIC, la ordenación trata cada elemento de matriz como una cadena de carácter y genera los resultados en orden Unicode.

Por ejemplo, en la matriz de valores [2005, 7, 35], si la constante NUMERIC *no* está incluida en el parámetro `options`, la matriz ordenada es [2005, 35, 7], pero si la constante NUMERIC *sí* está incluida, la matriz ordenada es [7, 35, 2005].

Tenga en cuenta que esta constante sólo se aplica a los números de la matriz, y no a las cadenas que contienen datos numéricos (como ["23", "5"]).

El valor de esta constante es 16.

Disponibilidad: ActionScript 1.0; Flash Player 7

Véase también

`sort (método Array.sort)`, `sortOn (método Array.sortOn)`

pop (método Array.pop)

```
public pop() : Object
```

Elimina el último elemento de una matriz y devuelve el valor de dicho elemento.

Disponibilidad: ActionScript 1.0; Flash Player 5

Valor devuelto

`Object` - El valor del último elemento de la matriz especificada.

Ejemplo

El código siguiente crea la matriz `myPets_array` que contiene cuatro elementos y, a continuación, elimina su último elemento:

```
var myPets_array:Array = new Array("cat", "dog", "bird", "fish");
var popped:Object = myPets_array.pop();
trace(popped); // Displays fish.
trace(myPets_array); // Displays cat,dog,bird.
```

Véase también

`push (método Array.push)`, `shift (método Array.shift)`, `unshift (método Array.unshift)`

push (método Array.push)

```
public push(value:Object) : Number
```

Añade uno o varios elementos al final de una matriz y devuelve la nueva longitud de la matriz.

Disponibilidad: ActionScript 1.0; Flash Player 5

Parámetros

`value``:Object` - Uno o más valores que se añadirán a la matriz.

Valor devuelto

`Number` - Un entero que representa la longitud de la nueva matriz.

Ejemplo

El ejemplo siguiente crea una matriz `myPets_array` con dos elementos definidos: `cat` y `dog`. La segunda línea añade dos elementos a la matriz.

Dado que el método `push()` devuelve la nueva longitud de la matriz, la sentencia `trace()` de la última línea envía la nueva longitud de `myPets_array` (`4`) al panel Salida.

```
var myPets_array:Array = new Array("cat", "dog");
var pushed:Number = myPets_array.push("bird", "fish");
trace(pushed); // Displays 4.
```

Véase también

`pop (método Array.pop)`, `shift (método Array.shift)`, `unshift (método Array.unshift)`

RETURNINDEXEDARRAY (propiedad Array.RETURNINDEXEDARRAY)

```
public static RETURNINDEXEDARRAY : Number
```

Especifica que una ordenación devuelve una matriz indexada como resultado de llamar al método `sort()` o `sortOn()`. Puede utilizar esta constante para el parámetro `options` en el método `sort()` o `sortOn()`. Proporciona funciones de vista previa o copia al devolver una matriz que representa los resultados de la ordenación y deja la matriz original sin modificar.

El valor de esta constante es 8.

Disponibilidad: ActionScript 1.0; Flash Player 7

Véase también

`sort (método Array.sort)`, `sortOn (método Array.sortOn)`

reverse (método Array.reverse)

```
public reverse() : Void
```

Invierte la colocación de la matriz.

Disponibilidad: ActionScript 1.0; Flash Player 5

Ejemplo

El ejemplo siguiente utiliza este método para invertir la matriz `numbers_array`:

```
var numbers_array:Array = new Array(1, 2, 3, 4, 5, 6);
trace(numbers_array); // Displays 1,2,3,4,5,6.
numbers_array.reverse();
trace(numbers_array); // Displays 6,5,4,3,2,1.
```

shift (método Array.shift)

```
public shift() : Object
```

Elimina el primer elemento de una matriz y devuelve dicho elemento.

Disponibilidad: ActionScript 1.0; Flash Player 5

Valor devuelto

`Object` - El primer elemento de una matriz.

Ejemplo

El código siguiente crea la matriz `myPets_array` y, a continuación, elimina el primer elemento de la matriz y lo asigna a la variable `shifted`:

```
var myPets_array:Array = new Array("cat", "dog", "bird", "fish");
var shifted:Object = myPets_array.shift();
trace(shifted); // Displays "cat".
trace(myPets_array); // Displays dog,bird,fish.
```

Véase también

`pop (método Array.pop)`, `push (método Array.push)`, `unshift (método Array.unshift)`

slice (método Array.slice)

```
public slice([startIndex:Number], [endIndex:Number]): Array
```

Devuelve una matriz nueva que consta de un rango de elementos de la matriz original sin modificar la matriz original. La matriz devuelta incluye el elemento `startIndex` y todos los elementos hasta `endIndex`, sin incluir éste.

Si no se pasa ningún parámetro, se creará un duplicado de la matriz original.

Disponibilidad: ActionScript 1.0; Flash Player 5

Parámetros

startIndex:Number [opcional] - Un número que especifica el índice del punto inicial para la división. Si *start* es un número negativo, el punto inicial comenzará al final de la matriz, donde -1 es el último elemento.

endIndex:Number [opcional] - Un número que especifica el índice del punto final para la división. Si omite este parámetro, la división incluirá todos los elementos desde el punto inicial hasta el final de la matriz. Si *end* es un número negativo, el punto final se especificará desde el final de la matriz, donde -1 es el último elemento.

Valor devuelto

Array - Una matriz que consta de un rango de elementos de la matriz original.

Ejemplo

El ejemplo siguiente crea una matriz de cinco mascotas y utiliza slice() para llenar una nueva matriz que sólo contiene las mascotas con cuatro patas:

```
var myPets_array:Array = new Array("cat", "dog", "fish", "canary",
  "parrot");
var myFourLeggedPets_array:Array = new Array();
var myFourLeggedPets_array = myPets_array.slice(0, 2);
trace(myFourLeggedPets_array); // Returns cat,dog.
trace(myPets_array); // Returns cat,dog,fish,canary,parrot.
```

El ejemplo siguiente crea una matriz de cinco mascotas y, a continuación, utiliza slice() con un parámetro start negativo para copiar los dos últimos elementos de la matriz:

```
var myPets_array:Array = new Array("cat", "dog", "fish", "canary",
  "parrot");
var myFlyingPets_array:Array = myPets_array.slice(-2);
trace(myFlyingPets_array); // Traces canary,parrot.
```

El ejemplo siguiente crea una matriz de cinco mascotas y utiliza slice() con un parámetro end negativo para copiar el elemento central de la matriz:

```
var myPets_array:Array = new Array("cat", "dog", "fish", "canary",
  "parrot");
var myAquaticPets_array:Array = myPets_array.slice(2,-2);
trace(myAquaticPets_array); // Returns fish.
```

sort (método Array.sort)

`public sort([compareFunction:Object], [options:Number]) : Array`

Ordena los elementos de una matriz. Flash ordena por valores Unicode. (ASCII es un subconjunto de Unicode.)

De manera predeterminada, `Array sort()` funciona de la forma descrita en la siguiente lista:

- La ordenación distingue entre mayúsculas y minúsculas (*Z* precede a *a*).
- La ordenación es ascendente (*a* precede a *b*).
- La matriz se modifica para reflejar el orden de clasificación; los elementos que tengan campos de ordenación idénticos se sitúan consecutivamente en la matriz ordenada sin seguir ningún orden específico.
- Los campos numéricos se ordenan como si fueran cadenas, de manera que 100 precede a 99, dado que "1" es un valor de cadena inferior a "9".

Si desea ordenar una matriz utilizando valores que se desvían de los predeterminados, puede utilizar una de las opciones de ordenación que se describen en la entrada del parámetro `options` o puede crear una función personalizada para realizar la ordenación. Si crea una función personalizada, podrá utilizarla llamando al método `sort()`, con el nombre de dicha función como primer parámetro (`compareFunction`).

Disponibilidad: ActionScript 1.0; Flash Player 5

Parámetros

`compareFunction``:Object` [opcional] - Una función de comparación utilizada para determinar el orden de los elementos de una matriz. Dados los elementos A y B, el resultado de `compareFunction` puede tener uno de los tres valores siguientes:

- -1, si A debe aparecer antes que B en la secuencia ordenada
- 0, si A es igual a B
- 1, si A debe aparecer después que B en la secuencia ordenada

`options``:Number` [opcional] - Uno o más números o nombres de constantes definidas, separados por el operador `|` (OR en modo bit), que cambian el comportamiento de la ordenación respecto al valor predeterminado. Estos son los valores válidos del parámetro `options`:

- `Array.CASEINSENSITIVE` o 1
- `Array.DESCENDING` o 2
- `Array.UNIQUESORT` o 4
- `Array.RETURNINDEXEDARRAY` o 8
- `Array.NUMERIC` o 16

Para más información sobre este parámetro, consulte el método `Array.sortOn()`.

Nota: `Array.sort()` se define en ECMA-262, pero las opciones de ordenación de matrices introducidas en Flash Player 7 son extensiones específicas de Flash para la especificación ECMA-262.

Valor devuelto

`Array` - El valor devuelto depende de si pasa algún parámetro, como se describe en la siguiente lista:

- Si especifica el valor 4 o `Array.UNIQUESORT` para el parámetro `options` y dos o más elementos ordenados tienen campos de ordenación idénticos, Flash devolverá el valor 0 y no modificará la matriz.
- Si especifica el valor 8 o `Array.RETURNINDEXEDARRAY` para el parámetro `options`, Flash devolverá una matriz que reflejará los resultados de la ordenación y no modificará la matriz.
- En cualquier otro caso, Flash no devuelve nada y modifica la matriz de manera que refleje el orden de clasificación.

Ejemplo

Sintaxis 1: En ejemplo siguiente muestra el uso de `Array.sort()` con y sin un valor pasado para `options`:

```
var fruits_array:Array = new Array("oranges", "apples", "strawberries",
  "pineapples", "cherries");
trace(fruits_array); // Displays
  oranges,apples,strawberries,pineapples,cherries.
fruits_array.sort();
trace(fruits_array); // Displays
  apples,cherries,oranges,pineapples,strawberries.
trace(fruits_array); // Writes
  apples,cherries,oranges,pineapples,strawberries.
fruits_array.sort(Array.DESCENDING);
trace(fruits_array); // Displays
  strawberries,pineapples,oranges,cherries,apples.
trace(fruits_array); // Writes
  strawberries,pineapples,oranges,cherries,apples.
```

Sintaxis 2: El ejemplo siguiente utiliza `Array.sort()` con una función compare: Las entradas se ordenan en el formato nombre:contraseña Ordene utilizando como criterio la parte del nombre solamente:

```
var passwords_array:Array = new Array("mom:glam", "ana:ring", "jay:mag",
  "anne:home", "regina:silly");
function order(a, b):Number {
  var name1:String = a.split(":")[0];
```

```
  var name2:String = b.split(":")[0];
  if (name1<name2) {
  return -1;
  } else if (name1>name2) {
  return 1;
  } else {
  return 0;
  }
}
trace("Unsorted:");
//Displays Unsorted:
trace(passwords_array);
//Displays mom:glam,ana:ring,jay:mag,anne:home,regina:silly.
//Writes mom:glam,ana:ring,jay:mag,anne:home,regina:silly
passwords_array.sort(order);
trace("Sorted:");
//Displays Sorted:
trace(passwords_array);
//Displays ana:ring,anne:home,jay:mag,mom:glam,regina:silly.
//Writes ana:ring,anne:home,jay:mag,mom:glam,regina:silly.
```

Véase también

Operador OR en modo bit (|), sortOn (método Array.sortOn)

sortOn (método Array.sortOn)

```
public sortOn(fieldName:Object, [options:Object]) : Array
```

Ordena los elementos de una matriz por uno o varios campos de la matriz. La matriz debe tener las siguientes características:

- La matriz es indexada, no asociativa.
- Cada elemento de la matriz contiene un objeto con una o varias propiedades.
- Todos los objetos tienen al menos una propiedad en común, cuyos valores pueden emplearse para ordenar la matriz. Esa propiedad se denomina *campo*.

Si pasa varios parámetros `fieldName`, el primer campo representará al campo de ordenación principal, el segundo representará al siguiente campo de ordenación, etc. Flash ordena por valores Unicode. (ASCII es un subconjunto de Unicode.) Si uno de los elementos que se están comparando no contiene el campo especificado en el parámetro `fieldName`, se dará por hecho que el campo no está definido (`undefined`) y los elementos se colocarán consecutivamente en la matriz ordenada sin seguir ningún orden determinado.

De manera predeterminada, `Array.sortOn()` funciona de la siguiente forma:

- La ordenación distingue entre mayúsculas y minúsculas (*Z* precede a *a*).
- La ordenación es ascendente (*a* precede a *b*).

- La matriz se modifica para reflejar el orden de clasificación; los elementos que tengan campos de ordenación idénticos se sitúan consecutivamente en la matriz ordenada sin seguir ningún orden específico.
- Los campos numéricos se ordenan como si fueran cadenas, de manera que 100 precede a 99, dado que "1" es un valor de cadena inferior a "9".

Flash Player 7 añadía el parámetro `options`, que puede utilizar para sobrescribir el comportamiento de ordenación anterior. Para ordenar una matriz sencilla (por ejemplo, una matriz con un solo campo) o si desea especificar un orden de clasificación que el parámetro `options` no admite, utilice `Array.sort()`.

Para pasar varios indicadores, sepárelos con el operador OR en modo de bit (`|`):

```
my_array.sortOn(someFieldName, Array.DESCENDING | Array.NUMERIC);
```

Flash Player 8 ofrece además la posibilidad de especificar una opción de ordenación diferente para cada campo cuando se realiza una ordenación por más de un campo. En Flash Player 8, el parámetro `options` acepta una matriz de opciones de ordenación de manera que cada opción de ordenación corresponde a un campo de ordenación en el parámetro `fieldName`. El ejemplo siguiente ordena el campo de ordenación principal `a` en orden descendente, el campo de ordenación secundario `b` empleando una ordenación numérica y el campo de ordenación terciario `c` utilizando una ordenación sin distinción entre mayúsculas y minúsculas:

```
Array.sortOn (["a", "b", "c"], [Array.DESCENDING, Array.NUMERIC,
  Array.CASEINSENSITIVE]);
```

Nota: Las matrices `fieldName` y `options` deben tener el mismo número de elementos; de lo contrario la matriz `options` se ignora. Asimismo, las opciones `Array.UNIQUESORT` y `Array.RETURNINDEXEDARRAY` sólo pueden utilizarse como primer elemento de la matriz y, en caso contrario, se ignoran.

Disponibilidad: ActionScript 1.0; Flash Player 6

Parámetros

`fieldName``:Object` - Una cadena que identifica un campo que se va a utilizar como valor de ordenación, o una matriz en la cual el primer elemento representa el campo de ordenación principal, el segundo representa el campo de ordenación secundario, etc.

`options``:Object` [opcional] - Uno o más números o nombres de constantes definidas, separados por el operador `bitwise OR (|)`, que cambian el comportamiento de la ordenación. Estos son los valores válidos del parámetro `options`:

- `Array.CASEINSENSITIVE` o 1
- `Array.DESCENDING` o 2
- `Array.UNIQUESORT` o 4

- `Array.RETURNINDEXEDARRAY` o 8
- `Array.NUMERIC` o 16

Las sugerencias están activadas si se utiliza la forma de cadena del indicador (por ejemplo, `DESCENDING`) en lugar de su forma numérica (2).

Valor devuelto

`Array` - El valor devuelto depende de si se pasa algún parámetro:

- Si especifica el valor 4 o `Array.UNIQUESORT` para el parámetro `options` y dos o más elementos ordenados tienen campos de ordenación idénticos, se devolverá el valor 0 y no se modificará la matriz.
- Si especifica el valor 8 o `Array.RETURNINDEXEDARRAY` para el parámetro `options`, se devolverá una matriz que reflejará los resultados de la ordenación y no se modificará la matriz.
- En cualquier otro caso, no se devuelve nada y se modifica la matriz de manera que refleje el orden de clasificación.

Ejemplo

El ejemplo siguiente crea una matriz nueva y la ordena de acuerdo con los campos `name` y `city`. La primera ordenación utiliza `name` como primer valor de ordenación y `city` como segundo. La segunda ordenación utiliza `city` como primer valor de ordenación y `name` como segundo.

```
var rec_array:Array = new Array();
rec_array.push({name: "john", city: "omaha", zip: 68144});
rec_array.push({name: "john", city: "kansas city", zip: 72345});
rec_array.push({name: "bob", city: "omaha", zip: 94010});
for(i=0; i<rec_array.length; i++){
  trace(rec_array[i].name + ", " + rec_array[i].city);
}
// Results:
// john, omaha
// john, kansas city
// bob, omaha

rec_array.sortOn(["name", "city"]);
for(i=0; i<rec_array.length; i++){
  trace(rec_array[i].name + ", " + rec_array[i].city);
}
// Results:
// bob, omaha
// john, kansas city
// john, omaha
```

```
rec_array.sortOn(["city", "name" ]);
for(i=0; i<rec_array.length; i++){
  trace(rec_array[i].name + ", " + rec_array[i].city);
}
// Results:
// john, kansas city
// bob, omaha
// john, omaha
```

La matriz de objetos siguiente se utiliza en los ejemplos restantes, que muestran cómo utilizar el parámetro `options`:

```
var my_array:Array = new Array();
my_array.push({password: "Bob", age:29});
my_array.push({password: "abcd", age:3});
my_array.push({password: "barb", age:35});
my_array.push({password: "catchy", age:4});
```

La ordenación predeterminada por el campo password produce el siguiente resultado:

```
my_array.sortOn("password");
// Bob
// abcd
// barb
// catchy
```

La ordenación con distinción entre mayúsculas y minúsculas por el campo password produce el siguiente resultado:

```
my_array.sortOn("password", Array.CASEINSENSITIVE);
// abcd
// barb
// Bob
// catchy
```

La ordenación con distinción entre mayúsculas y minúsculas en orden descendente por el campo password produce el siguiente resultado:

```
my_array.sortOn("password", Array.CASEINSENSITIVE | Array.DESCENDING);
// catchy
// Bob
// barb
// abcd
```

La ordenación predeterminada por el campo age produce el siguiente resultado:

```
my_array.sortOn("age");
// 29
// 3
// 35
// 4
```

La ordenación numérica por el campo age produce el siguiente resultado:

```
my_array.sortOn("age", Array.NUMERIC);
// my_array[0].age = 3
```

```
// my_array[1].age = 4
// my_array[2].age = 29
// my_array[3].age = 35
```

La ordenación numérica descendente por el campo age produce el siguiente resultado:

```
my_array.sortOn("age", Array.DESCENDING | Array.NUMERIC);
// my_array[0].age = 35
// my_array[1].age = 29
// my_array[2].age = 4
// my_array[3].age = 3
```

Cuando utilice la opción de ordenación `Array.RETURNEDINDEXARRAY`, deberá asignar el valor devuelto a otra matriz. La matriz original no se modifica.

```
var indexArray:Array = my_array.sortOn("age", Array.RETURNINDEXEDARRAY);
```

Véase también

Operador OR en modo bit (|), sort (método Array.sort)

splice (método Array.splice)

```
public splice(startIndex:Number, [deleteCount:Number], [value:Object]) :
  Array
```

Añade y elimina elementos de una matriz. Este método modifica la matriz sin realizar ninguna copia de ella.

Disponibilidad: ActionScript 1.0; Flash Player 5

Parámetros

`startIndex``:Number` - Un entero que especifica el índice del elemento de la matriz donde comienza la inserción o eliminación. Puede especifica un entero negativo para establecer una posición relativa al final de la matriz (por ejemplo, -1 es el último elemento de la matriz).

`deleteCount``:Number` [opcional] - Un entero que especifica el número de elementos que se va a eliminar. Este número incluye el elemento especificado en el parámetro `startIndex`. Si no se ha especificado ningún valor para el parámetro `deleteCount`, el método elimina todos los valores desde el elemento `startIndex` hasta el último elemento de la matriz. Si el valor es 0, no se eliminará ningún elemento.

`value``:Object` [opcional] - Especifica el valor que se va a insertar en la matriz en el punto de inserción especificado en el parámetro `startIndex`.

Valor devuelto

`Array` - Una matriz que contiene los elementos eliminados de la matriz original.

Ejemplo

El ejemplo siguiente crea una matriz y la divide utilizando el índice de elemento 1 para el parámetro `startIndex`. De este modo se eliminan todos los elementos de la matriz a partir del segundo elemento, dejando únicamente el elemento de índice 0 en la matriz original:

```
var myPets_array:Array = new Array("cat", "dog", "bird", "fish");
trace( myPets_array.splice(1) ); // Displays dog,bird,fish.
trace( myPets_array ); // cat
```

El ejemplo siguiente crea una matriz y la divide utilizando el índice de elemento 1 para el parámetro `startIndex` y el número 2 para el parámetro `deleteCount`. De este modo se eliminan dos los elementos de la matriz a partir del segundo elemento, dejando el primero y el últimos elementos en la matriz original:

```
var myFlowers_array:Array = new Array("roses", "tulips", "lilies",
  "orchids");
trace( myFlowers_array.splice(1,2 ) ); // Displays tulips,lilies.
trace( myFlowers_array ); // roses,orchids
```

El ejemplo siguiente crea una matriz y la divide utilizando el índice de elemento 1 para el parámetro `startIndex`, el número 0 para el parámetro `deleteCount` y la cadena `chair` para el parámetro `value`. No se elimina nada de la matriz original y añade la cadena `chair` en el índice 1:

```
var myFurniture_array:Array = new Array("couch", "bed", "desk", "lamp");
trace( myFurniture_array.splice(1,0, "chair" ) ); // Displays empty array.
trace( myFurniture_array ); // displays couch,chair,bed,desk,lamp
```

toString (método Array.toString)

```
public toString() : String
```

Devuelve un valor de cadena que representa los elementos del objeto Array especificado. Cada elemento de la matriz, comenzando por el índice 0 y terminando por el índice más alto, se convierte en una cadena concatenada separada por comas. Para especificar un separador personalizado, utilice el método `Array.join()`.

Disponibilidad: ActionScript 1.0; Flash Player 5

Valor devuelto

`String` - Una cadena.

Ejemplo

El ejemplo siguiente crea `my_array` y la convierte en una cadena.

```
var my_array:Array = new Array();
my_array[0] = 1;
my_array[1] = 2;
my_array[2] = 3;
my_array[3] = 4;
my_array[4] = 5;
trace(my_array.toString()); // Displays 1,2,3,4,5.
```

Este ejemplo produce 1,2,3,4,5 como resultado de la sentencia `trace`.

Véase también

`split (método String.split)`, `join (método Array.join)`

UNIQUESORT (propiedad Array.UNIQUESORT)

```
public static UNIQUESORT : Number
```

En los métodos de ordenación, esta constante especifica requisito de ordenación exclusiva. Puede utilizar esta constante para el parámetro options del método `sort()` o `sortOn()`. La opción de ordenación exclusiva anula la ordenación si dos elementos o campos que se están ordenando tienen valores idénticos.

El valor de esta constante es 4.

Disponibilidad: ActionScript 1.0; Flash Player 7

Véase también

`sort (método Array.sort)`, `sortOn (método Array.sortOn)`

Clase Accessibility

unshift (método Array.unshift)

```
public unshift(value:Object) : Number
```

Añade uno o varios elementos al comienzo de una matriz y devuelve la nueva longitud de la matriz.

Disponibilidad: ActionScript 1.0; Flash Player 5

Parámetros

value:Object - Uno o más números, elementos o variables que se insertarán al principio de la matriz.

Valor devuelto

Number - Un entero que representa la nueva longitud de la matriz.

Ejemplo

El ejemplo siguiente muestra el uso del método Array.unshift():

```
var pets_array:Array = new Array("dog", "cat", "fish");
trace( pets_array ); // Displays dog,cat,fish.
pets_array.unshift("ferrets", "gophers", "engineers");
trace( pets_array ); // Displays ferrets,gophers,engineers,dog,cat,fish.
```

Véase también

pop (método Array.pop), push (método Array.push), shift (método Array.shift)

AsBroadcaster

```
Object
  |
  +-AsBroadcaster
```

public class *AsBroadcaster*
extends Object

Ofrece funciones de notificación de eventos y administración de detectores que se pueden añadir a objetos definidos por el usuario. Esta clase se ha diseñado para usuarios avanzados que desean crear mecanismos de gestión de eventos personalizados. Es posible utilizar esta clase para convertir a cualquier objeto en un difusor de eventos y para crear uno o varios objetos detectores que reciben notificación en el momento que el objeto difusor llame al método broadcastMessage().

No hay ninguna función constructora para la clase AsBroadcaster. Para utilizar esta clase, siga este proceso:

- Seleccione o cree un objeto para que actúe como difusor de eventos.
- Convierta el objeto en un difusor de eventos llamando al método estático AsBroadcaster.initialize(obj:Object), donde el parámetro obj es el nombre del objeto seleccionado como difusor.
- Seleccione o cree uno o varios objetos detectores. Los objetos detectores reciben notificación cada vez que el objeto difusor difunde un mensaje.

- Defina un método detector para cada uno de los objetos detectores. El método detector ejecutará código ActionScript en respuesta a la notificación de eventos. El nombre del método debe coincidir con el nombre del evento que difunde el objeto difusor.
- Registre cada objeto detector con el difusor de eventos llamando a `myBroadcaster.addListener(myListener)`, donde `myBroadcaster` es el nombre del objeto difusor de eventos y `myListener` es el nombre de objeto detector. Cada difusor de eventos almacena una lista de objetos detectores a los que se notificará cuando se difunda un mensaje. Utilice el `addListener()` para añadir detectores a la lista, y `removeListener()` para quitar detectores de la lista.
- Por último, para difundir un mensaje, llame al método `myBroadcaster.broadcastMessage(eventName:String)`, donde `myBroadcaster` es el nombre del difusor de eventos y `eventName` es el nombre del evento que coincide con el nombre del método detector.

Nota: Un error frecuente consiste en poner en mayúscula la segunda letra *de AsBroadcaster*. Cuando realice una llamada al método `AsBroadcaster.initialize()`, asegúrese de que la segunda letra está en minúscula. Si se escribe incorrectamente *AsBroadcaster* se producirá un error sin mensaje.

Disponibilidad: ActionScript 1.0; Flash Player 6

Resumen de propiedades

Modificadores	Propiedad	Descripción
	`_listeners:Array` [read-only]	Una lista de referencias a todos los objetos detectores registrados.

Propiedades heredadas de la clase Object

`constructor (propiedad Object.constructor), __proto__ (propiedad Object.__proto__), prototype (propiedad Object.prototype), __resolve (propiedad Object.__resolve)`

Resumen de métodos

Modificadores	Firma	Descripción
	`addListener(listenerObj:Object) : Boolean`	Registra un objeto para recibir mensajes de notificación de eventos.
	`broadcastMessage(eventName:String) : Void`	Envía un mensaje de evento a cada objeto de la lista de detectores.

Modificadores	Firma	Descripción
`static`	`initialize(obj:Object) : Void`	Añade funcionalidad de notificación de eventos y administración de detectores a un objeto determinado.
	`removeListener(listenerObj:Object) : Boolean`	Quita un objeto de la lista de objetos que reciben mensajes de notificación de eventos.

Métodos heredados de la clase Object

`addProperty (método Object.addProperty)`, `hasOwnProperty (método Object.hasOwnProperty)`, `isPropertyEnumerable (método Object.isPropertyEnumerable)`, `isPrototypeOf (método Object.isPrototypeOf)`, `registerClass (método Object.registerClass)`, `toString (método Object.toString)`, `unwatch (método Object.unwatch)`, `valueOf (método Object.valueOf)`, `watch (método Object.watch)`

addListener (método AsBroadcaster.addListener)

`public addListener(listenerObj:Object) : Boolean`

Registra un objeto para recibir mensajes de notificación de eventos. Este método se llama en el objeto difusor, y el objeto detector se envía como argumento.

Disponibilidad: ActionScript 1.0; Flash Player 6

Parámetros

`listenerObj``:Object` - El nombre del objeto detector que recibe notificación de eventos.

Valor devuelto

`Boolean` - Si bien este método devuelve técnicamente un valor booleano, en la práctica devuelve `Void` porque siempre devuelve el valor de `true`.

Ejemplo

El ejemplo siguiente es un extracto del ejemplo completo suministrado en la entrada del método `AsBroadcaster.initialize()`.

```
someObject.addListener(myListener1); // Register myListener1 as listener.
someObject.addListener(myListener2); // Register myListener2 as listener.
```

Véase también

`initialize (método AsBroadcaster.initialize)`, `removeListener (método AsBroadcaster.removeListener)`

broadcastMessage (método AsBroadcaster.broadcastMessage)

```
public broadcastMessage(eventName:String) : Void
```

Envía un mensaje de evento a cada objeto de la lista de detectores. Cuando el objeto detector recibe el mensaje, Flash Player intenta invocar una función del mismo nombre en el objeto detector. Supongamos que el objeto difunde un mensaje de evento como este:

```
obj.broadcastMessage("onAlert");
```

Cuando se recibe este mensaje, Flash Player invoca un método llamado `onAlert()` en el objeto detector que lo recibe.

Nota: Puede pasar argumentos a las funciones detectoras incluyendo argumentos adicionales en el método `broadcastMessage()`. El método detector recibirá como argumentos los argumentos que aparezcan después del parámetro `eventName`.

Sólo puede llamar a este método desde un objeto inicializado mediante el método `AsBroadcaster.initialize()`.

Disponibilidad: ActionScript 1.0; Flash Player 6

Parámetros

`eventName``:String` - El nombre del evento que se va a difundir. El nombre de los métodos detectores debe coincidir con este parámetro para recibir el evento de difusión. Puede pasar argumentos a los métodos detectores incluyendo argumentos adicionales después de `eventName`.

Ejemplo

El ejemplo siguiente es un extracto del ejemplo completo suministrado en la entrada del método `AsBroadcaster.initialize()`:

```
someObject.broadcastMessage("someEvent"); // Broadcast the "someEvent"
  message.
```

El ejemplo siguiente es un extracto del segundo ejemplo completo suministrado en la entrada del método `AsBroadcaster.initialize()`. Muestra cómo enviar argumentos a los métodos de detección.

```
someObject.broadcastMessage("someEvent", 3, "arbitrary string");
```

Véase también

`initialize (método AsBroadcaster.initialize)`, `removeListener (método AsBroadcaster.removeListener)`

initialize (método AsBroadcaster.initialize)

```
public static initialize(obj:Object) : Void
```

Añade funcionalidad de notificación de eventos y administración de detectores a un objeto determinado. Es un método estático; debe llamarse utilizando la clase AsBroadcaster (donde `someObject` es el nombre del objeto que se va a inicializar como difusor de eventos):

```
AsBroadcaster.initialize(someObject);
```

Nota: Un error frecuente consiste en poner en mayúscula la segunda letra *de AsBroadcaster*. Cuando realice una llamada al método `AsBroadcaster.initialize()`, asegúrese de que la segunda letra está en minúscula. Si se escribe incorrectamente *AsBroadcaster* se producirá un error sin mensaje.

Este método añade la propiedad `_listeners` junto con los tres métodos siguientes al objeto especificado por el parámetro `obj`:

- `obj.addListener()`
- `obj.removeListener()`
- `obj.broadcastMessage()`

Disponibilidad: ActionScript 1.0; Flash Player 6

Parámetros

`obj``:Object` - Un objeto que actuará como objeto difusor.

Ejemplo

El ejemplo siguiente crea un objeto genérico, `someObject`, y lo convierte en un difusor de eventos. El resultado deberá ser las cadenas mostradas en las dos sentencias `trace()`:

```
var someObject:Object = new Object(); // Creates broadcast object.

var myListener1:Object = new Object(); // Creates listener object.
var myListener2:Object = new Object(); // Creates listener object.

myListener1.someEvent = function() { // Creates listener method.
  trace("myListener1 received someEvent");
}
myListener2.someEvent = function() { // Createz listener method.
  trace("myListener2 received someEvent");
}

AsBroadcaster.initialize(someObject); // Makes someObject an event
  broadcaster.
someObject.addListener(myListener1); // Registers myListener1 as listener.
someObject.addListener(myListener2); // Registers myListener2 as listener.
```

```
someObject.broadcastMessage("someEvent"); // Broadcasts the "someEvent"
  message.
```

El ejemplo siguiente muestra cómo pasar argumentos adicionales a un método detector empleando el método `broadcastMessage()` El resultado deberá ser las tres cadenas mostradas en las sentencias `trace()`, que también incluyen los argumentos pasados a través del método `broadcastMessage()`.

```
var someObject:Object = new Object();

var myListener:Object = new Object();
myListener.someEvent = function(param1:Number, param2:String) {
  trace("myListener received someEvent");
  trace("param1: " + param1);
  trace("param2: " + param2);
}

AsBroadcaster.initialize(someObject);
someObject.addListener(myListener);
someObject.broadcastMessage("someEvent", 3, "arbitrary string");
```

_listeners (propiedad AsBroadcaster._listeners)

```
public _listeners : Array [read-only]
```

Una lista de referencias a todos los objetos detectores registrados. Esta propiedad se ha diseñado para uso interno, no para manipulación directa. Los objetos se añaden y quitan de esta matriz mediante llamadas a los métodos `addListener()` y `removelistener()`.

Sólo puede llamar a esta propiedad desde un objeto inicializado mediante el método `AsBroadcaster.initialize()`.

Disponibilidad: ActionScript 1.0; Flash Player 6

Ejemplo

El ejemplo siguiente muestra cómo utilizar la propiedad `length` para comprobar el número de objetos detectores registrados actualmente en un difusor de eventos. El código siguiente funciona si se añade al primer ejemplo completo de la sección Ejemplos de la entrada `AsBroadcaster.initialize()`.

```
trace(someObject._listeners.length); // Output: 2
```

Para usuarios avanzados, el ejemplo siguiente muestra cómo utilizar la propiedad `_listeners` para presentar todos los detectores registrados con un difusor de eventos, junto con todas las propiedades de cada objeto detector. El ejemplo siguiente crea dos métodos detectores distintos para el primer objeto detector.

```
var someObject:Object = new Object(); // create broadcast object
```

```
var myListener1:Object = new Object(); // create listener object
var myListener2:Object = new Object(); // create listener object

myListener1.someEvent = function() { // create listener method
  trace("myListener1 received someEvent");
}
myListener1.anotherEvent = function() { // create another listener method
  trace("myListener1 received anotherEvent");
}
myListener2.someEvent = function() { // create listener method
  trace("myListener2 received someEvent");
}

AsBroadcaster.initialize(someObject); // make someObject an event
  broadcaster
someObject.addListener(myListener1); // register myListener1 as listener
someObject.addListener(myListener2); // register myListener2 as listener

var numListeners:Number = someObject._listeners.length; // get number of
  registered listeners

// cycle through all listener objects, listing all properties of each
  listener object
for (var i:Number = 0; i < numListeners; i++) {
  trace("Listener " + i + " listens for these events:");
  for (item in someObject._listeners[i]) {
    trace (" " + item + ": " + someObject._listeners[i][item]);
  }
}
```

Véase también

`initialize (método AsBroadcaster.initialize)`

removeListener (método AsBroadcaster.removeListener)

```
public removeListener(listenerObj:Object) : Boolean
```

Quita un objeto de la lista de objetos que reciben mensajes de notificación de eventos.

Sólo puede llamar a este método desde un objeto inicializado mediante el método `AsBroadcaster.initialize()`.

Disponibilidad: ActionScript 1.0; Flash Player 6

Parámetros

listenerObj:Object - El nombre de un objeto detector que está registrado para recibir notificación de eventos desde el objeto difusor.

Valor devuelto

Boolean - Si el objeto detector se ha eliminado, devuelve true, en caso contrario devuelve false.

Ejemplo

El ejemplo siguiente muestra cómo quitar un detector de la lista de detectores registrados. El código siguiente funciona si se añade al primer ejemplo completo de la sección Ejemplos de la entrada AsBroadcaster.initialize(). Las sentencias trace() sólo se incluyen para verificar que el número de detectores registrados se reduce en uno después de llamar al método removeListener().

```
trace(someObject._listeners.length); // Output: 2
someObject.removeListener(myListener1);
trace(someObject._listeners.length); // Output: 1
```

Véase también

addListener (método AsBroadcaster.addListener), initialize (método AsBroadcaster.initialize)

BevelFilter (flash.filters.BevelFilter)

```
Object
  |
  +-flash.filters.BitmapFilter
     |
     +-flash.filters.BevelFilter
```

public class ***BevelFilter***
extends BitmapFilter

La clase BevelFilter permite añadir un efecto de biselado a diversos objetos en Flash. El efecto de biselado da aspecto tridimensional a objetos como los botones. Puede personalizar el aspecto del biselado con distintos colores de resaltado y sombreado, la cantidad de desenfoque del biselado, el ángulo del biselado, la ubicación del biselado y un efecto de extractor.

El uso de filtros depende del objeto al que se aplique el filtro:

- Para aplicar filtros a clips de película, campos de texto y botones en tiempo de ejecución, utilice la propiedad `filters`. El establecimiento de la propiedad `filters` de un objeto no modifica el objeto y se puede deshacer borrando la propiedad `filters`.
- Para aplicar filtros a instancias de BitmapData, utilice el método `BitmapData.applyFilter()`. La llamada a `applyFilter` en un objeto BitmapData toma el objeto BitmapData de origen y objeto de filtro y genera una imagen filtrada.

También puede aplicar efectos de filtro a imágenes y vídeo en tiempo de edición. Para más información, consulte la documentación de edición.

Si aplica un filtro a un clip de película o a un botón, la propiedad `cacheAsBitmap` del clip de película o del botón se establecerá como `true`. Si borra todos los filtros, se restaurará el valor original de `cacheAsBitmap`.

Este filtro admite escalado de escenario. Sin embargo, no admite escalado genera, rotación ni sesgo. Cuando se escala el propio objeto (si `_xscale` y `_yscale` no son 100%), no se escala el filtro. Sólo se escalará cuando se acerque el escenario.

El filtro no se aplica si la imagen resultante supera los 2880 píxeles de anchura o altura. Por ejemplo, si acerca un clip de película con un filtro aplicado, éste se desactiva si la imagen resultante supera el límite de 2880 píxeles.

Disponibilidad: ActionScript 1.0; Flash Player 8

Véase también

`filters (propiedad MovieClip.filters)`, `cacheAsBitmap (propiedad MovieClip.cacheAsBitmap)`, `filters (propiedad Button.filters)`, `cacheAsBitmap (propiedad Button.cacheAsBitmap)`, `filters (propiedad TextField.filters)`, `applyFilter (método BitmapData.applyFilter)`, `MovieClip`

Resumen de propiedades

Modificadores	Propiedad	Descripción
	`angle:Number`	El ángulo del bisel.
	`blurX:Number`	Cantidad de desenfoque horizontal, expresada en píxeles.
	`blurY:Number`	Cantidad de desenfoque vertical, expresada en píxeles.
	`distance:Number`	Distancia de desplazamiento del bisel.
	`highlightAlpha:Number`	Valor de transparencia alfa del color de resaltado.

Modificadores	Propiedad	Descripción
	highlightColor:Number	Color de resaltado del bisel.
	knockout:Boolean	Aplica un efecto de extractor (true), que hace que el relleno del objeto sea transparente y muestra el color de fondo del documento.
	quality:Number	Número de veces que debe aplicarse el filtro.
	shadowAlpha:Number	Valor de transparencia alfa del color de la sombra.
	shadowColor:Number	Color de sombra del bisel.
	strength:Number	Intensidad de la impresión o extensión.
	type:String	Tipo de bisel.

Propiedades heredadas de la clase Object

```
constructor (propiedad Object.constructor), __proto__ (propiedad
Object.__proto__), prototype (propiedad Object.prototype), __resolve
(propiedad Object.__resolve)
```

Resumen de constructores

Firma	Descripción
BevelFilter([distance:Number], [angle:Number], [highlightColor:Number], [highlightAlpha:Number], [shadowColor:Number], [shadowAlpha:Number], [blurX:Number], [blurY:Number], [strength:Number], [quality:Number], [type:String], [knockout:Boolean])	Inicializa una instancia BevelFilter nueva con los parámetros especificados.

Resumen de métodos

Modificadores	Firma	Descripción
	clone() : BevelFilter	Devuelve una copia de este objeto de filtro.

Métodos heredados de la clase BitmapFilter

```
clone (método BitmapFilter.clone)
```

addProperty (método Object.addProperty), hasOwnProperty (método Object.hasOwnProperty), isPropertyEnumerable (método Object.isPropertyEnumerable), isPrototypeOf (método Object.isPrototypeOf), registerClass (método Object.registerClass), toString (método Object.toString), unwatch (método Object.unwatch), valueOf (método Object.valueOf), watch (método Object.watch)

angle (propiedad BevelFilter.angle)

```
public angle : Number
```

El ángulo del bisel. Los valores válidos son de 0 a 360 grados. El valor predeterminado es 45.

El valor de angle representa el ángulo de la fuente de luz teórica que ilumina el objeto y determina la posición del efecto con respecto al objeto Si la distancia se define como 0, el efecto no se desplaza del objeto y el resultado es que la propiedad angle no se aplica.

Disponibilidad: ActionScript 1.0; Flash Player 8

Ejemplo

En el ejemplo siguiente se cambia la propiedad `angle` de una instancia MovieClip existente (`rect`) cuando un usuario hace clic en ella:

```
import flash.filters.BevelFilter;

var rect:MovieClip = createBevelRectangle("BevelDistance");
rect.onRelease = function() {
  var filter:BevelFilter = this.filters[0];
  filter.angle = 225;
  this.filters = new Array(filter);
}

function createBevelRectangle(name:String):MovieClip {
  var w:Number = 100;
  var h:Number = 100;
  var bgColor:Number = 0x00CC00;

  var rect:MovieClip = this.createEmptyMovieClip(name,
  this.getNextHighestDepth());
  rect.beginFill(bgColor);
  rect.lineTo(w, 0);
  rect.lineTo(w, h);
  rect.lineTo(0, h);
  rect.lineTo(0, 0);
  rect._x = 20;
  rect._y = 20;
```

```
  var filter:BevelFilter = new BevelFilter(5, 45, 0xFFFF00, .8, 0x0000FF,
  .8, 20, 20, 1, 3, "inner", false);
  rect.filters = new Array(filter);
  return rect;
}
```

Contructor BevelFilter

```
public BevelFilter([distance:Number], [angle:Number],
  [highlightColor:Number], [highlightAlpha:Number], [shadowColor:Number],
  [shadowAlpha:Number], [blurX:Number], [blurY:Number], [strength:Number],
  [quality:Number], [type:String], [knockout:Boolean])
```

Inicializa una instancia BevelFilter nueva con los parámetros especificados.

Disponibilidad: ActionScript 1.0; Flash Player 8

Parámetros

distance:Number [opcional] - Distancia de desplazamiento del bisel, en píxeles (coma flotante). El valor predeterminado es 4.

angle:Number [opcional] - Ángulo del bisel, de 0 a 360 grados. El valor predeterminado es 45.

highlightColor:Number [opcional] - Color de resaltado del bisel, 0x00RRGGBB. El valor predeterminado es 0xFFFFFF.

highlightAlpha:Number [opcional] - Valor de transparencia alfa del color de resaltado. Los valores válidos son de 0 a 1. Por ejemplo, 0,25 define un valor de transparencia de 25%. El valor predeterminado es 1.

shadowColor:Number [opcional] - Color de sombra del bisel, *0xRRGGBB*. El valor predeterminado es 0x000000.

shadowAlpha:Number [opcional] - Valor de transparencia alfa del color de sombra. Los valores válidos son de 0 a 1. Por ejemplo, 0,25 define un valor de transparencia de 25%. El valor predeterminado es 1.

blurX:Number [opcional] - Cantidad de desenfoque horizontal, expresada en píxeles. Los valores válidos son de 0 a 255 (coma flotante). El valor predeterminado es 4. Los valores que son múltiplos de 2 (como 2, 4, 8, 16 y 32) se optimizan para representarse más rápidamente que otros valores.

blurY:Number [opcional] - Cantidad de desenfoque vertical, expresada en píxeles. Los valores válidos van de 0 a 255 (coma flotante). El valor predeterminado es 4. Los valores que son múltiplos de 2 (como 2, 4, 8, 16 y 32) se optimizan para representarse más rápidamente que otros valores.

strength:Number [opcional] - Intensidad de la impresión o extensión. Cuanto más alto sea el valor, más color se imprimirá y mayor será el contraste entre el bisel y el fondo. Los valores válidos son de 0 a 255. El valor predeterminado es 1.

quality:Number [opcional] - Número de veces que debe aplicarse el filtro. El valor predeterminado es 1, que equivale a calidad baja. Un valor 2 equivale a calidad media y un valor 3 a alta.

type:String [opcional] - Tipo de bisel. Los valores válidos son "inner", "outer" y "full". El valor predeterminado es "inner".

knockout:Boolean [opcional] - Aplica un efecto de extractor (true), que hace que el relleno del objeto sea transparente y muestra el color de fondo del documento. El valor predeterminado es false (sin extractor).

Ejemplo

En el ejemplo siguiente se crea una instancia BevelFilter nueva y la aplica a la instancia MovieClip (rect):

```
import flash.filters.BevelFilter;

var distance:Number = 5;
var angleInDegrees:Number = 45;
var highlightColor:Number = 0xFFFF00;
var highlightAlpha:Number = .8;
var shadowColor:Number = 0x0000FF;
var shadowAlpha:Number = .8;
var blurX:Number = 5;
var blurY:Number = 5;
var strength:Number = 5;
var quality:Number = 3;
var type:String = "inner";
var knockout:Boolean = false;

var filter:BevelFilter = new BevelFilter(distance,
                         angleInDegrees,
                         highlightColor,
                         highlightAlpha,
                         shadowColor,
                         shadowAlpha,
                         blurX,
                         blurY,
                         strength,
                         quality,
                         type,
                         knockout);
```

```
var rect:MovieClip = createRectangle(100, 100, 0x00CC00,
  "bevelFilterExample");
rect.filters = new Array(filter);

function createRectangle(w:Number, h:Number, bgColor:Number,
  name:String):MovieClip {
  var rect:MovieClip = this.createEmptyMovieClip(name,
  this.getNextHighestDepth());
  rect.beginFill(bgColor);
  rect.lineTo(w, 0);
  rect.lineTo(w, h);
  rect.lineTo(0, h);
  rect.lineTo(0, 0);
  rect._x = 20;
  rect._y = 20;
  return rect;
}
```

blurX (propiedad BevelFilter.blurX)

```
public blurX : Number
```

Cantidad de desenfoque horizontal, expresada en píxeles. Los valores válidos son de 0 a 255 (coma flotante). El valor predeterminado es 4. Los valores que son múltiplos de 2 (como 2, 4, 8, 16 y 32) se optimizan para representarse más rápidamente que otros valores.

Disponibilidad: ActionScript 1.0; Flash Player 8

Ejemplo

En el ejemplo siguiente se cambia la propiedad `blurX` de la instancia MovieClip existente (`rect`) cuando un usuario hace clic en ella:

```
import flash.filters.BevelFilter;

var rect:MovieClip = createBevelRectangle("BevelBlurX");
rect.onRelease = function() {
  var filter:BevelFilter = this.filters[0];
  filter.blurX = 10;
  this.filters = new Array(filter);
}

function createBevelRectangle(name:String):MovieClip {
  var w:Number = 100;
  var h:Number = 100;
  var bgColor:Number = 0x00CC00;

  var rect:MovieClip = this.createEmptyMovieClip(name,
  this.getNextHighestDepth());
  rect.beginFill(bgColor);
```

```
  rect.lineTo(w, 0);
  rect.lineTo(w, h);
  rect.lineTo(0, h);
  rect.lineTo(0, 0);
  rect._x = 20;
  rect._y = 20;

  var filter:BevelFilter = new BevelFilter(5, 45, 0xFFFF00, .8, 0x0000FF,
  .8, 20, 20, 1, 3, "inner", false);
  rect.filters = new Array(filter);
  return rect;
}
```

blurY (propiedad BevelFilter.blurY)

```
public blurY : Number
```

Cantidad de desenfoque vertical, expresada en píxeles. Los valores válidos son de 0 a 255 (coma flotante). El valor predeterminado es 4. Los valores que son múltiplos de 2 (como 2, 4, 8, 16 y 32) se optimizan para representarse más rápidamente que otros valores.

Disponibilidad: ActionScript 1.0; Flash Player 8

Ejemplo

En el ejemplo siguiente se cambia la propiedad `blurY` de la instancia MovieClip existente (`rect`) cuando un usuario hace clic en ella:

```
import flash.filters.BevelFilter;

var rect:MovieClip = createBevelRectangle("BevelBlurY");
rect.onRelease = function() {
  var filter:BevelFilter = this.filters[0];
  filter.blurY = 10;
  this.filters = new Array(filter);
}

function createBevelRectangle(name:String):MovieClip {
  var w:Number = 100;
  var h:Number = 100;
  var bgColor:Number = 0x00CC00;

  var rect:MovieClip = this.createEmptyMovieClip(name,
  this.getNextHighestDepth());
  rect.beginFill(bgColor);
  rect.lineTo(w, 0);
  rect.lineTo(w, h);
  rect.lineTo(0, h);
  rect.lineTo(0, 0);
  rect._x = 20;
```

```
  rect._y = 20;

  var filter:BevelFilter = new BevelFilter(5, 45, 0xFFFF00, .8, 0x0000FF,
  .8, 20, 20, 1, 3, "inner", false);
  rect.filters = new Array(filter);
  return rect;
}
```

clone (método BevelFilter.clone)

```
public clone() : BevelFilter
```

Devuelve una copia de este objeto de filtro.

Disponibilidad: ActionScript 1.0; Flash Player 8

Valor devuelto

`flash.filters.BevelFilter` - Instancia BevelFilter nueva con las mismas propiedades que la instancia BevelFilter original.

Ejemplo

En el ejemplo siguiente se crean y se comparan tres objetos BevelFilter. Puede crear el objeto `filter_1` utilizando el constructor BevelFilter. Para crear el objeto `filter_2`, defínalo como igual que `filter_1`. Para crear `clonedFilter`, copie `filter_1`. Observe que mientras que `filter_2` se evalúa como igual que `filter_1`, no ocurre igual con `clonedFilter`, aunque contiene el mismo valor que `filter_1`.

```
import flash.filters.BevelFilter;

var filter_1:BevelFilter = new BevelFilter(5, 45, 0xFFFF00, .8, 0x0000FF,
  .8, 20, 20, 1, 3, "inner", false);
var filter_2:BevelFilter = filter_1;
var clonedFilter:BevelFilter = filter_1.clone();

trace(filter_1 == filter_2); // true
trace(filter_1 == clonedFilter); // false

for(var i in filter_1) {
  trace(">> " + i + ": " + filter_1[i]);
  // >> clone: [type Function]
  // >> type: inner
  // >> blurY: 20
  // >> blurX: 20
  // >> knockout: false
  // >> strength: 1
  // >> quality: 3
    // >> shadowAlpha: 0.8
```

```
  // >> shadowColor: 255
  // >> highlightAlpha: 0.8
  // >> highlightColor: 16776960
  // >> angle: 45
  // >> distance: 5
}

for(var i in clonedFilter) {
  trace(">> " + i + ": " + clonedFilter[i]);
  // >> clone: [type Function]
  // >> type: inner
  // >> blurY: 20
  // >> blurX: 20
  // >> knockout: false
  // >> strength: 1
  // >> quality: 3
  // >> shadowAlpha: 0.8
  // >> shadowColor: 255
  // >> highlightAlpha: 0.8
  // >> highlightColor: 16776960
  // >> angle: 45
  // >> distance: 5
}
```

Para demostrar mejor la relación entre `filter_1`, `filter_2` y `clonedFilter`, en el ejemplo siguiente se modifica la propiedad `knockout` de `filter_1`. La modificación de `knockout` demuestra que el método `clone()` crea una instanacia basada en valores de `filter_1` en lugar de hacer referencia a los valores.

```
import flash.filters.BevelFilter;

var filter_1:BevelFilter = new BevelFilter(5, 45, 0xFFFF00, .8, 0x0000FF,
  .8, 20, 20, 1, 3, "inner", false);
var filter_2:BevelFilter = filter_1;
var clonedFilter:BevelFilter = filter_1.clone();

trace(filter_1.knockout); // false
trace(filter_2.knockout); // false
trace(clonedFilter.knockout); // false

filter_1.knockout = true;

trace(filter_1.knockout); // true
trace(filter_2.knockout); // true
trace(clonedFilter.knockout); // false
```

distance (propiedad BevelFilter.distance)

```
public distance : Number
```

Distancia de desplazamiento del bisel. Los valores válidos son en píxeles (coma flotante). El valor predeterminado es 4.

Disponibilidad: ActionScript 1.0; Flash Player 8

Ejemplo

En el ejemplo siguiente se cambia la propiedad `distance` de la instancia MovieClip existente (`rect`) cuando un usuario hace clic en ella:

```
import flash.filters.BevelFilter;

var rect:MovieClip = createBevelRectangle("BevelDistance");
rect.onRelease = function() {
  var filter:BevelFilter = this.filters[0];
  filter.distance = 3;
  this.filters = new Array(filter);
}

function createBevelRectangle(name:String):MovieClip {
  var w:Number = 100;
  var h:Number = 100;
  var bgColor:Number = 0x00CC00;

  var rect:MovieClip = this.createEmptyMovieClip(name,
  this.getNextHighestDepth());
  rect.beginFill(bgColor);
  rect.lineTo(w, 0);
  rect.lineTo(w, h);
  rect.lineTo(0, h);
  rect.lineTo(0, 0);
  rect._x = 20;
  rect._y = 20;

  var filter:BevelFilter = new BevelFilter(5, 45, 0xFFFF00, .8, 0x0000FF,
  .8, 20, 20, 1, 3, "inner", false);
  rect.filters = new Array(filter);
  return rect;
}
```

highlightAlpha (propiedad BevelFilter.highlightAlpha)

```
public highlightAlpha : Number
```

Valor de transparencia alfa del color de resaltado. El valor se especifica como normalizado de 0 a 1. Por ejemplo, 0,25 establece un valor de transparencia del 25%. El valor predeterminado es 1.

Disponibilidad: ActionScript 1.0; Flash Player 8

Ejemplo

En el ejemplo siguiente se cambia la propiedad `highlightAlpha` de la instancia MovieClip existente (`rect`) cuando un usuario hace clic en ella:

```
import flash.filters.BevelFilter;

var rect:MovieClip = createBevelRectangle("BevelHighlightAlpha");
rect.onRelease = function() {
  var filter:BevelFilter = this.filters[0];
  filter.highlightAlpha = .2;
  this.filters = new Array(filter);
}

function createBevelRectangle(name:String):MovieClip {
  var w:Number = 100;
  var h:Number = 100;
  var bgColor:Number = 0x00CC00;

  var rect:MovieClip = this.createEmptyMovieClip(name,
  this.getNextHighestDepth());
  rect.beginFill(bgColor);
  rect.lineTo(w, 0);
  rect.lineTo(w, h);
  rect.lineTo(0, h);
  rect.lineTo(0, 0);
  rect._x = 20;
  rect._y = 20;

  var filter:BevelFilter = new BevelFilter(5, 45, 0xFFFF00, .8, 0x0000FF,
  .8, 20, 20, 1, 3, "inner", false);
  rect.filters = new Array(filter);
  return rect;
}
```

highlightColor (propiedad BevelFilter.highlightColor)

```
public highlightColor : Number
```

Color de resaltado del bisel. Los valores válidos tienen formato hexadecimal, *0xRRGGBB*. El valor predeterminado es 0xFFFFFF.

Disponibilidad: ActionScript 1.0; Flash Player 8

Ejemplo

En el ejemplo siguiente se cambia la propiedad `highlightColor` de la instancia MovieClip existente (`rect`) cuando un usuario hace clic en ella:

```
import flash.filters.BevelFilter;

var rect:MovieClip = createBevelRectangle("BevelHighlightColor");
rect.onRelease = function() {
  var filter:BevelFilter = this.filters[0];
  filter.highlightColor = 0x0000FF;
  this.filters = new Array(filter);
}

function createBevelRectangle(name:String):MovieClip {
  var w:Number = 100;
  var h:Number = 100;
  var bgColor:Number = 0x00CC00;

  var rect:MovieClip = this.createEmptyMovieClip(name,
  this.getNextHighestDepth());
  rect.beginFill(bgColor);
  rect.lineTo(w, 0);
  rect.lineTo(w, h);
  rect.lineTo(0, h);
  rect.lineTo(0, 0);
  rect._x = 20;
  rect._y = 20;

  var filter:BevelFilter = new BevelFilter(5, 45, 0xFFFF00, .8, 0x0000FF,
  .8, 20, 20, 1, 3, "inner", false);
  rect.filters = new Array(filter);
  return rect;
}
```

knockout (propiedad BevelFilter.knockout)

```
public knockout : Boolean
```

Aplica un efecto de extractor (`true`), que hace que el relleno del objeto sea transparente y muestra el color de fondo del documento. El valor predeterminado es `false` (sin extractor).

Disponibilidad: ActionScript 1.0; Flash Player 8

Ejemplo

En el ejemplo siguiente se cambia la propiedad `knockout` de la instancia MovieClip existente (`rect`) cuando un usuario hace clic en ella:

```
import flash.filters.BevelFilter;

var rect:MovieClip = createBevelRectangle("BevelKnockout");
rect.onRelease = function() {
  var filter:BevelFilter = this.filters[0];
  filter.knockout = true;
  this.filters = new Array(filter);
}

function createBevelRectangle(name:String):MovieClip {
  var w:Number = 100;
  var h:Number = 100;
  var bgColor:Number = 0x00CC00;

  var rect:MovieClip = this.createEmptyMovieClip(name,
  this.getNextHighestDepth());
  rect.beginFill(bgColor);
  rect.lineTo(w, 0);
  rect.lineTo(w, h);
  rect.lineTo(0, h);
  rect.lineTo(0, 0);
  rect._x = 20;
  rect._y = 20;

  var filter:BevelFilter = new BevelFilter(5, 45, 0xFFFF00, .8, 0x0000FF,
  .8, 20, 20, 1, 3, "inner", false);
  rect.filters = new Array(filter);
  return rect;
}
```

quality (propiedad BevelFilter.quality)

```
public quality : Number
```

Número de veces que debe aplicarse el filtro. El valor predeterminado es 1, que equivale a calidad baja (low). Un valor 2 equivale a calidad media y un valor 3 a alta. Los filtros con valores más bajos se representan con mayor rapidez.

Para la mayoría de las aplicaciones, es suficiente un valor `quality` de 1, 2 o 3. Aunque puede utilizar valores numéricos hasta 15 para conseguir efectos distintos, los valores más altos se representan más lentamente. En lugar de aumentar el valor de `quality`, habitualmente se consigue un efecto similar, y con una representación más rápida, sencillamente aumentando los valores de `blurX` y `blurY`.

Disponibilidad: ActionScript 1.0; Flash Player 8

Ejemplo

En el ejemplo siguiente se cambia la propiedad `quality` de la instancia MovieClip existente (`rect`) cuando un usuario hace clic en ella:

```
import flash.filters.BevelFilter;

var rect:MovieClip = createBevelRectangle("BevelQuality");
rect.onRelease = function() {
  var filter:BevelFilter = this.filters[0];
  filter.quality = 1;
  this.filters = new Array(filter);
}

function createBevelRectangle(name:String):MovieClip {
  var w:Number = 100;
  var h:Number = 100;
  var bgColor:Number = 0x00CC00;

  var rect:MovieClip = this.createEmptyMovieClip(name,
  this.getNextHighestDepth());
  rect.beginFill(bgColor);
  rect.lineTo(w, 0);
  rect.lineTo(w, h);
  rect.lineTo(0, h);
  rect.lineTo(0, 0);
  rect._x = 20;
  rect._y = 20;

  var filter:BevelFilter = new BevelFilter(5, 45, 0xFFFF00, .8, 0x0000FF,
  .8, 20, 20, 1, 3, "inner", false);
  rect.filters = new Array(filter);
  return rect;
}
```

shadowAlpha (propiedad BevelFilter.shadowAlpha)

```
public shadowAlpha : Number
```

Valor de transparencia alfa del color de la sombra. El valor se especifica como normalizado de 0 a 1. Por ejemplo, 0,25 establece un valor de transparencia del 25%. El valor predeterminado es 1.

Disponibilidad: ActionScript 1.0; Flash Player 8

Ejemplo

En el ejemplo siguiente se cambia la propiedad `shadowAlpha` de la instancia MovieClip existente (`rect`) cuando un usuario hace clic en ella:

```
import flash.filters.BevelFilter;

var rect:MovieClip = createBevelRectangle("BevelShadowAlpha");
rect.onRelease = function() {
  var filter:BevelFilter = this.filters[0];
  filter.shadowAlpha = .2;
  this.filters = new Array(filter);
}

function createBevelRectangle(name:String):MovieClip {
  var w:Number = 100;
  var h:Number = 100;
  var bgColor:Number = 0x00CC00;

  var rect:MovieClip = this.createEmptyMovieClip(name,
  this.getNextHighestDepth());
  rect.beginFill(bgColor);
  rect.lineTo(w, 0);
  rect.lineTo(w, h);
  rect.lineTo(0, h);
  rect.lineTo(0, 0);
  rect._x = 20;
  rect._y = 20;

  var filter:BevelFilter = new BevelFilter(5, 45, 0xFFFF00, .8, 0x0000FF,
  .8, 20, 20, 1, 3, "inner", false);
  rect.filters = new Array(filter);
  return rect;
}
```

shadowColor (propiedad BevelFilter.shadowColor)

```
public shadowColor : Number
```

Color de sombra del bisel. Los valores válidos tienen formato hexadecimal, *0xRRGGBB*. El valor predeterminado es 0x000000.

Disponibilidad: ActionScript 1.0; Flash Player 8

Ejemplo

En el ejemplo siguiente se cambia la propiedad `shadowColor` de la instancia MovieClip existente (`rect`) cuando un usuario hace clic en ella:

```
import flash.filters.BevelFilter;

var rect:MovieClip = createBevelRectangle("BevelShadowColor");
rect.onRelease = function() {
  var filter:BevelFilter = this.filters[0];
  filter.shadowColor = 0xFFFF00;
  this.filters = new Array(filter);
}

function createBevelRectangle(name:String):MovieClip {
  var w:Number = 100;
  var h:Number = 100;
  var bgColor:Number = 0x00CC00;

  var rect:MovieClip = this.createEmptyMovieClip(name,
  this.getNextHighestDepth());
  rect.beginFill(bgColor);
  rect.lineTo(w, 0);
  rect.lineTo(w, h);
  rect.lineTo(0, h);
  rect.lineTo(0, 0);
  rect._x = 20;
  rect._y = 20;

  var filter:BevelFilter = new BevelFilter(5, 45, 0xFFFF00, .8, 0x0000FF,
  .8, 20, 20, 1, 3, "inner", false);
  rect.filters = new Array(filter);
  return rect;
}
```

strength (propiedad BevelFilter.strength)

```
public strength : Number
```

Intensidad de la impresión o extensión. Los valores válidos son de 0 a 255. Cuanto más alto sea el valor, más color se imprimirá y mayor será el contraste entre el bisel y el fondo. El valor predeterminado es 1.

Disponibilidad: ActionScript 1.0; Flash Player 8

Ejemplo

En el ejemplo siguiente se cambia la propiedad `strength` de la instancia MovieClip existente (`rect`) cuando un usuario hace clic en ella:

```
import flash.filters.BevelFilter;

var rect:MovieClip = createBevelRectangle("BevelStrength");
rect.onRelease = function() {
  var filter:BevelFilter = this.filters[0];
  filter.strength = 10;
  this.filters = new Array(filter);
}

function createBevelRectangle(name:String):MovieClip {
  var w:Number = 100;
  var h:Number = 100;
  var bgColor:Number = 0x00CC00;

  var rect:MovieClip = this.createEmptyMovieClip(name,
  this.getNextHighestDepth());
  rect.beginFill(bgColor);
  rect.lineTo(w, 0);
  rect.lineTo(w, h);
  rect.lineTo(0, h);
  rect.lineTo(0, 0);
  rect._x = 20;
  rect._y = 20;

  var filter:BevelFilter = new BevelFilter(5, 45, 0xFFFF00, .8, 0x0000FF,
  .8, 20, 20, 1, 3, "inner", false);
  rect.filters = new Array(filter);
  return rect;
}
```

type (propiedad BevelFilter.type)

```
public type : String
```

Tipo de bisel. Los valores válidos son `"inner"`, `"outer"` y `"full"`.

Disponibilidad: ActionScript 1.0; Flash Player 8

Ejemplo

En el ejemplo siguiente se cambia la propiedad `type` de la instancia MovieClip existente (`rect`) cuando un usuario hace clic en ella:

```
import flash.filters.BevelFilter;

var rect:MovieClip = createBevelRectangle("BevelType");
rect.onRelease = function() {
  var filter:BevelFilter = this.filters[0];
  filter.type = "outer";
  this.filters = new Array(filter);
}

function createBevelRectangle(name:String):MovieClip {
  var w:Number = 100;
  var h:Number = 100;
  var bgColor:Number = 0x00CC00;

  var rect:MovieClip = this.createEmptyMovieClip(name,
  this.getNextHighestDepth());
  rect.beginFill(bgColor);
  rect.lineTo(w, 0);
  rect.lineTo(w, h);
  rect.lineTo(0, h);
  rect.lineTo(0, 0);
  rect._x = 20;
  rect._y = 20;

  var filter:BevelFilter = new BevelFilter(5, 45, 0xFFFF00, .8, 0x0000FF,
  .8, 20, 20, 1, 3, "inner", false);
  rect.filters = new Array(filter);
  return rect;
}
```

BitmapData (flash.display.BitmapData)

```
Object
  |
  +-flash.display.BitmapData
```

```
public class BitmapData
extends Object
```

La clase BitmapData permite crear imágenes de mapa de bits transparentes u opacas de tamaño arbitrario y manipularlas de distintas formas en tiempo de ejecución.

Esta clase permite separar operaciones de representación de mapa de bits de la rutinas internas de actualización de la visualización de Flash Player. Manipulando un objeto BitmapData directamente es posible crear imágenes muy complejas sin recargar cada fotograma al redibujar constantemente el contenido de datos vectoriales.

Los métodos de la clase BitmapData admiten diversos efectos que no están disponibles a través de la interfaz genérica de filtros.

Un objeto BitmapData contiene una matriz de datos de píxel. Estos datos pueden representar un mapa de bits completamente opaco o un mapa de bits transparente que contiene datos del canal alfa. Ambos tipos de objetos BitmapData se almacenan como búfer de enteros de 32 bits. Cada entero de 32 bits determina las propiedades de un píxel único del mapa de bits.

Cada entero de 32 bits es una combinación de cuatro valores de canal de 8 bits (de cero a 255) que describen los valores de transparencia alfa y rojo, verde y azul (ARGB) del píxel.

Los cuatro canales (rojo, verde, azul y alfa) se representan como números cuando se utilizan con el método `BitmapData.copyChannel()` o las propiedades `DisplacementMapFilter.componentX` y `DisplacementMapFilter.componentY`, como se indica a continuación:

- `1` (rojo)
- `2` (verde)
- `4` (azul)
- `8` (alfa)

Los objetos BitmapData se pueden asociar a un objeto MovieClip mediante el método `MovieClip.attachBitmap()`.

Puede utilizar un objeto BitmapData para rellenar un área de un clip de película utilizando el método `MovieClip.beginBitmapFill()`.

Un objeto BitmapData puede tener una altura y una anchura máximas de 2.880 píxeles.

Disponibilidad: ActionScript 1.0; Flash Player 8

Véase también

attachBitmap (método MovieClip.attachBitmap), beginBitmapFill (método MovieClip.beginBitmapFill)

Resumen de propiedades

Modificadores	Propiedad	Descripción
	height:Number [read-only]	Altura de la imagen de mapa de bits, expresada en píxeles.
	rectangle:Rectangle [read-only]	Rectángulo que define el tamaño y la ubicación de la imagen de mapa de bits.
	transparent:Boolean [read-only]	Define si la imagen de mapa de bits admite transparencia por píxel.
	width:Number [read-only]	Anchura de la imagen de mapa de bits, expresada en píxeles.

Propiedades heredadas de la clase Object

constructor (propiedad Object.constructor), __proto__ (propiedad Object.__proto__), prototype (propiedad Object.prototype), __resolve (propiedad Object.__resolve)

Resumen de constructores

Firma	Descripción
BitmapData(width:Number, height:Number, [transparent:Boolean], [fillColor:Number])	Crea un nuevo objeto BitmapData con una determinada anchura y altura.

Resumen de métodos

Modificadores	Firma	Descripción
	applyFilter(sourceBitmap:BitmapData, sourceRect:Rectangle, destPoint:Point, filter:BitmapFilter) : Number	Toma una imagen de origen y un objeto de filtro y genera la imagen filtrada.
	clone() : BitmapData	Devuelve un objeto BitmapData nuevo que es un clon de la instancia original con una copia exacta del mapa de bits contenido.

Modificadores	Firma	Descripción
	`colorTransform(rect:Rectangle, colorTransform:ColorTransform) : Void`	Ajusta los valores de color en una determinada área de una imagen de mapa de bits mediante un objeto ColorTransform.
	`copyChannel(sourceBitmap:BitmapData, sourceRect:Rectangle, destPoint:Point, sourceChannel:Number, destChannel:Number) : Void`	Transfiere datos de un canal de otro objeto BitmapData o del objeto BitmapData actual a un canal del objeto BitmapData actual.
	`copyPixels(sourceBitmap:BitmapData, sourceRect:Rectangle, destPoint:Point, [alphaBitmap:BitmapData], [alphaPoint:Point], [mergeAlpha:Boolean]) : Void`	Proporciona una rutina rápida para la manipulación de píxeles entre imágenes sin efectos de color, rotación ni expansión.
	`dispose() : Void`	Libera memoria que se utiliza para almacenar el objeto BitmapData.
	`draw(source:Object, [matrix:Matrix], [colorTransform:ColorTransform], [blendMode:Object], [clipRect:Rectangle], [smooth:Boolean]) : Void`	Dibuja una imagen de origen o un clip de película en una imagen de destino, mediante el procesador de vectores de Flash Player.
	`fillRect(rect:Rectangle, color:Number) : Void`	Rellena un área rectangular de píxeles con un determinado color ARGB.
	`floodFill(x:Number, y:Number, color:Number) : Void`	Realiza una operación de relleno en una imagen empezando por una coordenada (*x*, *y*) y rellenando con un determinado color.

Modificadores	Firma	Descripción
	`generateFilterRect(sourceRect:Rectangle, filter:BitmapFilter) : Rectangle`	Determina el rectángulo de destino al que afecta la llamada `applyFilter()`, dados un objeto BitmapData, un rectángulo de origen y un objeto de filtro.
	`getColorBoundsRect(mask:Number, color:Number, [findColor:Boolean]) : Rectangle`	Determina una región rectangular que encierra completamente todos los píxeles de un determinado color en la imagen de mapa de bits.
	`getPixel(x:Number, y:Number) : Number`	Devuelve un entero que representa un valor de píxel RGB de un objeto BitmapData en un punto específico (*x*, *y*).
	`getPixel32(x:Number, y:Number) : Number`	Devuelve un valor de color ARGB que contiene datos del canal alfa y datos de RGB.
	`hitTest(firstPoint:Point, firstAlphaThreshold:Number, secondObject:Object, [secondBitmapPoint:Point], [secondAlphaThreshold:Number]) : Boolean`	Detecta la zona activa a nivel de píxeles entre una imagen de mapa de bits y un punto, un rectángulo u otra imagen de mapa de bits.
`static`	`loadBitmap(id:String) : BitmapData`	Devuelve un nuevo objeto BitmapData que contiene una representación de imagen de mapa de bits del símbolo que se identifica mediante un ID de vínculo especificado en la biblioteca.
	`merge(sourceBitmap:BitmapData, sourceRect:Rectangle, destPoint:Point, redMult:Number, greenMult:Number, blueMult:Number, alphaMult:Number) : Void`	Realiza una mezcla por canal de una imagen de origen a una imagen de destino.

Modificadores	Firma	Descripción
	`noise(randomSeed:Number, [low:Number], [high:Number], [channelOptions:Number], [grayScale:Boolean]) : Void`	Rellena una imagen con píxeles que representan ruido aleatorio.
	`paletteMap(sourceBitmap:BitmapData, sourceRect:Rectangle, destPoint:Point, [redArray:Array], [greenArray:Array], [blueArray:Array], [alphaArray:Array]) : Void`	Reasigna los valores del canal de color en una imagen proporcionada a cuatro conjuntos de datos de la paleta de colores, una por cada canal.
	`perlinNoise(baseX:Number, baseY:Number, numOctaves:Number, randomSeed:Number, stitch:Boolean, fractalNoise:Boolean, [channelOptions:Number], [grayScale:Boolean], [offsets:Object]) : Void`	Genera una imagen de ruido Perlin.
	`pixelDissolve(sourceBitmap:BitmapData, sourceRect:Rectangle, destPoint:Point, [randomSeed:Number], [numberOfPixels:Number], [fillColor:Number]) : Number`	Realiza una disolución de píxeles de una imagen de origen a una imagen de destino o utilizando la misma imagen.
	`scroll(x:Number, y:Number) : Void`	Desplaza una imagen una determinada cantidad de píxeles (*x*, *y*).

Modificadores	Firma	Descripción
	`setPixel(x:Number, y:Number, color:Number) : Void`	Establece el color de un solo píxel de un objeto BitmapData.
	`setPixel32(x:Number, y:Number, color:Number) : Void`	Establece los valores de color y transparencia alfa de un solo píxel de un objeto BitmapData.
	`threshold(sourceBitmap:BitmapData, sourceRect:Rectangle, destPoint:Point, operation:String, threshold:Number, [color:Number], [mask:Number], [copySource:Boolean]) : Number`	Prueba valores de píxel en una imagen con un umbral especificado y define los píxeles que se pasan para probar los valores del nuevo color.

Métodos heredados de la clase Object

`addProperty (método Object.addProperty)`, `hasOwnProperty (método Object.hasOwnProperty)`, `isPropertyEnumerable (método Object.isPropertyEnumerable)`, `isPrototypeOf (método Object.isPrototypeOf)`, `registerClass (método Object.registerClass)`, `toString (método Object.toString)`, `unwatch (método Object.unwatch)`, `valueOf (método Object.valueOf)`, `watch (método Object.watch)`

applyFilter (método BitmapData.applyFilter)

```
public applyFilter(sourceBitmap:BitmapData, sourceRect:Rectangle,
  destPoint:Point, filter:BitmapFilter) : Number
```

Toma una imagen de origen y un objeto de filtro y genera la imagen filtrada.

Este método se basa en el comportamiento de los objetos de filtro incorporado, que tienen código para determinar el rectángulo de destino que se ve afectado por un rectángulo de origen de entrada.

Después de aplicar un filtro, la imagen resultante puede ser mayor que la imagen de entrada. Por ejemplo, si se utiliza una clase BlurFilter para desenfocar un rectángulo de origen de entrada (50,50,100,100) y un punto de destino de (10,10), el área que cambia en la imagen de destino será mayor que (10,10,60,60) debido al desenfoque. Esto sucede internamente durante la llamada a `applyFilter()`.

Si el parámetro `sourceRect` del parámetro `sourceBitmapData` es una región interior, como (50,50,100,100) en una imagen de 200 x 200, el filtro utilizará los píxeles de origen fuera del parámetro `sourceRect` para generar el rectángulo de destino.

Disponibilidad: ActionScript 1.0; Flash Player 8

Parámetros

`sourceBitmap``:flash.display.BitmapData` - La imagen de mapa de bits de entrada que se va a utilizar. La imagen de origen puede ser otro objeto BitmapData o puede hacer referencia a la instancia BitmapData actual.

`sourceRect``:flash.geom.Rectangle` - Un rectángulo que define el área de la imagen de origen que se va a utilizar como entrada.

`destPoint``:flash.geom.Point` - El punto de la imagen de destino (la instancia BitmapData actual) que corresponde a la esquina superior izquierda del rectángulo de origen.

`filter``:flash.filters.BitmapFilter` - El objeto de filtro que se utilizará para realizar la operación de filtrado. Cada tipo de filtro tiene determinados requisitos, a saber:

- *BlurFilter* - Este filtro puede utilizar imágenes de origen y destino que son opacas o transparentes. Si no coinciden los formatos de las imágenes, la copia de la imagen de origen que se realiza durante el filtrado coincide con el formato de la imagen de destino.
- *BevelFilter, DropShadowFilter, GlowFilter* - La imagen de destino de estos filtros debe ser una imagen transparente. Al llamar a DropShadowFilter o GlowFilter se crea una imagen que contiene los datos del canal alfa de la sombra o del iluminado. No se crea la sombra sobre la imagen de destino. Si se utiliza alguno de estos filtros con una imagen de destino opaco, se devuelve un valor de código de error de -6.
- *ConvolutionFilter* - Este filtro puede utilizar imágenes de origen y destino que son opacas o transparentes.
- *ColorMatrixFilter* - Este filtro puede utilizar imágenes de origen y destino que son opacas o transparentes.
- *DisplacementMapFilter* - Este filtro puede utilizar imágenes de origen y destino que son opacas o transparentes, pero el formato de las imágenes de origen y destino debe ser el mismo.

Valor devuelto

`Number` - Un número que indica si el filtro se ha aplicado correctamente. Si se devuelve cero, el filtro se ha aplicado correctamente. Si se devuelve un número negativo, se ha producido un error durante la aplicación del filtro.

Ejemplo

En el ejemplo siguiente se muestra cómo aplicar un filtro de bisel a una instancia BitmapData:

```
import flash.display.BitmapData;
import flash.filters.BevelFilter;
import flash.geom.Point;

var myBitmapData:BitmapData = new BitmapData(100, 80, true, 0xCCCCCCCC);

var mc:MovieClip = this.createEmptyMovieClip("mc",
  this.getNextHighestDepth());
mc.attachBitmap(myBitmapData, this.getNextHighestDepth());

var filter:BevelFilter = new BevelFilter(5, 45, 0xFFFF00, .8, 0x0000FF, .8,
  20, 20, 1, 3, "inner", false);

mc.onPress = function() {
  myBitmapData.applyFilter(myBitmapData, myBitmapData.rectangle, new
  Point(0, 0), filter);
}
```

Véase también

BevelFilter (flash.filters.BevelFilter), BlurFilter (flash.filters.BlurFilter), ColorMatrixFilter (flash.filters.ColorMatrixFilter), ConvolutionFilter (flash.filters.ConvolutionFilter), DisplacementMapFilter (flash.filters.DisplacementMapFilter), DropShadowFilter (flash.filters.DropShadowFilter), GlowFilter (flash.filters.GlowFilter), filters (propiedad MovieClip.filters)

Constructor BitmapData

```
public BitmapData(width:Number, height:Number, [transparent:Boolean],
  [fillColor:Number])
```

Crea un nuevo objeto BitmapData con una determinada anchura y altura. Si especifica un valor para el parámetro `fillColor`, todos los píxeles del mapa de bits utilizan ese color.

De forma predeterminada, el mapa de bits se crea como opaco, a menos que pase el valor `true` para el parámetro `transparent`. Una vez se haya creado el mapa de bits, no puede cambiarlo a uno transparente. Todos los píxeles de un mapa de bits opaco utilizan 24 bits de información del canal de color. Si define el mapa de bits como `transparent`, todos los píxeles utilizan 32 bits de información del canal de color, incluido un canal de transparencia alfa.

Un objeto BitmapData puede tener una altura y una anchura máximas de 2.880 píxeles. Si especifica un valor de anchura o altura mayor que 2.880, no se creará una instancia nueva.

Disponibilidad: ActionScript 1.0; Flash Player 8

Parámetros

width:Number - Anchura de la imagen de mapa de bits, expresada en píxeles.

height:Number - Altura de la imagen de mapa de bits, expresada en píxeles.

transparent:Boolean [opcional] - Define si la imagen de mapa de bits admite transparencia por píxel. El valor predeterminado es `true` (transparente). Para crear un mapa de bits totalmente transparente, defina el valor del parámetro `transparent` a `true` y el del parámetro `fillColor` a `0x00000000` (o a 0).

fillColor:Number [opcional] - Un valor de color ARGB de 32 bits que se utiliza para rellenar el área de la imagen de mapa de bits. El valor predeterminado es 0xFFFFFFFF (blanco sólido).

Ejemplo

El ejemplo siguiente crea un nuevo objeto BitmapData. Los valores de este ejemplo son los predeterminados para los parámetros `transparent` y `fillColor`; podría llamar al constructor sin estos parámetros y obtendría el mismo resultado.

```
import flash.display.BitmapData;

var width:Number = 100;
var height:Number = 80;
var transparent:Boolean = true;
var fillColor:Number = 0xFFFFFFFF;

var bitmap_1:BitmapData = new BitmapData(width, height, transparent,
  fillColor);

trace(bitmap_1.width); // 100
trace(bitmap_1.height); // 80
trace(bitmap_1.transparent); // true

var bitmap_2:BitmapData = new BitmapData(width, height);

trace(bitmap_2.width); // 100
trace(bitmap_2.height); // 80
trace(bitmap_2.transparent); // true
```

clone (método BitmapData.clone)

```
public clone() : BitmapData
```

Devuelve un objeto BitmapData nuevo que es un clon de la instancia original con una copia exacta del mapa de bits contenido.

Disponibilidad: ActionScript 1.0; Flash Player 8

Valor devuelto

`flash.display.BitmapData` - Un objeto BitmapData nuevo que es idéntico al original.

Ejemplo

En el ejemplo siguiente se crean y se comparan tres objetos BitmapData. Puede crear el objeto `bitmap_1` utilizando el constructor `BitmapData`. Para crear la instancia `bitmap_2`, defínala como igual que `bitmap_1`. Para crear `clonedBitmap`, copie `bitmap_1`. Observe que mientras que `bitmap_2` se evalúa como igual que `bitmap_1`, no ocurre igual con `clonedBitmap`, aunque contiene el mismo valor que `bitmap_1`.

```
import flash.display.BitmapData;

var bitmap_1:BitmapData = new BitmapData(100, 80, false, 0x000000);
var bitmap_2:BitmapData = bitmap_1;
var clonedBitmap:BitmapData = bitmap_1.clone();

trace(bitmap_1 == bitmap_2); // true
trace(bitmap_1 == clonedBitmap); // false

for(var i in bitmap_1) {
  trace(">> " + i + ": " + bitmap_1[i]);
  // >> generateFilterRect: [type Function]
  // >> dispose: [type Function]
  // >> clone: [type Function]
  // >> copyChannel: [type Function]
  // >> noise: [type Function]
  // >> merge: [type Function]
  // >> paletteMap: [type Function]
  // >> hitTest: [type Function]
  // >> colorTransform: [type Function]
  // >> perlinNoise: [type Function]
  // >> getColorBoundsRect: [type Function]
  // >> floodFill: [type Function]
  // >> setPixel32: [type Function]
  // >> getPixel32: [type Function]
  // >> pixelDissolve: [type Function]
  // >> draw: [type Function]
  // >> threshold: [type Function]
  // >> scroll: [type Function]
```

```
  // >> applyFilter: [type Function]
  // >> copyPixels: [type Function]
  // >> fillRect: [type Function]
  // >> setPixel: [type Function]
  // >> getPixel: [type Function]
  // >> transparent: false
  // >> rectangle: (x=0, y=0, w=100, h=80)
  // >> height: 80
  // >> width: 100
}

for(var i in clonedBitmap) {
  trace(">> " + i + ": " + clonedBitmap[i]);
  // >> generateFilterRect: [type Function]
  // >> dispose: [type Function]
  // >> clone: [type Function]
  // >> copyChannel: [type Function]
  // >> noise: [type Function]
  // >> merge: [type Function]
  // >> paletteMap: [type Function]
  // >> hitTest: [type Function]
  // >> colorTransform: [type Function]
  // >> perlinNoise: [type Function]
  // >> getColorBoundsRect: [type Function]
  // >> floodFill: [type Function]
  // >> setPixel32: [type Function]
  // >> getPixel32: [type Function]
  // >> pixelDissolve: [type Function]
  // >> draw: [type Function]
  // >> threshold: [type Function]
  // >> scroll: [type Function]
  // >> applyFilter: [type Function]
  // >> copyPixels: [type Function]
  // >> fillRect: [type Function]
  // >> setPixel: [type Function]
  // >> getPixel: [type Function]
  // >> transparent: false
  // >> rectangle: (x=0, y=0, w=100, h=80)
  // >> height: 80
  // >> width: 100
}
```

Para demostrar mejor la relación entre `bitmap_1`, `bitmap_2` y `clonedBitmap`, en el ejemplo siguiente se modifica el valor de píxel en (1, 1) de `bitmap_1`. La modificación del valor de píxel en (1, 1) demuestra que el método `clone()` crea una instancia basada en los valores de la instancia `bitmap_1` en lugar de hacer referencia a los valores.

```
import flash.display.BitmapData;

var bitmap_1:BitmapData = new BitmapData(100, 80, false, 0x000000);
```

```
var bitmap_2:BitmapData = bitmap_1;
var clonedBitmap:BitmapData = bitmap_1.clone();

trace(bitmap_1.getPixel32(1, 1)); // -16777216
trace(bitmap_2.getPixel32(1, 1)); // -16777216
trace(clonedBitmap.getPixel32(1, 1)); // -16777216

bitmap_1.setPixel32(1, 1, 0xFFFFFF);

trace(bitmap_1.getPixel32(1, 1)); // -1
trace(bitmap_2.getPixel32(1, 1)); // -1
trace(clonedBitmap.getPixel32(1, 1)); // -16777216
```

colorTransform (método BitmapData.colorTransform)

```
public colorTransform(rect:Rectangle, colorTransform:ColorTransform) : Void
```

Ajusta los valores de color en una determinada área de una imagen de mapa de bits mediante un objeto ColorTransform. Si el rectángulo coincide con los límites de la imagen de mapa de bits, este método transformará los valores de color de toda la imagen.

Disponibilidad: ActionScript 1.0; Flash Player 8

Parámetros

rect`:flash.geom.Rectangle` - Un objeto Rectangle que define el área de la imagen en la que se aplica el objeto ColorTransform.

colorTransform`:flash.geom.ColorTransform` - Un objeto ColorTransform que describen los valores de transformación de color que se van a aplicar.

Ejemplo

En el ejemplo siguiente se muestra cómo aplicar una operación de transformación de color a una instancia BitmapData:

```
import flash.display.BitmapData;
import flash.geom.ColorTransform;

var myBitmapData:BitmapData = new BitmapData(100, 80, false, 0x00CCCCCC);

var mc:MovieClip = this.createEmptyMovieClip("mc",
  this.getNextHighestDepth());
mc.attachBitmap(myBitmapData, this.getNextHighestDepth());
```

```
mc.onPress = function() {
  myBitmapData.colorTransform(myBitmapData.rectangle, new
  ColorTransform(1, 0, 0, 1, 255, 0, 0, 0));
}
```

Véase también

ColorTransform (flash.geom.ColorTransform), Rectangle (flash.geom.Rectangle)

copyChannel (método BitmapData.copyChannel)

```
public copyChannel(sourceBitmap:BitmapData, sourceRect:Rectangle,
  destPoint:Point, sourceChannel:Number, destChannel:Number) : Void
```

Transfiere datos de un canal de otro objeto BitmapData o del objeto BitmapData actual a un canal del objeto BitmapData actual. Se mantendrán todos los datos de los demás canales del objeto BitmapData de destino.

Los valores del canal de origen y del canal de destino pueden ser los siguientes:

- 1 (rojo)
- 2 (verde)
- 4 (azul)
- 8 (alfa)

Disponibilidad: ActionScript 1.0; Flash Player 8

Parámetros

sourceBitmap:flash.display.BitmapData - La imagen de mapa de bits de entrada que se va a utilizar. La imagen de origen puede ser otro objeto BitmapData o puede hacer referencia al objeto BitmapData actual.

sourceRect:flash.geom.Rectangle - El objeto Rectangle de origen. Especifique un rectángulo de origen con un tamaño menor que el tamaño global del objeto BitmapData si sólo desea copiar los datos del canal de un área menor del mapa de bits.

destPoint:flash.geom.Point - El objeto Point de destino que representa la esquina superior izquierda del área rectangular donde se situarán los nuevos datos del canal. Especifique un punto distinto de (0,0) si desea copiar los datos del canal de un área a otra área en la imagen de destino.

sourceChannel:Number - El canal de origen. Utilice un valor del conjunto (1,2,4,8), que representan los canales rojo, verde, azul y alfa, respectivamente.

destChannel:Number - El canal de destino. Utilice un valor del conjunto (1,2,4,8), que representan los canales rojo, verde, azul y alfa, respectivamente.

Ejemplo

En el ejemplo siguiente se muestra cómo copiar un canal ARGB de origen desde un objeto `BitmapData` de nuevo al propio canal en una ubicación distinta:

```
import flash.display.BitmapData;
import flash.geom.Rectangle;
import flash.geom.Point;

var myBitmapData:BitmapData = new BitmapData(100, 80, false, 0x00CCCCCC);

var mc:MovieClip = this.createEmptyMovieClip("mc",
  this.getNextHighestDepth());
mc.attachBitmap(myBitmapData, this.getNextHighestDepth());

mc.onPress = function() {
  myBitmapData.copyChannel(myBitmapData, new Rectangle(0, 0, 50, 80), new
  Point(51, 0), 3, 1);
}
```

Véase también

`Rectangle (flash.geom.Rectangle)`

copyPixels (método BitmapData.copyPixels)

```
public copyPixels(sourceBitmap:BitmapData, sourceRect:Rectangle,
  destPoint:Point, [alphaBitmap:BitmapData], [alphaPoint:Point],
  [mergeAlpha:Boolean]) : Void
```

Proporciona una rutina rápida para la manipulación de píxeles entre imágenes sin efectos de color, rotación ni expansión. Este método copia un área rectangular de una imagen de origen en un área rectangular del mismo tamaño en el punto de destino del objeto BitmapData de destino.

Si incluye los parámetros `alphaBitmap` y `alphaPoint` puede utilizar una imagen secundaria como origen de alfa para la imagen de origen. Si la imagen de origen tiene datos alfa, se utilizarán los dos conjuntos de datos alfa para componer píxeles de la imagen de origen en la imagen de destino. El parámetro `alphaPoint` es el punto de la imagen alfa que corresponde a la esquina superior izquierda del rectángulo de origen. Los píxeles que queden fuera de la intersección de la imagen de origen y la imagen alfa no se copiarán en la imagen de destino.

La propiedad `mergeAlpha` controla si se utiliza el canal alfa cuando se copia una imagen transparente sobre otra imagen transparente. Para copiar píxeles (sin utilizar alfa), establezca la propiedad `mergeAlpha` como `false`. A continuación, se copiarán todos los píxeles desde el origen hasta el destino. De forma predeterminada, la propiedad `mergeAlpha` es `true`.

Disponibilidad: ActionScript 1.0; Flash Player 8

Parámetros

sourceBitmap:flash.display.BitmapData - La imagen de mapa de bits de entrada desde la que se van a copiar píxeles. La imagen de origen puede ser otra instancia BitmapData o puede hacer referencia a la instancia BitmapData actual.

sourceRect:flash.geom.Rectangle - Un rectángulo que define el área de la imagen de origen que se va a utilizar como entrada.

destPoint:flash.geom.Point - El punto de destino, que representa la esquina superior izquierda del área rectangular donde se sitúan los nuevos píxeles.

alphaBitmap:flash.display.BitmapData [opcional] - Un origen secundario del objeto BitmapData de alfa.

alphaPoint:flash.geom.Point [opcional] - El punto del origen del objeto BitmapData de alfa que corresponde a la esquina superior izquierda del parámetro sourceRect.

mergeAlpha:Boolean [opcional] - Un valor booleano; para utilizar el canal alfa, defina el valor como true. Para copiar píxeles sin el canal alfa, defina el valor como false.

Ejemplo

En el ejemplo siguiente se muestra cómo copiar píxeles desde una instancia BitmapData a otra.

```
import flash.display.BitmapData;
import flash.geom.Rectangle;
import flash.geom.Point;

var bitmapData_1:BitmapData = new BitmapData(100, 80, false, 0x00CCCCCC);
var bitmapData_2:BitmapData = new BitmapData(100, 80, false, 0x00FF0000);

var mc_1:MovieClip = this.createEmptyMovieClip("mc",
  this.getNextHighestDepth());
mc_1.attachBitmap(bitmapData_1, this.getNextHighestDepth());

var mc_2:MovieClip = this.createEmptyMovieClip("mc",
  this.getNextHighestDepth());
mc_2.attachBitmap(bitmapData_2, this.getNextHighestDepth());
mc_2._x = 101;

mc_1.onPress = function() {
  bitmapData_2.copyPixels(bitmapData_1, new Rectangle(0, 0, 50, 80), new
  Point(51, 0));
}

mc_2.onPress = function() {
  bitmapData_1.copyPixels(bitmapData_2, new Rectangle(0, 0, 50, 80), new
  Point(51, 0));
}
```

dispose (método BitmapData.dispose)

```
public dispose() : Void
```

Libera memoria que se utiliza para almacenar el objeto BitmapData.

Cuando se llama a este método en una imagen, la anchura y altura de la imagen se define como 0. Después de vaciar la memoria del objeto BitmapData, fallarán las llamadas de acceso al método y propiedad del objeto, devolviendo un valor de -1.

Disponibilidad: ActionScript 1.0; Flash Player 8

Ejemplo

En el ejemplo siguiente se muestra cómo liberar la memoria de una instancia BitmapData, dando como resultado una instancia sin contenido.

```
import flash.display.BitmapData;

var myBitmapData:BitmapData = new BitmapData(100, 80, false, 0x00CCCCCC);

var mc:MovieClip = this.createEmptyMovieClip("mc",
  this.getNextHighestDepth());
mc.attachBitmap(myBitmapData, this.getNextHighestDepth());

mc.onPress = function() {
  myBitmapData.dispose();

  trace(myBitmapData.width); // -1
  trace(myBitmapData.height); // -1
  trace(myBitmapData.transparent); // -1
}
```

draw (método BitmapData.draw)

```
public draw(source:Object, [matrix:Matrix],
  [colorTransform:ColorTransform], [blendMode:Object],
  [clipRect:Rectangle], [smooth:Boolean]) : Void
```

Dibuja una imagen de origen o un clip de película en una imagen de destino, mediante el procesador de vectores de Flash Player. Puede utilizar los objetos Matrix, ColorTransform, BlendMode y un objeto Rectangle de destino para controlar cómo se realiza la representación. También puede especificar si el mapa de bits se debe suavizar cuando se escala. Eso sólo funciona si el objeto de origen es un objeto BitmapData.

Este método corresponde directamente a cómo se dibujan los objetos utilizando el procesador de vectores estándar para objetos en la interfaz de la herramienta de edición.

Un objeto MovieClip de origen no utiliza ninguna de sus transformaciones en el escenario para esta llamada. Se procesa tal como se encuentra en la biblioteca o el archivo, sin transformación de matriz, transformación de color ni modo de mezcla. Si desea dibujar un clip de película empleando sus propias propiedades de transformación, puede utilizar su objeto Transform para transferir las distintas propiedades de transformación.

Disponibilidad: ActionScript 1.0; Flash Player 8

Parámetros

source:Object - El objeto que se va a dibujar.

matrix:flash.geom.Matrix [opcional] - Un objeto Matrix empleado para escalar, rotar o convertir las coordenadas del mapa de bits. Si no se proporciona ningún objeto, no se transformará la imagen de mapa de bits. Defina ese parámetro a una matrix de identidad, creada mediante el constructor `new Matrix()` predeterminado, si tiene que pasar este parámetro pero no desea transformar la imagen.

colorTransform:flash.geom.ColorTransform [opcional] - Un objeto ColorTransform empleado para ajustar los valores de color del mapa de bits. Si no se proporciona ningún objeto, no se transformarán los colores de la imagen de mapa de bits. Defina ese parámetro a un objeto ColorTransform creada mediante el constructor `new ColorTransform()` predeterminado, si tiene que pasar este parámetro pero no desea transformar la imagen.

blendMode:Object [opcional] - Un objeto BlendMode.

clipRect:flash.geom.Rectangle [opcional] - Un objeto Rectangle. Si no proporciona este valor, no se produce recorte.

smooth:Boolean [opcional] - Un valor booleano que determina si se suavizará un objeto BitmapData cuando se escale. El valor predeterminado es `false`.

Ejemplo

En el ejemplo siguiente se muestra cómo dibujar una instancia MovieClip de origen en un objeto BitmapData.

```
import flash.display.BitmapData;
import flash.geom.Rectangle;
import flash.geom.Matrix;
import flash.geom.ColorTransform;

var myBitmapData:BitmapData = new BitmapData(100, 80, false, 0x00CCCCCC);

var mc_1:MovieClip = this.createEmptyMovieClip("mc",
  this.getNextHighestDepth());
mc_1.attachBitmap(myBitmapData, this.getNextHighestDepth());
```

```
var mc_2:MovieClip = createRectangle(50, 40, 0xFF0000);
mc_2._x = 101;

var myMatrix:Matrix = new Matrix();
myMatrix.rotate(Math.PI/2);

var translateMatrix:Matrix = new Matrix();
translateMatrix.translate(70, 15);

myMatrix.concat(translateMatrix);

var myColorTransform:ColorTransform = new ColorTransform(0, 0, 1, 1, 0, 0,
  255, 0);
var blendMode:String = "normal";

var myRectangle:Rectangle = new Rectangle(0, 0, 100, 80);
var smooth:Boolean = true;

mc_1.onPress = function() {
  myBitmapData.draw(mc_2, myMatrix, myColorTransform, blendMode,
  myRectangle, smooth);
}

function createRectangle(width:Number, height:Number,
  color:Number):MovieClip {
  var depth:Number = this.getNextHighestDepth();
  var mc:MovieClip = this.createEmptyMovieClip("mc_" + depth, depth);
  mc.beginFill(color);
  mc.lineTo(0, height);
  mc.lineTo(width, height);
  mc.lineTo(width, 0);
  mc.lineTo(0, 0);
  return mc;
}
```

fillRect (método BitmapData.fillRect)

```
public fillRect(rect:Rectangle, color:Number) : Void
```

Rellena un área rectangular de píxeles con un determinado color ARGB.

Disponibilidad: ActionScript 1.0; Flash Player 8

Parámetros

rect:`flash.geom.Rectangle` - El área rectangular que se va a rellenar.

color:`Number` - El valor de color ARGB que rellena el área. Los colores ARGB se especifican a menudo en formato hexadecimal, como 0xFF336699.

Ejemplo

En el ejemplo siguiente se muestra cómo rellenar con un color un área que se define mediante un `Rectangle` dentro de un `BitmapData`.

```
import flash.display.BitmapData;
import flash.geom.Rectangle;

var myBitmapData:BitmapData = new BitmapData(100, 80, false, 0x00CCCCCC);

var mc:MovieClip = this.createEmptyMovieClip("mc",
  this.getNextHighestDepth());
mc.attachBitmap(myBitmapData, this.getNextHighestDepth());

mc.onPress = function() {
  myBitmapData.fillRect(new Rectangle(0, 0, 50, 40), 0x00FF0000);
}
```

Véase también

`Rectangle (flash.geom.Rectangle)`

floodFill (método BitmapData.floodFill)

```
public floodFill(x:Number, y:Number, color:Number) : Void
```

Realiza una operación de relleno en una imagen empezando por una coordenada (*x*, *y*) y rellenando con un determinado color. El método `floodFill()` es similar a la herramienta cubo de pintura que incorporan muchos programas de dibujo. El color es un color ARGB que contiene información alfa y de color.

Disponibilidad: ActionScript 1.0; Flash Player 8

Parámetros

x:`Number` - La coordenada *x* de la imagen.

y:`Number` - La coordenada *y* de la imagen.

color:`Number` - El color ARGB que se va a utilizar como relleno. Los colores ARGB se especifican a menudo en formato hexadecimal, como 0xFF336699.

Ejemplo

En el siguiente ejemplo se muestra cómo aplicar un relleno de color a una imagen comenzando en el punto en el que el usuario hace clic dentro de un objeto BitmapData.

```
import flash.display.BitmapData;
import flash.geom.Rectangle;

var myBitmapData:BitmapData = new BitmapData(100, 80, false, 0x00CCCCCC);
```

```
var mc:MovieClip = this.createEmptyMovieClip("mc",
  this.getNextHighestDepth());
mc.attachBitmap(myBitmapData, this.getNextHighestDepth());

myBitmapData.fillRect(new Rectangle(0, 0, 50, 40), 0x00FF0000);

mc.onPress = function() {
  myBitmapData.floodFill(_xmouse, _ymouse, 0x000000FF);
}
```

generateFilterRect (método BitmapData.generateFilterRect)

```
public generateFilterRect(sourceRect:Rectangle, filter:BitmapFilter) :
  Rectangle
```

Determina el rectángulo de destino al que afecta la llamada `applyFilter()`, dados un objeto BitmapData, un rectángulo de origen y un objeto de filtro.

Por ejemplo, un filtro de desenfoque suele afectar a un área mayor que la imagen original. Una imagen de 100 x 200 píxeles que se filtra mediante una instancia BlurFilter predeterminada `blurX = blurY = 4` genera un rectángulo de destino de `(-2,-2,104,204)`. El método `generateFilterRect()` permite conocer el tamaño de este rectángulo de destino de antemano para poder ajustar el tamaño de la imagen de destino convenientemente antes de una operación de filtro.

Algunos filtros recortan su rectángulo de destino de acuerdo con el tamaño de la imagen de origen. Por ejemplo, un `DropShadow` interior no genera un resultado mayor que su imagen de origen. En esta API, se utiliza como límite de origen el objeto BitmapData y no el parámetro `rect` de origen.

Disponibilidad: ActionScript 1.0; Flash Player 8

Parámetros

`sourceRect``:flash.geom.Rectangle` - Un rectángulo que define el área de la imagen de origen que se va a utilizar como entrada.

`filter``:flash.filters.BitmapFilter` - Objeto de filtro que se va a utilizar para calcular el rectángulo de destino.

Valor devuelto

`flash.geom.Rectangle` - Rectángulo de destino calculado empleando una imagen, el parámetro `sourceRect` y un filtro.

Ejemplo

En el ejemplo siguiente se muestra cómo determinar el rectángulo de destino al que afecta el método `applyfilter()`:

```
import flash.display.BitmapData;
import flash.filters.BevelFilter;
import flash.geom.Rectangle;

var myBitmapData:BitmapData = new BitmapData(100, 80, true, 0xCCCCCCCC);

var filter:BevelFilter = new BevelFilter(5, 45, 0xFFFF00, .8, 0x0000FF, .8,
  20, 20, 1, 3, "outter", false);

var filterRect:Rectangle =
  myBitmapData.generateFilterRect(myBitmapData.rectangle, filter);

trace(filterRect); // (x=-31, y=-31, w=162, h=142)
```

getColorBoundsRect (método BitmapData.getColorBoundsRect)

```
public getColorBoundsRect(mask:Number, color:Number, [findColor:Boolean]) :
  Rectangle
```

Determina una región rectangular que encierra completamente todos los píxeles de un determinado color en la imagen de mapa de bits.

Por ejemplo, si tiene una imagen de origen y desea determinar el rectángulo de la imagen que contiene un canal alfa distinto de cero, pasaría `{mask: 0xFF000000, color: 0x00000000}` como parámetros. Se buscan en toda la imagen los límites de los píxeles con `(value & mask) != color`. ara determinar el espacio blanco alrededor de una imagen, se pasaría `{mask: 0xFFFFFFFF, color: 0xFFFFFFFF}` para buscar los límites de los píxeles no blancos.

Disponibilidad: ActionScript 1.0; Flash Player 8

Parámetros

`mask``:Number` - Un valor de color hexadecimal.

`color``:Number` - Un valor de color hexadecimal.

`findColor``:Boolean` [opcional] - Si el valor es `true`, devuelve los límites de un valor de color de una imagen. Si es `false`, devuelve los límites de donde no existe este color en la imagen. El valor predeterminado es `true`.

Valor devuelto

`flash.geom.Rectangle` - La región de la imagen que es del color especificado.

Ejemplo

El el ejemplo siguiente se muestra cómo determinar una región rectangular que encierra completamente todos los píxeles de un determinado color en la imagen de mapa de bits.

```
import flash.display.BitmapData;
import flash.geom.Rectangle;

var myBitmapData:BitmapData = new BitmapData(100, 80, false, 0x00CCCCCC);

var mc:MovieClip = this.createEmptyMovieClip("mc",
  this.getNextHighestDepth());
mc.attachBitmap(myBitmapData, this.getNextHighestDepth());
myBitmapData.fillRect(new Rectangle(0, 0, 50, 40), 0x00FF0000);

mc.onPress = function() {
  var colorBoundsRect:Rectangle =
  myBitmapData.getColorBoundsRect(0x00FFFFFF, 0x00FF0000, true);
  trace(colorBoundsRect); // (x=0, y=0, w=50, h=40)
}
```

getPixel (método BitmapData.getPixel)

```
public getPixel(x:Number, y:Number) : Number
```

Devuelve un entero que representa un valor de píxel RGB de un objeto BitmapData en un punto específico (*x*, *y*). El método `getPixel()` devuelve un valor de píxel no multiplicado. No se devuelve información de alfa.

Todos los píxeles de un objeto BitmapData se almacenan como valores de color premultiplicados. Un píxel de imagen premultiplicado ya tiene los valores de canal de color rojo, verde y azul multiplicados por los datos alfa. Por ejemplo, si el alfa es cero, los canales RGB también son cero, con independencia de sus valores no multiplicados.

Esta pérdida de datos puede ocasionar algunos problemas cuando se realizan operaciones. Todos los métodos de Flash Player toman y devuelven valores no multiplicados. La representación interna de píxeles es no multiplicada antes de que se devuelva como valor. Durante una operación dada, el valor de píxel se premultiplicada antes de establecer el píxel de imagen en bruto.

Disponibilidad: ActionScript 1.0; Flash Player 8

Parámetros

`x:Number` - La posición *x* del píxel.

`y:Number` - La posición *y* del píxel.

Valor devuelto

`Number` - Un número que representa un valor de píxel RGB. Si las coordenadas (*x*, *y*) se encuentran fuera de los límites de la imagen, se devuelve 0.

Ejemplo

En el ejemplo siguiente se utiliza el método `getPixel()` para recuperar el valor RGB de un píxel en una posición *x* e *y* específicas.

```
import flash.display.BitmapData;

var myBitmapData:BitmapData = new BitmapData(100, 80, false, 0x00CCCCCC);

var mc:MovieClip = this.createEmptyMovieClip("mc",
  this.getNextHighestDepth());
mc.attachBitmap(myBitmapData, this.getNextHighestDepth());
trace("0x" + myBitmapData.getPixel(0, 0).toString(16)); // 0xcccccc
```

Véase también

`getPixel32 (método BitmapData.getPixel32)`

getPixel32 (método BitmapData.getPixel32)

```
public getPixel32(x:Number, y:Number) : Number
```

Devuelve un valor de color ARGB que contiene datos del canal alfa y datos de RGB. Este método es similar al método `getPixel()`, que devuelve un color RGB sin datos del canal alfa.

Disponibilidad: ActionScript 1.0; Flash Player 8

Parámetros

x`:Number` - La posición *x* del píxel.

y`:Number` - La posición *y* del píxel.

Valor devuelto

`Number` - Un número que representa un valor de píxel ARGB. Si las coordenadas (*x*, *y*) se encuentran fuera de los límites de la imagen, se devuelve 0. Si el mapa de bits se creó como un mapa de bits opaco y no como uno transparente, el método devolverá un código de error de -1.

Ejemplo

En el ejemplo siguiente se utiliza el método `getPixel32()` para recuperar el valor ARGB de un píxel en una posición *x* e *y* específicas:

```
import flash.display.BitmapData;

var myBitmapData:BitmapData = new BitmapData(100, 80, true, 0xFFAACCEE);

var mc:MovieClip = this.createEmptyMovieClip("mc",
  this.getNextHighestDepth());
mc.attachBitmap(myBitmapData, this.getNextHighestDepth());

var alpha:String = (myBitmapData.getPixel32(0, 0) >> 24 &
  0xFF).toString(16);
trace(">> alpha: " + alpha); // ff

var red:String = (myBitmapData.getPixel32(0, 0) >> 16 & 0xFF).toString(16);
trace(">> red: " + red); // aa

var green:String = (myBitmapData.getPixel32(0, 0) >> 8 &
  0xFF).toString(16);
trace(">> green: " + green); // cc

var blue:String = (myBitmapData.getPixel32(0, 0) & 0xFF).toString(16);
trace(">> blue: " + blue); // ee

trace("0x" + alpha + red + green + blue); // 0xffaaccee
```

Véase también

`getPixel (método BitmapData.getPixel)`

height (propiedad BitmapData.height)

```
public height : Number [read-only]
```

Altura de la imagen de mapa de bits, expresada en píxeles.

Disponibilidad: ActionScript 1.0; Flash Player 8

Ejemplo

En el ejemplo siguiente se demuestra que la propiedad `height` de la instancia `BitmapData` es de sólo lectura intentando definirla sin conseguirlo:

```
import flash.display.BitmapData;

var myBitmapData:BitmapData = new BitmapData(100, 80, false, 0x00CCCCCC);
```

```
var mc:MovieClip = this.createEmptyMovieClip("mc",
  this.getNextHighestDepth());
mc.attachBitmap(myBitmapData, this.getNextHighestDepth());
trace(myBitmapData.height); // 80

myBitmapData.height = 999;
trace(myBitmapData.height); // 80
```

hitTest (método BitmapData.hitTest)

```
public hitTest(firstPoint:Point, firstAlphaThreshold:Number,
  secondObject:Object, [secondBitmapPoint:Point],
  [secondAlphaThreshold:Number]) : Boolean
```

Detecta la zona activa a nivel de píxeles entre una imagen de mapa de bits y un punto, un rectángulo u otra imagen de mapa de bits. Cuando se realiza la prueba de zona activa no se tiene en cuenta expansión, rotación ni ninguna otra transformación de ninguno de los objetos.

Si una imagen es opaca, se considerará un rectángulo totalmente opaco para este método. Las dos imágenes deben ser transparentes para realizar pruebas de zona activa a nivel de píxeles que tengan en cuenta la transparencia. Cuando se comprueban dos imágenes transparentes, los parámetros de umbral alfa controlan qué valores de canal, de 0 a 255, se consideran opacos.

Disponibilidad: ActionScript 1.0; Flash Player 8

Parámetros

`firstPoint``:flash.geom.Point` - Punto que define una ubicación de píxel en la instancia BitmapData actual.

`firstAlphaThreshold``:Number` - El valor de canal alfa que se considera opaco para esta prueba.

`secondObject``:Object` - Un rectángulo, punto o un objeto BitmapData.

`secondBitmapPoint``:flash.geom.Point` [opcional] - Punto que define una ubicación de píxel en la segunda instancia BitmapData. Utilice este parámetro únicamente cuando el valor de `secondObject` sea un objeto BitmapData.

`secondAlphaThreshold``:Number` [opcional] - El valor de canal alfa que se considera opaco en el segundo objeto BitmapData. Utilice este parámetro únicamente cuando el valor de `secondObject` sea un objeto BitmapData, y los dos objetos BitmapData sean transparentes.

Valor devuelto

`Boolean` - Valor booleano. Si hay una zona activa, devuelve un valor de `true`; de lo contrario, devuelve `false`.

Ejemplo

En el ejemplo siguiente se muestra cómo determinar si un objeto BitmapData está colisionando con un `MovieClip`.

```
import flash.display.BitmapData;
import flash.geom.Point;

var myBitmapData:BitmapData = new BitmapData(100, 80, false, 0x00CCCCCC);

var mc_1:MovieClip = this.createEmptyMovieClip("mc",
  this.getNextHighestDepth());
mc_1.attachBitmap(myBitmapData, this.getNextHighestDepth());

var mc_2:MovieClip = createRectangle(20, 20, 0xFF0000);

var destPoint:Point = new Point(myBitmapData.rectangle.x,
  myBitmapData.rectangle.y);
var currPoint:Point = new Point();

mc_1.onEnterFrame = function() {
  currPoint.x = mc_2._x;
  currPoint.y = mc_2._y;
  if(myBitmapData.hitTest(destPoint, 255, currPoint)) {
    trace(">> Collision at y:" + currPoint.x + " and y:" + currPoint.y);
  }
}

mc_2.startDrag(true);

function createRectangle(width:Number, height:Number,
  color:Number):MovieClip {
  var depth:Number = this.getNextHighestDepth();
  var mc:MovieClip = this.createEmptyMovieClip("mc_" + depth, depth);
  mc.beginFill(color);
  mc.lineTo(0, height);
  mc.lineTo(width, height);
  mc.lineTo(width, 0);
  mc.lineTo(0, 0);
  return mc;
}
```

loadBitmap (método BitmapData.loadBitmap)

```
public static loadBitmap(id:String) : BitmapData
```

Devuelve un nuevo objeto BitmapData que contiene una representación de imagen de mapa de bits del símbolo que se identifica mediante un ID de vínculo especificado en la biblioteca.

Disponibilidad: ActionScript 1.0; Flash Player 8

Parámetros

id:String - Un ID de vinculación de un símbolo de la biblioteca.

Valor devuelto

flash.display.BitmapData - Una representación de imagen de mapa de bits del símbolo.

Ejemplo

En el ejemplo siguiente se carga un mapa de bits con el linkageId (ID de vínculo) libraryBitmap de la biblioteca. Debe vincularlo a un objeto MovieClip para darle una representación visual.

```
import flash.display.BitmapData;

var linkageId:String = "libraryBitmap";
var myBitmapData:BitmapData = BitmapData.loadBitmap(linkageId);
trace(myBitmapData instanceof BitmapData); // true

var mc:MovieClip = this.createEmptyMovieClip("mc",
  this.getNextHighestDepth());
mc.attachBitmap(myBitmapData, this.getNextHighestDepth());
```

merge (método BitmapData.merge)

```
public merge(sourceBitmap:BitmapData, sourceRect:Rectangle,
  destPoint:Point, redMult:Number, greenMult:Number, blueMult:Number,
  alphaMult:Number) : Void
```

Realiza una mezcla por canal de una imagen de origen a una imagen de destino. Se utiliza la fórmula siguiente para cada canal:

```
new red dest = (red source * redMult) + (red dest * (256 - redMult) / 256;
```

Los valores de redMult, greenMult, blueMult y alphaMult son los multiplicadores empleados para cada canal de color. Su rango válido va de 0 a 256.

Disponibilidad: ActionScript 1.0; Flash Player 8

Parámetros

sourceBitmap:flash.display.BitmapData - La imagen de mapa de bits de entrada que se va a utilizar. La imagen de origen puede ser otro objeto BitmapData o puede hacer referencia al objeto BitmapData actual.

sourceRect:flash.geom.Rectangle - Un rectángulo que define el área de la imagen de origen que se va a utilizar como entrada.

destPoint:flash.geom.Point - El punto de la imagen de destino (la instancia BitmapData actual) que corresponde a la esquina superior izquierda del rectángulo de origen.

redMult:Number - Un número por el que se multiplica el valor del canal rojo.

greenMult:Number - Un número por el que se multiplica el valor del canal verde.

blueMult:Number - Un número por el que se multiplica el valor del canal azul.

alphaMult:Number - Un número por el que se multiplica el valor de transparencia alfa.

Ejemplo

En el ejemplo siguiente se muestra cómo mezclar parte de un objeto BitmapData con otro.

```
import flash.display.BitmapData;
import flash.geom.Rectangle;
import flash.geom.Point;

var bitmapData_1:BitmapData = new BitmapData(100, 80, false, 0x00CCCCCC);
var bitmapData_2:BitmapData = new BitmapData(100, 80, false, 0x00FF0000);

var mc_1:MovieClip = this.createEmptyMovieClip("mc",
  this.getNextHighestDepth());
mc_1.attachBitmap(bitmapData_1, this.getNextHighestDepth());

var mc_2:MovieClip = this.createEmptyMovieClip("mc",
  this.getNextHighestDepth());
mc_2.attachBitmap(bitmapData_2, this.getNextHighestDepth());
mc_2._x = 101;

mc_1.onPress = function() {
  bitmapData_1.merge(bitmapData_2, new Rectangle(0, 0, 50, 40), new
  Point(25, 20), 128, 0, 0, 0);
}
```

noise (método BitmapData.noise)

```
public noise(randomSeed:Number, [low:Number], [high:Number],
  [channelOptions:Number], [grayScale:Boolean]) : Void
```

Rellena una imagen con píxeles que representan ruido aleatorio.

Disponibilidad: ActionScript 1.0; Flash Player 8

Parámetros

randomSeed:Number - El valor de inicialización aleatorio que se va a utilizar.

low:Number [opcional] - El valor más bajo que se va a generar para cada canal (de 0 a 255). El valor predeterminado es 0.

high:Number [opcional] - El valor más alto que se va a generar para cada canal (de 0 a 255). El valor predeterminado es 255.

channelOptions:Number [opcional] - Un número que puede ser una combinación de cualquiera de los cuatro valores de canal: 1 (rojo), 2 (verde), 4 (azul) y 8(alfa). Puede utilizar el operador lógico OR | para combinar valores de canal. El valor predeterminado es (1 | 2 | 4).

grayScale:Boolean [opcional] - Valor booleano. Si es true, se creará una imagen en escala de grises estableciendo todos los canales de color con el mismo valor. La selección del canal alfa no se ve afectada por el establecimiento de este parámetro como true. El valor predeterminado es false.

Ejemplo

En el ejemplo siguiente se muestra cómo aplicar ruido de píxel a un objeto BitmapData para un mapa de bits en color y para uno en blanco y negro.

```
import flash.display.BitmapData;
import flash.geom.Rectangle;
import flash.geom.Point;

var bitmapData_1:BitmapData = new BitmapData(100, 80, false, 0x00CCCCCC);
var bitmapData_2:BitmapData = new BitmapData(100, 80, false, 0x00FF0000);

var mc_1:MovieClip = this.createEmptyMovieClip("mc",
  this.getNextHighestDepth());
mc_1.attachBitmap(bitmapData_1, this.getNextHighestDepth());

var mc_2:MovieClip = this.createEmptyMovieClip("mc",
  this.getNextHighestDepth());
mc_2.attachBitmap(bitmapData_2, this.getNextHighestDepth());
mc_2._x = 101;
```

```
mc_1.onPress = function() {
    bitmapData_1.merge(bitmapData_2, new Rectangle(0, 0, 50, 40), new
  Point(25, 20), 128, 0, 0, 0);
}

mc_1.onPress = function() {
    bitmapData_1.noise(128, 0, 255, 1, true);
}

mc_2.onPress = function() {
    bitmapData_2.noise(128);
}
```

paletteMap (método BitmapData.paletteMap)

```
public paletteMap(sourceBitmap:BitmapData, sourceRect:Rectangle,
  destPoint:Point, [redArray:Array], [greenArray:Array], [blueArray:Array],
  [alphaArray:Array]) : Void
```

Reasigna los valores del canal de color en una imagen proporcionada a cuatro conjuntos de datos de la paleta de colores, una por cada canal.

Flash Player utiliza la fórmula siguiente para generar la imagen resultante.

Después de calcular los valores rojo, verde, azul y alfa, se suman utilizando aritmética estándar de enteros de 32 bits. Los valores del canal rojo, azul, verde y alfa de cada píxel se extraen en un valor entre 0 y 255 independiente. A continuación se utilizan estos valores para consultar nuevos valores de color en la matriz adecuada: `redArray`, `greenArray`, `blueArray` y `alphaArray`. Cada una de estas cuatro matrices debe contener 256 valores. Después de recuperar los cuatro valores nuevos de canal, se combinan en un valor ARGB estándar que se aplica al píxel.

Este método admite efectos intercanal. Cada matriz de entrada puede contener valores completos de 32 bits, y no se produce desplazamiento cuando se suman los valores. Esta rutina no admite fijación de canales individuales.

Si no se especifica ninguna matriz para un canal, el canal de color se copia simplemente desde la imagen de origen hasta la de destino.

Puede emplear este método para diversos efectos, como asignación de paleta general (tomando un canal y convirtiéndolo en una imagen de color falso). También puede utilizarlo para una gran variedad de algoritmos de manipulación de color avanzados, como gamma, curvas, niveles y cuantificación.

Disponibilidad: ActionScript 1.0; Flash Player 8

Parámetros

sourceBitmap:flash.display.BitmapData - La imagen de mapa de bits de entrada que se va a utilizar. La imagen de origen puede ser otro objeto BitmapData o puede hacer referencia al objeto BitmapData actual.

sourceRect:flash.geom.Rectangle - Un rectángulo que define el área de la imagen de origen que se va a utilizar como entrada.

destPoint:flash.geom.Point - El punto de la imagen de destino (el objeto BitmapData actual) que corresponde a la esquina superior izquierda del rectángulo de origen.

redArray:Array [opcional] - Si redArray no es null, red = redArray[source red value] else red = source rect value.

greenArray:Array [opcional] - Si greenArray no es null, green = greenArray[source green value] else green = source green value.

blueArray:Array [opcional] - Si blueArray no es null, blue = blueArray[source blue value] else blue = source blue value.

alphaArray:Array [opcional] - Si alphaArray no es null, alpha = alphaArray[source alpha value] else alpha = source alpha value.

Ejemplo

En el ejemplo siguiente se muestra cómo utilizar un mapa de paleta para convertir rojo sólido en verde y verde sólido en rojo en un único objeto BitmapData.

```
import flash.display.BitmapData;
import flash.geom.Rectangle;
import flash.geom.Point;

var myBitmapData:BitmapData = new BitmapData(100, 80, false, 0x00FF0000);

var mc:MovieClip = this.createEmptyMovieClip("mc",
  this.getNextHighestDepth()); mc.attachBitmap(myBitmapData,
  this.getNextHighestDepth());

myBitmapData.fillRect(new Rectangle(51, 0, 50, 80), 0x0000FF00);

mc.onPress = function() {
  var redArray:Array = new Array(256);
  var greenArray:Array = new Array(256);

  for(var i = 0; i < 255; i++) {
    redArray[i] = 0x00000000;
    greenArray[i] = 0x00000000;
  }

  redArray[0xFF] = 0x0000FF00;
```

```
  greenArray[0xFF] = 0x00FF0000;

  myBitmapData.paletteMap(myBitmapData, new Rectangle(0, 0, 100, 40), new
  Point(0, 0), redArray, greenArray, null, null);
}
```

perlinNoise (método BitmapData.perlinNoise)

```
public perlinNoise(baseX:Number, baseY:Number, numOctaves:Number,
  randomSeed:Number, stitch:Boolean, fractalNoise:Boolean,
  [channelOptions:Number], [grayScale:Boolean], [offsets:Object]) : Void
```

Genera una imagen de ruido Perlin.

El algoritmo de generación de ruido Perlin interpola y combina funciones individuales de ruido aleatorio (denominadas octavas) en una función única que genera ruido aleatorio de aspecto más natural. Como las octavas musicales, cada función de octava es dos veces la frecuencia de la anterior. El ruido Perlin se ha descrito como una "suma fractal de ruido", porque que combina múltiples conjuntos de datos de ruido con distintos niveles de detalle.

Las funciones de ruido Perlin pueden emplearse para simular fenómenos naturales y paisajes, como el grano de la madera, nubes o montañas. En la mayoría de los casos, el resultado de una función de ruido Perlin no se muestra directamente, pero se utiliza para generar otras imágenes y darles variaciones pseudoaleatorias.

Las funciones digitales sencillas de ruido aleatorio producen a menudo imágenes con puntos contrastados y discordantes. Este tipo de contraste no suele darse en la naturaleza. El algoritmo de ruido Perlin combina varias funciones de ruido que funcionan en distintos niveles de detalle. Este algoritmo produce menores variaciones entre los valores de píxel colindantes.

Nota: El algoritmo de ruido Perlin recibe su nombre de Ken Perlin, que lo desarrolló después de generar gráficos para la película *Tron* en 1982. En 1997, Perlin ganó un Oscar en la categoría de logros tecnológicos por la función de ruido que él había desarrollado.

Disponibilidad: ActionScript 1.0; Flash Player 8

Parámetros

baseX: `Number` - La frecuencia que se va a utilizar la dirección *x*. Por ejemplo, para generar un ruido con el tamaño ajustado para una imagen de 64 x 128, pase 64 como valor `baseX`.

baseY: `Number` - La frecuencia que se va a utilizar la dirección *y*. Por ejemplo, para generar un ruido con el tamaño ajustado para una imagen de 64 x 128, pase 128 como valor `baseY`.

numOctaves:Number - Número de octavas o funciones de ruido individuales que se va a combinar para crear este ruido. Cuanto mayor sea el número de octavas más detalladas serán las imágenes. Asimismo, un número elevado de octavas requiere más tiempo de procesamiento.

randomSeed:Number - El número de inicialización aleatorio que se va a utilizar. Si mantiene iguales todos los demás parámetros, podrá generar resultados pseudoaleatorios variando el valor de inicialización aleatorio. La función de ruido Perlin es una función de asignación y no una verdadera función de generación de ruido aleatorio, por lo que producirá siempre los mismos resultados con el mismo valor de inicialización aleatorio.

stitch:Boolean - Valor booleano. Si es `true`, intenta suavizar los bordes de transición de la imagen para crear texturas totalmente integradas para mosaico como relleno de mapa de bits.

fractalNoise:Boolean - Valor booleano. Si el valor es `true`, el método genera ruido fractal; en caso contrario, genera turbulencia. Una imagen con turbulencia presenta discontinuidades visibles en el degradado que pueden producir efectos visuales más nítidos, como llamas u olas del mar.

channelOptions:Number [opcional] - Un número que indica uno o varios canales de color. Para crear este valor, puede utilizar o combinar cualquier de los cuatro valores de canal de color: `1` (rojo), `2` (verde), `4` (azul) y `8`(alfa). Puede combinar los valores de canal utilizando el operador lógico OR, por ejemplo, puede combinar los canales rojo y verde con el código siguiente: `1 | 2`.

grayScale:Boolean [opcional] - Valor booleano. Si es `true`, se creará una imagen en escala de grises utilizando valores idénticos para todos los canales de color rojo, verde y azul. El valor del canal alfa no se ve afectado si este valor se establece como `true`. El valor predeterminado es `false`.

offsets:Object [opcional] - Una matriz de puntos que corresponde a desplazamientos *x* e *y* para cada octava. La manipulación de los valores de desplazamiento permite desplazarse con suavidad por una imagen con perlinNoise. Cada punto de la matriz de desplazamiento afecta a una función de ruido de octava específica.

Ejemplo

En el ejemplo siguiente se muestra cómo aplicar un ruido Perlin a una instancia BitmapData.

```
import flash.display.BitmapData;

var bitmapData_1:BitmapData = new BitmapData(100, 80, false, 0x00CCCCCC);
var bitmapData_2:BitmapData = new BitmapData(100, 80, false, 0x00FF0000);

var mc_1:MovieClip = this.createEmptyMovieClip("mc",
  this.getNextHighestDepth());
mc_1.attachBitmap(bitmapData_1, this.getNextHighestDepth());
```

```
var mc_2:MovieClip = this.createEmptyMovieClip("mc",
  this.getNextHighestDepth());
mc_2.attachBitmap(bitmapData_2, this.getNextHighestDepth());
mc_2._x = 101;

mc_1.onPress = function() {
  var randomNum:Number = Math.floor(Math.random() * 10);
    bitmapData_1.perlinNoise(100, 80, 6, randomNum, false, true, 1, true,
  null);
}

mc_2.onPress = function() {
  var randomNum:Number = Math.floor(Math.random() * 10);
    bitmapData_2.perlinNoise(100, 80, 4, randomNum, false, false, 15,
  false, null);
}
```

pixelDissolve (método BitmapData.pixelDissolve)

```
public pixelDissolve(sourceBitmap:BitmapData, sourceRect:Rectangle,
  destPoint:Point, [randomSeed:Number], [numberOfPixels:Number],
  [fillColor:Number]) : Number
```

Realiza una disolución de píxeles de una imagen de origen a una imagen de destino o utilizando la misma imagen. Flash Player utiliza un valor `randomSeed` para generar una disolución de píxeles aleatoria. El valor devuelto por la función debe pasarse en llamadas posteriores para continuar la disolución de píxeles hasta terminar.

Si la imagen de origen no es igual que la de destino, los píxeles se copiarán desde el origen hasta el destino utilizando todas las propiedades. Esto permite disolver desde una imagen en blanco hasta una imagen completamente llena.

Si las imágenes de origen y destino son iguales, los píxeles se rellenan con el parámetro `color`. Esto permite disolver desde una imagen completamente llena. En este modo se ignora el parámetro `point` de destino.

Disponibilidad: ActionScript 1.0; Flash Player 8

Parámetros

`sourceBitmap``:flash.display.BitmapData` - La imagen de mapa de bits de entrada que se va a utilizar. La imagen de origen puede ser otro objeto BitmapData o puede hacer referencia a la instancia BitmapData actual.

`sourceRect``:flash.geom.Rectangle` - Un rectángulo que define el área de la imagen de origen que se va a utilizar como entrada.

destPoint:flash.geom.Point - El punto de la imagen de destino (la instancia BitmapData actual) que corresponde a la esquina superior izquierda del rectángulo de origen.

randomSeed:Number [opcional] - El valor de inicialización aleatorio que se va a utilizar para comenzar la disolución de píxeles. El valor predeterminado es 0.

numberOfPixels:Number [opcional] - El valor predeterminado es 1/30 del área de origen (ancho x alto).

fillColor:Number [opcional] - Un valor de color ARGB que se utiliza para rellenar los píxeles con un valor de origen igual al de destino. El valor predeterminado es 0.

Valor devuelto

Number - El nuevo valor de inicialización aleatorio que se va a utilizar para llamadas posteriores.

Ejemplo

En el ejemplo siguiente se utiliza pixelDissolve() para convertir un objeto BitmapData gris a uno rojo disolviendo 40 píxeles a la vez hasta que la totalidad de los 8000 píxeles han cambiado de color:

```
import flash.display.BitmapData;
import flash.geom.Point;

var myBitmapData:BitmapData = new BitmapData(100, 80, false, 0x00CCCCCC);

var mc:MovieClip = this.createEmptyMovieClip("mc",
  this.getNextHighestDepth());
mc.attachBitmap(myBitmapData, this.getNextHighestDepth());

mc.onPress = function() {
  var randomNum:Number = Math.floor(Math.random() * 10);
  dissolve(randomNum);
}

var intervalId:Number;
var totalDissolved:Number = 0;
var totalPixels:Number = 8000;

function dissolve(randomNum:Number) {
  var newNum:Number = myBitmapData.pixelDissolve(myBitmapData,
  myBitmapData.rectangle, new Point(0, 0), randomNum, 40, 0x00FF0000);
  clearInterval(intervalId);
  if(totalDissolved < totalPixels) {
    intervalId = setInterval(dissolve, 10, newNum);
  }
  totalDissolved += 40;
}
```

rectangle (propiedad BitmapData.rectangle)

```
public rectangle : Rectangle [read-only]
```

Rectángulo que define el tamaño y la ubicación de la imagen de mapa de bits. Las partes superior e izquierda del rectángulo son 0; el ancho y el alto son iguales al ancho y al alto en píxeles del objeto BitmapData.

Disponibilidad: ActionScript 1.0; Flash Player 8

Ejemplo

En el ejemplo siguiente se demuestra que la propiedad `rectangle` de la instancia `Bitmap` es de sólo lectura intentando definirla sin conseguirlo:

```
import flash.display.BitmapData;
import flash.geom.Rectangle;

var myBitmapData:BitmapData = new BitmapData(100, 80, false, 0x00CCCCCC);

var mc:MovieClip = this.createEmptyMovieClip("mc",
  this.getNextHighestDepth());
mc.attachBitmap(myBitmapData, this.getNextHighestDepth());
trace(myBitmapData.rectangle); // (x=0, y=0, w=100, h=80)

myBitmapData.rectangle = new Rectangle(1, 2, 4, 8);
trace(myBitmapData.rectangle); // (x=0, y=0, w=100, h=80)
```

scroll (método BitmapData.scroll)

```
public scroll(x:Number, y:Number) : Void
```

Desplaza una imagen una determinada cantidad de píxeles (*x*, *y*). Las regiones del borde fuera del área de desplazamiento permanecen sin modificar.

Disponibilidad: ActionScript 1.0; Flash Player 8

Parámetros

x`:Number` - La cantidad en la que se realiza desplazamiento horizontal.

y`:Number` - La cantidad en la que se realiza desplazamiento vertical.

Ejemplo

El ejemplo siguiente muestra cómo desplazar un objeto BitmapData.

```
import flash.display.BitmapData;
import flash.geom.Rectangle;

var myBitmapData:BitmapData = new BitmapData(100, 80, false, 0x00CCCCCC);
```

```
var mc:MovieClip = this.createEmptyMovieClip("mc",
  this.getNextHighestDepth());
mc.attachBitmap(myBitmapData, this.getNextHighestDepth());

myBitmapData.fillRect(new Rectangle(0, 0, 25, 80), 0x00FF0000);

mc.onPress = function() {
  myBitmapData.scroll(25, 0);
}
```

setPixel (método BitmapData.setPixel)

```
public setPixel(x:Number, y:Number, color:Number) : Void
```

Establece el color de un solo píxel de un objeto BitmapData. Durante esta operación se mantiene el valor del canal alfa actual del píxel de la imagen. El valor del parámetro de color RGB se trata como un valor de color no multiplicado.

Disponibilidad: ActionScript 1.0; Flash Player 8

Parámetros

x:Number - La posición *x* del píxel cuyo valor cambia.

y:Number - La posición *x* del píxel cuyo valor cambia.

color:Number - El color RGB con el que se establece el píxel.

Ejemplo

En el ejemplo siguiente se utiliza el método `getPixel()` para asignar el valor RGB de un píxel en una posición *x* e *y* específicas. Puede dibujar en el mapa de bits creado en 0x000000 arrastrando el ratón.

```
import flash.display.BitmapData;

var myBitmapData:BitmapData = new BitmapData(100, 80, false, 0x00CCCCCC);

var mc:MovieClip = this.createEmptyMovieClip("mc",
  this.getNextHighestDepth());
mc.attachBitmap(myBitmapData, this.getNextHighestDepth());

mc.onPress = function() {
  this.onEnterFrame = sketch;
}

mc.onRelease = function() {
  delete this.onEnterFrame;
}
```

```
function sketch() {
  myBitmapData.setPixel(_xmouse, _ymouse, 0x000000);
}
```

Véase también

getPixel (método BitmapData.getPixel), setPixel32 (método BitmapData.setPixel32)

setPixel32 (método BitmapData.setPixel32)

```
public setPixel32(x:Number, y:Number, color:Number) : Void
```

Establece los valores de color y transparencia alfa de un solo píxel de un objeto BitmapData. Este método es similar al método `setPixel()`; la diferencia principal radica en que el método `setPixel32()` toma un valor de color ARGB que contiene información del canal alfa.

Disponibilidad: ActionScript 1.0; Flash Player 8

Parámetros

x:`Number` - La posición *x* del píxel cuyo valor cambia.

y:`Number` - La posición *x* del píxel cuyo valor cambia.

color:`Number` - El color ARGB con el que se establece el píxel. Si ha creado un mapa de bits opaco (no uno transparente), se ignora la porción de transparencia alfa de este valor.

Ejemplo

En el ejemplo siguiente se utiliza el método `setPixel32()`para asignar el valor ARGB de un píxel en una posición x e y específicas. Puede dibujar en el mapa de bits creado en 0x000000 sin un valor alfa manteniendo pulsado el botón del ratón y arrastrándolo.

```
import flash.display.BitmapData;

var myBitmapData:BitmapData = new BitmapData(100, 80, true, 0xFFCCCCCC);

var mc:MovieClip = this.createEmptyMovieClip("mc",
  this.getNextHighestDepth());
mc.attachBitmap(myBitmapData, this.getNextHighestDepth());

mc.onPress = function() {
  this.onEnterFrame = sketch;
}

mc.onRelease = function() {
  delete this.onEnterFrame;
}
```

```
function sketch() {
  myBitmapData.setPixel32(_xmouse, _ymouse, 0x00000000);
}
```

Véase también

getPixel32 (método BitmapData.getPixel32), setPixel (método BitmapData.setPixel)

threshold (método BitmapData.threshold)

```
public threshold(sourceBitmap:BitmapData, sourceRect:Rectangle,
  destPoint:Point, operation:String, threshold:Number, [color:Number],
  [mask:Number], [copySource:Boolean]) : Number
```

Prueba valores de píxel en una imagen con un umbral especificado y define los píxeles que se pasan para probar los valores del nuevo color. El método `threshold()` permite aislar y reemplazar rangos de color en una imagen y realizar otras operaciones lógicas en los píxeles de la imagen.

La lógica de la prueba de umbral es la siguiente:

```
if ((pixelValue & mask) operation (threshold & mask)) then

    set pixel to color

 else

    if (copySource) then

       set pixel to corresponding pixel value from sourceBitmap
```

El parámetro `operation` especifica el operador de comparación que se va a utilizar para la prueba de umbral. Por ejemplo, utilizando "==", puede aislar un color específico de una imagen. Asimismo, utilizando `{operation: "<", mask: 0xFF000000, threshold: 0x7f000000, color: 0x00000000}`, puede establecer todos los píxeles de destino como totalmente transparentes cuando el alfa del píxel de la imagen de origen es menor que 0x7F. Puede utilizar esta técnica para transiciones animadas y otros efectos.

Disponibilidad: ActionScript 1.0; Flash Player 8

Parámetros

`sourceBitmap``:flash.display.BitmapData` - La imagen de mapa de bits de entrada que se va a utilizar. La imagen de origen puede ser otro objeto BitmapData o puede hacer referencia a la instancia BitmapData actual.

sourceRect:flash.geom.Rectangle - Un rectángulo que define el área de la imagen de origen que se va a utilizar como entrada.

destPoint:flash.geom.Point - El punto de la imagen de destino (la instancia BitmapData actual) que corresponde a la esquina superior izquierda del rectángulo de origen.

operation:String - Uno de los siguientes operadores de comparación, pasados como una cadena: "<", "<=", ">", ">=", "==", "!="

threshold:Number - El valor con el que se prueba cada píxel para comprobar si está dentro o supera el umbral.

color:Number [opcional] - El valor de color al que se define un píxel si se supera la prueba de umbral. El valor predeterminado es 0x00000000.

mask:Number [opcional] - La máscara que se va a emplear para aislar un componente de color. El valor predeterminado es 0xFFFFFFFF.

copySource:Boolean [opcional] - Valor booleano. Si el valor es `true`, los valores de píxel de la imagen de origen se copian al destino cuando falla la prueba de umbral. Si es `false`, la imagen de origen no se copia cuando falla la prueba de umbral. El valor predeterminado es `false`.

Valor devuelto

`Number` - El número de píxeles modificados.

Ejemplo

En el ejemplo siguiente se muestra cómo cambiar el valor de color de los píxeles con un valor de color mayor o igual que un umbral concreto.

```
import flash.display.BitmapData;
import flash.geom.Rectangle;
import flash.geom.Point;

var myBitmapData:BitmapData = new BitmapData(100, 80, false, 0x00CCCCCC);

var mc:MovieClip = this.createEmptyMovieClip("mc",
  this.getNextHighestDepth());
mc.attachBitmap(myBitmapData, this.getNextHighestDepth());

myBitmapData.fillRect(new Rectangle(0, 0, 50, 80), 0x00FF0000);

mc.onPress = function() {
  myBitmapData.threshold(myBitmapData, new Rectangle(0, 0, 100, 40), new
  Point(0, 0), ">=", 0x00CCCCCC, 0x000000FF, 0x00FF0000, false);
}
```

transparent (propiedad BitmapData.transparent)

```
public transparent : Boolean [read-only]
```

Define si la imagen de mapa de bits admite transparencia por píxel. Solamente puede definir este valor cuando cree un objeto BitmapData pasando `true` para el parámetro `transparent`. Tras crear el objeto BitmapData, puede comprobar si admite transparencia por píxel observando si el valor de la propiedad `transparent` es `true`.

Disponibilidad: ActionScript 1.0; Flash Player 8

Ejemplo

En el ejemplo siguiente se demuestra que la propiedad `transparent` de la instancia `Bitmap` es de sólo lectura intentando definirla sin conseguirlo:

```
import flash.display.BitmapData;

var myBitmapData:BitmapData = new BitmapData(100, 80, false, 0x00CCCCCC);

var mc:MovieClip = this.createEmptyMovieClip("mc",
  this.getNextHighestDepth());
mc.attachBitmap(myBitmapData, this.getNextHighestDepth());
trace(myBitmapData.transparent); // false

myBitmapData.transparent = true;
trace(myBitmapData.transparent); // false
```

width (propiedad BitmapData.width)

```
public width : Number [read-only]
```

Anchura de la imagen de mapa de bits, expresada en píxeles.

Disponibilidad: ActionScript 1.0; Flash Player 8

Ejemplo

En el ejemplo siguiente se demuestra que la propiedad `width` de la instancia `Bitmap` es de sólo lectura intentando definirla sin conseguirlo:

```
import flash.display.BitmapData;

var myBitmapData:BitmapData = new BitmapData(100, 80, false, 0x00CCCCCC);

var mc:MovieClip = this.createEmptyMovieClip("mc",
  this.getNextHighestDepth());
mc.attachBitmap(myBitmapData, this.getNextHighestDepth());
trace(myBitmapData.width); // 100
```

```
myBitmapData.width = 999;
trace(myBitmapData.width); // 100
```

BitmapFilter (flash.filters.BitmapFilter)

```
Object
  |
  +-flash.filters.BitmapFilter
```

public class *BitmapFilter*
extends Object

La clase base BitmapFilter para todos los efectos de filtro de imagen.

Las clases BevelFilter, BlurFilter, ColorMatrixFilter, ConvolutionFilter, DisplacementMapFilter, DropShadowFilter, GlowFilter, GradientBevelFilter y GradientGlowFilter amplían la clase BitmapFilter. Estos efectos de filtro se pueden aplicar a mapas de bits o instancias MovieClip.

Solamente puede crear subclases para las subclases que preceden a la clase BitmapFilter.

Disponibilidad: ActionScript 1.0; Flash Player 8

Resumen de propiedades

Propiedades heredadas de la clase Object

constructor (propiedad Object.constructor), __proto__ (propiedad Object.__proto__), prototype (propiedad Object.prototype), __resolve (propiedad Object.__resolve)

Resumen de métodos

Modificadores	Firma	Descripción
	clone() : BitmapFilter	Devuelve un objeto BitmapFilter que es una copia exacta del objeto BitmapFilter original.

Métodos heredados de la clase Object

addProperty (método Object.addProperty), hasOwnProperty (método Object.hasOwnProperty), isPropertyEnumerable (método Object.isPropertyEnumerable), isPrototypeOf (método Object.isPrototypeOf), registerClass (método Object.registerClass), toString (método Object.toString), unwatch (método Object.unwatch), valueOf (método Object.valueOf), watch (método Object.watch)

clone (método BitmapFilter.clone)

```
public clone() : BitmapFilter
```

Devuelve un objeto BitmapFilter que es una copia exacta del objeto BitmapFilter original.

Disponibilidad: ActionScript 1.0; Flash Player 8

Valor devuelto

`flash.filters.BitmapFilter` - Un objeto BitmapFilter.

BlurFilter (flash.filters.BlurFilter)

```
Object
  |
  +-flash.filters.BitmapFilter
    |
    +-flash.filters.BlurFilter
```

```
public class BlurFilter
extends BitmapFilter
```

La clase BlurFilter permite aplicar un efecto visual de desenfoque a diversos objetos en Flash. El efecto de desenfoque suaviza los detalles de una imagen. El desenfoque puede oscilar desde un ligero efecto hasta el desenfoque gaussiano, con un aspecto brumoso como el que se obtiene al mirar a través de un cristal semiopaco. Cuando la propiedad `quality` de este filtro se define como 1, el resultado es un ligero efecto. Cuando la propiedad `quality` se define como 3, se aproxima a un filtro de desenfoque gaussiano.

El uso de filtros depende del objeto al que se aplique el filtro:

- Para aplicar filtros a clips de película, campos de texto y botones en tiempo de ejecución, utilice la propiedad `filters`. El establecimiento de la propiedad `filters` de un objeto no modifica el objeto y se puede deshacer borrando la propiedad `filters`.
- Para aplicar filtros a instancias de BitmapData, utilice el método `BitmapData.applyFilter()`. La llamada a `applyFilter` en un objeto BitmapData toma el objeto BitmapData de origen y el objeto de filtro y genera una imagen filtrada.

También puede aplicar efectos de filtro a imágenes y vídeo en tiempo de edición. Para más información, consulte la documentación de edición.

Si aplica un filtro a un clip de película o a un botón, la propiedad `cacheAsBitmap` del clip de película o del botón se establecerá como `true`. Si borra todos los filtros, se restaurará el valor original de `cacheAsBitmap`.

Este filtro admite escalado de escenario. Sin embargo, no admite escalado genera, rotación ni sesgo. Cuando se escala el propio objeto (`_xscale` y `_yscale` no son 100%), no se escala el filtro. Sólo se escalará cuando se acerque el escenario.

El filtro no se aplica si la imagen resultante supera los 2880 píxeles de anchura o altura. Por ejemplo, si acerca un clip de película con un filtro aplicado, éste se desactiva si la imagen resultante supera el límite de 2880 píxeles.

Disponibilidad: ActionScript 1.0; Flash Player 8

Véase también

`filters (propiedad MovieClip.filters)`, `cacheAsBitmap (propiedad MovieClip.cacheAsBitmap)`, `filters (propiedad Button.filters)`, `cacheAsBitmap (propiedad Button.cacheAsBitmap)`, `filters (propiedad TextField.filters)`, `applyFilter (método BitmapData.applyFilter)`

Resumen de propiedades

Modificadores	Propiedad	Descripción
	`blurX:Number`	Cantidad de desenfoque horizontal.
	`blurY:Number`	Cantidad de desenfoque vertical.
	`quality:Number`	Número de veces que debe realizarse el desenfoque.

Propiedades heredadas de la clase Object

`constructor (propiedad Object.constructor)`, `__proto__ (propiedad Object.__proto__)`, `prototype (propiedad Object.prototype)`, `__resolve (propiedad Object.__resolve)`

Resumen de constructores

Firma	Descripción
`BlurFilter([blurX:Number], [blurY:Number], [quality:Number])`	Inicializa el filtro con los parámetros especificados.

Resumen de métodos

Modificadores	Firma	Descripción
	`clone() : BlurFilter`	Devuelve una copia de este objeto de filtro.

Métodos heredados de la clase BitmapFilter

clone (método BitmapFilter.clone)

Métodos heredados de la clase Object

addProperty (método Object.addProperty), hasOwnProperty (método Object.hasOwnProperty), isPropertyEnumerable (método Object.isPropertyEnumerable), isPrototypeOf (método Object.isPrototypeOf), registerClass (método Object.registerClass), toString (método Object.toString), unwatch (método Object.unwatch), valueOf (método Object.valueOf), watch (método Object.watch)

Constructor BlurFilter

`public BlurFilter([blurX:Number], [blurY:Number], [quality:Number])`

Inicializa el filtro con los parámetros especificados. Los valores predeterminados crean una imagen suave y desenfocada.

Disponibilidad: ActionScript 1.0; Flash Player 8

Parámetros

`blurX``:Number` [opcional] - La cantidad que se va a desenfocar en horizontal. Los valores válidos son de 0 a 255 (valor de coma flotante). El valor predeterminado es 4. Los valores que son múltiplos de 2 (como 2, 4, 8, 16 y 32) se optimizan para representarse más rápidamente que otros valores.

`blurY``:Number` [opcional] - La cantidad que se va a desenfocar en vertical. Los valores válidos son de 0 a 255 (valor de coma flotante). El valor predeterminado es 4. Los valores que son múltiplos de 2 (como 2, 4, 8, 16 y 32) se optimizan para representarse más rápidamente que otros valores.

`quality``:Number` [opcional] - Número de veces que debe aplicarse el filtro. El valor predeterminado es 1, que equivale a calidad baja. Un valor 2 equivale a calidad media y un valor 3 a alta, y se aproxima a un desenfoque gaussiano.

Ejemplo

En el ejemplo siguiente se crea una instancia de un constructor `BlurFilter` nuevo y se aplica a una forma plana y rectangular:

```
import flash.filters.BlurFilter;
var rect:MovieClip = createRectangle(100, 100, 0x003366,
  "BlurFilterExample");
```

```
var blurX:Number = 30;
var blurY:Number = 30;
var quality:Number = 3;

var filter:BlurFilter = new BlurFilter(blurX, blurY, quality);
var filterArray:Array = new Array();
filterArray.push(filter);
rect.filters = filterArray;

function createRectangle(w:Number, h:Number, bgColor:Number,
  name:String):MovieClip {
  var mc:MovieClip = this.createEmptyMovieClip(name,
  this.getNextHighestDepth());
  mc.beginFill(bgColor);
  mc.lineTo(w, 0);
  mc.lineTo(w, h);
  mc.lineTo(0, h);
  mc.lineTo(0, 0);
  mc._x = 20;
  mc._y = 20;
  return mc;
}
```

blurX (propiedad BlurFilter.blurX)

```
public blurX : Number
```

Cantidad de desenfoque horizontal. Los valores válidos son de 0 a 255 (coma flotante). El valor predeterminado es 4. Los valores que son múltiplos de 2 (como 2, 4, 8, 16 y 32) se optimizan para representarse más rápidamente que otros valores.

Disponibilidad: ActionScript 1.0; Flash Player 8

Ejemplo

En el ejemplo siguiente se cambia la propiedad `blurX` de una instancia MovieClip existente cuando un usuario hace clic en ella:

```
import flash.filters.BlurFilter;
var mc:MovieClip = createBlurFilterRectangle("BlurFilterBlurX");
mc.onRelease = function() {
  var filter:BlurFilter = this.filters[0];
  filter.blurX = 200;
  this.filters = new Array(filter);
}

function createBlurFilterRectangle(name:String):MovieClip {
  var rect:MovieClip = this.createEmptyMovieClip(name,
  this.getNextHighestDepth());
  var w:Number = 100;
```

```
  var h:Number = 100;
  rect.beginFill(0x003366);
  rect.lineTo(w, 0);
  rect.lineTo(w, h);
  rect.lineTo(0, h);
  rect.lineTo(0, 0);
  rect._x = 20;
  rect._y = 20;

  var filter:BlurFilter = new BlurFilter(30, 30, 2);
  var filterArray:Array = new Array();
  filterArray.push(filter);
  rect.filters = filterArray;
  return rect;
}
```

blurY (propiedad BlurFilter.blurY)

```
public blurY : Number
```

Cantidad de desenfoque vertical. Los valores válidos son de 0 a 255 (coma flotante). El valor predeterminado es 4. Los valores que son múltiplos de 2 (como 2, 4, 8, 16 y 32) se optimizan para representarse más rápidamente que otros valores.

Disponibilidad: ActionScript 1.0; Flash Player 8

Ejemplo

En el ejemplo siguiente se cambia la propiedad `blurY` de una instancia MovieClip existente cuando un usuario hace clic en ella:

```
import flash.filters.BlurFilter;
var mc:MovieClip = createBlurFilterRectangle("BlurFilterBlurY");
mc.onRelease = function() {
  var filter:BlurFilter = this.filters[0];
  filter.blurY = 200;
  this.filters = new Array(filter);
}

function createBlurFilterRectangle(name:String):MovieClip {
  var rect:MovieClip = this.createEmptyMovieClip(name,
  this.getNextHighestDepth());
  var w:Number = 100;
  var h:Number = 100;
  rect.beginFill(0x003366);
  rect.lineTo(w, 0);
  rect.lineTo(w, h);
  rect.lineTo(0, h);
  rect.lineTo(0, 0);
  rect._x = 20;
```

```
  rect._y = 20;

  var filter:BlurFilter = new BlurFilter(30, 30, 2);
  var filterArray:Array = new Array();
  filterArray.push(filter);
  rect.filters = filterArray;
  return rect;
}
```

clone (método BlurFilter.clone)

```
public clone() : BlurFilter
```

Devuelve una copia de este objeto de filtro.

Disponibilidad: ActionScript 1.0; Flash Player 8

Valor devuelto

`flash.filters.BlurFilter` - BitmapFilter Instancia BlurFilter nueva con las mismas propiedades que la instancia BlurFilter original.

Ejemplo

En el ejemplo siguiente se crean y se comparan tres objetos BlurFilter. Puede crear el objeto `filter_1` utilizando el constructor `BlurFilter`. Para crear el objeto `filter_2`, defínalo como igual que `filter_1`. Para crear `clonedFilter`, copie `filter_1`. Observe que mientras que `filter_2` se evalúa como igual que `filter_1`, no ocurre igual con `clonedFilter`, aunque contiene el mismo valor que `filter_1`.

```
import flash.filters.BlurFilter;

var filter_1:BlurFilter = new BlurFilter(30, 30, 2);
var filter_2:BlurFilter = filter_1;
var clonedFilter:BlurFilter = filter_1.clone();

trace(filter_1 == filter_2); // true
trace(filter_1 == clonedFilter); // false

for(var i in filter_1) {
  trace(">> " + i + ": " + filter_1[i]);
  // >> clone: [type Function]
  // >> quality: 2
  // >> blurY: 30
  // >> blurX: 30
}

for(var i in clonedFilter) {
  trace(">> " + i + ": " + clonedFilter[i]);
```

```
  // >> clone: [type Function]
  // >> quality: 2
  // >> blurY: 30
  // >> blurX: 30
}
```

Para demostrar mejor la relación entre `filter_1`, `filter_2` y `clonedFilter`, en el ejemplo siguiente se modifica la propiedad `quality` de `filter_1`. La modificación de `quality` demuestra que el método `clone()` crea una instancia basada en valores de `filter_1` en lugar de hacer referencia a los valores.

```
import flash.filters.BlurFilter;

var filter_1:BlurFilter = new BlurFilter(30, 30, 2);
var filter_2:BlurFilter = filter_1;
var clonedFilter:BlurFilter = filter_1.clone();

trace(filter_1.quality); // 2
trace(filter_2.quality); // 2
trace(clonedFilter.quality); // 2

filter_1.quality = 1;

trace(filter_1.quality); // 1
trace(filter_2.quality); // 1
trace(clonedFilter.quality); // 2
```

quality (propiedad BlurFilter.quality)

```
public quality : Number
```

Número de veces que debe realizarse el desenfoque. Los valores válidos son del 0 al 15. El valor predeterminado es 1, que equivale a la calidad baja. El valor 2 equivale a la calidad media. Un valor de 3 equivale a la calidad alta y se aproxima a un desenfoque gaussiano.

Para la mayoría de las aplicaciones, es suficiente un valor `quality` de 1, 2 o 3. Aunque puede utilizar valores numéricos hasta 15 para aumentar el número de veces que se aplica el desenfoque, y obtener así un mayor efecto de desenfoque, tenga en cuenta que los valores mayores se representan más lentamente. En lugar de aumentar el valor de `quality`, habitualmente se consigue un efecto similar, y con una representación más rápida, sencillamente aumentando los valores de `blurX` y `blurY`.

Disponibilidad: ActionScript 1.0; Flash Player 8

Ejemplo

En el ejemplo siguiente se crea un rectángulo y se aplica al mismo un filtro de desenfoque con un valor de calidad (`quality`) de 1. Cuando hace clic en el rectángulo, `quality` aumenta a 3, y el rectángulo se hace más borroso.

```
import flash.filters.BlurFilter;
var mc:MovieClip = createBlurFilterRectangle("BlurFilterQuality");
mc.onRelease = function() {
  var filter:BlurFilter = this.filters[0];
  filter.quality = 3;
  this.filters = new Array(filter);
}

function createBlurFilterRectangle(name:String):MovieClip {
  var rect:MovieClip = this.createEmptyMovieClip(name,
  this.getNextHighestDepth());
  var w:Number = 100;
  var h:Number = 100;
  rect.beginFill(0x003366);
  rect.lineTo(w, 0);
  rect.lineTo(w, h);
  rect.lineTo(0, h);
  rect.lineTo(0, 0);
  rect._x = 20;
  rect._y = 20;

  var filter:BlurFilter = new BlurFilter(30, 30, 1);
  var filterArray:Array = new Array();
  filterArray.push(filter);
  rect.filters = filterArray;
  return rect;
}
```

Boolean

```
Object
  |
  +-Boolean
```

```
public class Boolean
extends Object
```

La clase Boolean es un envolvente con la misma funcionalidad que el objeto Boolean estándar de JavaScript. Utilice la clase Boolean para recuperar el tipo de datos simple o la representación de cadena de un objeto booleano.

Debe utilizar el constructor new `Boolean()` para crear un objeto booleano antes de llamar a sus métodos.

Disponibilidad: ActionScript 1.0; Flash Player 5

Resumen de propiedades

Propiedades heredadas de la clase Object

`constructor (propiedad Object.constructor), __proto__ (propiedad Object.__proto__), prototype (propiedad Object.prototype), __resolve (propiedad Object.__resolve)`

Resumen de constructores

Firma	Descripción
`Boolean([value:Object])`	Crea un objeto Boolean.

Resumen de métodos

Modificadores	Firma	Descripción
	`toString() : String`	Devuelve la representación de cadena (`"true"` o `"false"`) del objeto Boolean.
	`valueOf() : Boolean`	Devuelve `true` si el tipo de valor simple del objeto Boolean especificado es true; en cualquier otro caso, devuelve `false`

Métodos heredados de la clase Object

`addProperty (método Object.addProperty), hasOwnProperty (método Object.hasOwnProperty), isPropertyEnumerable (método Object.isPropertyEnumerable), isPrototypeOf (método Object.isPrototypeOf), registerClass (método Object.registerClass), toString (método Object.toString), unwatch (método Object.unwatch), valueOf (método Object.valueOf), watch (método Object.watch)`

Constructor Boolean

```
public Boolean([value:Object])
```

Crea un objeto Boolean. Si omite el parámetro `value`, el objeto Boolean se inicializa con el valor `false`. Si especifica un valor para el parámetro `value`, el método lo evaluará y devolverá el resultado como un valor booleano conforme a las reglas de la función `Boolean()`.

Disponibilidad: ActionScript 1.0; Flash Player 5

Parámetros

`value``:Object` [opcional] - Cualquier expresión. El valor predeterminado es `false`.

Ejemplo

El código siguiente crea un nuevo objeto Boolean vacío llamado `myBoolean`:

```
var myBoolean:Boolean = new Boolean();
```

toString (método Boolean.toString)

```
public toString() : String
```

Devuelve la representación de cadena (`"true"` o `"false"`) del objeto Boolean.

Disponibilidad: ActionScript 1.0; Flash Player 5

Valor devuelto

`String` - Una cadena; `"true"` o `"false"`.

Ejemplo

Este ejemplo crea una variable de tipo booleano y utiliza `toString()` para convertir el valor en una cadena para utilizarla en la sentencia trace:

```
var myBool:Boolean = true;
trace("The value of the Boolean myBool is: " + myBool.toString());
myBool = false;
trace("The value of the Boolean myBool is: " + myBool.toString());
```

valueOf (método Boolean.valueOf)

```
public valueOf() : Boolean
```

Devuelve `true` si el tipo de valor simple del objeto Boolean especificado es true; en cualquier otro caso, devuelve `false`

Disponibilidad: ActionScript 1.0; Flash Player 5

Valor devuelto

`Boolean` - Valor booleano.

Ejemplo

El ejemplo siguiente muestra cómo funciona este método y también que el tipo de valor simple de un objeto Boolean nuevo es `false`:

```
var x:Boolean = new Boolean();
trace(x.valueOf());   // false
x = (6==3+3);
trace(x.valueOf());   // true
```

Button

```
Object
  |
  +-Button
```

```
public class Button
extends Object
```

Todos los símbolos de botón de un archivo SWF son instancias del objeto `Button`. Puede asignar a un botón un nombre de instancia en el inspector de propiedades y utilizar los métodos y propiedades de la clase Button para manipular botones con ActionScript. Los nombres de instancias de Button se muestran en el explorador de películas y en el cuadro de diálogo Insertar ruta de destino del panel Acciones.

La clase Button hereda de la clase Object.

Disponibilidad: ActionScript 1.0; Flash Player 6

Véase también

`Object`

Resumen de propiedades

Modificadores	Propiedad	Descripción
	`_alpha:Number`	Valor de transparencia alfa del botón especificado por `my_btn`.
	`blendMode:Object`	Modo de mezcla de este botón.
	`cacheAsBitmap:Boolean`	Si se define como `true`, Flash Player deja en caché una representación de mapa de bits interno del botón.
	`enabled:Boolean`	Un valor booleano que especifica si está activado un botón.
	`filters:Array`	Matriz indexada que contiene todos los objetos de filtro actualmente asociados con el botón.
	`_focusrect:Boolean`	Valor booleano que especifica si un botón debe mostrar un rectángulo amarillo a su alrededor cuando se selecciona con el teclado.
	`_height:Number`	Altura del botón, expresada en píxeles.
	`_highquality:Number`	*Desfasada* desde Flash Player 7. Esta propiedad está desfasada y en su lugar debe utilizarse `Button._quality`. Especifica el nivel de visualización suavizada que se aplica al archivo SWF actual.
	`menu:ContextMenu`	Asocia el objeto `ContextMenu` `contextMenu` al objeto de botón `my_button`.
	`_name:String`	Nombre de instancia del botón especificado por `my_btn`.
	`_parent:MovieClip`	Referencia al clip de película u objeto que contiene el clip de película u objeto actual.
	`_quality:String`	Propiedad (global); establece o recupera la calidad de representación utilizada por un archivo SWF.
	`_rotation:Number`	Giro del botón, expresado en grados, con respecto a su orientación original.
	`scale9Grid:Rectangle`	Región rectangular que define las nueve regiones de escala del botón.
	`_soundbuftime:Number`	La propiedad que especifica el número de segundos que un sonido debe acumularse previamente en el búfer antes de que comience a reproducirse sin interrupción.

Modificadores	Propiedad	Descripción
	`tabEnabled:Boolean`	Especifica si `my_btn` está incluido en el orden de tabulación automático.
	`tabIndex:Number`	Permite personalizar el orden de tabulación de los objetos de un archivo SWF.
	`_target:String` [read-only]	Devuelve la ruta de destino de la instancia de botón especificada por `my_btn`.
	`trackAsMenu:Boolean`	Valor booleano que indica si otros botones o clips de película pueden recibir eventos de liberación del botón del ratón.
	`_url:String` [read-only]	Recupera la URL del archivo SWF que creó el botón.
	`useHandCursor:Boolean`	Valor booleano que, cuando se establece como `true` (configuración predeterminada), indica si debe aparecer una mano que señala (cursor de mano) cuando el ratón pasa por encima de un botón.
	`_visible:Boolean`	Valor booleano que indica si el botón especificado por `my_btn` es visible.
	`_width:Number`	Anchura del botón, expresada en píxeles.
	`_x:Number`	Entero que establece la coordenada x de un botón como relativa a las coordenadas locales del clip de película principal.
	`_xmouse:Number` [read-only]	Devuelve la coordenada *x* de la posición del ratón relativa al botón.
	`_xscale:Number`	Escala horizontal del botón aplicada desde el punto de registro del botón, expresada como porcentaje.
	`_y:Number`	Coordenada *y* del botón relativa a las coordenadas locales del clip de película principal.
	`_ymouse:Number` [read-only]	Indica la coordenada *y* de la posición del ratón relativa al botón.
	`_yscale:Number`	Escala vertical del botón aplicada desde el punto de registro del botón, expresada como porcentaje.

Propiedades heredadas de la clase Object

```
constructor (propiedad Object.constructor), __proto__ (propiedad
Object.__proto__), prototype (propiedad Object.prototype), __resolve
(propiedad Object.__resolve)
```

Resumen de eventos

Evento	Descripción
`onDragOut = function() {}`	Se invoca cuando se hace clic con el botón del ratón sobre el botón y luego se arrastra el puntero hacia el exterior del botón.
`onDragOver = function() {}`	Se invoca cuando el usuario presiona y arrastra el botón del ratón fuera del botón y, seguidamente, por encima de éste.
`onKeyDown = function() {}`	Se invoca cuando un botón se ha seleccionado con el teclado y se presiona una tecla.
`onKeyUp = function() {}`	Se invoca cuando un botón tiene la selección de entrada y se suelta una tecla.
`onKillFocus = function(newFocus:Object) {}`	Se invoca cuando un botón deja de estar seleccionado con el teclado.
`onPress = function() {}`	Se invoca cuando se presiona una tecla.
`onRelease = function() {}`	Se invoca cuando se suelta un botón.
`onReleaseOutside = function() {}`	Se invoca cuando se suelta el botón del ratón mientras el puntero está fuera del botón después de haber hecho clic mientras el puntero estaba dentro del botón.
`onRollOut = function() {}`	Se invoca cuando el puntero se desplaza fuera del área de un botón.
`onRollOver = function() {}`	Se invoca cuando el puntero se desplaza sobre el área de un botón.
`onSetFocus = function(oldFocus:Object) {}`	Se invoca cuando un botón queda seleccionado con el teclado.

Resumen de métodos

Modificadores	Firma	Descripción
	`getDepth() : Number`	Devuelve la profundidad de una instancia de botón.

addProperty (método Object.addProperty), hasOwnProperty (método Object.hasOwnProperty), isPropertyEnumerable (método Object.isPropertyEnumerable), isPrototypeOf (método Object.isPrototypeOf), registerClass (método Object.registerClass), toString (método Object.toString), unwatch (método Object.unwatch), valueOf (método Object.valueOf), watch (método Object.watch)

_alpha (propiedad Button._alpha)

```
public _alpha : Number
```

Valor de transparencia alfa del botón especificado por `my_btn`. Los valores válidos son los comprendidos entre 0 (totalmente transparente) y 100 (totalmente opaco). El valor predeterminado es 100. Los objetos existentes en un botón que tenga configurado `_alpha` con el valor 0 continuarán activos aunque no sean visibles.

Disponibilidad: ActionScript 1.0; Flash Player 6

Ejemplo

El código siguiente establece la `_alpha` de un botó llamado myBtn_btn al 50% cuando el usuario hace clic en el botón. En primer lugar añada una instancia Button al escenario. A continuación, asigne a la instancia el nombre myBtn_btn. Por último, con el fotograma 1 seleccionado, introduzca el código siguiente en el panel Acciones:

```
myBtn_btn.onRelease = function(){
  this._alpha = 50;
};
```

Véase también

_alpha (propiedad MovieClip._alpha), _alpha (propiedad TextField._alpha)

blendMode (propiedad Button.blendMode)

`public blendMode : Object`

Modo de mezcla de este botón. El modo de mezcla afecta al aspecto del botón cuando se encuentra en una capa encima de otro objeto de la pantalla.

Flash Player aplica la propiedad `blendMode` a cada píxel del botón. Cada píxel consta de los tres colores primarios (o rojo, verde y azul). Cada color primario tiene un valor comprendido entre 0x00 y 0xFF. Flash Player compara cada color primario de un píxel del botón con el color correspondiente del píxel en el fondo. Por ejemplo, si `blendMode` se define como `"lighten"`, Flash Player compara el valor del rojo botón con el valor del rojo del fondo y utiliza el color más claro de los dos como valor del componente rojo para el color de visualización.

La tabla siguiente describe los valores de `blendMode`: Para definir la propiedad `blendMode`, puede emplear un entero del 1 al 14 o una cadena. Las ilustraciones de la tabla muestran los valores `blendMode` aplicados a un botón (2) superpuesto en otro objeto (1) de la pantalla.

Valor entero	Valor de la cadena	Ilustración	Descripción
1	`"normal"`		El botón aparece delante del fondo. Los valores de píxel del botón anulan a los del fondo. Cuando el botón sea transparente, el fondo será visible.
2	`"layer"`		Fuerza la creación de un búfer temporal para la composición previa del botón. Esto se hace automáticamente si hay más de un objeto secundario en un botón y se ha seleccionado un modo `blendMode` diferente de `"normal"` para el objeto secundario.

Valor entero	Valor de la cadena	Ilustración	Descripción
3	`"multiply"`		Multiplica los valores de los colores primarios del botón por los del color del fondo, a continuación, normaliza el resultado dividiéndolo por 0xFF, con esto se obtienen unos colores más oscuros. Este valor se suele utilizar para efectos de sombras y profundidad. Por ejemplo, si un color primario (como el rojo) de un píxel del botón y el color correspondiente del píxel del fondo tienen ambos el valor 0x88, el resultado de la multiplicación será 0x4840. Al dividir por 0xFF el resultado es 0x48 para dicho color primario, consiguiéndose un tono más oscuro que el del botón o el del fondo.
4	`"screen"`		Multiplica el complemento (inverso) del color del botón por el complemento del color de fondo, con lo que se obtiene un efecto de decoloración. Este valor se suele utilizar para los resaltes o para eliminar áreas de color negro del botón.
5	`"lighten"`		Selecciona el color primario más claro del botón y los del fondo (los que tengan valores mayores). Este valor suele emplearse para el tipo de superposición. Por ejemplo, si el botón tiene un píxel con un valor RGB de 0xFFCC33, y el píxel del fondo tiene un valor RGB de 0xDDF800, el valor RGB resultante para el píxel mostrado es de 0xFFF833 (debido a que 0xFF > 0xDD, 0xCC < 0xF8 y 0x33 > 0x00 = 33).

Valor entero	Valor de la cadena	Ilustración	Descripción
6	`"darken"`		Selecciona el color primario más oscuro del botón y los del fondo (los que tengan valores menores). Este valor suele emplearse para el tipo de superposición. Por ejemplo, si el botón tiene un píxel con un valor RGB de 0xFFCC33, y el píxel del fondo tiene un valor RGB de 0xDDF800, el valor RGB resultante para el píxel mostrado es de 0xDDCC00 (debido a que 0xFF > 0xDD, 0xCC < 0xF8 y 0x33 > 0x00 = 33).
7	`"difference"`		Compara los colores primarios del botón con los del fondo y resta el valor más oscuro del valor más claro de los colores primarios. Este valor suele emplearse en los colores más vivos. Por ejemplo, si el botón tiene un píxel con un valor RGB de 0xFFCC33, y el píxel del fondo tiene un valor RGB de 0xDDF800, el valor RGB resultante para el píxel mostrado es de 0x222C33 (debido a que 0xFF - 0xDD = 0x22, 0xF8 - 0xCC = 0x2C y 0x33 - 0x00 = 0x33).
8	`"add"`		Agrega los valores de los colores primarios del botón a los del fondo con un límite de 0xFF. Este valor suele emplearse para animar una disolución de aclarado entre dos objetos. Por ejemplo, si el botón tiene un píxel con un valor RGB de 0xAAA633, y el píxel del fondo tiene un valor RGB de 0xDD2200, el valor RGB resultante para el píxel mostrado es de 0xFFC833 (debido a que 0xAA + 0xDD > 0xFF, 0xA6 + 0x22 = 0xC8 y 0x33 + 0x00 = 0x33).

Valor entero	Valor de la cadena	Ilustración	Descripción
9	`"subtract"`		Resta los valores de los colores primarios del botón a los del fondo, aplicando un límite inferior de 0. Suele utilizarse para animar una disolución de oscurecimiento entre dos imágenes. Por ejemplo, si el botón tiene un píxel con un valor RGB de 0xAA2233, y el píxel del fondo tiene un valor RGB de 0xDDA600, el valor RGB resultante para el píxel mostrado es de 0x338400 (debido a que 0xDD - 0xAA = 0x33, 0xA6 - 0x22 = 0x84 y 0x00 - 0x33 < 0x00).
10	`"invert"`		Invierte el fondo.
11	`"alpha"`		Aplica el valor alfa de cada píxel del botón al fondo. Esto requiere que se aplique `blendMode` de "capa" al botón principal. Por ejemplo, en la ilustración, el botón principal, un fondo blanco, tiene `blendMode = "layer"`.
12	`"erase"`		Borra el fondo en función del valor alfa del botón. Esto requiere que se aplique `"layer" blendMode` al botón principal. Por ejemplo, en la ilustración, el botón principal, un fondo blanco, tiene `blendMode = "layer"`.

Valor entero	Valor de la cadena	Ilustración	Descripción
13	`"overlay"`		Ajusta el color de cada mapa de bits en función de la oscuridad del fondo. Si el fondo es más claro que un 50% de gris, los colores del botón y del fondo se tamizan, consiguiéndose un color más claro. Si el fondo es más oscuro que un 50% de gris, los colores del clip de película y del fondo se multiplican, consiguiéndose un color más oscuro. Este valor suele emplearse para conseguir efectos de sombreado.
14	`"hardlight"`		Ajusta el color de cada mapa de bits en función de la oscuridad del fondo. Si el botón es más claro que un 50% de gris, los colores del botón y del fondo se tamizan, consiguiéndose un color más claro. Si el botón es más oscuro que un 50% de gris, los colores se multiplican, consiguiéndose un color más oscuro. Este valor suele emplearse para conseguir efectos de sombreado.

Si intenta establecer la propiedad `blendMode` en cualquier otro valor, Flash Player la establece en `"normal"`.

Disponibilidad: ActionScript 1.0; Flash Player 8

Ejemplo

En los siguientes ejemplos puede ver que si define la propiedad en un entero, Flash Player convierte el valor en la versión de cadena correspondiente:

```
my_button.blendMode = 8;
trace (my_button.blendMode) // add
```

Para ver un ejemplo relacionado, consulte la descripción de la propiedad `blendMode` de la clase MovieClip.

Véase también

`blendMode (propiedad MovieClip.blendMode)`

cacheAsBitmap (propiedad Button.cacheAsBitmap)

`public cacheAsBitmap : Boolean`

Si se define como `true`, Flash Player deja en caché una representación de mapa de bits interno del botón. Esto puede aumentar el rendimiento de los botones con contenido vectorial complejo.

Para un botón que tiene `cacheAsBitmap` definido como `true`, Flash Player almacena una representación de mapa de bits para cada uno de los cuatro estados de botón.

Todos los datos vectoriales de un botón con un mapa de bits en caché se dibujan en el mapa de bits, no en el escenario. A continuación, el mapa de bits se copia en el escenario principal como píxeles no expandidos ni rotados, pero ajustados a los límites del píxel más cercano. Los píxeles se asignan de uno a uno con el objeto principal. Si los límites del mapa de bits cambian, el mapa de bits se vuelve a crear en lugar de expandirse.

No se crea ningún mapa de bits interno a menos que la propiedad `cacheAsBitmap` esté definida en `true`.

Después de definir la propiedad `cacheAsBitmap` de un botón como `true`, la representación no varía, pero el botón realiza automáticamente el ajuste en píxeles. La velocidad de animación puede ser notablemente mayor según la complejidad del contenido vectorial.

La propiedad `cacheAsBitmap` se establece automáticamente en `true` siempre que se aplica un filtro a un botón (cuando su matriz `filter` no está vacía) y, si un botón tiene aplicado un filtro, `cacheAsBitmap` será `true` para ese botón, aun cuando la propiedad se haya configurado como `false`. Si borra todos los filtros para un botón, el valor `cacheAsBitmap` cambia a su última configuración..

En los casos siguientes y aunque la propiedad `cacheAsBitmap` se haya definido como `true`, un botón no utiliza un mapa de bits sino que se representa a partir de datos vectoriales:

- Cuando el mapa de bits es excesivamente grande, es decir, tiene más de 2880 píxeles en cualquier dirección..
- Cuando no se puede asignar memoria para el mapa de bits (se produce un error de memoria insuficiente).

La propiedad `cacheAsBitmap` es ideal con botones que tienen sobre todo contenido estático y no cambian de escala ni giran con frecuencia. Con estos botones, `cacheAsBitmap` puede aumentar el rendimiento cuando se convierte el botón (cuando se modifica la posición de *x* e *y*).

Disponibilidad: ActionScript 1.0; Flash Player 8

Ejemplo

El ejemplo siguiente añade una sombra a una instancia Button, llamada `myButton`. Después, averigua el valor de `cacheAsBitmap` que se define como `true` cuando se aplica un filtro.

```
import flash.filters.DropShadowFilter;
trace(myButton.cacheAsBitmap); // false
var dropShadow:DropShadowFilter = new DropShadowFilter(6, 45, 0x000000, 50,
  5, 5, 1, 2, false, false, false);
myButton.filters = new Array(dropShadow);
trace(myButton.cacheAsBitmap); // true
```

enabled (propiedad Button.enabled)

```
public enabled : Boolean
```

Un valor booleano que especifica si está activado un botón. Cuando un botón está desactivado (la propiedad enabled está configurada como `false`), el botón está visible pero no se puede hacer clic en él. El valor predeterminado es `true`. Esta propiedad resulta útil si desea desactivar parte de la navegación; por ejemplo, puede que desee desactivar un botón de la página mostrada actualmente para que no se pueda hacer clic en él y no pueda volver a cargarse la página.

Disponibilidad: ActionScript 1.0; Flash Player 6

Ejemplo

El ejemplo siguiente demuestra cómo desactivar y activar botones para hacer clic en ellos. Dos botones, `myBtn1_btn` y `myBtn2_btn`,, se encuentran en el escenario y se añade el código ActionScript siguiente de modo que no se pueda hacer clic en el botón `myBtn2_btn`: En primer lugar añada dos instancias Button al escenario. A continuación, asigne a las instancias los nombres `myBtn1_btn` y `myBtn2_btn`. Por último, introduzca el siguiente código en el fotograma 1 para activar o desactivar los botones.

```
myBtn1_btn.enabled = true;
myBtn2_btn.enabled = false;

//button code
// the following function will not get called
// because myBtn2_btn.enabled was set to false
myBtn1_btn.onRelease = function() {
  trace( "you clicked : " + this._name );
};
myBtn2_btn.onRelease = function() {
  trace( "you clicked : " + this._name );
};
```

filters (propiedad Button.filters)

```
public filters : Array
```

Matriz indexada que contiene todos los objetos de filtro actualmente asociados con el botón. El paquete `flash.filters` contiene varias clases que definen filtros específicos que puede utilizar.

Los filtros pueden aplicarse en tiempo de diseño en la herramienta del entorno de edición de Flash o en tiempo de ejecución con código ActionScript. Para aplicar un filtro con ActionScript, debe realizar una copia temporal de toda la matriz `Button.filters`, modificar la matriz temporal y volver a asignar el valor de la matriz temporal a la matriz `Button.filters`. A la matriz `Button.filters` no se le pueden añadir directamente objetos filter nuevos. El código siguiente no afecta al botón de destino, denominado `myButton`:

```
myButton.filters[0].push(myDropShadow);
```

Para añadir un filtro con ActionScript, debe seguir estos pasos (suponiendo que el botón de destino se denomina `myButton`):

- Cree un nuevo objeto filter con la función constructora de la clase de filtro que haya elegido.
- Asigne el valor de la matriz `myButton.filters` a una matriz temporal, denominada por ejemplo `myFilters`.
- Añada el nuevo objeto filter a la matriz temporal, `myFilters`.
- Asigne el valor de la matriz temporal a la matriz `myButton.filters`.

Si la matriz `filters` está vacía, no tendrá que utilizar una matriz temporal. En lugar de ello, podrá asignar directamente un literal de matriz que contenga uno o varios objetos filter que haya creado.

Para modificar un objeto filter existente, creado tanto en tiempo de diseño o como en tiempo de ejecución, debe utilizar la técnica de modificar una copia de la matriz `filters`:

- Asigne el valor de la matriz `myButton.filters` a una matriz temporal, denominada por ejemplo `myFilters`.
- Modifique la propiedad utilizando la matriz temporal, `myFilters`. Por ejemplo, si desea establecer la propiedad `quality` del primer filtro de la matriz, podría utilizar el código siguiente: `myList[0].quality = 1;`
- Asigne el valor de la matriz temporal a la matriz `myButton.filters`.

Para borrar los filtros de botón, defina `filters` con una matriz vacía (`[]`).

Durante la carga, si un botón tiene un filtro asociado, se marca para guardarlo en caché como mapa de bits transparente. Desde ese momento, siempre y cuando el botón tenga una lista de filtros válida, el reproductor guardará el botón en caché como mapa de bits. Este mapa de bits de origen se utiliza como imagen de origen para los efectos de filtro. Cada botón tienen normalmente dos conjuntos de mapas de bits: uno con el botón de origen sin filtrar original y otro para las imágenes finales (en cada uno de los cuatro estados de botón) tras filtrar. La imagen final es la que se utiliza en la representación. Mientras el botón no cambie, la imagen final no necesita actualización.

Si está trabajando con una matriz `filters` que contiene varios filtros y necesita averiguar el tipo de filtro asignado a cada índice de matriz, puede conservar su propia matriz `filters` y utilizar una estructura de datos aparte para averiguar el tipo de filtro asociado a cada índice de la matriz. No hay ninguna forma sencilla de determinar el tipo de filtro asociado a cada índice de la matriz `filters`.

Disponibilidad: ActionScript 1.0; Flash Player 8

Ejemplo

El ejemplo siguiente añade un filtro de sombra a un botón denominado `myButton`:

```
import flash.filters.DropShadowFilter;
var myDropFilter:DropShadowFilter = new DropShadowFilter(6, 45, 0x000000,
  50, 5, 5, 1, 2, false, false, false);
var myFilters:Array = myButton.filters;
myFilters.push(myDropFilter);
myButton.filters = myFilters;
```

El ejemplo siguiente cambia la configuración de `quality` del primer filtro de la matriz a 15 (este ejemplo sólo funciona si el campo de texto `myButton` tiene asociado como mínimo un objeto filter):

```
var myList:Array = myButton.filters;
myList[0].quality = 15;
myButton.filters = myList;
```

Véase también

, `cacheAsBitmap (propiedad Button.cacheAsBitmap)`

_focusrect (propiedad Button._focusrect)

```
public _focusrect : Boolean
```

Valor booleano que especifica si un botón debe mostrar un rectángulo amarillo a su alrededor cuando se selecciona con el teclado. Esta propiedad puede sustituir a la propiedad global `_focusrect`. De manera predeterminada, la propiedad `_focusrect` de una instancia de botón es nula (null), lo que significa que la instancia de botón no sustituye a la propiedad global `_focusrect`. Si la propiedad `_focusrect` de una instancia de botón se establece como `true` o `false`, ésta sustituirá a la configuración de la propiedad global `_focusrect` para la instancia de botón en cuestión.

En archivos SWF de Flash Player 4 o Flash Player 5, la propiedad `_focusrect` controla la propiedad global `_focusrect`. Se trata de un valor booleano. Este comportamiento cambió en Flash Player 6 y versiones posteriores para permitir la personalización de la propiedad `_focusrect` en un clip de película concreto.

Si la propiedad `_focusrect` se estable como `false`, la navegación con el teclado para ese botón quedará limitada a la tecla Tabulador. Todas las demás teclas, incluida Intro y las teclas de flecha, quedan anuladas. Para restablecer el desplazamiento completo con el teclado, es preciso configurar `_focusrect` con el valor `true`.

Disponibilidad: ActionScript 1.0; Flash Player 6

Ejemplo

Este ejemplo muestra cómo ocultar el rectángulo amarillo alrededor de una instancia de botón especificada en un archivo SWF cuando está seleccionada en una ventana del navegador. Cree tres botones llamados `myBtn1_btn`, `myBtn2_btn` y `myBtn3_btn`, y añada el código ActionScript siguiente al fotograma 1 de la línea de tiempo:

```
myBtn2_btn._focusrect = false;
```

Cambie la configuración de publicación a Flash Player 6 y compruebe el archivo SWF en una ventana del navegador seleccionando Archivo> Vista previa de publicación > HTML. Seleccione el SWF haciendo clic en él en la ventana del navegador y utilice la tecla Tabulador para seleccionar cada instancia. No podrá ejecutar el código de este botón pulsando Intro o la barra espaciadora cuando `_focusrect` esté desactivado.

getDepth (método Button.getDepth)

```
public getDepth() : Number
```

Devuelve la profundidad de una instancia de botón.

Cada clip de película, botón y campo de texto tiene asociada una profundidad exclusiva que determina cómo aparece el objeto delante o detrás de otros objetos. Los objetos con mayor profundidad aparecen delante.

Disponibilidad: ActionScript 1.0; Flash Player 6

Valor devuelto

`Number` - La profundidad de la instancia de botón.

Ejemplo

Si crea `myBtn1_btn` y `myBtn2_btn` en el escenario, podrá realizar un seguimiento de su profundidad empleando el código ActionScript siguiente:

```
trace(myBtn1_btn.getDepth());
trace(myBtn2_btn.getDepth());
```

Si carga un archivo SWF llamado buttonMovie.swf en este documento, podrá realizar un seguimiento de la profundidad de un botón, `myBtn4_btn`, dentro de ese archivo SWF empleando otro botón del SWF principal:

```
this.createEmptyMovieClip("myClip_mc", 999);
myClip_mc.loadMovie("buttonMovie.swf");
myBtn3_btn.onRelease = function(){
  trace(myClip_mc.myBtn4_btn.getDepth());
};
```

Quizá haya observado que dos de estos botones tienen el mismo valor de profundidad, uno en el archivo SWF principal y otro en el archivo SWF cargado. Esto resulta engañoso, porque buttonMovie.swf se cargó con una profundidad de 999, lo que significa que el botón que contiene también tendrá una profundidad de 999 en relación con los botones del archivo SWF principal. Tenga en cuenta que cada clip de película tiene su propio orden z interno, lo que significa que cada clip de película tiene su propio conjunto de valores de profundidad. Los dos botones pueden tener el mismo valor de profundidad, pero los valores sólo tienen significado en relación con otros objetos del mismo orden z. En este caso, los botones tienen el mismo valor de profundidad, pero los valores hacen referencia a clips de película distintos. Por ejemplo, el valor de profundidad del botón del archivo SWF principal hace referencia al orden z de la línea de tiempo principal, mientras que el valor de profundidad del botón del archivo SWF cargado se refiere al orden z interno del clip de película `myClip_mc`.

Véase también

`getDepth (método MovieClip.getDepth)`, `getDepth (método TextField.getDepth)`,

_height (propiedad Button._height)

```
public _height : Number
```

Altura del botón, expresada en píxeles.

Disponibilidad: ActionScript 1.0; Flash Player 6

Ejemplo

El ejemplo siguiente establece la altura y anchura de un botón llamado `my_btn` con unos valores especificados.

```
my_btn._width = 500;
my_btn._height = 200;
```

_highquality (propiedad Button._highquality)

```
public _highquality : Number
```

Desfasada desde Flash Player 7. Esta propiedad está desfasada y en su lugar debe utilizarse `Button._quality`.

Especifica el nivel de visualización suavizada que se aplica al archivo SWF actual. Especifique 2 (mejor calidad) para aplicar alta calidad con el suavizado de mapa de bits siempre activado. Especifique 1 (alta calidad) para aplicar la visualización suavizada, que suaviza los mapas de bits si el archivo SWF no contiene animación y es el valor predeterminado. Especifique 0 (baja calidad) para evitar el suavizado.

Disponibilidad: ActionScript 1.0; Flash Player 6

Ejemplo

Añada una instancia de botón al escenario y asígnele el nombre myBtn_btn. Dibuje un óvalo en el escenario utilizando la herramienta Óvalo con un color de trazo y relleno. Seleccione el fotograma 1 y añada el código ActionScript siguiente a través del panel Acciones:

```
myBtn_btn.onRelease = function() {
 myBtn_btn._highquality = 0;
};
```

Cuando hace clic en myBtn_btn, el trazo del círculo adquiere un aspecto dentado. En su lugar puede añadir el código ActionScript siguiente para que afecte el SWF globalmente:

```
_quality = 0;
```

Véase también

_quality (propiedad Button._quality), Propiedad _quality

menu (propiedad Button.menu)

```
public menu : ContextMenu
```

Asocia el objeto `ContextMenu` `contextMenu` al objeto de botón `my_button`. La clase ContextMenu le permite modificar el menú contextual que se muestra cuando un usuario hace clic con el botón derecho (Windows) o hace clic mientras presiona la tecla Control (Macintosh) en Flash Player.

Disponibilidad: ActionScript 1.0; Flash Player 7

Ejemplo

El ejemplo siguiente asigna un objeto ContextMenu a una instancia de botón llamada `myBtn_btn`. El objeto ContextMenu contiene un único elemento de menú (con la etiqueta "Save...") con una función de controlador callback llamada `doSave`.

Añada la instancia de botón al escenario y asígnele el nombre myBtn_btn.

```
var menu_cm:ContextMenu = new ContextMenu();
menu_cm.customItems.push(new ContextMenuItem("Save...", doSave));
function doSave(menu:Object, obj:Object):Void {
  trace( " You selected the 'Save...' menu item ");
}
myBtn_btn.menu = menu_cm;
```

Seleccione Control > Probar película para probar el archivo SWF. Con el puntero sobre `myBtn_btn`, haga clic con el botón derecho o haga clic mientras pulsa Control. Aparecerá el menú contextual con Save. Cuando seleccione Save en el menú, aparecerá el panel Salida.

Véase también

ContextMenu, ContextMenuItem, menu (propiedad MovieClip.menu), menu (propiedad TextField.menu)

_name (propiedad Button._name)

```
public _name : String
```

Nombre de instancia del botón especificado por `my_btn`.

Disponibilidad: ActionScript 1.0; Flash Player 6

Ejemplo

El ejemplo siguiente realiza un seguimiento de todos los nombres de instancia de cualquier instancia Button que se encuentre en la línea de tiempo actual de un archivo SWF.

```
for (i in this) {
  if (this[i] instanceof Button) {
  trace(this[i]._name);
  }
}
```

onDragOut (controlador Button.onDragOut)

```
onDragOut = function() {}
```

Se invoca cuando se hace clic con el botón del ratón sobre el botón y luego se arrastra el puntero hacia el exterior del botón. Debe definir la función que ha de ejecutarse cuando se invoca el controlador de eventos.

Disponibilidad: ActionScript 1.0; Flash Player 6

Ejemplo

El ejemplo siguiente demuestra cómo puede ejecutar sentencias cuando se arrastra el puntero fuera de un botón. Cree un botón llamado `my_btn` en el escenario e introduzca el código ActionScript siguiente en un fotograma de la línea de tiempo:

```
my_btn.onDragOut = function() {
  trace("onDragOut: "+this._name);
};
my_btn.onDragOver = function() {
  trace("onDragOver: "+this._name);
};
```

onDragOver (controlador Button.onDragOver)

```
onDragOver = function() {}
```

Se invoca cuando el usuario presiona y arrastra el botón del ratón fuera del botón y, seguidamente, por encima de éste. Debe definir la función que ha de ejecutarse cuando se invoca el controlador de eventos.

Disponibilidad: ActionScript 1.0; Flash Player 6

Ejemplo

El ejemplo siguiente define una función del controlador onDragOver que envía una sentencia `trace()` al panel Salida. Cree un botón llamado `my_btn` en el escenario e introduzca el código ActionScript siguiente en la línea de tiempo:

```
my_btn.onDragOut = function() {
  trace("onDragOut: "+this._name);
};
my_btn.onDragOver = function() {
  trace("onDragOver: "+this._name);
};
```

Cuando compruebe el archivo SWF, arrastre el puntero fuera de la instancia de botón. A continuación, mientras mantiene pulsado el botón del ratón, vuelva a arrastrarlo sobre la instancia de botón. Observe que el panel Salida realiza un seguimiento de los movimientos.

Véase también

`onDragOut (controlador Button.onDragOut)`

onKeyDown (controlador Button.onKeyDown)

```
onKeyDown = function() {}
```

Se invoca cuando un botón se ha seleccionado con el teclado y se presiona una tecla. El controlador de eventos `onKeyDown` se invoca sin parámetros. Puede utilizar los métodos `Key.getAscii()` y `Key.getCode()` para determinar qué tecla ha presionado el usuario. Debe definir la función que ha de ejecutarse cuando se invoca el controlador de eventos.

Disponibilidad: ActionScript 1.0; Flash Player 6

Ejemplo

En el ejemplo siguiente se define una función que envía texto al panel Salida para el controlador `onKeyDown`. Cree un botón llamado `my_btn` en el escenario e introduzca el código ActionScript siguiente en un fotograma de la línea de tiempo:

```
my_btn.onKeyDown = function() {
  trace("onKeyDown: "+this._name+" (Key: "+getKeyPressed()+")");
};
function getKeyPressed():String {
  var theKey:String;
  switch (Key.getAscii()) {
  case Key.BACKSPACE :
    theKey = "BACKSPACE";
    break;
  case Key.SPACE :
    theKey = "SPACE";
    break;
  default :
    theKey = chr(Key.getAscii());
  }
  return theKey;
}
```

Seleccione Control > Probar película para probar el archivo SWF. Seleccione Control > Deshabilitar métodos abreviados de teclado en el entorno de prueba. A continuación, pulse la tecla Tabulador hasta que el botón quede seleccionado (aparecerá un rectángulo amarillo alrededor de la instancia `my_btn`) y comience a pulsar teclas en el teclado. Cuando pulsa teclas se muestran en el panel Salida.

Véase también

`onKeyUp (controlador Button.onKeyUp)`, `getAscii (método Key.getAscii)`, `getCode (método Key.getCode)`

onKeyUp (controlador Button.onKeyUp)

```
onKeyUp = function() {}
```

Se invoca cuando un botón tiene la selección de entrada y se suelta una tecla. El controlador de eventos `onKeyUp` se invoca sin parámetros. Puede utilizar los métodos `Key.getAscii()` y `Key.getCode()` para determinar qué tecla ha presionado el usuario.

Disponibilidad: ActionScript 1.0; Flash Player 6

Ejemplo

En el ejemplo siguiente se define una función que envía texto al panel Salida para el controlador `onKeyDown handler`. Cree un botón llamado `my_btn` en el escenario e introduzca el código ActionScript siguiente en un fotograma de la línea de tiempo:

```
my_btn.onKeyDown = function() {
  trace("onKeyDown: "+this._name+" (Key: "+getKeyPressed()+")");
};
my_btn.onKeyUp = function() {
  trace("onKeyUp: "+this._name+" (Key: "+getKeyPressed()+")");
};
function getKeyPressed():String {
  var theKey:String;
  switch (Key.getAscii()) {
  case Key.BACKSPACE :
    theKey = "BACKSPACE";
    break;
  case Key.SPACE :
    theKey = "SPACE";
    break;
  default :
    theKey = chr(Key.getAscii());
  }
  return theKey;
}
```

Pulse Control+Intro para comprobar el archivo SWF. Seleccione Control > Deshabilitar métodos abreviados de teclado en el entorno de prueba. A continuación, pulse la tecla Tabulador hasta que el botón quede seleccionado (aparecerá un rectángulo amarillo alrededor de la instancia `my_btn`) y comience a pulsar teclas en el teclado. Cuando pulsa teclas se muestran en el panel Salida.

Véase también

`onKeyDown (controlador Button.onKeyDown)`, `getAscii (método Key.getAscii)`, `getCode (método Key.getCode)`

onKillFocus (controlador Button.onKillFocus)

```
onKillFocus = function(newFocus:Object) {}
```

Se invoca cuando un botón deja de estar seleccionado con el teclado. El controlador `onKillFocus` recibe un parámetro, `newFocus`, que es un objeto que representa al nuevo objeto seleccionado. Si no hay ningún objeto seleccionado con el teclado, `newFocus` contendrá el valor `null` (nulo).

Disponibilidad: ActionScript 1.0; Flash Player 6

Parámetros

newFocus:Object - El objeto que recibe la selección.

Ejemplo

El ejemplo siguiente demuestra cómo pueden ejecutarse sentencias cuando un botón deja de estar seleccionado. Cree una instancia de botón llamada my_btn en el escenario y añada el código ActionScript siguiente al fotograma 1 de la línea de tiempo:

```
this.createTextField("output_txt", this.getNextHighestDepth(), 0, 0, 300,
  200);
output_txt.wordWrap = true;
output_txt.multiline = true;
output_txt.border = true;
my_btn.onKillFocus = function() {
  output_txt.text = "onKillFocus: "+this._name+newline+output_txt.text;
};
```

Compruebe el archivo SWF en una ventana del navegador y utilice la tecla Tabulador para pasar por los elementos de la ventana. Cuando la instancia de botón deja de estar seleccionada, se envía texto al campo de texto output_txt.

El método MovieClip.getNextHighestDepth() utilizado en este ejemplo necesita Flash Player 7 o una versión posterior. Si el archivo SWF incluye un componente de la versión 2, utilice la clase DepthManager de los componentes de la versión 2 en lugar del método MovieClip.getNextHighestDepth().

onPress (controlador Button.onPress)

```
onPress = function() {}
```

Se invoca cuando se presiona una tecla. Debe definir la función que ha de ejecutarse cuando se invoca el controlador de eventos.

Disponibilidad: ActionScript 1.0; Flash Player 6

Ejemplo

En el ejemplo siguiente se define una función que envía una sentencia trace() al panel Salida para el controlador onPress:

```
my_btn.onPress = function () {
  trace ("onPress called");
};
```

onRelease (controlador Button.onRelease)

```
onRelease = function() {}
```

Se invoca cuando se suelta un botón. Debe definir la función que ha de ejecutarse cuando se invoca el controlador de eventos.

Disponibilidad: ActionScript 1.0; Flash Player 6

Ejemplo

En el ejemplo siguiente se define una función que envía una sentencia trace() al panel Salida para el controlador `onRelease`:

```
my_btn.onRelease = function () {
  trace ("onRelease called");
};
```

onReleaseOutside (controlador Button.onReleaseOutside)

```
onReleaseOutside = function() {}
```

Se invoca cuando se suelta el botón del ratón mientras el puntero está fuera del botón después de haber hecho clic mientras el puntero estaba dentro del botón. Debe definir la función que ha de ejecutarse cuando se invoca el controlador de eventos.

Disponibilidad: ActionScript 1.0; Flash Player 6

Ejemplo

En el ejemplo siguiente se define una función que envía una sentencia trace() al panel Salida para el controlador `onReleaseOutside`:

```
my_btn.onReleaseOutside = function () {
  trace ("onReleaseOutside called");
};
```

onRollOut (controlador Button.onRollOut)

```
onRollOut = function() {}
```

Se invoca cuando el puntero se desplaza fuera del área de un botón. Debe definir la función que ha de ejecutarse cuando se invoca el controlador de eventos.

Disponibilidad: ActionScript 1.0; Flash Player 6

Ejemplo

En el ejemplo siguiente se define una función que envía una sentencia trace() al panel Salida para el controlador `onRollOut`:

```
my_btn.onRollOut = function () {
  trace ("onRollOut called");
};
```

onRollOver (controlador Button.onRollOver)

```
onRollOver = function() {}
```

Se invoca cuando el puntero se desplaza sobre el área de un botón. Debe definir la función que ha de ejecutarse cuando se invoca el controlador de eventos.

Disponibilidad: ActionScript 1.0; Flash Player 6

Ejemplo

En el ejemplo siguiente se define una función que envía una sentencia trace() al panel Salida para el controlador `onRollOver`:

```
my_btn.onRollOver = function () {
  trace ("onRollOver called");
};
```

onSetFocus (controlador Button.onSetFocus)

```
onSetFocus = function(oldFocus:Object) {}
```

Se invoca cuando un botón queda seleccionado con el teclado. El parámetro `oldFocus` es el objeto que deja de estar seleccionado. Por ejemplo, si el usuario presiona la tecla Tabulador parar desplazar la selección del teclado de un campo de texto a un botón, oldFocus contendrá la instancia del campo de texto.

Si anteriormente no había ningún objeto seleccionado con el teclado, `oldFocus` contendrá un valor null (nulo).

Disponibilidad: ActionScript 1.0; Flash Player 6

Parámetros

`oldFocus``:Object` - El objeto que deja de estar seleccionado.

Ejemplo

El ejemplo siguiente demuestra cómo ejecutar sentencias cuando el usuario de un archivo SWF desplaza la selección de un botón a otro. Cree dos botones, `btn1_btn` y `btn2_btn`, e introduzca el código ActionScript siguiente en el fotograma 1 de la línea de tiempo:

```
Selection.setFocus(btn1_btn);
trace(Selection.getFocus());
btn2_btn.onSetFocus = function(oldFocus) {
  trace(oldFocus._name + "lost focus");
};
```

Compruebe el archivo SWF pulsando Control+Intro. Seleccione Control > Deshabilitar métodos abreviados de teclado si no está seleccionado. La selección pasará a `btn1_btn`. Cuando `btn1_btn` deja de estar seleccionado y `btn2_btn` pasa a estar seleccionado, se muestra información en el panel Salida.

_parent (propiedad Button._parent)

```
public _parent : MovieClip
```

Referencia al clip de película u objeto que contiene el clip de película u objeto actual. El objeto actual es el que contiene el código ActionScript que hace referencia a `_parent`.

Utilice `_parent` para especificar una ruta de acceso relativa a los clips de película u objetos que se encuentran por encima del clip de película u objeto actual. Puede utilizar `_parent` para subir múltiples niveles en la lista de visualización, como se muestra a continuación:

```
this._parent._parent._alpha = 20;
```

Disponibilidad: ActionScript 1.0; Flash Player 6

Ejemplo

En el ejemplo siguiente, un botón llamado `my_btn` se sitúa dentro de un clip de película llamado `my_mc`. El código siguiente muestra cómo utilizar la propiedad `_parent` para obtener una referencia al clip de película `my_mc`:

```
trace(my_mc.my_btn._parent);
```

El panel Salida muestra lo siguiente:

```
_level0.my_mc
```

Véase también

`_parent (propiedad MovieClip._parent)`, `_target (propiedad MovieClip._target)`, `Propiedad _root`

_quality (propiedad Button._quality)

```
public _quality : String
```

Propiedad (global); establece o recupera la calidad de representación utilizada por un archivo SWF. Las fuentes de dispositivo siempre se muestran dentadas y, por consiguiente, no se ven afectadas por la propiedad `_quality`.

La propiedad `_quality` puede configurarse con los siguientes valores:

- `"LOW"` Calidad de representación baja. No se suavizan ni los gráficos ni los mapas de bits.
- `"MEDIUM"` Calidad de representación media. Los gráficos se suavizan empleando una cuadrícula de 2 x 2 píxeles, pero los mapas de bits no se suavizan. Esta configuración resulta adecuada para películas que no contengan texto.
- `"HIGH"` Calidad de representación alta. Los gráficos se suavizan empleando una cuadrícula de 4 x 4 píxeles, mientras que los mapas de bits se suavizan si la película es estática. Esta es la calidad de representación predeterminada de Flash.
- `"BEST"` Calidad de representación muy alta. Los gráficos se suavizan empleando una cuadrícula de 4 x 4 píxeles y los mapas de bits se suavizan siempre.

Nota: Aunque puede especificar esta propiedad para un objeto Button, se trata en realidad de una propiedad global, por lo que puede especificar su valor simplemente como `_quality`.

Disponibilidad: ActionScript 1.0; Flash Player 6

Ejemplo

Este ejemplo establece como `LOW` la calidad de representación de un botón llamado `my_btn`:

```
my_btn._quality = "LOW";
```

_rotation (propiedad Button._rotation)

```
public _rotation : Number
```

Giro del botón, expresado en grados, con respecto a su orientación original. Los valores comprendidos entre 0 y 180 representan un giro en el sentido de las agujas del reloj, mientras que los comprendidos entre 0 y -180 representan un giro en sentido contrario al de las agujas del reloj. Los valores situados fuera de este rango se suman o restan de 360 para obtener un valor que sí esté comprendido en el rango. Por ejemplo, la sentencia `my_btn._rotation = 450` es igual que `my_btn._rotation = 90`.

Disponibilidad: ActionScript 1.0; Flash Player 6

Ejemplo

El ejemplo siguiente gira dos botones en el escenario. Cree dos botones en el escenario llamados `control_btn` y `my_btn`. Procure que `my_btn` no sea completamente redondo, para poder ver cómo gira. A continuación, introduzca el código ActionScript siguiente en el fotograma 1 de la línea de tiempo:

```
var control_btn:Button;
var my_btn:Button;
control_btn.onRelease = function() {
  my_btn._rotation += 10;
};
```

Ahora puede crear otro botón en el escenario llamado `myOther_btn`, procurando que no sea completamente redondo (para poder ver cómo gira). Introduzca el código ActionScript siguiente en el fotograma 1 de la línea de tiempo:

```
var myOther_btn:Button;
this.createEmptyMovieClip("rotater_mc", this.getNextHighestDepth());
rotater_mc.onEnterFrame = function() {
  myOther_btn._rotation += 2;
};
```

El método `MovieClip.getNextHighestDepth()` utilizado en este ejemplo necesita Flash Player 7 o una versión posterior. Si el archivo SWF incluye un componente de la versión 2, utilice la clase DepthManager de los componentes de la versión 2 en lugar del método `MovieClip.getNextHighestDepth()`.

Véase también

`_rotation (propiedad MovieClip._rotation)`, `_rotation (propiedad TextField._rotation)`

scale9Grid (propiedad Button.scale9Grid)

```
public scale9Grid : Rectangle
```

Región rectangular que define las nueve regiones de escala del botón. Si se establece como `null`, todo el botón se escalará normalmente cuando se aplique cualquier transformación de escala.

Cuando se define una propiedad `scale9Grid` para un botón, éste se divide en una cuadrícula con nueve regiones, basadas en el rectángulo `scale9Grid`, que define la región central de la cuadrícula. La cuadrícula contiene otras ocho regiones:

- El área de la esquina superior izquierda fuera del rectángulo
- El área situada por encima del rectángulo
- El área de la esquina superior derecha fuera del rectángulo

- El área a la izquierda del rectángulo
- El área a la derecha del rectángulo
- El área de la esquina inferior izquierda fuera del rectángulo
- El área situada por debajo del rectángulo
- El área de la esquina inferior derecha fuera del rectángulo

Puede considerar las ocho regiones externas al centro (que define el rectángulo) como si fueran el marco de un cuadro con reglas especiales cuando se escala el botón.

Cuando se establece la propiedad `scale9Grid` y se escala un botón, todo el texto y los degradados se escalan normalmente. Sin embargo, se aplican las reglas siguientes a otros tipos de objetos:

- El contenido de la región central se escala normalmente.
- El contenido de las esquinas no se escala.
- El contenido de las regiones superior e inferior se escala sólo en horizontal. El contenido de las regiones izquierda y derecha se escala sólo en vertical.

Si se gira un botón, todo el escalado posterior será normal y se ignorará la propiedad `scale9Grid`.

La propiedad `scale9Grid` suele utilizarse para crear un botón en el que las líneas de los bordes conserven la misma anchura cuando se escale el botón.

Para más información, incluidas ilustraciones y un ejemplo relacionado, consulte MovieClip.scale9Grid.

Disponibilidad: ActionScript 1.0; Flash Player 8

Véase también

`Rectangle (flash.geom.Rectangle)`, `scale9Grid (propiedad MovieClip.scale9Grid)`

_soundbuftime (propiedad Button._soundbuftime)

```
public _soundbuftime : Number
```

La propiedad que especifica el número de segundos que un sonido debe acumularse previamente en el búfer antes de que comience a reproducirse sin interrupción.

Nota: Aunque puede especificar esta propiedad para un objeto Button, se trata en realidad de una propiedad global que se aplica a todos los sonidos cargados y puede especificar su valor simplemente como `_soundbuftime`. Al establecer esta propiedad para un objeto Button en realidad se establece la propiedad global.

Para más información y ver un ejemplo, consulte la propiedad `_soundbuftime` global.

Disponibilidad: ActionScript 1.0; Flash Player 6

Véase también

`Propiedad _soundbuftime`

tabEnabled (propiedad Button.tabEnabled)

```
public tabEnabled : Boolean
```

Especifica si `my_btn` está incluido en el orden de tabulación automático. De manera predeterminada, tiene el valor `undefined`.

Si la propiedad `tabEnabled` es `undefined` o `true`, el objeto se incluirá en el orden de tabulación automático. Si la propiedad `tabIndex` también está configurada con un valor, el objeto se incluirá también en el orden de tabulación personalizado. Si `tabEnabled` tiene el valor `false`, el objeto no se incluirá en el orden de tabulación automático ni en el personalizado aunque se establezca la propiedad `tabIndex`.

Disponibilidad: ActionScript 1.0; Flash Player 6

Ejemplo

El código ActionScript siguiente se utiliza para establecer la propiedad `tabEnabled` como `false` para un botón de cuatro. Sin embargo, los cuatro botones (`one_btn`, `two_btn`, `three_btn`, y `four_btn`) se colocan en un orden de tabulación personalizado utilizando `tabIndex`. Aunque se establece `tabIndex` para `three_btn`, no se incluye `three_btn` en un orden de tabulación personalizado o automático, ya que `tabEnabled` se establece como `false` para esa instancia. Para establecer el orden de tabulación para los cuatro botones, añada el código ActionScript siguiente al fotograma 1 de la línea de tiempo:

```
three_btn.tabEnabled = false;
two_btn.tabIndex = 1;
four_btn.tabIndex = 2;
three_btn.tabIndex = 3;
one_btn.tabIndex = 4;
```

Asegúrese de desactivar los métodos abreviados de teclado cuando compruebe el archivo SWF seleccionando Control > Deshabilitar métodos abreviados de teclado en el entorno de pruebas.

Véase también

`tabIndex (propiedad Button.tabIndex)`, `tabEnabled (propiedad MovieClip.tabEnabled)`, `tabEnabled (propiedad TextField.tabEnabled)`

tabIndex (propiedad Button.tabIndex)

```
public tabIndex : Number
```

Permite personalizar el orden de tabulación de los objetos de un archivo SWF. Puede establecer la propiedad `tabIndex` de un botón, un clip de película o una instancia de campo de texto; la configuración predeterminada es `undefined` (no definido).

Si algún objeto mostrado actualmente en el archivo SWF contiene una propiedad `tabIndex`, se desactivará el orden de tabulación automático y el orden de tabulación se calculará a partir de las propiedades `tabIndex` de los objetos del archivo SWF. El orden de tabulación personalizado sólo incluye objetos que tienen propiedades `tabIndex`.

La propiedad `tabIndex` puede ser un entero no negativo. Los objetos se ordenan conforme a lo que indiquen las propiedades `tabIndex` y en orden ascendente. Un objeto que tenga un valor `tabIndex` de `1` precederá a otro objeto que tenga un valor `tabIndex` de `2`. Si hay dos objetos con el mismo valor `tabIndex`, el que precede al otro en el orden de tabulación es `undefined`.

El orden de tabulación personalizado definido por la propiedad `tabIndex` es *flat* (plano o no jerárquico). Esto significa que no se tienen en cuenta las relaciones jerárquicas entre los objetos del archivo SWF. Todos los objetos del archivo SWF con propiedades `tabIndex` tendrán su lugar en el orden de tabulación, que se determinará por el orden de los valores de `tabIndex`. Si dos objetos tienen el mismo valor de `tabIndex`, el que va en primer lugar no estará definido (`undefined`). No debe utilizar el mismo valor de `tabIndex` para varios objetos.

Disponibilidad: ActionScript 1.0; Flash Player 6

Ejemplo

El código ActionScript siguiente se utiliza para establecer la propiedad `tabEnabled` como `false` para un botón de cuatro. Sin embargo, los cuatro botones (`one_btn`, `two_btn`, `three_btn`, y `four_btn`) se colocan en un orden de tabulación personalizado utilizando `tabIndex`. Aunque se establece `tabIndex` para `three_btn`, no se incluye `three_btn` en un orden de tabulación personalizado o automático, ya que `tabEnabled` se establece como `false` para esa instancia. Para establecer el orden de tabulación para los cuatro botones, añada el código ActionScript siguiente al fotograma 1 de la línea de tiempo:

```
three_btn.tabEnabled = false;
two_btn.tabIndex = 1;
four_btn.tabIndex = 2;
three_btn.tabIndex = 3;
one_btn.tabIndex = 4;
```

Asegúrese de desactivar los métodos abreviados de teclado cuando compruebe el archivo SWF seleccionando Control > Deshabilitar métodos abreviados de teclado en el entorno de pruebas.

Véase también

tabEnabled (propiedad Button.tabEnabled), tabChildren (propiedad MovieClip.tabChildren), tabEnabled (propiedad MovieClip.tabEnabled), tabIndex (propiedad MovieClip.tabIndex), tabIndex (propiedad TextField.tabIndex)

_target (propiedad Button._target)

```
public _target : String [read-only]
```

Devuelve la ruta de destino de la instancia de botón especificada por `my_btn`.

Disponibilidad: ActionScript 1.0; Flash Player 6

Ejemplo

Añada una instancia de botón al escenario con el nombre `my_btn` y añada el código siguiente al fotograma 1 de la línea de tiempo:

```
trace(my_btn._target); //displays /my_btn
```

Seleccione my_btn y conviértalo en un clip de película. Asigne al clip de película nuevo el nombre de instancia `my_mc`. Elimine el código ActionScript existente en el fotograma 1 de la línea de tiempo y sustitúyalo por:

```
my_mc.my_btn.onRelease = function(){
  trace(this._target); //displays /my_mc/my_btn
};
```

Para convertir la notación de barras en notación de puntos, modifique el ejemplo del código anterior por el siguiente:

```
my_mc.my_btn.onRelease = function(){
  trace(eval(this._target)); //displays _level0.my_mc.my_btn
};
```

Este código permite acceder a métodos y parámetros del objeto de destino, como:

```
my_mc.my_btn.onRelease = function(){
  var target_btn:Button = eval(this._target);
trace(target_btn._name); //displays my_btn
};
```

Véase también

_target (propiedad MovieClip._target)

trackAsMenu (propiedad Button.trackAsMenu)

```
public trackAsMenu : Boolean
```

Valor booleano que indica si otros botones o clips de película pueden recibir eventos de liberación del botón del ratón. Si arrastra un botón y lo suelta sobre un segundo botón, el evento `onRelease` se registrará para el segundo botón. Esto le permite crear menús. Puede establecer la propiedad `trackAsMenu` en cualquier botón u objeto del clip de película. Si no se ha definido la propiedad `trackAsMenu`, el comportamiento predeterminado será `false`.

Puede cambiar la propiedad `trackAsMenu` en cualquier momento; el botón modificado adoptará de inmediato el nuevo comportamiento.

Disponibilidad: ActionScript 1.0; Flash Player 6

Ejemplo

El ejemplo siguiente demuestra cómo realizar el seguimiento de dos botones como un menú. Sitúe dos instancias de botón en el escenario, llamadas `one_btn` y `two_btn`. Introduzca el código ActionScript siguiente en la línea de tiempo:

```
var one_btn:Button;
var two_btn:Button;
one_btn.trackAsMenu = true;
two_btn.trackAsMenu = true
one_btn.onRelease = function() {
  trace("clicked one_btn");
};
two_btn.onRelease = function() {
  trace("clicked two_btn");
};
```

Compruebe el archivo SWF haciendo clic en `one_btn` en el escenario, manteniendo pulsado el botón del ratón y liberándolo sobre `two_btn`. A continuación, marque como comentario las dos líneas de ActionScript que contienen `trackAsMenu` y pruebe el archivo SWF otra vez para ver la diferencia en el comportamiento del botón.

Véase también

`trackAsMenu (propiedad MovieClip.trackAsMenu)`

_url (propiedad Button._url)

```
public _url : String [read-only]
```

Recupera la URL del archivo SWF que creó el botón.

Disponibilidad: ActionScript 1.0; Flash Player 6

Ejemplo

Cree dos instancias de botón en el escenario, llamadas `one_btn` y `two_btn`. Introduzca el siguiente código ActionScript en el fotograma 1 de la línea de tiempo:

```
var one_btn:Button;
var two_btn:Button;
this.createTextField("output_txt", 999, 0, 0, 100, 22);
output_txt.autoSize = true;
one_btn.onRelease = function() {
  trace("clicked one_btn");
  trace(this._url);
};
two_btn.onRelease = function() {
  trace("clicked "+this._name);
  var url_array:Array = this._url.split("/");
  var my_str:String = String(url_array.pop());
  output_txt.text = unescape(my_str);
};
```

Cuando hace clic en cada botón, se mostrará en el panel Salida el nombre del archivo SWF que contiene los botones.

useHandCursor (propiedad Button.useHandCursor)

```
public useHandCursor : Boolean
```

Valor booleano que, cuando se establece como `true` (configuración predeterminada), indica si debe aparecer una mano que señala (cursor de mano) cuando el ratón pasa por encima de un botón. Si esta propiedad se establece con `false`, se utilizará el puntero de flecha.

Puede cambiar la propiedad `useHandCursor` en cualquier momento; el botón modificado adoptará de inmediato el nuevo comportamiento. La propiedad `useHandCursor` puede leerse de un objeto prototipo.

Disponibilidad: ActionScript 1.0; Flash Player 6

Ejemplo

Cree dos botones en el escenario con los nombres de instancia `myBtn1_btn` y `myBtn2_btn`. Introduzca el siguiente código ActionScript en el fotograma 1 de la línea de tiempo:

```
myBtn1_btn.useHandCursor = false;
myBtn1_btn.onRelease = buttonClick;
myBtn2_btn.onRelease = buttonClick;
function buttonClick() {
  trace(this._name);
}
```

Cuando se coloca el ratón sobre `myBtn1_btn` y se hace clic, no aparece una mano que señala. Sin embargo, la mano aparecerá cuando se coloque sobre el botón y haga clic en `myBtn2_btn`.

_visible (propiedad Button._visible)

```
public _visible : Boolean
```

Valor booleano que indica si el botón especificado por `my_btn` es visible. Los botones no visibles (que tienen la propiedad `_visible` configurada como `false`) se desactivan.

Disponibilidad: ActionScript 1.0; Flash Player 6

Ejemplo

Cree dos botones en el escenario con los nombres de instancia `myBtn1_btn` y `myBtn2_btn`. Introduzca el siguiente código ActionScript en el fotograma 1 de la línea de tiempo:

```
myBtn1_btn.onRelease = function() {
  this._visible = false;
  trace("clicked "+this._name);
};
myBtn2_btn.onRelease = function() {
  this._alpha = 0;
  trace("clicked "+this._name);
};
```

Observe que puede seguir haciendo clic en `myBtn2_btn` después de establecer el alfa en 0.

Véase también

`_visible (propiedad MovieClip._visible)`, `_visible (propiedad TextField._visible)`

_width (propiedad Button._width)

```
public _width : Number
```

Anchura del botón, expresada en píxeles.

Disponibilidad: ActionScript 1.0; Flash Player 6

Ejemplo

El ejemplo siguiente incrementa la propiedad width de un botón llamado `my_btn`, y muestra la anchura en el panel Salida. Introduzca el siguiente código ActionScript en el fotograma 1 de la línea de tiempo:

```
my_btn.onRelease = function() {
  trace(this._width);
  this._width ~= 1.1;
};
```

Véase también

`_width (propiedad MovieClip._width)`

_x (propiedad Button._x)

```
public _x : Number
```

Entero que establece la coordenada x de un botón como relativa a las coordenadas locales del clip de película principal. Si un botón se encuentra en la línea de tiempo principal, su sistema de coordenadas hará referencia a la esquina superior izquierda del escenario como (0, 0). Si el botón está dentro de un clip de película que incluye transformaciones, el botón estará en el sistema de coordenadas local del clip de película en el que está contenido. Por consiguiente, en el caso de un clip de película con un giro de 90 grados en sentido contrario al de las agujas del reloj, el botón contenido hereda un sistema de coordenadas con un giro de 90 grados en sentido contrario al de las agujas del reloj. Las coordenadas del botón hacen referencia a la posición del punto de registro.

Disponibilidad: ActionScript 1.0; Flash Player 6

Ejemplo

El ejemplo siguiente establece las coordenadas de `my_btn` en 0 en el escenario. Cree un botón llamado `my_btn` e introduzca el código ActionScript siguiente en el fotograma 1 de la línea de tiempo:

```
my_btn._x = 0;
my_btn._y = 0;
```

Véase también

_xscale (propiedad Button._xscale), _y (propiedad Button._y), _yscale (propiedad Button._yscale)

_xmouse (propiedad Button._xmouse)

```
public _xmouse : Number [read-only]
```

Devuelve la coordenada *x* de la posición del ratón relativa al botón.

Disponibilidad: ActionScript 1.0; Flash Player 6

Ejemplo

El ejemplo siguiente muestra la posición xmouse para el escenario y un botón llamado `my_btn` que se sitúa en el escenario. Introduzca el siguiente código ActionScript en el fotograma 1 de la línea de tiempo:

```
this.createTextField("mouse_txt", 999, 5, 5, 150, 40);
mouse_txt.html = true;
mouse_txt.wordWrap = true;
mouse_txt.border = true;
mouse_txt.autoSize = true;
mouse_txt.selectable = false;
//
var mouseListener:Object = new Object();
mouseListener.onMouseMove = function() {
  var table_str:String = "<textformat tabstops='[50,100]'>";
  table_str += "<b>Stage</b>\t"+"x:"+_xmouse+"\t"+"y:"+_ymouse+newline;
  table_str += "<b>Button</
  b>\t"+"x:"+my_btn._xmouse+"\t"+"y:"+my_btn._ymouse+newline;
  table_str += "</textformat>";
  mouse_txt.htmlText = table_str;
};
Mouse.addListener(mouseListener);
```

Véase también

_ymouse (propiedad Button._ymouse)

_xscale (propiedad Button._xscale)

```
public _xscale : Number
```

Escala horizontal del botón aplicada desde el punto de registro del botón, expresada como porcentaje. El punto de registro predeterminado es (0,0).

La escala del sistema de coordenadas local afecta a la configuración de las propiedades _x e _y, que se definen en píxeles. Por ejemplo, si se aplica una escala del 50% al clip de película principal, la configuración de la propiedad _x desplazará un objeto situado en el botón la mitad de píxeles que si el archivo SWF tuviera una escala del 100%.

Disponibilidad: ActionScript 1.0; Flash Player 6

Ejemplo

El ejemplo siguiente escala un botón llamado my_btn. Cuando hace clic y suelta el botón, crece un 10% en los ejes *x* e *y*. Introduzca el siguiente código ActionScript en el fotograma 1 de la línea de tiempo:

```
my_btn.onRelease = function(){
  this._xscale ~= 1.1;
  this._yscale ~= 1.1;
};
```

Véase también

_x (propiedad Button._x), _y (propiedad Button._y), _yscale (propiedad Button._yscale)

_y (propiedad Button._y)

```
public _y : Number
```

Coordenada *y* del botón relativa a las coordenadas locales del clip de película principal. Si un botón se encuentra en la línea de tiempo principal, su sistema de coordenadas hará referencia a la esquina superior izquierda del escenario como (0, 0). Si el botón está dentro de otro clip de película que incluye transformaciones, el botón estará en el sistema de coordenadas local del clip de película en el que está contenido. Por consiguiente, en el caso de un clip de película con un giro de 90 grados en sentido contrario al de las agujas del reloj, el botón contenido hereda un sistema de coordenadas con un giro de 90 grados en sentido contrario al de las agujas del reloj. Las coordenadas del botón hacen referencia a la posición del punto de registro.

Disponibilidad: ActionScript 1.0; Flash Player 6

Ejemplo

El ejemplo siguiente establece las coordenadas de my_btn en 0 en el escenario. Cree un botón llamado my_btn e introduzca el código ActionScript siguiente en el fotograma 1 de la línea de tiempo:

```
my_btn._x = 0;
my_btn._y = 0;
```

Véase también

_x (propiedad Button._x), _xscale (propiedad Button._xscale), _yscale (propiedad Button._yscale)

_ymouse (propiedad Button._ymouse)

```
public _ymouse : Number [read-only]
```

Indica la coordenada *y* de la posición del ratón relativa al botón.

Disponibilidad: ActionScript 1.0; Flash Player 6

Ejemplo

El ejemplo siguiente muestra la posición ymouse para el escenario y un botón llamado `my_btn` que se sitúa en el escenario. Introduzca el siguiente código ActionScript en el fotograma 1 de la línea de tiempo:

```
this.createTextField("mouse_txt", 999, 5, 5, 150, 40);
mouse_txt.html = true;
mouse_txt.wordWrap = true;
mouse_txt.border = true;
mouse_txt.autoSize = true;
mouse_txt.selectable = false;
//
var mouseListener:Object = new Object();
mouseListener.onMouseMove = function() {
  var table_str:String = "<textformat tabstops='[50,100]'>";
  table_str += "<b>Stage</b>\t"+"x:"+_xmouse+"\t"+"y:"+_ymouse+newline;
  table_str += "<b>Button</
  b>\t"+"x:"+my_btn._xmouse+"\t"+"y:"+my_btn._ymouse+newline;
  table_str += "</textformat>";
  mouse_txt.htmlText = table_str;
};
Mouse.addListener(mouseListener);
```

Véase también

_xmouse (propiedad Button._xmouse)

_yscale (propiedad Button._yscale)

```
public _yscale : Number
```

Escala vertical del botón aplicada desde el punto de registro del botón, expresada como porcentaje. El punto de registro predeterminado es (0,0).

Disponibilidad: ActionScript 1.0; Flash Player 6

Ejemplo

El ejemplo siguiente escala un botón llamado `my_btn`. Cuando hace clic y suelta el botón, crece un 10% en los ejes *x* e *y*. Introduzca el siguiente código ActionScript en el fotograma 1 de la línea de tiempo:

```
my_btn.onRelease = function(){
  this._xscale ~= 1.1;
  this._yscale ~ 1.1;
};
```

Véase también

`_y (propiedad Button._y)`, `_x (propiedad Button._x)`, `_xscale (propiedad Button._xscale)`

Camera

```
Object
  |
  +-Camera
```

`public class` ***Camera***
`extends Object`

La clase Camera está pensada principalmente para ser utilizada con Macromedia Flash Communication Server, aunque puede utilizarse de forma limitada sin el servidor.

La clase Camera le permite capturar vídeo de una cámara de vídeo conectada al equipo que ejecuta Macromedia Flash Player (por ejemplo, para controlar la salida de vídeo de una webcam conectada a su sistema local). (Flash ofrece funciones de audio similares; para más información, consulte la entrada de la clase Microphone.)

Advertencia: Cuando un archivo SWF intenta acceder a la cámara devuelta por `Camera.get()`, Flash Player muestra un cuadro de diálogo Privacidad que permite al usuario autorizar o denegar el acceso a la cámara. (Asegúrese de que el tamaño del escenario sea de al menos 215 x 138 píxeles para los ejemplos de la clase Camera; este es el tamaño mínimo que exige Flash para mostrar el cuadro de diálogo.) Los usuarios finales y los usuarios con derechos administrativos también pueden desactivar el acceso de cámara para cada sitio o de forma global.

Para crear o hacer referencia a un objeto Camera, utilice el método `Camera.get()`.

Disponibilidad: ActionScript 1.0; Flash Player 6

Resumen de propiedades

Modificadores	Propiedad	Descripción
	`activityLevel:Number` [read-only]	Valor numérico que especifica la cantidad de movimiento que está detectando la cámara.
	`bandwidth:Number` [read-only]	Entero que especifica la cantidad máxima de ancho de banda, expresada en bytes, que puede utilizar la salida de vídeo actual.
	`currentFps:Number` [read-only]	Velocidad a la que está capturando datos la cámara, expresada en fotogramas por segundo.
	`fps:Number` [read-only]	Velocidad máxima a la que desea que la cámara capture datos, expresada en fotogramas por segundo.
	`height:Number` [read-only]	Altura de captura actual, expresada en píxeles.
	`index:Number` [read-only]	Entero basado en cero que especifica el índice de la cámara, como se refleja en la matriz devuelta por `Camera.names`.
	`motionLevel:Number` [read-only]	Valor numérico que especifica la cantidad de movimiento necesario para invocar `Camera.onActivity(true)`.
	`motionTimeOut:Number` [read-only]	Número de milisegundos que transcurren entre el momento en que la cámara deja de detectar movimiento y la invocación de `Camera.onActivity (false)`.
	`muted:Boolean` [read-only]	Valor booleano que especifica si el usuario ha denegado el acceso a la cámara (`true`) o lo ha autorizado (`false`) en el panel Parámetros de privacidad de Flash Player.
	`name:String` [read-only]	Cadena que especifica el nombre de la cámara actual tal y como lo ha devuelto el hardware de la cámara.
`static`	`names:Array` [read-only]	Recupera una matriz de cadenas que refleja los nombres de todas las cámaras disponibles sin mostrar el panel Parámetros de privacidad de Flash Player.
	`quality:Number` [read-only]	Entero que especifica el nivel necesario de calidad de imagen, tal y como lo determina la cantidad de compresión aplicada a cada fotograma de vídeo.
	`width:Number` [read-only]	Anchura de captura actual, expresada en píxeles.

Propiedades heredadas de la clase Object

constructor (propiedad Object.constructor), __proto__ (propiedad Object.__proto__), prototype (propiedad Object.prototype), __resolve (propiedad Object.__resolve)

Resumen de eventos

Evento	Descripción
onActivity = function(active:Boolean) {}	Se invoca cuando la cámara comienza o deja de detectar movimiento.
onStatus = function(infoObject:Object) {}	Se invoca cuando el usuario autoriza o deniega el acceso a la cámara.

Resumen de métodos

Modificadores	Firma	Descripción
static	get([index:Number]) : Camera	Devuelve una referencia a un objeto Camera para capturar vídeo.
	setMode([width:Number], [height:Number], [fps:Number], [favorArea:Boolean]) : Void	Establece el modo de captura de cámara con el modo nativo que mejor se adapte a los requisitos especificados.
	setMotionLevel([motionLevel:Number], [timeOut:Number]) : Void	Especifica la cantidad de movimiento necesaria para invocar Camera.onActivity(true).
	setQuality([bandwidth:Number], [quality:Number]) : Void	Establece la cantidad máxima de ancho de banda por segundo o la calidad de imagen requerida para la salida de vídeo actual.

addProperty (método Object.addProperty), hasOwnProperty (método Object.hasOwnProperty), isPropertyEnumerable (método Object.isPropertyEnumerable), isPrototypeOf (método Object.isPrototypeOf), registerClass (método Object.registerClass), toString (método Object.toString), unwatch (método Object.unwatch), valueOf (método Object.valueOf), watch (método Object.watch)

activityLevel (propiedad Camera.activityLevel)

`public activityLevel : Number [read-only]`

Valor numérico que especifica la cantidad de movimiento que está detectando la cámara. El rango de valores válidos es de 0 (no se detecta movimiento) a 100 (se detecta gran cantidad de movimiento). El valor de esta propiedad puede ayudarle a determinar si es preciso pasar un valor de configuración para `Camera.setMotionLevel()`.

Si la cámara está disponible pero no se está utilizando todavía porque no se ha llamado a `Video.attachVideo()`, esta propiedad se establece con el valor -1.

Si está reproduciendo sin interrupción vídeo local sin comprimir, esta propiedad sólo se establece si ha asignado una función al controlador de eventos `Camera.onActivity`. En caso contrario, no está definida.

Disponibilidad: ActionScript 1.0; Flash Player 6

Ejemplo

Este ejemplo detecta la cantidad de movimiento que detecta la cámara utilizando la propiedad `activityLevel` y una instancia ProgressBar. Cree una instancia de vídeo nueva seleccionando Nuevo vídeo en el menú Opciones de Biblioteca. Añada una instancia al escenario y asígnele el nombre de instancia `my_video`. Añada una instancia de componente ProgressBar al escenario y asígnele el nombre `activity_pb`. A continuación, añada el código ActionScript siguiente al fotograma 1 de la línea de tiempo:

```
// video instance on the Stage.
var my_video:Video;
var activity_pb:mx.controls.ProgressBar;
var my_cam:Camera = Camera.get();
my_video.attachVideo(my_cam);
activity_pb.mode = "manual";
activity_pb.label = "Activity %3%%";

this.onEnterFrame = function() {
  activity_pb.setProgress(my_cam.activityLevel, 100);
};
```

```
my_cam.onActivity = function(isActive:Boolean) {
  var themeColor:String = (isActive) ? "haloGreen" : "haloOrange";
  activity_pb.setStyle("themeColor", themeColor);
};
```

Véase también

motionLevel (propiedad Camera.motionLevel), setMotionLevel (método Camera.setMotionLevel)

bandwidth (propiedad Camera.bandwidth)

```
public bandwidth : Number [read-only]
```

Entero que especifica la cantidad máxima de ancho de banda, expresada en bytes, que puede utilizar la salida de vídeo actual. El valor 0 indica que el vídeo Flash puede utilizar el ancho de banda que sea necesario para mantener la calidad de fotogramas deseada.

Para definir esta propiedad, llame a `Camera.setQuality()`.

Disponibilidad: ActionScript 1.0; Flash Player 6

Ejemplo

El ejemplo siguiente cambia la cantidad máxima de ancho de banda empleada por la cámara. Cree una instancia de vídeo nueva seleccionando Nuevo vídeo en el menú Opciones de Biblioteca. Añada una instancia al escenario y asígnele el nombre de instancia `my_video`. Añada una instancia de componente NumericStepper al escenario y asígnele el nombre `bandwidth_nstep`. A continuación, añada el código ActionScript siguiente al fotograma 1 de la línea de tiempo:

```
var bandwidth_nstep:mx.controls.NumericStepper;
var my_video:Video;
var my_cam:Camera = Camera.get();
my_video.attachVideo(my_cam);
this.createTextField("bandwidth_txt", this.getNextHighestDepth(), 0, 0,
  100, 22);
bandwidth_txt.autoSize = true;
this.onEnterFrame = function() {
  bandwidth_txt.text = "Camera is currently using "+my_cam.bandwidth+"
  bytes ("+Math.round(my_cam.bandwidth/1024)+" KB) bandwidth.";
};
//
bandwidth_nstep.minimum = 0;
bandwidth_nstep.maximum = 128;
bandwidth_nstep.stepSize = 16;
bandwidth_nstep.value = my_cam.bandwidth/1024;
```

```
function changeBandwidth(evt:Object) {
  my_cam.setQuality(evt.target.value 1024, 0);
}
bandwidth_nstep.addEventListener("change", changeBandwidth);
```

El método `MovieClip.getNextHighestDepth()` utilizado en este ejemplo necesita Flash Player 7 o una versión posterior. Si el archivo SWF incluye un componente de la versión 2, utilice la clase DepthManager de los componentes de la versión 2 en lugar del método `MovieClip.getNextHighestDepth()`.

Véase también

`setQuality (método Camera.setQuality)`

currentFps (propiedad Camera.currentFps)

```
public currentFps : Number [read-only]
```

Velocidad a la que está capturando datos la cámara, expresada en fotogramas por segundo. Esta propiedad no puede establecerse; sin embargo, puede utilizar el método `Camera.setMode()` para establecer una propiedad relacionada (`Camera.fps`) que especifica la velocidad de fotogramas máxima con la que desea que la cámara capture datos.

Disponibilidad: ActionScript 1.0; Flash Player 6

Ejemplo

El ejemplo siguiente detecta la velocidad en fotogramas por segundo con la que la cámara captura datos, utilizando la propiedad `currentFps` y una instancia ProgressBar: Cree una instancia de vídeo nueva seleccionando Nuevo vídeo en el menú Opciones de Biblioteca. Añada una instancia al escenario y asígnele el nombre de instancia `my_video`. Añada una instancia de componente ProgressBar al escenario y asígnele el nombre de instancia fps_pb. A continuación, añada el código ActionScript siguiente al fotograma 1 de la línea de tiempo:

```
var my_video:Video;
var fps_pb:mx.controls.ProgressBar;
var my_cam:Camera = Camera.get();
my_video.attachVideo(my_cam);
this.onEnterFrame = function() {
  fps_pb.setProgress(my_cam.fps-my_cam.currentFps, my_cam.fps);
};

fps_pb.setStyle("fontSize", 10);
fps_pb.setStyle("themeColor", "haloOrange");
fps_pb.labelPlacement = "top";
fps_pb.mode = "manual";
fps_pb.label = "FPS: %2 (%3%% dropped)";
```

Véase también

setMode (método Camera.setMode), fps (propiedad Camera.fps)

fps (propiedad Camera.fps)

```
public fps : Number [read-only]
```

Velocidad máxima a la que desea que la cámara capture datos, expresada en fotogramas por segundo. Es posible que la velocidad máxima dependa de las prestaciones de la cámara; es decir, si la cámara no admite el valor que usted establezca aquí, no se alcanzará dicha velocidad de fotogramas.

- Para establecer el valor que desee para esta propiedad, utilice `Camera.setMode()`.
- Para determinar la velocidad a la que está capturando datos la cámara, utilice la propiedad `Camera.currentFps`.

Disponibilidad: ActionScript 1.0; Flash Player 6

Ejemplo

El ejemplo siguiente detecta la velocidad en fotogramas por segundo con la que la cámara captura datos, utilizando la propiedad `currentFps` y una instancia ProgressBar: Cree una instancia de vídeo nueva seleccionando Nuevo vídeo en el menú Opciones de Biblioteca. Añada una instancia al escenario y asígnele el nombre de instancia `my_video`. Añada una instancia de componente ProgressBar al escenario y asígnele el nombre de instancia fps_pb. A continuación, añada el código ActionScript siguiente al fotograma 1 de la línea de tiempo:

```
var my_video:Video;
var fps_pb:mx.controls.ProgressBar;
var my_cam:Camera = Camera.get();
my_video.attachVideo(my_cam);
this.onEnterFrame = function() {
  fps_pb.setProgress(my_cam.fps-my_cam.currentFps, my_cam.fps);
};

fps_pb.setStyle("fontSize", 10);
fps_pb.setStyle("themeColor", "haloOrange");
fps_pb.labelPlacement = "top";
fps_pb.mode = "manual";
fps_pb.label = "FPS: %2 (%3%% dropped)";
```

Nota: La función `setMode()` no garantiza la configuración solicitada de fps; establece los fps que ha solicitado o los fps más rápidos disponibles.

Véase también

currentFps (propiedad Camera.currentFps), setMode (método Camera.setMode)

get (método Camera.get)

```
public static get([index:Number]) : Camera
```

Devuelve una referencia a un objeto Camera para capturar vídeo. Para comenzar a capturar vídeo, deberá asociar el objeto Camera a un objeto Video (véase `Video.attachVideo()`).

A diferencia de los objetos que cree utilizando el constructor `new`, varias llamadas a `Camera.get()` hacen referencia a la misma cámara. Por consiguiente, si el script contiene las líneas `first_cam = Camera.get()` y `second_cam = Camera.get()`, tanto `first_cam` como `second_cam` harán referencia a la misma cámara (predeterminada).

En general, no deberá pasar un valor para *`index`* ; simplemente utilice `Camera.get()` para devolver una referencia a la cámara predeterminada. A través del panel de configuración de cámara (descrito posteriormente en esta sección), el usuario puede especificar la cámara que debe utilizar Flash. Si pasa un valor para *`index`* , es posible que esté intentando hacer referencia a una cámara diferente a la que desea el usuario. Podría utilizar *`index`* en casos poco frecuentes (por ejemplo, si la aplicación está capturando vídeo de dos cámaras a la vez.

Cuando un archivo SWF intenta acceder a la cámara devuelta por `Camera.get()` Flash Player muestra un cuadro de diálogo Privacidad que permite al usuario autorizar o denegar el acceso a la cámara. (Asegúrese de que el tamaño del escenario sea de al menos 215 x 138 píxeles; este es el tamaño mínimo que exige Flash para mostrar el cuadro de diálogo.)

Cuando el usuario responde a este cuadro de diálogo, el controlador de eventos `Camera.onStatus` devuelve un objeto de información que indica la respuesta del usuario. Para determinar si el usuario ha denegado o autorizado el acceso a la cámara sin procesar este controlador de eventos, utilice la propiedad `Camera.muted`.

El usuario también puede especificar una configuración de privacidad permanente para un dominio concreto haciendo clic con el botón derecho del ratón (Windows) o haciendo clic con la tecla Control presionada (Macintosh) mientras se reproduce un archivo SWF, seleccionando Configuración, abriendo el panel Privacidad y seleccionando Recordar.

No puede utilizar ActionScript para establecer el valor Allow (autorizar) o Deny (denegar) para un usuario, pero puede mostrar el panel Privacidad para el usuario utilizando `System.showSettings(0)`. Si el usuario selecciona Recordar, Flash Player ya no mostrará el cuadro de diálogo Privacidad para archivos SWF de este dominio.

Si `Camera.get` devuelve el valor `null` (nulo), indicará que la cámara está siendo utilizada por otra aplicación o que no hay cámaras instaladas en el sistema. Para comprobar si hay cámaras instaladas, utilice `Camera.names.length`. Para ver el panel de configuración de la cámara de Flash Player, en el que el usuario puede elegir la cámara a la que hará referencia `Camera.get()`, utilice `System.showSettings(3)`.

La exploración del hardware en busca de cámaras lleva tiempo. Cuando Flash encuentra al menos una cámara, el hardware no vuelve a explorarse durante toda la vida de la instancia del reproductor. No obstante, si Flash no encuentra ninguna cámara, la buscará cada vez que se llame a `Camera.get`. Esto resulta útil si un usuario ha olvidado conectar la cámara; si el archivo SWF ofrece un botón Intentar de nuevo que llama a `Camera.get`, Flash podrá localizar la cámara sin que el usuario tenga que reiniciar el archivo SWF.

Nota: La sintaxis correcta es `Camera.get()`. Para asignar el objeto Camera a una variable, utilice una sintaxis del tipo `active_cam = Camera.get()`.

Disponibilidad: ActionScript 1.0; Flash Player 6

Parámetros

`index``:Number` [opcional] - Entero basado en cero que especifica qué cámara se obtiene, en función de la matriz devuelta por la propiedad `Camera.names`. Para obtener la cámara predeterminada (que se recomienda para la mayoría de las aplicaciones), omita este parámetro.

Valor devuelto

`Camera` - Si no se especifica `index`, este método devuelve una referencia a la cámara predeterminada o, si está siendo utilizada por otra aplicación, a la primera cámara disponible. (Si hay varias cámaras instaladas, el usuario puede especificar la cámara predeterminada en el panel Parámetros de la cámara de Flash Player.) Si no hay cámaras disponibles o instaladas, el método devuelve el valor `null` (nulo). Si se especifica `index`, este método devuelve una referencia al micrófono solicitado, o `null` si no está disponible.

Ejemplo

El ejemplo siguiente permite seleccionar una cámara activa para utilizar desde un instancia ComboBox. La cámara activa actualmente se mostrará en una instancia Label. Cree una instancia de vídeo nueva seleccionando Nuevo vídeo en el menú Opciones de Biblioteca. Añada una instancia al escenario y asígnele el nombre de instancia `my_video`. Añada una instancia de componente al escenario y asígnele el nombre de instancia camera_lbl, y una instancia de componente ComboBox y asígnele el nombre de instancia `cameras_cb`. A continuación, añada el código ActionScript siguiente al fotograma 1 de la línea de tiempo:

```
var my_cam:Camera = Camera.get();
var my_video:Video;
my_video.attachVideo(my_cam);
var camera_lbl:mx.controls.Label;
var cameras_cb:mx.controls.ComboBox;
camera_lbl.text = my_cam.name;
cameras_cb.dataProvider = Camera.names;
function changeCamera():Void {
  my_cam = Camera.get(cameras_cb.selectedIndex);
```

```
  my_video.attachVideo(my_cam);
  camera_lbl.text = my_cam.name;
}
cameras_cb.addEventListener("change", changeCamera);
camera_lbl.setStyle("fontSize", 9);
cameras_cb.setStyle("fontSize", 9);
```

Véase también

index (propiedad Camera.index), muted (propiedad Camera.muted), names (propiedad Camera.names), onStatus (controlador Camera.onStatus), setMode (método Camera.setMode), showSettings (método System.showSettings), attachVideo (método Video.attachVideo)

height (propiedad Camera.height)

```
public height : Number [read-only]
```

Altura de captura actual, expresada en píxeles. Para establecer un valor para esta propiedad, utilice `Camera.setMode()`.

Disponibilidad: ActionScript 1.0; Flash Player 6

Ejemplo

El código siguiente muestra la anchura, la altura y los FPS actuales de una instancia de vídeo en una instancia de componente Label del escenario. Cree una instancia de vídeo nueva seleccionando Nuevo vídeo en el menú Opciones de Biblioteca. Añada una instancia al escenario y asígnele el nombre de instancia `my_video`. Añada una instancia de componente Label al escenario y asígnele el nombre de instancia dimensions_lbl. A continuación, añada el código ActionScript siguiente al fotograma 1 de la línea de tiempo:

```
var my_cam:Camera = Camera.get();
var my_video:Video;
my_video.attachVideo(my_cam);
var dimensions_lbl:mx.controls.Label;
dimensions_lbl.setStyle("fontSize", 9);
dimensions_lbl.setStyle("fontWeight", "bold");
dimensions_lbl.setStyle("textAlign", "center");
dimensions_lbl.text = "width: "+my_cam.width+", height: "+my_cam.height+",
  FPS: "+my_cam.fps;
```

Consulte también el ejemplo de `Camera.setMode()`.

Véase también

width (propiedad Camera.width), setMode (método Camera.setMode)

index (propiedad Camera.index)

```
public index : Number [read-only]
```

Entero basado en cero que especifica el índice de la cámara, como se refleja en la matriz devuelta por `Camera.names`.

Disponibilidad: ActionScript 1.0; Flash Player 6

Ejemplo

El ejemplo siguiente muestra una matriz de cámaras en un campo de texto que se crea en tiempo de ejecución e indica qué cámara se está utilizando actualmente. Cree una instancia de vídeo nueva seleccionando Nuevo vídeo en el menú Opciones de Biblioteca. Añada una instancia al escenario y asígnele el nombre de instancia `my_video`. Añada una instancia de componente Label al escenario y asígnele el nombre de instancia camera_lbl. A continuación, añada el código ActionScript siguiente al fotograma 1 de la línea de tiempo:

```
var camera_lbl:mx.controls.Label;
var my_cam:Camera = Camera.get();
var my_video:Video;
my_video.attachVideo(my_cam);

camera_lbl.text = my_cam.index+". "+my_cam.name;
this.createTextField("cameras_txt", this.getNextHighestDepth(), 25, 160,
  160, 80);
cameras_txt.html = true;
cameras_txt.border = true;
cameras_txt.wordWrap = true;
cameras_txt.multiline = true;
for (var i = 0; i<Camera.names.length; i++) {
  cameras_txt.htmlText += "<li><u><a
  href=\"asfunction:changeCamera,"+i+"\">"+Camera.names[i]+"</a></u></
  li>";
}
function changeCamera(index:Number) {
  my_cam = Camera.get(index);
  my_video.attachVideo(my_cam);
  camera_lbl.text = my_cam.index+". "+my_cam.name;
}
```

El método `MovieClip.getNextHighestDepth()` utilizado en este ejemplo necesita Flash Player 7 o una versión posterior. Si el archivo SWF incluye un componente de la versión 2, utilice la clase DepthManager de los componentes de la versión 2 en lugar del método `MovieClip.getNextHighestDepth()`.

Véase también

`names (propiedad Camera.names)`, `get (método Camera.get)`

motionLevel (propiedad Camera.motionLevel)

```
public motionLevel : Number [read-only]
```

Valor numérico que especifica la cantidad de movimiento necesario para invocar `Camera.onActivity(true)`. El rango de valores admitidos va de 0 a 100. El valor predeterminado es 50.

Puede mostrarse vídeo con independencia del valor de la propiedad `motionLevel`. Para más información, consulte `Camera.setMotionLevel()`.

Disponibilidad: ActionScript 1.0; Flash Player 6

Ejemplo

El ejemplo siguiente detecta continuamente el nivel de movimiento de una cámara. Cree una instancia de vídeo nueva seleccionando Nuevo vídeo en el menú Opciones de Biblioteca. Añada una instancia al escenario y asígnele el nombre de instancia `my_video`. Añada una instancia de componente Label al escenario y asígnele el nombre motionLevel_lbl, un componente NumericStepper con el nombre de instancia `motionLevel_nstep` y un componente ProgressBar con el nombre de instancia motion_pb. A continuación, añada el código ActionScript siguiente al fotograma 1 de la línea de tiempo:

```
var my_cam:Camera = Camera.get();
var my_video:Video;
my_video.attachVideo(my_cam);

// configure the ProgressBar component instance
var motion_pb:mx.controls.ProgressBar;
motion_pb.mode = "manual";
motion_pb.label = "Motion: %3%%";

var motionLevel_lbl:mx.controls.Label;
// configure the NumericStepper component instance
var motionLevel_nstep:mx.controls.NumericStepper;
motionLevel_nstep.minimum = 0;
motionLevel_nstep.maximum = 100;
motionLevel_nstep.stepSize = 5;
motionLevel_nstep.value = my_cam.motionLevel;

// Continuously update the progress of the ProgressBar component instance to
  the activityLevel
// of the current Camera instance, which is defined in my_cam
this.onEnterFrame = function() {
  motion_pb.setProgress(my_cam.activityLevel, 100);
};
```

```
// When the level of activity goes above or below the number defined in
  Camera.motionLevel,
// trigger the onActivity event handler.
my_cam.onActivity = function(isActive:Boolean) {
  // If isActive equals true, set the themeColor variable to "haloGreen".
  // Otherwise set the themeColor to "haloOrange".
  var themeColor:String = (isActive) ? "haloGreen" : "haloOrange";
  motion_pb.setStyle("themeColor", themeColor);
};

function changeMotionLevel() {
  // Set the motionLevel property for my_cam Camera instance to the value
  of the NumericStepper
  // component instance. Maintain the current motionTimeOut value of the
  my_cam Camera instance.
  my_cam.setMotionLevel(motionLevel_nstep.value, my_cam.motionTimeOut);
}
motionLevel_nstep.addEventListener("change", changeMotionLevel);
```

Véase también

onActivity (controlador Camera.onActivity), onStatus (controlador Camera.onStatus), setMotionLevel (método Camera.setMotionLevel), activityLevel (propiedad Camera.activityLevel)

motionTimeOut (propiedad Camera.motionTimeOut)

```
public motionTimeOut : Number [read-only]
```

Número de milisegundos que transcurren entre el momento en que la cámara deja de detectar movimiento y la invocación de `Camera.onActivity(false)`. El valor predeterminado es 2000 (2 segundos).

Para definir este valor, utilice `Camera.setMotionLevel()`.

Disponibilidad: ActionScript 1.0; Flash Player 6

Ejemplo

En el ejemplo siguiente, la instancia ProgressBar cambia el color de su tema Halo cuando el nivel de actividad se sitúa por debajo del nivel de movimiento. Puede establecer el número de segundos de la propiedad `motionTimeout` utilizando una instancia NumericStepper. Cree una instancia de vídeo nueva seleccionando Nuevo vídeo en el menú Opciones de Biblioteca. Añada una instancia al escenario y asígnele el nombre de instancia `my_video`. Añada una instancia de componente Label al escenario y asígnele el nombre de instancia motionLevel_lbl, un componente NumericStepper con el nombre de instancia motionTimeOut_nstep y un componente ProgressBar con el nombre de instancia motion_pb. A continuación, añada el código ActionScript siguiente al fotograma 1 de la línea de tiempo:

```
var motionLevel_lbl:mx.controls.Label;
var motion_pb:mx.controls.ProgressBar;
var motionTimeOut_nstep:mx.controls.NumericStepper;
var my_cam:Camera = Camera.get();
var my_video:Video;
my_video.attachVideo(my_cam);

this.onEnterFrame = function() {
  motionLevel_lbl.text = "activityLevel: "+my_cam.activityLevel;
};

motion_pb.indeterminate = true;
my_cam.onActivity = function(isActive:Boolean) {
  if (isActive) {
  motion_pb.setStyle("themeColor", "haloGreen");
  motion_pb.label = "Motion is above "+my_cam.motionLevel;
  } else {
  motion_pb.setStyle("themeColor", "haloOrange");
  motion_pb.label = "Motion is below "+my_cam.motionLevel;
  }
};
function changeMotionTimeOut() {
  my_cam.setMotionLevel(my_cam.motionLevel, motionTimeOut_nstep.value
  1000);
}
motionTimeOut_nstep.addEventListener("change", changeMotionTimeOut);
motionTimeOut_nstep.value = my_cam.motionTimeOut/1000;
```

Véase también

setMotionLevel (método Camera.setMotionLevel), onActivity (controlador Camera.onActivity)

muted (propiedad Camera.muted)

```
public muted : Boolean [read-only]
```

Valor booleano que especifica si el usuario ha denegado el acceso a la cámara (`true`) o lo ha autorizado (`false`) en el panel Parámetros de privacidad de Flash Player. Cuando este valor cambia, se invoca `Camera.onStatus`. Para más información, consulte `Camera.get()`.

Disponibilidad: ActionScript 1.0; Flash Player 6

Ejemplo

En el ejemplo siguiente se puede mostrar un mensaje de error si `my_cam.muted` da como resultado true. Cree una instancia de vídeo nueva seleccionando Nuevo vídeo en el menú Opciones de Biblioteca. Añada una instancia al escenario y asígnele el nombre de instancia `my_video`. A continuación, añada el código ActionScript siguiente al fotograma 1 de la línea de tiempo:

```
var my_cam:Camera = Camera.get();
var my_video:Video;
my_video.attachVideo(my_cam);
my_cam.onStatus = function(infoObj:Object) {
  if (my_cam.muted) {
  // If user is denied access to their Camera, you can display an error
  message here. You can display the user's Camera/Privacy settings again
  using System.showSettings(0);
  trace("User denied access to Camera");
  System.showSettings(0);
  }
};
```

Véase también

`get (método Camera.get)`, `onStatus (controlador Camera.onStatus)`

name (propiedad Camera.name)

```
public name : String [read-only]
```

Cadena que especifica el nombre de la cámara actual tal y como lo ha devuelto el hardware de la cámara.

Disponibilidad: ActionScript 1.0; Flash Player 6

Ejemplo

En el ejemplo siguiente se muestra en un cuadro de texto el nombre de la cámara predeterminada: En Windows, este nombre es el mismo que el del dispositivo que figura en Escáneres y cámaras del Panel de control. Cree una instancia de vídeo nueva seleccionando Nuevo vídeo en el menú Opciones de Biblioteca. Añada una instancia al escenario y asígnele el nombre de instancia `my_video`. A continuación, añada el código ActionScript siguiente al fotograma 1 de la línea de tiempo:

```
var my_cam:Camera = Camera.get();
var my_video:Video;
my_video.attachVideo(my_cam);

this.createTextField("name_txt", this.getNextHighestDepth(), 0, 0, 100,
  22);
name_txt.autoSize = true;
name_txt.text = my_cam.name;
```

El método `MovieClip.getNextHighestDepth()` utilizado en este ejemplo necesita Flash Player 7 o una versión posterior. Si el archivo SWF incluye un componente de la versión 2, utilice la clase DepthManager de los componentes de la versión 2 en lugar del método `MovieClip.getNextHighestDepth()`.

Véase también

`get (método Camera.get)`, `names (propiedad Camera.names)`

names (propiedad Camera.names)

```
public static names : Array [read-only]
```

Recupera una matriz de cadenas que refleja los nombres de todas las cámaras disponibles sin mostrar el panel Parámetros de privacidad de Flash Player. Esta matriz se comporta de la misma forma que cualquier otra matriz ActionScript y proporciona implícitamente el índice basado en cero de cada cámara y el número de cámaras existentes en el sistema (mediante `Camera.names.length`). Para más información, consulte la entrada de la clase de matriz `Camera.names`.

La llamada a la propiedad `Camera.names` exige un examen amplio del hardware y puede tardar varios segundos en crear la matriz. En la mayoría de los casos, podrá utilizar simplemente la cámara predeterminada.

Nota: La sintaxis correcta es `Camera.names`. Para asignar el valor devuelto a una variable, utilice una sintaxis del tipo *`cam_array`* `= Camera.names`. Para determinar el nombre de la cámara actual, utilice *`active_cam`* `.name`.

Disponibilidad: ActionScript 1.0; Flash Player 6

Ejemplo

El ejemplo siguiente utiliza la cámara predeterminada a menos que haya disponibles varias cámaras, en cuyo caso el usuario puede elegir cuál de ellas desea establecer como predeterminada. Si sólo hay una cámara, se utilizará la cámara predeterminada. Cree una instancia de vídeo nueva seleccionando Nuevo vídeo en el menú Opciones de Biblioteca. Añada una instancia al escenario y asígnele el nombre de instancia `my_video`. A continuación, añada el código ActionScript siguiente al fotograma 1 de la línea de tiempo:

```
var my_video:Video;
var cam_array:Array = Camera.names;
if (cam_array.length>1) {
  System.showSettings(3);
}
var my_cam:Camera = Camera.get();
my_video.attachVideo(my_cam);
```

Véase también

`get (método Camera.get)`, `index (propiedad Camera.index)`, `name (propiedad Camera.name)`

onActivity (controlador Camera.onActivity)

```
onActivity = function(active:Boolean) {}
```

Se invoca cuando la cámara comienza o deja de detectar movimiento. Si desea responder a este controlador de eventos, deberá crear una función que procese su valor de actividad.

Para especificar la cantidad de movimiento necesario para invocar `Camera.onActivity(true)` y el tiempo que debe transcurrir sin que exista actividad para que se invoque Camera.on`Activity(false)`, utilice `Camera.setMotionLevel()`.

Disponibilidad: ActionScript 1.0; Flash Player 6

Parámetros

`active``:Boolean` - Un valor booleano definido en true cuando la cámara comienza detectar movimiento y en false cuando deja de hacerlo.

Ejemplo

El ejemplo siguiente muestra true o false en el panel Salida cuando la cámara comienza o deja de detectar movimiento:

```
// Assumes a Video object named "myVideoObject" is on the Stage
active_cam = Camera.get();
myVideoObject.attachVideo(active_cam);
active_cam.setMotionLevel(10, 500);
```

```
active_cam.onActivity = function(mode)
{
  trace(mode);
}
```

Véase también

setMotionLevel (método Camera.setMotionLevel)

onStatus (controlador Camera.onStatus)

```
onStatus = function(infoObject:Object) {}
```

Se invoca cuando el usuario autoriza o deniega el acceso a la cámara. Si desea responder a este controlador de eventos, deberá crear una función que procese el objeto de información generado por la cámara.

Cuando un archivo SWF intenta acceder a la cámara, Flash Player muestra un cuadro de diálogo Privacidad que permite al usuario autorizar o denegar el acceso.

- Si el usuario autoriza el acceso, la propiedad `Camera.muted` se establece como false y se invoca su controlador con un objeto de información cuya propiedad de código es "Camera.Unmuted" y cuya propiedad de nivel es "Status".
- Si el usuario deniega el acceso, la propiedad `Camera.muted` se establece como true y se invoca su controlador con un objeto de información cuya propiedad de código es "Camera.Muted" y cuya propiedad de nivel es "Status".

Para determinar si el usuario ha denegado o autorizado el acceso a la cámara sin procesar este controlador de eventos, utilice la propiedad `Camera.muted`.

Nota: Si el usuario opta por permitir o denegar permanentemente el acceso a todos los archivos SWF de un determinado dominio, no se invocará este controlador para archivos SWF de dicho dominio a no ser que el usuario posteriormente cambie la configuración de privacidad. Para más información, consulte `Camera.get()`.

Disponibilidad: ActionScript 1.0; Flash Player 6

Parámetros

`infoObject``:Object` - Un parámetro definido de acuerdo con el mensaje de estado.

Ejemplo

El código ActionScript siguiente se utiliza para mostrar un mensaje cada vez que el usuario permite o rechaza el acceso a la cámara:

```
var my_cam:Camera = Camera.get();
var my_video:Video;
```

```
my_video.attachVideo(my_cam);
my_cam.onStatus = function(infoObj:Object) {
  switch (infoObj.code) {
  case 'Camera.Muted' :
  trace("Camera access is denied");
  break;
  case 'Camera.Unmuted' :
  trace("Camera access granted");
  break;
  }
}
```

Véase también

`get (método Camera.get)`, `muted (propiedad Camera.muted)`, `showSettings (método System.showSettings)`, `onStatus (controlador System.onStatus)`

quality (propiedad Camera.quality)

```
public quality : Number [read-only]
```

Entero que especifica el nivel necesario de calidad de imagen, tal y como lo determina la cantidad de compresión aplicada a cada fotograma de vídeo. Los valores de calidad (quality) admitidos van del 1 (la calidad más baja, máxima compresión) al 100 (la calidad más alta, sin compresión). El valor predeterminado es 0, que hace que la calidad de imagen varíe si es preciso para no superar el ancho de banda disponible.

Disponibilidad: ActionScript 1.0; Flash Player 6

Ejemplo

El ejemplo siguiente utiliza una instancia NumericStepper para especificar la cantidad de compresión aplicada a la cámara. Cree una instancia de vídeo nueva seleccionando Nuevo vídeo en el menú Opciones de Biblioteca. Añada una instancia al escenario y asígnele el nombre de instancia `my_video`. Añada un componente NumericStepper con el nombre de instancia quality_nstep. A continuación, añada el código ActionScript siguiente al fotograma 1 de la línea de tiempo:

```
var quality_nstep:mx.controls.NumericStepper;

var my_cam:Camera = Camera.get();
var my_video:Video;
my_video.attachVideo(my_cam);

quality_nstep.minimum = 0;
quality_nstep.maximum = 100;
quality_nstep.stepSize = 5;
quality_nstep.value = my_cam.quality;
```

```
function changeQuality() {
  my_cam.setQuality(my_cam.bandwidth, quality_nstep.value);
}
quality_nstep.addEventListener("change", changeQuality);
```

Véase también

setQuality (método Camera.setQuality)

setMode (método Camera.setMode)

```
public setMode([width:Number], [height:Number], [fps:Number],
  [favorArea:Boolean]) : Void
```

Establece el modo de captura de cámara con el modo nativo que mejor se adapte a los requisitos especificados. Si la cámara no dispone de ningún modo nativo que coincida con todos los parámetros que usted ha pasado, Flash seleccionará el modo de captura que mejor sintetice el modo solicitado. Esta manipulación puede implicar el recorte de la imagen y la eliminación de fotogramas.

De manera predeterminada, Flash elimina los fotogramas que sean necesarios para mantener el tamaño de la imagen. Para minimizar el número de fotogramas eliminados, aunque ello implique la reducción del tamaño de la imagen, pase el valor `false` para el parámetro `favorArea`.

Al elegir un modo nativo, Flash intenta mantener la proporción solicitada siempre que ello sea posible. Por ejemplo, si emite el comando *`active_cam`*`.setMode(400, 400, 30)` y los valores máximos de anchura y altura disponibles en la cámara son 320 y 288, Flash establece tanto la anchura como al altura en 288; al establecer estas propiedades con el mismo valor, Flash mantiene la proporción 1:1 solicitada.

Para determinar los valores asignados a estas propiedades después de que Flash seleccione el modo más próximo a los valores que ha solicitado, utilice `Camera.width`, `Camera.height` y `Camera.fps`.

Disponibilidad: ActionScript 1.0; Flash Player 6

Parámetros

`width``:Number` [opcional] - Anchura de captura solicitada, expresada en píxeles. El valor predeterminado es 160.

`height``:Number` [opcional] - Altura de captura solicitada, expresada en píxeles. El valor predeterminado es 120.

fps:Number [opcional] - Velocidad solicitada a la que la cámara debería capturar los datos, expresada en fotogramas por segundo. El valor predeterminado es 15.

favorArea:Boolean [opcional] - Valor booleano que especifica cómo manipular la anchura, altura y velocidad de reproducción de fotogramas si la cámara no tiene un modo nativo que cumpla los requisitos especificados. El valor predeterminado es `true`, lo que significa que se favorece el mantenimiento del tamaño de captura; al utilizar este parámetro se selecciona el modo más próximo a los valores de *width* y *height*, aunque afecte negativamente al rendimiento debido a la reducción en la velocidad de reproducción de fotogramas. Para maximizar la velocidad de reproducción de fotogramas a costa de la altura y anchura de la cámara, pase `false` para el parámetro `favorArea`.

Ejemplo

El ejemplo siguiente establece el modo de captura de la cámara. Puede escribir una velocidad de reproducción de fotogramas en una instancia TextInput y pulsar Intro o Retorno para aplicar la velocidad de reproducción de fotogramas. Cree una instancia de vídeo nueva seleccionando Nuevo vídeo en el menú Opciones de Biblioteca. Añada una instancia al escenario y asígnele el nombre de instancia `my_video`. Añada una instancia de componente TextInput con el nombre de instancia fps_ti. A continuación, añada el código ActionScript siguiente al fotograma 1 de la línea de tiempo:

```
var my_cam:Camera = Camera.get();
var my_video:Video;
my_video.attachVideo(my_cam);

fps_ti.maxChars = 2;
fps_ti.restrict = [0-9];
fps_lbl.text = "Current: "+my_cam.fps+" fps";

function changeFps():Void {
  my_cam.setMode(my_cam.width, my_cam.height, fps_ti.text);
  fps_lbl.text = "Current: "+my_cam.fps+" fps";
  fps_ti.text = my_cam.fps;
  Selection.setSelection(0,2);
}
fps_ti.addEventListener("enter", changeFps);
```

Véase también

fps (propiedad Camera.fps), height (propiedad Camera.height), width (propiedad Camera.width), currentFps (propiedad Camera.currentFps)

setMotionLevel (método Camera.setMotionLevel)

```
public setMotionLevel([motionLevel:Number], [timeOut:Number]) : Void
```

Especifica la cantidad de movimiento necesaria para invocar `Camera.onActivity(true)`. Opcionalmente, establezca el número de milisegundos que deben transcurrir sin que exista actividad para que Flash considere que el movimiento ha desaparecido e invoque `Camera.onActivity(false)`.

Nota: Puede mostrarse vídeo con independencia del valor del parámetro `sensitivity`. Este parámetro sólo determina cuándo y en qué circunstancias debe invocarse `Camera.onActivity` (no determina si realmente se está capturando o mostrando vídeo).

- Para impedir que la cámara detecte movimiento , pase el valor 100 para *`sensitivity`* ; `Camera.onActivity` nunca se invocará. (Lo más probable es que desee utilizar este valor solamente para realizar pruebas, por ejemplo, para desactivar temporalmente las acciones configuradas para que tengan lugar cuando se invoque `Camera.onActivity`.)
- Para determinar la cantidad de movimiento que está detectando actualmente la cámara, utilice la propiedad `Camera.activityLevel`.
- Los valores de sensibilidad al movimiento se corresponden directamente con los valores de actividad. La ausencia total de movimiento tiene un valor de actividad 0, mientras que el movimiento constante tiene el valor de actividad 100. Su valor de actividad es inferior al valor de sensibilidad al movimiento cuando no se está movimiento; cuando se está movimiento, los valores de actividad normalmente superan al valor de sensibilidad al movimiento.
- Este método tiene una finalidad similar a la de `Microphone.setSilenceLevel()`; ambos métodos se utilizan para especificar cuándo debe invocarse el controlador de eventos `onActivity`. No obstante, estos métodos tienen efectos muy distintos en la publicación de flujos:
- `Microphone.setSilenceLevel()` está diseñado para optimizar el ancho de banda. Cuando se considera que un flujo de audio está en silencio, no se envían datos de audio. Por el contrario, se envía un único mensaje que indica que el silencio ha comenzado.
- `Camera.setMotionLevel()` está diseñado para detectar movimiento y no afecta al uso del ancho de banda. Aunque un flujo de vídeo no detecte movimiento, el vídeo continúa enviándose.

Disponibilidad: ActionScript 1.0; Flash Player 6

Parámetros

motionLevel:Number [opcional] - Valor numérico que especifica la cantidad de movimiento necesario para invocar Camera.onActivity(true). El rango de valores admitidos va de 0 a 100. El valor predeterminado es 50.

timeOut:Number [opcional] - Parámetro numérico que especifica cuántos milisegundos deben transcurrir sin actividad antes de que Flash considere que la actividad se ha detenido e invoque al controlador de eventos Camera.onActivity(false). El valor predeterminado es 2000 (2 segundos).

Ejemplo

El ejemplo siguiente envía mensajes al panel Salida cuando se inicia o detiene la actividad de vídeo. Cambie el valor de sensibilidad al movimiento de 30 a un valor mayor o menor para comprobar cómo afectan a la detección del movimiento.

```
// Assumes a Video object named "myVideoObject" is on the Stage
active_cam = Camera.get();
x = 0;
function motion(mode) {
  trace(x + ": " + mode);
  x++;
}
active_cam.onActivity = function(mode) {
  motion(mode);
}
active_cam.setMotionLevel(30, 500);
myVideoObject.attachVideo(active_cam);
```

Véase también

motionLevel (propiedad Camera.motionLevel), motionTimeOut (propiedad Camera.motionTimeOut), onActivity (controlador Camera.onActivity), activityLevel (propiedad Camera.activityLevel)

setQuality (método Camera.setQuality)

```
public setQuality([bandwidth:Number], [quality:Number]) : Void
```

Establece la cantidad máxima de ancho de banda por segundo o la calidad de imagen requerida para la salida de vídeo actual. Este método sólo es aplicable con carácter general si está transmitiendo vídeo mediante Flash Communication Server.

Utilice este método para especificar qué elemento de la salida de vídeo es más importante para su aplicación: el uso del ancho de banda o la calidad de imagen.

- Para indicar que el uso del ancho de banda tiene prioridad, pase un valor para *bandwidth* y 0 para *frameQuality*. Flash transmitirá vídeo con la mayor calidad que permita el ancho de banda especificado. Si es preciso, Flash reducirá la calidad de imagen para evitar superar el ancho de banda especificado. En general, a mayor movimiento, menor calidad.
- Para indicar que la calidad tiene prioridad, pase 0 para *bandwidth* y un valor numérico para *frameQuality*. Flash utilizará el ancho de banda que sea necesario para mantener la calidad especificada. Si es preciso, Flash reducirá la velocidad de fotogramas para mantener la calidad de imagen. En general, a mayor movimiento, mayor uso del ancho de banda.
- Para especificar que el uso del ancho de banda y la calidad tienen idéntica importancia, pase valores numéricos para ambos parámetros. Flash transmitirá vídeo con la calidad especificada sin superar el ancho de banda indicado. Si es preciso, Flash reducirá la velocidad de fotogramas para mantener la calidad de imagen sin superar el ancho de banda especificado.

Disponibilidad: ActionScript 1.0; Flash Player 6

Parámetros

bandwidth:Number [opcional] - Entero que especifica la cantidad máxima de ancho de banda, expresada en bytes por segundo, que puede utilizar la salida de vídeo actual. Para especificar que el vídeo de Flash puede utilizar todo el ancho de vídeo necesario para mantener el valor de *frameQuality*, pase 0 para *bandwidth*. El valor predeterminado es 16384.

quality:Number [opcional] - Entero que especifica el nivel necesario de calidad de imagen, tal y como lo determina la cantidad de compresión aplicada a cada fotograma de vídeo. Los valores admitidos van del 1 (la calidad más baja, máxima compresión) al 100 (la calidad más alta, sin compresión). Para especificar que la calidad de vídeo puede variar según las necesidades para evitar superar el ancho de vídeo, pase 0 para *quality*. El valor predeterminado es 0.

Ejemplo

Los ejemplos siguientes ilustran cómo utilizar este método para controlar el uso del ancho de banda y la calidad de imagen.

```
// Ensure that no more than 8192 (8K/second) is used to send video
active_cam.setQuality(8192,0);

// Ensure that no more than 8192 (8K/second) is used to send video
// with a minimum quality of 50
```

```
active_cam.setQuality(8192,50);

// Ensure a minimum quality of 50, no matter how much bandwidth it takes
active_cam.setQuality(0,50);
```

Véase también

get (método Camera.get), quality (propiedad Camera.quality), bandwidth (propiedad Camera.bandwidth)

width (propiedad Camera.width)

```
public width : Number [read-only]
```

Anchura de captura actual, expresada en píxeles. Para establecer el valor que desee para esta propiedad, utilice `Camera.setMode()`.

Disponibilidad: ActionScript 1.0; Flash Player 6

Ejemplo

El código siguiente muestra la anchura, la altura y los FPS actuales de una instancia de vídeo en una instancia de componente Label del escenario. Cree una instancia de vídeo nueva seleccionando Nuevo vídeo en el menú Opciones de Biblioteca. Añada una instancia al escenario y asígnele el nombre de instancia `my_video`. Añada una instancia de componente Label al escenario y asígnele el nombre de instancia dimensions_lbl. A continuación, añada el código ActionScript siguiente al fotograma 1 de la línea de tiempo:

```
var my_cam:Camera = Camera.get();
var my_video:Video;
my_video.attachVideo(my_cam);
var dimensions_lbl:mx.controls.Label;
dimensions_lbl.setStyle("fontSize", 9);
dimensions_lbl.setStyle("fontWeight", "bold");
dimensions_lbl.setStyle("textAlign", "center");
dimensions_lbl.text = "width: "+my_cam.width+", height: "+my_cam.height+",
  FPS: "+my_cam.fps;
```

Consulte también el ejemplo de `Camera.setMode()`.

Véase también

height (propiedad Camera.height), setMode (método Camera.setMode)

capabilities (System.capabilities)

```
Object
  |
  +-System.capabilities
```

```
public class capabilities
extends Object
```

La clase Capabilities determina la capacidad del sistema y del reproductor que alojan un archivo SWF, lo que permite adaptar el contenido a los diferentes formatos. Por ejemplo, la pantalla de un teléfono móvil (en blanco y negro, 100 píxeles cuadrados) es diferente a una pantalla de PC en color de 1000 píxeles cuadrados. Con el fin de ofrecer el contenido adecuado al mayor número posible de usuarios, puede utilizar el objeto `System.capabilities` para determinar el tipo de dispositivo de que dispone un usuario. Luego podrá indicar al servidor que envíe archivos SWF diferentes en función de las prestaciones del dispositivo o indicar al archivo SWF que modifique su presentación dependiendo de las prestaciones del dispositivo.

Puede enviar información de prestaciones utilizando el método HTTP `GET` o `POST`. En el siguiente ejemplo se muestra una cadena de servidor para un equipo compatible con MP3, con resolución de 1600 x 1200 píxeles y que ejecuta Windows XP y Flash Player 8 (8.0.0.0):

```
A=t&SA=t&SV=t&EV=t&MP3=t&AE=t&VE=t&ACC=f&PR=t&SP=t&

SB=f&DEB=t&V=WIN%208%2C0%2C0%2C0&M=Macromedia%20Windows&

R=1600x1200&DP=72&COL=color&AR=1.0&OS=Windows%20XP&

L=en&PT=External&AVD=f&LFD=f&WD=f"
```

Todas la propiedades del objeto `System.capabilities` son de sólo lectura.

Disponibilidad: ActionScript 1.0; Flash Player 6

Resumen de propiedades

Modificadores	Propiedad	Descripción
static	avHardwareDisable:Boolean [read-only]	Valor booleano que especifica si el acceso a la cámara y al micrófono del usuario ha sido prohibido (true) o autorizado (false) administrativamente.
static	hasAccessibility:Boolean [read-only]	Valor booleano que da como resultado true si el reproductor se está ejecutando en un entorno que admite la comunicación entre Flash Player y las ayudas de accesibilidad; en caso contrario, da como resultado false.
static	hasAudio:Boolean [read-only]	Especifica si el sistema tiene prestaciones de audio.
static	hasAudioEncoder:Boolean [read-only]	Especifica si Flash Player puede codificar un flujo de audio.
static	hasEmbeddedVideo:Boolean [read-only]	Valor booleano que da como resultado true si el reproductor se está ejecutando en un sistema que admite vídeo incorporado; en caso contrario, da como resultado false.
static	hasIME:Boolean [read-only]	Indica si hay un editor de método de entrada (IME) instalado en el sistema.
static	hasMP3:Boolean [read-only]	Especifica si el sistema dispone de un descodificador de MP3.
static	hasPrinting:Boolean [read-only]	Valor booleano que da como resultado true si el reproductor se está ejecutando en un sistema que admite impresión; en caso contrario, da como resultado false.
static	hasScreenBroadcast:Boolean [read-only]	Valor booleano que da como resultado true si el reproductor admite el desarrollo de aplicaciones de difusión de pantallas para su ejecución a través de Flash Communication Server; en caso contrario, da como resultado false.
static	hasScreenPlayback:Boolean [read-only]	Valor booleano que da como resultado true si el reproductor admite la reproducción de aplicaciones de difusión de pantallas que se están ejecutando a través de Flash Communication Server; en caso contrario, da como resultado false.

Modificadores	Propiedad	Descripción
static	hasStreamingAudio:Boolean [read-only]	Valor booleano que da como resultado true si el reproductor es capaz de reproducir audio sin interrupción; en caso contrario, da como resultado false.
static	hasStreamingVideo:Boolean [read-only]	Valor booleano que da como resultado true si el reproductor es capaz de reproducir vídeo sin interrupción; en caso contrario, da como resultado false.
static	hasVideoEncoder:Boolean [read-only]	Especifica si Flash Player puede codificar un flujo de vídeo.
static	isDebugger:Boolean [read-only]	Valor booleano que especifica si el reproductor es de una versión lanzada oficialmente (false) o una versión especial de depuración (true).
static	language:String [read-only]	Indica el idioma del sistema en el que se está ejecutando el reproductor.
static	localFileReadDisable:Boolean [read-only]	Valor booleano que indica si el acceso de lectura al disco duro del usuario ha sido prohibido (true) o autorizado (false) administrativamente.
static	manufacturer:String [read-only]	Cadena que indica el fabricante de Flash Player, con el formato Macromedia *OSName*" (*OSName* puede ser "Windows", "Macintosh", "Linux" o "Other OS Name").
static	os:String [read-only]	Cadena que indica el sistema operativo actual.
static	pixelAspectRatio:Number [read-only]	Entero que indica la proporción de aspecto de píxeles de la pantalla.
static	playerType:String [read-only]	Cadena que indica el tipo de reproductor.
static	screenColor:String [read-only]	Cadena que indica el color de la pantalla.
static	screenDPI:Number [read-only]	Número que indica la resolución de puntos por pulgada (ppp) de la pantalla expresada en píxeles.
static	screenResolutionX:Number [read-only]	Entero que indica la resolución horizontal máxima de la pantalla.
static	screenResolutionY:Number [read-only]	Entero que indica la resolución vertical máxima de la pantalla.

Modificadores	Propiedad	Descripción
`static`	`serverString:String` [read-only]	Cadena con codificación URL que especifica valores para cada propiedad `System.capabilities`.
`static`	`version:String` [read-only]	Cadena que contiene la información de plataforma y versión de Flash Player (por ejemplo, `"WIN 8,0,0,0"`).

Propiedades heredadas de la clase Object

`constructor (propiedad Object.constructor), __proto__ (propiedad Object.__proto__), prototype (propiedad Object.prototype), __resolve (propiedad Object.__resolve)`

Resumen de métodos

Métodos heredados de la clase Object

`addProperty (método Object.addProperty), hasOwnProperty (método Object.hasOwnProperty), isPropertyEnumerable (método Object.isPropertyEnumerable), isPrototypeOf (método Object.isPrototypeOf), registerClass (método Object.registerClass), toString (método Object.toString), unwatch (método Object.unwatch), valueOf (método Object.valueOf), watch (método Object.watch)`

avHardwareDisable (propiedad capabilities.avHardwareDisable)

`public static avHardwareDisable : Boolean [read-only]`

Valor booleano que especifica si el acceso a la cámara y al micrófono del usuario ha sido prohibido (`true`) o autorizado (`false`) administrativamente. La cadena de servidor es `AVD`.

Disponibilidad: ActionScript 1.0; Flash Player 7

Ejemplo

El ejemplo siguiente realiza un seguimiento del valor de esta propiedad de sólo lectura:

```
trace(System.capabilities.avHardwareDisable);
```

Véase también

`get (método Camera.get), get (propiedad Microphone.get), showSettings (método System.showSettings)`

hasAccessibility (propiedad capabilities.hasAccessibility)

```
public static hasAccessibility : Boolean [read-only]
```

Valor booleano que da como resultado `true` si el reproductor se está ejecutando en un entorno que admite la comunicación entre Flash Player y las ayudas de accesibilidad; en caso contrario, da como resultado `false`. La cadena de servidor es `ACC`.

Disponibilidad: ActionScript 1.0; Flash Player 6

Ejemplo

El ejemplo siguiente realiza un seguimiento del valor de esta propiedad de sólo lectura:

```
trace(System.capabilities.hasAccessibility);
```

Véase también

`isActive (Método Accessibility.isActive)`, `updateProperties (método Accessibility.updateProperties)`,

hasAudio (propiedad capabilities.hasAudio)

```
public static hasAudio : Boolean [read-only]
```

Especifica si el sistema tiene prestaciones de audio. Valor booleano que da como resultado `true` si el reproductor se está ejecutando en un sistema con prestaciones de audio; en caso contrario, da como resultado `false`. La cadena de servidor es `A`.

Disponibilidad: ActionScript 1.0; Flash Player 6

Ejemplo

El ejemplo siguiente realiza un seguimiento del valor de esta propiedad de sólo lectura:

```
trace(System.capabilities.hasAudio);
```

hasAudioEncoder (propiedad capabilities.hasAudioEncoder)

```
public static hasAudioEncoder : Boolean [read-only]
```

Especifica si Flash Player puede codificar un flujo de audio. Valor booleano que da como resultado `true` si el reproductor permite codificar un flujo de audio, como, por ejemplo, el procedente de un micrófono; en caso contrario, da como resultado `false`. La cadena de servidor es `AE`.

Disponibilidad: ActionScript 1.0; Flash Player 6

Ejemplo

El ejemplo siguiente realiza un seguimiento del valor de esta propiedad de sólo lectura:

```
trace(System.capabilities.hasAudioEncoder);
```

hasEmbeddedVideo (propiedad capabilities.hasEmbeddedVideo)

```
public static hasEmbeddedVideo : Boolean [read-only]
```

Valor booleano que da como resultado `true` si el reproductor se está ejecutando en un sistema que admite vídeo incorporado; en caso contrario, da como resultado `false`. La cadena de servidor es `EV`.

Disponibilidad: ActionScript 1.0; Flash Player 6,0,65,0

Ejemplo

El ejemplo siguiente realiza un seguimiento del valor de esta propiedad de sólo lectura:

```
trace(System.capabilities.hasEmbeddedVideo);
```

hasIME (propiedad capabilities.hasIME)

```
public static hasIME : Boolean [read-only]
```

Indica si hay un editor de método de entrada (IME) instalado en el sistema. Un valor de `true` indica que el reproductor se está ejecutando en un sistema con IME instalado; un valor de `false` indica que no hay ningún IME instalado. La cadena de servidor es `IME`.

Disponibilidad: ActionScript 1.0; Flash Player 8

Ejemplo

El ejemplo siguiente define el IME como `ALPHANUMERIC_FULL` si el reproductor se ejecuta en un sistema que tiene un IME instalado.

```
if(System.capabilities.hasIME) {
  trace(System.IME.getConversionMode());
  System.IME.setConversionMode(System.IME.ALPHANUMERIC_FULL);
  trace(System.IME.getConversionMode());
}
```

hasMP3 (propiedad capabilities.hasMP3)

```
public static hasMP3 : Boolean [read-only]
```

Especifica si el sistema dispone de un descodificador de MP3. Valor booleano que da como resultado `true` si el reproductor se está ejecutando en un sistema con descodificador MP3; en caso contrario, el resultado es `false`. La cadena de servidor es `MP3`.

Disponibilidad: ActionScript 1.0; Flash Player 6

Ejemplo

El ejemplo siguiente realiza un seguimiento del valor de esta propiedad de sólo lectura:

```
trace(System.capabilities.hasMP3);
```

hasPrinting (propiedad capabilities.hasPrinting)

```
public static hasPrinting : Boolean [read-only]
```

Valor booleano que da como resultado `true` si el reproductor se está ejecutando en un sistema que admite impresión; en caso contrario, da como resultado `false`. La cadena de servidor es `PR`.

Disponibilidad: ActionScript 1.0; Flash Player 6,0,65,0

Ejemplo

El ejemplo siguiente realiza un seguimiento del valor de esta propiedad de sólo lectura:

```
trace(System.capabilities.hasPrinting);
```

hasScreenBroadcast (propiedad capabilities.hasScreenBroadcast)

```
public static hasScreenBroadcast : Boolean [read-only]
```

Valor booleano que da como resultado `true` si el reproductor admite el desarrollo de aplicaciones de difusión de pantallas para su ejecución a través de Flash Communication Server; en caso contrario, da como resultado `false`. La cadena de servidor es `SB`.

Disponibilidad: ActionScript 1.0; Flash Player 6,0,79,0

Ejemplo

El ejemplo siguiente realiza un seguimiento del valor de esta propiedad de sólo lectura:

```
trace(System.capabilities.hasScreenBroadcast);
```

hasScreenPlayback (propiedad capabilities.hasScreenPlayback)

```
public static hasScreenPlayback : Boolean [read-only]
```

Valor booleano que da como resultado `true` si el reproductor admite la reproducción de aplicaciones de difusión de pantallas que se están ejecutando a través de Flash Communication Server; en caso contrario, da como resultado `false`. La cadena de servidor es `SP`.

Disponibilidad: ActionScript 1.0; Flash Player 6,0,79,0

Ejemplo

El ejemplo siguiente realiza un seguimiento del valor de esta propiedad de sólo lectura:

```
trace(System.capabilities.hasScreenPlayback);
```

hasStreamingAudio (propiedad capabilities.hasStreamingAudio)

```
public static hasStreamingAudio : Boolean [read-only]
```

Valor booleano que da como resultado `true` si el reproductor es capaz de reproducir audio sin interrupción; en caso contrario, da como resultado `false`. La cadena de servidor es `SA`.

Disponibilidad: ActionScript 1.0; Flash Player 6,0,65,0

Ejemplo

El ejemplo siguiente realiza un seguimiento del valor de esta propiedad de sólo lectura:

```
trace(System.capabilities.hasStreamingAudio);
```

hasStreamingVideo (propiedad capabilities.hasStreamingVideo)

```
public static hasStreamingVideo : Boolean [read-only]
```

Valor booleano que da como resultado `true` si el reproductor es capaz de reproducir vídeo sin interrupción; en caso contrario, da como resultado `false`. La cadena de servidor es `SV`.

Disponibilidad: ActionScript 1.0; Flash Player 6,0,65,0

Ejemplo

El ejemplo siguiente realiza un seguimiento del valor de esta propiedad de sólo lectura:

```
trace(System.capabilities.hasStreamingVideo);
```

hasVideoEncoder (propiedad capabilities.hasVideoEncoder)

```
public static hasVideoEncoder : Boolean [read-only]
```

Especifica si Flash Player puede codificar un flujo de vídeo. Valor booleano que da como resultado `true` si el reproductor permite codificar un flujo de vídeo, como, por ejemplo, el procedente de una cámara; en caso contrario, el resultado es `false`. La cadena de servidor es `VE`.

Disponibilidad: ActionScript 1.0; Flash Player 6

Ejemplo

El ejemplo siguiente realiza un seguimiento del valor de esta propiedad de sólo lectura:

```
trace(System.capabilities.hasVideoEncoder);
```

isDebugger (propiedad capabilities.isDebugger)

```
public static isDebugger : Boolean [read-only]
```

Valor booleano que especifica si el reproductor es de una versión lanzada oficialmente (`false`) o una versión especial de depuración (`true`). La cadena de servidor es `DEB`.

Disponibilidad: ActionScript 1.0; Flash Player 6

Ejemplo

El ejemplo siguiente realiza un seguimiento del valor de esta propiedad de sólo lectura:

```
trace(System.capabilities.isDebugger);
```

language (propiedad capabilities.language)

```
public static language : String [read-only]
```

Indica el idioma del sistema en el que se está ejecutando el reproductor. Esta propiedad se especifica como un código de idioma de dos letras en minúsculas de ISO 639-1. En el caso del idioma chino, una subetiqueta adicional de código de dos letras de país de ISO 3166 permite distinguir si se trata de chino simplificado o chino tradicional. Los idiomas en sí se denominan por sus etiquetas en inglés. Por ejemplo, `fr` especifica francés.

Esta propiedad cambió de dos formas en Flash Player 7. En primer lugar, el código de idioma de los sistemas en inglés ya no incluye el código de idioma. En Flash Player 6, todos los sistemas en inglés devolvían el código de idioma y la subetiqueta de código de país de dos letras (`en-US`). En Flash Player 7, los sistemas en inglés sólo devuelven el código de idioma (`en`). En segundo lugar, en sistemas Microsoft Windows, esta propiedad ahora devuelve el idioma de la interfaz de usuario. En Flash Player 6 ejecutado en la plataforma Microsoft Windows, `System.capabilities.language` devolvía la localización de idioma/país del usuario, que controla la configuración de formato para fechas, horas, divisa y números grandes. En Flash Player 7 ejecutado en la plataforma Microsoft Windows, esta propiedad devuelve ahora el idioma de la interfaz de usuario, que hace referencia al idioma empleado en todos los menús, cuadros de diálogo, mensajes de error y archivos de ayuda. En la siguiente tabla se enumeran los valores posibles:

Idioma	**Etiqueta**
Checo	cs
Danés	da
Holandés	nl
Inglés	en
Finlandés	fi
Francés	fr
Alemán	de
Húngaro	hu
Italiano	it
Japonés	ja
Coreano	ko
Noruego	no
Otro/desconocido	xu
Polaco	pl
Portugués	pt
Ruso	ru
Chino simplificado	zh-CN
Español	es
Sueco	sv

Idioma	Etiqueta
Chino tradicional	zh-TW
Turco	tr

Disponibilidad: ActionScript 1.0; Flash Player 6

Ejemplo

El ejemplo siguiente realiza un seguimiento del valor de esta propiedad de sólo lectura:

```
trace(System.capabilities.language);
```

localFileReadDisable (propiedad capabilities.localFileReadDisable)

```
public static localFileReadDisable : Boolean [read-only]
```

Valor booleano que indica si el acceso de lectura al disco duro del usuario ha sido prohibido (`true`) o autorizado (`false`) administrativamente. Si es `true`,, Flash Player no podrá leer archivos (incluido el primer archivo SWF con el que se inicia Flash Player) del disco duro del usuario. Por ejemplo, los intentos de lectura de un archivo del disco duro del usuario mediante `XML.load()`, `LoadMovie()` o `LoadVars.load()` fallarán si esta propiedad se establece como `true`.

La lectura de bibliotecas compartidas en tiempo de ejecución también se bloqueará si esta propiedad se establece como `true`, aunque se autorizará la lectura de objetos compartidos locales con independencia de cuál sea el valor de esta propiedad. La cadena de servidor es `LFD`.

Disponibilidad: ActionScript 1.0; Flash Player 7

Ejemplo

El ejemplo siguiente realiza un seguimiento del valor de esta propiedad de sólo lectura:

```
trace(System.capabilities.localFileReadDisable);
```

manufacturer (propiedad capabilities.manufacturer)

```
public static manufacturer : String [read-only]
```

Cadena que indica el fabricante de Flash Player, con el formato `Macromedia` *`OSName`* " (*`OSName`* puede ser "`Windows`", "`Macintosh`", "`Linux`" o "`Other OS Name`"). La cadena de servidor es `M`.

Disponibilidad: ActionScript 1.0; Flash Player 6

Ejemplo

El ejemplo siguiente realiza un seguimiento del valor de esta propiedad de sólo lectura:

```
trace(System.capabilities.manufacturer);
```

os (propiedad capabilities.os)

```
public static os : String [read-only]
```

Cadena que indica el sistema operativo actual. La propiedad `os` puede devolver las siguientes cadenas: `"Windows XP"`, `"Windows 2000"`, `"Windows NT"`, `"Windows 98/ME"`, `"Windows 95"`, `"Windows CE"` (sólo disponible en Flash Player SDK, no en la versión de escritorio), `"Linux"` y `"MacOS"`. La cadena de servidor es `OS`.

Disponibilidad: ActionScript 1.0; Flash Player 6

Ejemplo

El ejemplo siguiente realiza un seguimiento del valor de esta propiedad de sólo lectura:

```
trace(System.capabilities.os);
```

pixelAspectRatio (propiedad capabilities.pixelAspectRatio)

```
public static pixelAspectRatio : Number [read-only]
```

Entero que indica la proporción de aspecto de píxeles de la pantalla. La cadena de servidor es `AR`.

Disponibilidad: ActionScript 1.0; Flash Player 6

Ejemplo

El ejemplo siguiente realiza un seguimiento del valor de esta propiedad de sólo lectura:

```
trace(System.capabilities.pixelAspectRatio);
```

playerType (propiedad capabilities.playerType)

```
public static playerType : String [read-only]
```

Cadena que indica el tipo de reproductor. Esta propiedad puede tener uno de los siguientes valores:

- `"StandAlone"` para Flash StandAlone Player
- `"External"` para la versión de Flash Player empleada por el reproductor externo o el modo de Probar película.

- `"PlugIn"` para el plug-in de navegador de Flash Player
- `"ActiveX"` para el control ActiveX de Flash Player utilizado por Microsoft Internet Explorer

La cadena de servidor es `PT`.

Disponibilidad: ActionScript 1.0; Flash Player 7

Ejemplo

El ejemplo siguiente realiza un seguimiento del valor de esta propiedad de sólo lectura:

```
trace(System.capabilities.playerType);
```

screenColor (propiedad capabilities.screenColor)

```
public static screenColor : String [read-only]
```

Cadena que indica el color de la pantalla. Esta propiedad puede tener el valor `"color"`, `"gray"` o `"bw"`, que indican, respectivamente, color, escala de grises y blanco y negro. La cadena de servidor es `COL`.

Disponibilidad: ActionScript 1.0; Flash Player 6

Ejemplo

El ejemplo siguiente realiza un seguimiento del valor de esta propiedad de sólo lectura:

```
trace(System.capabilities.screenColor);
```

screenDPI (propiedad capabilities.screenDPI)

```
public static screenDPI : Number [read-only]
```

Número que indica la resolución de puntos por pulgada (ppp) de la pantalla expresada en píxeles. La cadena de servidor es `DP`.

Disponibilidad: ActionScript 1.0; Flash Player 6

Ejemplo

El ejemplo siguiente realiza un seguimiento del valor de esta propiedad de sólo lectura:

```
trace(System.capabilities.screenDPI);
```

screenResolutionX (propiedad capabilities.screenResolutionX)

```
public static screenResolutionX : Number [read-only]
```

Entero que indica la resolución horizontal máxima de la pantalla. La cadena de servidor es `R` (que devuelve la anchura y la altura de la pantalla).

Disponibilidad: ActionScript 1.0; Flash Player 6

Ejemplo

El ejemplo siguiente realiza un seguimiento del valor de esta propiedad de sólo lectura:

```
trace(System.capabilities.screenResolutionX);
```

screenResolutionY (propiedad capabilities.screenResolutionY)

```
public static screenResolutionY : Number [read-only]
```

Entero que indica la resolución vertical máxima de la pantalla. La cadena de servidor es `R` (que devuelve la anchura y la altura de la pantalla).

Disponibilidad: ActionScript 1.0; Flash Player 6

Ejemplo

El ejemplo siguiente realiza un seguimiento del valor de esta propiedad de sólo lectura:

```
trace(System.capabilities.screenResolutionY);
```

serverString (propiedad capabilities.serverString)

```
public static serverString : String [read-only]
```

Cadena con codificación URL que especifica valores para cada propiedad `System.capabilities`.

En el siguiente ejemplo se muestra una cadena con codificación URL:

```
A=t&SA=t&SV=t&EV=t&MP3=t&AE=t&VE=t&ACC=f&PR=t&SP=t&

 SB=f&DEB=t&V=WIN%208%2C0%2C0%2C0&M=Macromedia%20Windows&

 R=1600x1200&DP=72&COL=color&AR=1.0&OS=Windows%20XP&

 L=en&PT=External&AVD=f&LFD=f&WD=f
```

Disponibilidad: ActionScript 1.0; Flash Player 6

Ejemplo

El ejemplo siguiente realiza un seguimiento del valor de esta propiedad de sólo lectura:

```
trace(System.capabilities.serverString);
```

version (propiedad capabilities.version)

```
public static version : String [read-only]
```

Cadena que contiene la información de plataforma y versión de Flash Player (por ejemplo, `"WIN 8,0,0,0"`). La cadena de servidor es `V`.

Disponibilidad: ActionScript 1.0; Flash Player 6

Ejemplo

El ejemplo siguiente realiza un seguimiento del valor de esta propiedad de sólo lectura:

```
trace(System.capabilities.version);
```

Color

```
Object
  |
  +-Color
```

```
public class Color
extends Object
```

Desfasada desde Flash Player 8. La clase Color está desfasada y en su lugar debe utilizarse la clase flash.geom.ColorTransform.

La clase Color le permite establecer el valor de color RGB y la transformación de color de clips de película y recuperar dichos valores una vez que se hayan establecido.

Debe utilizar el constructor `new Color()` para crear un objeto Color antes de llamar a sus métodos.

Disponibilidad: ActionScript 1.0; Flash Player 5

Resumen de propiedades

Propiedades heredadas de la clase Object

`constructor (propiedad Object.constructor), __proto__ (propiedad Object.__proto__), prototype (propiedad Object.prototype), __resolve (propiedad Object.__resolve)`

Resumen de constructores

Firma	Descripción
`Color(target:Object)`	*Clase desfasada.* Crea un objeto Color para el clip de película especificado por el parámetro *target_mc* .

Resumen de métodos

Modificadores	Firma	Descripción
	`getRGB() : Number`	*Clase desfasada.* Devuelve la combinación R+G+B utilizada por el objeto de color.
	`getTransform() : Object`	*Clase desfasada.* Devuelve el valor de transformación establecido por la última llamada a `Color.setTransform()`.
	`setRGB(offset:Number) : Void`	*Clase desfasada.* Especifica un color RGB para un objeto Color.
	`setTransform(transformObject:Object) : Void`	*Clase desfasada.* Establece la información de transformación para un objeto Color.

Métodos heredados de la clase Object

`addProperty (método Object.addProperty), hasOwnProperty (método Object.hasOwnProperty), isPropertyEnumerable (método Object.isPropertyEnumerable), isPrototypeOf (método Object.isPrototypeOf), registerClass (método Object.registerClass), toString (método Object.toString), unwatch (método Object.unwatch), valueOf (método Object.valueOf), watch (método Object.watch)`

Constructor Color()

`public Color(target:Object)`

La clase Color está desfasada desde Flash Player 8.

Crea un objeto Color para el clip de película especificado por el parámetro *target_mc* . Posteriormente podrá utilizar los métodos de dicho objeto Color para cambiar el color de todo el clip de película de destino.

Disponibilidad: ActionScript 1.0; Flash Player 5

Parámetros

target:Object - Nombre de instancia de un clip de película.

Ejemplo

El ejemplo siguiente crea un objeto Color llamado my_color para el clip de película my_mc y establece su valor RGB en naranja:

```
var my_color:Color = new Color(my_mc);
my_color.setRGB(0xff9933);
```

getRGB (método Color.getRGB)

```
public getRGB() : Number
```

La clase Color está desfasada desde Flash Player 8.

Devuelve la combinación R+G+B utilizada por el objeto de color.

Disponibilidad: ActionScript 1.0; Flash Player 5

Valor devuelto

Number - Número que representa al valor de número RGB para el color especificado.

Ejemplo

El código siguiente recupera el valor RGB para el objeto my_color, convierte el valor en una cadena hexadecimal y la asigna a la variable myValue. Para ver cómo funciona este código, añada una instancia de clip de película al escenario y asígnele el nombre de instancia my_mc:

```
var my_color:Color = new Color(my_mc);
// set the color
my_color.setRGB(0xff9933);
var myValue:String = my_color.getRGB().toString(16);
// trace the color value
trace(myValue); // traces ff9933
```

Véase también

setRGB (método Color.setRGB)

getTransform (método Color.getTransform)

```
public getTransform() : Object
```

La clase Color está desfasada desde Flash Player 8.

Devuelve el valor de transformación establecido por la última llamada a `Color.setTransform()`.

Disponibilidad: ActionScript 1.0; Flash Player 5

Valor devuelto

`Object` - Objeto cuyas propiedades contienen los valores actuales de desplazamiento y porcentaje para el color especificado.

Ejemplo

El ejemplo siguiente obtiene el objeto de transformación y, a continuación, establece nuevos porcentajes para colores y alfa de `my_mc` en relación con sus valores actuales. Para ver cómo funciona esta código, sitúe un clip de película multicolor en el escenario con el nombre de instancia `my_mc`. A continuación, sitúe el código siguiente en el fotograma 1 de la línea de tiempo principal y seleccione Control > Probar película:

```
var my_color:Color = new Color(my_mc);
var myTransform:Object = my_color.getTransform();
myTransform = { ra: 50, ba: 50, aa: 30};
my_color.setTransform(myTransform);
```

Para obtener descripciones de los parámetros de un objeto de transformación de color, consulte `Color.setTransform()`.

Véase también

`setTransform (método Color.setTransform)`

setRGB (método Color.setRGB)

```
public setRGB(offset:Number) : Void
```

La clase Color está desfasada desde Flash Player 8.

Especifica un color RGB para un objeto Color. La llamada a este método provoca la sustitución de toda la configuración anterior de `Color.setTransform()`.

Disponibilidad: ActionScript 1.0; Flash Player 5

Parámetros

offset:Number - 0x *RRGGBB* Color RGB o hexadecimal que desea establecer. *RR*, *GG* y *BB* constan cada uno de dos dígitos hexadecimales que especifican el desplazamiento de cada componente de color. El 0x indica al compilador de ActionScript que el número es un valor hexadecimal.

Ejemplo

Este ejemplo establece el valor de color RGB del clip de película my_mc. Para ver cómo funciona esta código, sitúe un clip de película en el escenario con el nombre de instancia my_mc. A continuación, sitúe el código siguiente en el fotograma 1 de la línea de tiempo principal y seleccione Control > Probar película:

```
var my_color:Color = new Color(my_mc);
my_color.setRGB(0xFF0000); // my_mc turns red
```

Véase también

setTransform (método Color.setTransform)

setTransform (método Color.setTransform)

```
public setTransform(transformObject:Object) : Void
```

La clase Color está desfasada desde Flash Player 8.

Establece la información de transformación para un objeto Color. El parámetro *colorTransformObject* es un objeto genérico que usted crea desde el constructor new Object. Incluye parámetros que especifican los valores de porcentaje y desplazamiento para los componentes rojo, verde, azul y alfa (transparencia) de un color, introducidos en el formato 0xRRGGBBAA.

Los parámetros de un objeto de transformación de color corresponden a la configuración del cuadro de diálogo Efecto avanzado y se definen de la siguiente forma:

- *ra* es el porcentaje del componente rojo (de -100 a 100).
- *rb* es el desplazamiento del componente rojo (de -255 a 255).
- *ga* es el porcentaje del componente rojo (de -100 a 100).
- *gb* es el desplazamiento del componente rojo (de -255 a 255).
- *ba* es el porcentaje del componente rojo (de -100 a 100).
- *bb* es el desplazamiento del componente rojo (de -255 a 255).
- *aa* es el porcentaje de alfa (de -100 a 100).
- *ab* es el desplazamiento de alfa (de -255 a 255).

Un parámetro *colorTransformObject* se crea de la siguiente forma:

```
var myColorTransform:Object = new Object();
myColorTransform.ra = 50;
myColorTransform.rb = 244;
myColorTransform.ga = 40;
myColorTransform.gb = 112;
myColorTransform.ba = 12;
myColorTransform.bb = 90;
myColorTransform.aa = 40;
myColorTransform.ab = 70;
```

También puede utilizar la siguiente sintaxis para crear un parámetro *colorTransformObject*:

```
var myColorTransform:Object = { ra: 50, rb: 244, ga: 40, gb: 112, ba: 12, bb: 90, aa: 40, ab:
  70}
```

Disponibilidad: ActionScript 1.0; Flash Player 5

Parámetros

transformObject:Object - Un objeto creado con el constructor `new Object`. Esta instancia de la clase Object debe tener las siguientes propiedades que especifican valores de transformación de color: `ra`, `rb`, `ga`, `gb`, `ba`, `bb`, `aa`, `ab`. A continuación se explican estas propiedades.

Ejemplo

Este ejemplo crea un objeto Color para un archivo SWF de destino, crea un objeto genérico llamado `myColorTransform` con las propiedades definidas anteriormente y utiliza el método `setTransform()` para pasar *colorTransformObject* a un objeto Color. Para utilizar este código en un documento de Flash (FLA), sitúelo en el fotograma 1 de la línea de tiempo principal y sitúe un clip de película en el escenario con el nombre de instancia `my_mc`, como en el código siguiente:

```
// Create a color object called my_color for the target my_mc
var my_color:Color = new Color(my_mc);
// Create a color transform object called myColorTransform using
// Set the values for myColorTransform
var myColorTransform:Object = { ra: 50, rb: 244, ga: 40, gb: 112, ba: 12,
  bb: 90, aa: 40, ab: 70};
// Associate the color transform object with the Color object
// created for my_mc
my_color.setTransform(myColorTransform);
```

Véase también

`Object`

ColorMatrixFilter (flash.filters.ColorMatrixFilter)

```
Object
  |
  +-flash.filters.BitmapFilter
    |
    +-flash.filters.ColorMatrixFilter
```

`public class` *ColorMatrixFilter*
`extends BitmapFilter`

La clase ColorMatrixFilter permite aplicar una transformación de matriz 4 x 5 en los valores de color RGBA y alfa de cada píxel de la imagen de entrada para producir un resultados con un nuevo conjunto de valores de color RGBA y alfa. Permite cambios de saturación, rotación de matiz, luminancia a alfa y otros efectos diversos. Este filtro se puede aplicar a mapas de bits e instancias de clip de película.

El uso de filtros depende del objeto al que se aplique el filtro:

- Para aplicar filtros a clips de película en tiempo de ejecución, utilice la propiedad `filters`. El establecimiento de la propiedad `filters` de un objeto no modifica el objeto y se puede deshacer borrando la propiedad `filters`.
- Para aplicar filtros a instancias de BitmapData, utilice el método `BitmapData.applyFilter()`. La llamada a `applyFilter()` en un objeto BitmapData toma el objeto BitmapData de origen y el objeto de filtro y genera una imagen filtrada.

También puede aplicar efectos de filtro a imágenes y vídeo en tiempo de edición. Para más información, consulte la documentación de edición.

Si aplica un filtro a un clip de película o a un botón, la propiedad `cacheAsBitmap` del clip de película o del botón se establecerá como `true`. Si borra todos los filtros, se restaurará el valor original de `cacheAsBitmap`.

Se utiliza la fórmula siguiente, donde a[0]-a[19] corresponden a las entradas 0-19 en la matriz de propiedades de 20 elementos.

```
redResult = a[0] * srcR + a[1] * srcG + a[2] * srcB + a[3] * srcA + a[4]
greenResult = a[5] * srcR + a[6] * srcG + a[7] * srcB + a[8] * srcA + a[9]
blueResult = a[10] * srcR + a[11] * srcG + a[12] * srcB + a[13] * srcA +
 a[14]
alphaResult = a[15] * srcR + a[16] * srcG + a[17] * srcB + a[18] * srcA +
 a[19]
```

Este filtro separa cada píxel de origen en sus componentes rojo, verde, azul y alfa como srcR, srcG, srcB, srcA. Por último, combina cada componente de color en un único píxel y escribe el resultado.

Los cálculos se realizan con valores de color no multiplicados. Si el gráfico de entrada se compone de valores de color premultiplicados, esos valores se convertirán automáticamente en valores de color no multiplicados para esta operación.

Hay dos modos optimizados disponibles.

Sólo alfa. Cuando pasa al filtro una matriz que ajusta el componente alfa solamente, como se indica aquí, el filtro optimiza su rendimiento:

```
1 0 0 0 0
0 1 0 0 0
0 0 1 0 0
0 0 0 N 0 (where N is between 0.0 and 1.0)
```

Versión rápida. Disponible sólo con procesadores que disponen de acelerador SSE/Altivec, como Pentium 3 y posteriores, Apple G4 y posteriores. El acelerador se utiliza cuando los términos del multiplicador corresponden al rango comprendido entre -15.99 y 15.99, y los términos del sumador a[4], a[9], a[14] y a[19] corresponden al rango entre -8000 y8000.

El filtro no se aplica si la imagen resultante supera los 2880 píxeles de anchura o altura. Por ejemplo, si acerca un clip de película con un filtro aplicado, éste se desactiva si la imagen resultante supera el límite de 2880 píxeles.

Disponibilidad: ActionScript 1.0; Flash Player 8

Ejemplo

El ejemplo siguiente utiliza BitmapFilter para manipular la saturación de color de una imagen según la ubicación del puntero del ratón. Si sitúe el puntero en la esquina superior izquierda (0,0), la imagen debe quedar como está. A media que desplaza el puntero hacia la derecha, lo canales verde y azul se eliminan ambos de la imagen. Cuando lo desplaza hacia abajo, se elimina el canal rojo. Si el puntero se sitúa en la parte inferior derecha del escenario, la imagen debe quedar completamente negra. En este ejemplo se asume que tiene una imagen en su biblioteca con el identificador de vinculación definido como "YourImageLinkage".

```
import flash.filters.BitmapFilter;
import flash.filters.ColorMatrixFilter;

var image:MovieClip = this.attachMovie("YourImageLinkage", "YourImage",
  this.getNextHighestDepth());
image.cacheAsBitmap = true;

var listener:Object = new Object();
listener.image = image;
```

```
listener.onMouseMove = function() {
  var xPercent:Number = 1 - (_xmouse/Stage.width);
  var yPercent:Number = 1 - (_ymouse/Stage.height);
  var matrix:Array = new Array();
  matrix = matrix.concat([yPercent, 0, 0, 0, 0]); // red
  matrix = matrix.concat([0, xPercent, 0, 0, 0]); // green
  matrix = matrix.concat([0, 0, xPercent, 0, 0]); // blue
  matrix = matrix.concat([0, 0, 0, 1, 0]); // alpha

  var filter:BitmapFilter = new ColorMatrixFilter(matrix);
  image.filters = new Array(filter);
}

Mouse.addListener(listener);
listener.onMouseMove();
```

Véase también

getPixel (método BitmapData.getPixel), applyFilter (método BitmapData.applyFilter), filters (propiedad MovieClip.filters), cacheAsBitmap (propiedad MovieClip.cacheAsBitmap)

Resumen de propiedades

Modificadores	Propiedad	Descripción
	matrix:Array	Conjunto de 20 elementos para la transformación de color 4x5.

Propiedades heredadas de la clase Object

constructor (propiedad Object.constructor), __proto__ (propiedad Object.__proto__), prototype (propiedad Object.prototype), __resolve (propiedad Object.__resolve)

Resumen de constructores

Firma	Descripción
ColorMatrixFilter(matrix:Array)	Inicializa una instancia ColorMatrixFilter nueva con los parámetros especificados.

Resumen de métodos

Modificadores	Firma	Descripción
	clone() : ColorMatrixFilter	Devuelve una copia de este objeto de filtro.

Métodos heredados de la clase BitmapFilter

clone (método BitmapFilter.clone)

Métodos heredados de la clase Object

addProperty (método Object.addProperty), hasOwnProperty (método Object.hasOwnProperty), isPropertyEnumerable (método Object.isPropertyEnumerable), isPrototypeOf (método Object.isPrototypeOf), registerClass (método Object.registerClass), toString (método Object.toString), unwatch (método Object.unwatch), valueOf (método Object.valueOf), watch (método Object.watch)

clone (método ColorMatrixFilter.clone)

`public clone() : ColorMatrixFilter`

Devuelve una copia de este objeto de filtro.

Disponibilidad: ActionScript 1.0; Flash Player 8

Valor devuelto

`flash.filters.ColorMatrixFilter` - Una instancia ColorMatrixFilter nueva con las mismas propiedades que la original.

Ejemplo

En el ejemplo siguiente se crea una nueva instancia ColorMatrixFilter y la copia con el método `clone`. La propiedad `matrix` no se puede modificar directamente (por ejemplo, `clonedFilter.matrix[2] = 1;`). En su lugar, debe obtener una referencia a la matriz, realizar el cambio y restablecer el valor mediante `clonedFilter.matrix = changedMatrix`.

```
import flash.filters.ColorMatrixFilter;

var matrix:Array = new Array();
matrix = matrix.concat([1, 0, 0, 0, 0]); // red
matrix = matrix.concat([0, 1, 0, 0, 0]); // green
matrix = matrix.concat([0, 0, 1, 0, 0]); // blue
matrix = matrix.concat([0, 0, 0, 1, 0]); // alpha

var filter:ColorMatrixFilter = new ColorMatrixFilter(matrix);
trace("filter: " + filter.matrix);
```

```
var clonedFilter:ColorMatrixFilter = filter.clone();
matrix = clonedFilter.matrix;
matrix[2] = 1;
clonedFilter.matrix = matrix;
trace("clonedFilter: " + clonedFilter.matrix);
```

Constructor ColorMatrixFilter

```
public ColorMatrixFilter(matrix:Array)
```

Inicializa una instancia ColorMatrixFilter nueva con los parámetros especificados.

Disponibilidad: ActionScript 1.0; Flash Player 8

Parámetros

matrix:Array - Una matriz de 20 elementos organizada en una matriz de 4x5.

matrix (propiedad ColorMatrixFilter.matrix)

```
public matrix : Array
```

Conjunto de 20 elementos para la transformación de color 4x5.

Disponibilidad: ActionScript 1.0; Flash Player 8

Ejemplo

En el ejemplo siguiente se crea una nueva instancia ColorMatrixFilter y, a continuación, se cambia su propiedad `matrix`. No se puede modificar directamente el valor de la propiedad `matrix` (por ejemplo, `clonedFilter.matrix[2] = 1;`). En su lugar, debe obtener una referencia a la matriz, realizar el cambio en la referencia y restablecer el valor mediante `clonedFilter.matrix = changedMatrix`.

```
import flash.filters.ColorMatrixFilter;

var matrix:Array = new Array();
matrix = matrix.concat([1, 0, 0, 0, 0]); // red
matrix = matrix.concat([0, 1, 0, 0, 0]); // green
matrix = matrix.concat([0, 0, 1, 0, 0]); // blue
matrix = matrix.concat([0, 0, 0, 1, 0]); // alpha

var filter:ColorMatrixFilter = new ColorMatrixFilter(matrix);
trace("filter: " + filter.matrix);
var changedMatrix:Array = filter.matrix;
changedMatrix[2] = 1;
filter.matrix = changedMatrix;
trace("filter: " + filter.matrix);
```

ColorTransform (flash.geom.ColorTransform)

```
Object
  |
  +-flash.geom.ColorTransform
```

```
public class ColorTransform
extends Object
```

La clase ColorTransform permite ajustar matemáticamente todos los valores de color de un clip de película. La función de ajuste de color o *transformación de color* se puede a los cuatro canales: rojo, verde, azul o transparencia alfa.

Cuando se aplica un objeto ColorTransform a un clip de película, se calcula de este modo un valor nuevo para cada canal de color:

- Nuevo valor rojo = (valor rojo antiguo* `redMultiplier)` + `redOffset`
- Nuevo valor verde = (valor verde antiguo * `greenMultiplier)` + `greenOffset`
- Nuevo valor azul = (valor azul antiguo * `blueMultiplier)` + `blueOffset`
- Nuevo valor alfa = (valor alfa antiguo * `alphaMultiplier)` + `alphaOffset`

Si alguno de los valores de canales de color es superior a 255 después del cálculo, se establece como 255. Si es inferior a 0, se establece como 0.

Debe utilizar el constructor `new ColorTransform()` nuevo para crear un objeto ColorTransform antes de llamar a los métodos del objeto ColorTransform.

Las transformaciones de color no se aplican al color del fondo de un clip de película (como un objeto SWF cargado). Sólo se aplican a gráficos y símbolos asociados al clip de película.

Disponibilidad: ActionScript 1.0; Flash Player 8

Véase también

`getTransform (método Color.getTransform)`, `setTransform (método Color.setTransform)`, `Transform (flash.geom.Transform)`

Resumen de propiedades

Modificadores	Propiedad	Descripción
	`alphaMultiplier:Number`	Valor decimal que se multiplica por el valor del canal de transparencia alfa.
	`alphaOffset:Number`	Número de -255 a 255 que se añade al valor del canal de transparencia alfa después de multiplicarse por el valor de `alphaMultiplier`.

Modificadores	Propiedad	Descripción
	`blueMultiplier:Number`	Valor decimal que se multiplica por el valor del canal azul.
	`blueOffset:Number`	Número de -255 a 255 que se añade al valor del canal azul después de multiplicarse por el valor de `blueMultiplier`.
	`greenMultiplier:Number`	Valor decimal que se multiplica por el valor del canal verde.
	`greenOffset:Number`	Número de -255 a 255 que se añade al valor del canal verde después de multiplicarse por el valor de `greenMultiplier`.
	`redMultiplier:Number`	Valor decimal que se multiplica por el valor del canal rojo.
	`redOffset:Number`	Número de -255 a 255 que se añade al valor del canal rojo después de multiplicarse por el valor de `redMultiplier`.
	`rgb:Number`	Valor del color RGB de un objeto ColorTransform.

Propiedades heredadas de la clase Object

`constructor (propiedad Object.constructor), __proto__ (propiedad Object.__proto__), prototype (propiedad Object.prototype), __resolve (propiedad Object.__resolve)`

Resumen de constructores

Firma	Descripción
`ColorTransform([redMultiplier:Number], [greenMultiplier:Number], [blueMultiplier:Number], [alphaMultiplier:Number], [redOffset:Number], [greenOffset:Number], [blueOffset:Number], [alphaOffset:Number])`	Crea un objeto ColorTransform para un objeto de visualización con los valores de canales de color y alfa especificados.

Resumen de métodos

Modificadores	Firma	Descripción
	`concat(second:ColorTransform) : Void`	Aplica una segunda transformación adicional de color al clip de película.
	`toString() : String`	Formatea y devuelve una cadena que describe todas las propiedades del objeto ColorTransform.

Métodos heredados de la clase Object

addProperty (método Object.addProperty), hasOwnProperty (método Object.hasOwnProperty), isPropertyEnumerable (método Object.isPropertyEnumerable), isPrototypeOf (método Object.isPrototypeOf), registerClass (método Object.registerClass), toString (método Object.toString), unwatch (método Object.unwatch), valueOf (método Object.valueOf), watch (método Object.watch)

alphaMultiplier (propiedad ColorTransform.alphaMultiplier)

`public alphaMultiplier : Number`

Valor decimal que se multiplica por el valor del canal de transparencia alfa.

Si establece el valor de transparencia alfa de un clip de película utilizando directamente la propiedad `MovieClip._alpha`, afectará al valor de la propiedad `alphaMultiplier` del objeto `ColorTransform` de dicho clip de película.

Disponibilidad: ActionScript 1.0; Flash Player 8

Ejemplo

El ejemplo siguiente crea el objeto ColorTransform `colorTrans` y ajusta su valor de `alphaMultiplier` de 1 a 0,5.

```
import flash.geom.ColorTransform;
import flash.geom.Transform;

var colorTrans:ColorTransform = new ColorTransform();
trace(colorTrans.alphaMultiplier); // 1

colorTrans.alphaMultiplier = .5;
trace(colorTrans.alphaMultiplier); // .5

var rect:MovieClip = createRectangle(20, 80, 0x000000);
var trans:Transform = new Transform(rect);
trans.colorTransform = colorTrans;

function createRectangle(width:Number, height:Number, color:Number,
  scope:MovieClip):MovieClip {
  scope = (scope == undefined) ? this : scope;
  var depth:Number = scope.getNextHighestDepth();
  var mc:MovieClip = scope.createEmptyMovieClip("mc_" + depth, depth);
  mc.beginFill(color);
  mc.lineTo(0, height);
  mc.lineTo(width, height);
```

```
  mc.lineTo(width, 0);
  mc.lineTo(0, 0);
  return mc;
}
```

Véase también

`_alpha (propiedad MovieClip._alpha)`

alphaOffset (propiedad ColorTransform.alphaOffset)

```
public alphaOffset : Number
```

Número de -255 a 255 que se añade al valor del canal de transparencia alfa después de multiplicarse por el valor de `alphaMultiplier`.

Disponibilidad: ActionScript 1.0; Flash Player 8

Ejemplo

El ejemplo siguiente crea el objeto ColorTransform `colorTrans` y ajusta su valor de `alphaOffset` de 0 a -128.

```
import flash.geom.ColorTransform;
import flash.geom.Transform;

var colorTrans:ColorTransform = new ColorTransform();
trace(colorTrans.alphaOffset); // 0

colorTrans.alphaOffset = -128;
trace(colorTrans.alphaOffset); // -128

var rect:MovieClip = createRectangle(20, 80, 0x000000);
var trans:Transform = new Transform(rect);
trans.colorTransform = colorTrans;

function createRectangle(width:Number, height:Number, color:Number,
  scope:MovieClip):MovieClip {
  scope = (scope == undefined) ? this : scope;
  var depth:Number = scope.getNextHighestDepth();
  var mc:MovieClip = scope.createEmptyMovieClip("mc_" + depth, depth);
  mc.beginFill(color);
  mc.lineTo(0, height);
  mc.lineTo(width, height);
  mc.lineTo(width, 0);
  mc.lineTo(0, 0);
  return mc;
}
```

blueMultiplier (propiedad ColorTransform.blueMultiplier)

```
public blueMultiplier : Number
```

Valor decimal que se multiplica por el valor del canal azul.

Disponibilidad: ActionScript 1.0; Flash Player 8

Ejemplo

El ejemplo siguiente crea el objeto ColorTransform `colorTrans` y ajusta su valor de `blueMultiplier` de 1 a 0,5.

```
import flash.geom.ColorTransform;
import flash.geom.Transform;

var colorTrans:ColorTransform = new ColorTransform();
trace(colorTrans.blueMultiplier); // 1

colorTrans.blueMultiplier = .5;
trace(colorTrans.blueMultiplier); // .5

var rect:MovieClip = createRectangle(20, 80, 0x0000FF);
var trans:Transform = new Transform(rect);
trans.colorTransform = colorTrans;

function createRectangle(width:Number, height:Number, color:Number,
  scope:MovieClip):MovieClip {
  scope = (scope == undefined) ? this : scope;
  var depth:Number = scope.getNextHighestDepth();
  var mc:MovieClip = scope.createEmptyMovieClip("mc_" + depth, depth);
  mc.beginFill(color);
  mc.lineTo(0, height);
  mc.lineTo(width, height);
  mc.lineTo(width, 0);
  mc.lineTo(0, 0);
  return mc;
}
```

blueOffset (propiedad ColorTransform.blueOffset)

```
public blueOffset : Number
```

Número de -255 a 255 que se añade al valor del canal azul después de multiplicarse por el valor de `blueMultiplier`.

Disponibilidad: ActionScript 1.0; Flash Player 8

Ejemplo

El ejemplo siguiente crea el objeto ColorTransform `colorTrans` y ajusta su valor de `blueOffset` de 0 a 255.

```
import flash.geom.ColorTransform;
import flash.geom.Transform;

var colorTrans:ColorTransform = new ColorTransform();
trace(colorTrans.blueOffset); // 0

colorTrans.blueOffset = 255;
trace(colorTrans.blueOffset); // 255

var rect:MovieClip = createRectangle(20, 80, 0x000000);
var trans:Transform = new Transform(rect);
trans.colorTransform = colorTrans;

function createRectangle(width:Number, height:Number, color:Number,
  scope:MovieClip):MovieClip {
  scope = (scope == undefined) ? this : scope;
  var depth:Number = scope.getNextHighestDepth();
  var mc:MovieClip = scope.createEmptyMovieClip("mc_" + depth, depth);
  mc.beginFill(color);
  mc.lineTo(0, height);
  mc.lineTo(width, height);
  mc.lineTo(width, 0);
  mc.lineTo(0, 0);
  return mc;
}
```

Constructor ColorTransform

```
public ColorTransform([redMultiplier:Number], [greenMultiplier:Number],
  [blueMultiplier:Number], [alphaMultiplier:Number], [redOffset:Number],
  [greenOffset:Number], [blueOffset:Number], [alphaOffset:Number])
```

Crea un objeto ColorTransform para un objeto de visualización con los valores de canales de color y alfa especificados.

Disponibilidad: ActionScript 1.0; Flash Player 8

Parámetros

redMultiplier:Number [opcional] - El valor del multiplicador rojo, en el rango de 0 a 1. El valor predeterminado es 1.

greenMultiplier:Number [opcional] - El valor del multiplicador verde, en el rango de 0 a 1. El valor predeterminado es 1.

blueMultiplier:Number [opcional] - El valor del multiplicador azul, en el rango de 0 a 1. El valor predeterminado es 1.

alphaMultiplier:Number [opcional] - El valor del multiplicador de transparencia alfa, en el rango de 0 a 1. El valor predeterminado es 1.

redOffset:Number [opcional] - El desplazamiento del valor del canal de color rojo (de -255 a 255). El valor predeterminado es 0.

greenOffset:Number [opcional] - El desplazamiento del valor del canal de color verde (de -255 a 255). El valor predeterminado es 0.

blueOffset:Number [opcional] - El desplazamiento del valor del canal de color azul (de -255 a 255). El valor predeterminado es 0.

alphaOffset:Number [opcional] - El desplazamiento del valor del canal de transparencia alfa (de -255 a 255). El valor predeterminado es 0.

Ejemplo

En el ejemplo siguiente se crea un objeto ColorTransform denominado `greenTransform`:

```
var greenTransform:flash.geom.ColorTransform = new
  flash.geom.ColorTransform(0.5, 1.0, 0.5, 0.5, 10, 10, 10, 0);
```

El ejemplo siguiente crea el objeto ColorTransfrom `colorTrans_1` con valores de constructor predeterminados. El hecho de que `colorTrans_1` y `colorTrans_2` obtengan los mismos valores demuestra que se utilizan los valores predeterminados del constructor.

```
import flash.geom.ColorTransform;

var colorTrans_1:ColorTransform = new ColorTransform(1, 1, 1, 1, 0, 0, 0,
  0);
trace(colorTrans_1);
//(redMultiplier=1, greenMultiplier=1, blueMultiplier=1, alphaMultiplier=1,
  redOffset=0, greenOffset=0, blueOffset=0, alphaOffset=0)

var colorTrans_2:ColorTransform = new ColorTransform();
trace(colorTrans_2);
//(redMultiplier=1, greenMultiplier=1, blueMultiplier=1, alphaMultiplier=1,
  redOffset=0, greenOffset=0, blueOffset=0, alphaOffset=0)
```

concat (método ColorTransform.concat)

```
public concat(second:ColorTransform) : Void
```

Aplica una segunda transformación adicional de color al clip de película. La segunda serie de parámetros de transformación se aplica a los colores del clip de película cuando se ha completado la primera transformación.

Disponibilidad: ActionScript 1.0; Flash Player 8

Parámetros

second:flash.geom.ColorTransform - Un segundo objeto ColorTransform para combinarlo con el objeto ColorTransform actual.

Ejemplo

El ejemplo siguiente concatena el objeto ColorTransform `colorTrans_2` con `colorTrans_1`, lo que da como resultado un desplazamiento rojo completo combinado con un multiplicador de alfa 0,5.

```
import flash.geom.ColorTransform;
import flash.geom.Transform;

var colorTrans_1:ColorTransform = new ColorTransform(1, 1, 1, 1, 255, 0, 0,
  0);
trace(colorTrans_1);
// (redMultiplier=1, greenMultiplier=1, blueMultiplier=1,
  alphaMultiplier=1, redOffset=255, greenOffset=0, blueOffset=0,
  alphaOffset=0)

var colorTrans_2:ColorTransform = new ColorTransform(1, 1, 1, .5, 0, 0, 0,
  0);
trace(colorTrans_2);
// (redMultiplier=1, greenMultiplier=1, blueMultiplier=1,
  alphaMultiplier=0.5, redOffset=0, greenOffset=0, blueOffset=0,
  alphaOffset=0)

colorTrans_1.concat(colorTrans_2);
trace(colorTrans_1);
// (redMultiplier=1, greenMultiplier=1, blueMultiplier=1,
  alphaMultiplier=0.5, redOffset=255, greenOffset=0, blueOffset=0,
  alphaOffset=0)

var rect:MovieClip = createRectangle(20, 80, 0x000000);
var trans:Transform = new Transform(rect);
trans.colorTransform = colorTrans_1;

function createRectangle(width:Number, height:Number, color:Number,
  scope:MovieClip):MovieClip {
```

```
  scope = (scope == undefined) ? this : scope;
  var depth:Number = scope.getNextHighestDepth();
  var mc:MovieClip = scope.createEmptyMovieClip("mc_" + depth, depth);
  mc.beginFill(color);
  mc.lineTo(0, height);
  mc.lineTo(width, height);
  mc.lineTo(width, 0);
  mc.lineTo(0, 0);
  return mc;
}
```

greenMultiplier (propiedad ColorTransform.greenMultiplier)

```
public greenMultiplier : Number
```

Valor decimal que se multiplica por el valor del canal verde.

Disponibilidad: ActionScript 1.0; Flash Player 8

Ejemplo

El ejemplo siguiente crea el objeto ColorTransform `colorTrans` y ajusta su valor de `greenMultiplier` de 1 a 0,5.

```
import flash.geom.ColorTransform;
import flash.geom.Transform;

var colorTrans:ColorTransform = new ColorTransform();
trace(colorTrans.greenMultiplier); // 1

colorTrans.greenMultiplier = .5;
trace(colorTrans.greenMultiplier); // .5

var rect:MovieClip = createRectangle(20, 80, 0x00FF00);
var trans:Transform = new Transform(rect);
trans.colorTransform = colorTrans;

function createRectangle(width:Number, height:Number, color:Number,
  scope:MovieClip):MovieClip {
  scope = (scope == undefined) ? this : scope;
  var depth:Number = scope.getNextHighestDepth();
  var mc:MovieClip = scope.createEmptyMovieClip("mc_" + depth, depth);
  mc.beginFill(color);
  mc.lineTo(0, height);
  mc.lineTo(width, height);
  mc.lineTo(width, 0);
  mc.lineTo(0, 0);
  return mc;
}
```

greenOffset (propiedad ColorTransform.greenOffset)

`public greenOffset : Number`

Número de -255 a 255 que se añade al valor del canal verde después de multiplicarse por el valor de `greenMultiplier`.

Disponibilidad: ActionScript 1.0; Flash Player 8

Ejemplo

El ejemplo siguiente crea el objeto ColorTransform `colorTrans` y ajusta su valor de `greenOffset` de 0 a 255.

```
import flash.geom.ColorTransform;
import flash.geom.Transform;

var colorTrans:ColorTransform = new ColorTransform();
trace(colorTrans.greenOffset); // 0

colorTrans.greenOffset = 255;
trace(colorTrans.greenOffset); // 255

var rect:MovieClip = createRectangle(20, 80, 0x000000);
var trans:Transform = new Transform(rect);
trans.colorTransform = colorTrans;

function createRectangle(width:Number, height:Number, color:Number,
  scope:MovieClip):MovieClip {
  scope = (scope == undefined) ? this : scope;
  var depth:Number = scope.getNextHighestDepth();
  var mc:MovieClip = scope.createEmptyMovieClip("mc_" + depth, depth);
  mc.beginFill(color);
  mc.lineTo(0, height);
  mc.lineTo(width, height);
  mc.lineTo(width, 0);
  mc.lineTo(0, 0);
  return mc;
}
```

redMultiplier (propiedad ColorTransform.redMultiplier)

```
public redMultiplier : Number
```

Valor decimal que se multiplica por el valor del canal rojo.

Disponibilidad: ActionScript 1.0; Flash Player 8

Ejemplo

El ejemplo siguiente crea el objeto ColorTransform `colorTrans` y ajusta su valor de `redMultiplier` de 1 a 0,5.

```
import flash.geom.ColorTransform;
import flash.geom.Transform;

var colorTrans:ColorTransform = new ColorTransform();
trace(colorTrans.redMultiplier); // 1

colorTrans.redMultiplier = .5;
trace(colorTrans.redMultiplier); // .5

var rect:MovieClip = createRectangle(20, 80, 0xFF0000);
var trans:Transform = new Transform(rect);
trans.colorTransform = colorTrans;

function createRectangle(width:Number, height:Number, color:Number,
  scope:MovieClip):MovieClip {
  scope = (scope == undefined) ? this : scope;
  var depth:Number = scope.getNextHighestDepth();
  var mc:MovieClip = scope.createEmptyMovieClip("mc_" + depth, depth);
  mc.beginFill(color);
  mc.lineTo(0, height);
  mc.lineTo(width, height);
  mc.lineTo(width, 0);
  mc.lineTo(0, 0);
  return mc;
}
```

redOffset (propiedad ColorTransform.redOffset)

```
public redOffset : Number
```

Número de -255 a 255 que se añade al valor del canal rojo después de multiplicarse por el valor de `redMultiplier`.

Disponibilidad: ActionScript 1.0; Flash Player 8

Ejemplo

El ejemplo siguiente crea el objeto ColorTransform `colorTrans` y ajusta su valor de `redOffset` de 0 a 255.

```
import flash.geom.ColorTransform;
import flash.geom.Transform;

var colorTrans:ColorTransform = new ColorTransform();
trace(colorTrans.redOffset); // 0

colorTrans.redOffset = 255;
trace(colorTrans.redOffset); // 255

var rect:MovieClip = createRectangle(20, 80, 0x000000);
var trans:Transform = new Transform(rect);
trans.colorTransform = colorTrans;

function createRectangle(width:Number, height:Number, color:Number,
  scope:MovieClip):MovieClip {
  scope = (scope == undefined) ? this : scope;
  var depth:Number = scope.getNextHighestDepth();
  var mc:MovieClip = scope.createEmptyMovieClip("mc_" + depth, depth);
  mc.beginFill(color);
  mc.lineTo(0, height);
  mc.lineTo(width, height);
  mc.lineTo(width, 0);
  mc.lineTo(0, 0);
  return mc;
}
```

rgb (propiedad ColorTransform.rgb)

```
public rgb : Number
```

Valor del color RGB de un objeto ColorTransform.

Cuando define esta propiedad, cambia en consecuencia los tres valores de desplazamiento de color (`redOffset`, `greenOffset` y `blueOffset`) y define los tres valores de multiplicador de color (`redMultiplier`, `greenMultiplier` y `blueMultiplier`) como 0. El multiplicador y los valores de desplazamiento de alfa no se modifican.

Para pasar un valor para esta propiedad, utilice el formato: 0x*RRGGBB*. *RR*, *GG* y *BB* constan cada uno de dos dígitos hexadecimales que especifican el desplazamiento de cada componente de color. El 0x indica al compilador de ActionScript que el número es un valor hexadecimal.

Disponibilidad: ActionScript 1.0; Flash Player 8

Ejemplo

El ejemplo siguiente crea el objeto ColorTransform `colorTrans` y ajusta su valor de `rgb` como 0xFF0000.

```
import flash.geom.ColorTransform;
import flash.geom.Transform;

var colorTrans:ColorTransform = new ColorTransform();
trace(colorTrans.rgb); // 0

colorTrans.rgb = 0xFF0000;
trace(colorTrans.rgb); // 16711680
trace("0x" + colorTrans.rgb.toString(16)); // 0xff0000

var rect:MovieClip = createRectangle(20, 80, 0x000000);
var trans:Transform = new Transform(rect);
trans.colorTransform = colorTrans;

function createRectangle(width:Number, height:Number, color:Number,
  scope:MovieClip):MovieClip {
  scope = (scope == undefined) ? this : scope;
  var depth:Number = scope.getNextHighestDepth();
  var mc:MovieClip = scope.createEmptyMovieClip("mc_" + depth, depth);
  mc.beginFill(color);
  mc.lineTo(0, height);
  mc.lineTo(width, height);
  mc.lineTo(width, 0);
  mc.lineTo(0, 0);
  return mc;
}
```

Véase también

`setRGB (método Color.setRGB)`, `getRGB (método Color.getRGB)`

toString (método ColorTransform.toString)

```
public toString() : Cadena
```

Formatea y devuelve una cadena que describe todas las propiedades del objeto ColorTransform.

Disponibilidad: ActionScript 1.0; Flash Player 8

Valor devuelto

`String` - Una cadena que muestra todas las propiedades del objeto ColorTransform.

Ejemplo

En el ejemplo siguiente se crea el objeto ColorTransform `colorTrans` y se llama a su método `toSting()`. Este método da como resultado una cadena con el formato siguiente: (redMultiplier=RM, greenMultiplier=GM, blueMultiplier=BM, alphaMultiplier=AM, redOffset=RO, greenOffset=GO, blueOffset=BO, alphaOffset=AO).

```
import flash.geom.ColorTransform;

var colorTrans:ColorTransform = new ColorTransform(1, 2, 3, 4, -255, -128,
  128, 255);
trace(colorTrans.toString());
// (redMultiplier=1, greenMultiplier=2, blueMultiplier=3,
  alphaMultiplier=4, redOffset=-255, greenOffset=-128, blueOffset=128,
  alphaOffset=255)
```

ContextMenu

```
Object
  |
  +-ContextMenu
```

`public dynamic class` ***ContextMenu***
`extends Object`

La clase ContextMenu permite controlar durante la ejecución los elementos del menú contextual de Flash Player, que aparece cuando un usuario hace clic con el botón derecho del ratón (Windows) o mantiene presionada la tecla Control mientras hace clic (Macintosh) en Flash Player. Puede utilizar los métodos y las propiedades de la clase ContextMenu para añadir elementos de menú personalizados, controlar la visualización de los elementos del menú contextual incorporados (por ejemplo, Acercar e Imprimir) o crear copias de menús.

Puede asociar un objeto ContextMenu a un botón, un clip de película u objeto de campo de texto específico, o bien a todo un nivel de una película. Para ello, deberá utilizar la propiedad menu de las clases Button, MovieClip o TextField. Para más información sobre la propiedad `menu`, consulte `Button.menu`, `MovieClip.menu` y `TextField.menu`.

Para agregar nuevos elementos a un objeto `ContextMenu`, crea un objeto `ContextMenuItem` y, a continuación, añade ese objeto a la matriz `ContextMenu.customItems`. Para más información sobre la creación de elementos de menús contextuales, consulte la entrada de la clase ContextMenuItem.

Flash Player dispone de tres tipos de menús contextuales: el menú estándar (que aparece al hacer clic con el botón derecho del ratón en Flash Player), el menú de edición (que aparece al hacer clic con el botón derecho del ratón en un campo de texto seleccionable o editable) y un menú de error (que aparece cuando un archivo SWF no logra cargarse en Flash Player.) Sólo los menús estándar y de edición pueden modificarse con la clase ContextMenu.

Los elementos de menú personalizados siempre aparecen en la parte superior del menú contextual de Flash Player, por encima de los elementos de menú incorporados que estén visibles; una barra separadora permite distinguir los elementos de menú incorporados de los personalizados. Puede añadir un máximo de 15 elementos personalizados a un menú contextual. No es posible quitar el elemento de menú Configuración del menú contextual. El elemento de menú Configuración es necesario para que los usuarios de Flash puedan acceder a la configuración que afecta a la privacidad y el almacenamiento en sus equipos. El elemento de menú Acerca de tampoco se puede quitar del menú contextual, ya que es necesario para que los usuarios puedan conocer la versión de Flash Player que están utilizando.

Debe utilizar el constructor `ContextMenu()` para crear un objeto ContextMenu antes de llamar a sus métodos.

Disponibilidad: ActionScript 1.0; Flash Player 7

Véase también

`ContextMenuItem`, `menu (propiedad Button.menu)`, `menu (propiedad MovieClip.menu)`, `menu (propiedad TextField.menu)`

Resumen de propiedades

Modificadores	Propiedad	Descripción
	`builtInItems:Object`	Objeto que tiene las siguientes propiedades booleanas: `zoom`, `quality`, `play`, `loop`, `rewind`, `forward_back` y `print`.
	`customItems:Array`	Una matriz de objetos ContextMenuItem.

Propiedades heredadas de la clase Object

`constructor (propiedad Object.constructor)`, `__proto__ (propiedad Object.__proto__)`, `prototype (propiedad Object.prototype)`, `__resolve (propiedad Object.__resolve)`

Resumen de eventos

Evento	Descripción
`onSelect = function(item:Object, item_menu:Object) {}`	Se efectúa una llamada a éste cuando un usuario invoca el menú contextual de Flash Player, pero antes de que se muestre realmente el menú.

Resumen de constructores

Firma	Descripción
`ContextMenu([callbackFunction:Function])`	Crea un objeto ContextMenu.

Resumen de métodos

Modificadores	Firma	Descripción
	`copy() : ContextMenu`	Crea una copia del objeto ContextMenu especificado.
	`hideBuiltInItems() : Void`	Oculta todos los elementos de menú incorporados (salvo Configuración) del objeto ContextMenu especificado.

Métodos heredados de la clase Object

`addProperty (método Object.addProperty)`, `hasOwnProperty (método Object.hasOwnProperty)`, `isPropertyEnumerable (método Object.isPropertyEnumerable)`, `isPrototypeOf (método Object.isPrototypeOf)`, `registerClass (método Object.registerClass)`, `toString (método Object.toString)`, `unwatch (método Object.unwatch)`, `valueOf (método Object.valueOf)`, `watch (método Object.watch)`

builtInItems (propiedad ContextMenu.builtInItems)

`public builtInItems : Object`

Objeto que tiene las siguientes propiedades booleanas: `zoom`, `quality`, `play`, `loop`, `rewind`, `forward_back` y `print`. La configuración de estas variables con el valor `false` quita los elementos de menú correspondientes del objeto ContextMenu especificado. Estas propiedades son enumerables y están configuradas con el valor `true` de manera predeterminada.

Disponibilidad: ActionScript 1.0; Flash Player 7

Ejemplo

En este ejemplo, los elementos de menú incorporados Quality y Print están desactivados para el objeto ContextMenu `my_cm`, que está asociado a la línea de tiempo actual del archivo SWF.

```
var my_cm:ContextMenu = new ContextMenu ();
my_cm.builtInItems.quality=false;
my_cm.builtInItems.print=false;
this.menu = my_cm;
```

Nota: No es posible desactivar los elementos de menú Configuración o Acerca de del menú contextual.

En el ejemplo siguiente, un bucle `for..in` enumera todos los nombres y valores de los elementos de menú incorporados del objeto ContextMenu `my_cm`.

```
var my_cm:ContextMenu = new ContextMenu();
for(eachProp in my_cm.builtInItems) {
  var propName = eachProp;
  var propValue = my_cm.builtInItems[propName];
  trace(propName + ": " + propValue);
}
```

Constructor ContextMenu

```
public ContextMenu([callbackFunction:Function])
```

Crea un objeto ContextMenu. Puede especificar opcionalmente un identificador para un controlador de eventos cuando cree el objeto. Se efectúa una llamada a la función especificada cuando el usuario invoca el menú contextual, pero *antes* de que se muestre el menú. Resulta útil para personalizar el contenido de los menús en función del estado de la aplicación o en función del tipo de objeto (clip de película, campo de texto o botón) o de la línea de tiempo en la que el usuario hace clic con el botón derecho del ratón o hace clic mientras presiona Control. (Para obtener un ejemplo de creación de un controlador de eventos, consulte `ContextMenu.onSelect`.)

Disponibilidad: ActionScript 1.0; Flash Player 7

Parámetros

`callbackFunction``:Function` [opcional] - Una referencia a una función a la que se llama cuando el usuario hace clic con el botón derecho del ratón o hace clic mientras pulsa la tecla Control, antes de que se muestre el menú.

Ejemplo

El ejemplo siguiente oculta todos los objetos incorporados en el menú contextual. (Sin embargo, los elementos Configuración y Acerca de siguen apareciendo, ya que no se pueden desactivar.)

```
var newMenu:ContextMenu = new ContextMenu();
newMenu.hideBuiltInItems();
this.menu = newMenu;
```

En este ejemplo, el controlador de eventos especificado, `menuHandler`, activa o desactiva un elemento de menú (utilizando la matriz `ContextMenu.customItems`) según el valor de una variable booleana llamada `showItem`. Si es `false`, el elemento de menú personalizado se desactiva; en caso contrario, se activa.

```
var showItem = true; // Change this to false to remove
var my_cm:ContextMenu = new ContextMenu(menuHandler);
my_cm.customItems.push(new ContextMenuItem("Hello", itemHandler));
function menuHandler(obj, menuObj) {
  if (showItem == false) {
  menuObj.customItems[0].enabled = false;
  } else {
  menuObj.customItems[0].enabled = true;
  }
}
function itemHandler(obj, item) {
  //...put code here...
  trace("selected!");
}
this.menu = my_cm;
```

Cuando el usuario hace clic con el botón derecho del ratón o hace clic mientras pulsa la tecla Control, se mostrará el menú personalizado.

Véase también

`menu (propiedad Button.menu)`, `onSelect (controlador ContextMenu.onSelect)`, `customItems (propiedad ContextMenu.customItems)`, `hideBuiltInItems (método ContextMenu.hideBuiltInItems)`, `menu (propiedad MovieClip.menu)`, `menu (propiedad TextField.menu)`

copy (método ContextMenu.copy)

```
public copy() : ContextMenu
```

Crea una copia del objeto ContextMenu especificado. La copia hereda todas las propiedades del objeto de menú original.

Disponibilidad: ActionScript 1.0; Flash Player 7

Valor devuelto

`ContextMenu` - Un objeto ContextMenu.

Ejemplo

Este ejemplo crea una copia del objeto ContextMenu llamado `my_cm` cuyos elementos de menú incorporados están ocultos, y añade un elemento de menú con el texto "Save...". A continuación crea una copia de `my_cm` y la asigna a la variable `clone_cm`, que hereda todas las propiedades del menú original.

```
var my_cm:ContextMenu = new ContextMenu();
my_cm.hideBuiltInItems();
var menuItem_cmi:ContextMenuItem = new ContextMenuItem("Save...",
  saveHandler);
my_cm.customItems.push(menuItem_cmi);
function saveHandler(obj, menuItem) {
  // saveDocument();
  // custom function (not shown)
  trace("something");
}
clone_cm = my_cm.copy();
this.menu = my_cm;
for (var i in clone_cm.customItems) {
  trace("clone_cm-> "+clone_cm.customItems[i].caption);
}
for (var i in my_cm.customItems) {
  trace("my_cm-> "+my_cm.customItems[i].caption);
}
```

customItems (propiedad ContextMenu.customItems)

```
public customItems : Array
```

Una matriz de objetos ContextMenuItem. Cada objeto de la matriz representa un elemento del menú contextual que ha definido. Utilice esta propiedad para añadir, quitar o modificar estos elementos de menú personalizados.

Para añadir nuevos elementos al menú, cree en primer lugar un nuevo objeto ContextMenuItem y luego añádalo a la matriz *`menu_mc`* `.customItems` (por ejemplo, utilizando `Array.push()`. Para más información sobre la creación de nuevos elementos de menú, consulte la entrada de la clase ContextMenuItem.

Disponibilidad: ActionScript 1.0; Flash Player 7

Ejemplo

El ejemplo siguiente crea un nuevo elemento de menú personalizado llamado `menuItem_cmi` con el título "Send e-mail" y un controlador callback llamado `emailHandler`. El nuevo elemento de menú se añade a continuación al objeto ContextMenu `my_cm` empleando la matriz `customItems`. Por último, el nuevo menú se asocia a un clip de película llamado `email_mc`. Para que este ejemplo funcione, cree una instancia de clip de película en el escenario y utilice el inspector de propiedades para asignarle el nombre `email_mc` a la instancia. En modo Probar película, el nuevo elemento de menú contextual aparecerá si abre el menú contextual mientras el puntero se encuentra sobre el clip de película `email_mc`.

```
var my_cm:ContextMenu = new ContextMenu();
var menuItem_cmi:ContextMenuItem = new ContextMenuItem("Send e-mail",
  emailHandler);
my_cm.customItems.push(menuItem_cmi);
email_mc.menu = my_cm;
function emailHandler() {
  trace("sending email");
}
```

Véase también

`menu (propiedad Button.menu)`, `menu (propiedad MovieClip.menu)`, `menu (propiedad TextField.menu)`, `push (método Array.push)`

hideBuiltInItems (método ContextMenu.hideBuiltInItems)

```
public hideBuiltInItems() : Void
```

Oculta todos los elementos de menú incorporados (salvo Configuración) del objeto ContextMenu especificado. Si se está en ejecución el Reproductor de depuración de Flash, se mostrará el elemento de menú Depuración, aunque éste estará atenuado para archivos SWF que no tengan activada la depuración remota.

Este método oculta sólo los elementos de menú que aparecen en el menú contextual estándar; no afecta a los elementos que aparecen en los menús de edición o de error.

Este método funciona configurando todos los miembros booleanos de *`my_cm`*`.builtInItems` como `false`. Puede seleccionar los elementos incorporados que deben estar visibles configurando su correspondiente miembro en *`my_cm`*`.builtInItems` como `true` (como se muestra en el siguiente ejemplo).

Disponibilidad: ActionScript 1.0; Flash Player 7

Ejemplo

El ejemplo siguiente crea un objeto ContextMenu nuevo llamado `my_cm` cuyos elementos de menú incorporados están ocultos, salvo Print. El objeto de menú está asociado a la línea de tiempo actual.

```
var my_cm:ContextMenu = new ContextMenu();
my_cm.hideBuiltInItems();
my_cm.builtInItems.print = true;
this.menu = my_cm;
```

onSelect (controlador ContextMenu.onSelect)

```
onSelect = function(item:Object, item_menu:Object) {}
```

Se efectúa una llamada a éste cuando un usuario invoca el menú contextual de Flash Player, pero antes de que se muestre realmente el menú. Este controlador de eventos permite personalizar el contenido del menú contextual en función del estado actual de la aplicación.

También es posible especificar el controlador callback para un objeto ContextMenu al construir un objeto ContextMenu nuevo. Para más información, consulte la entrada de ContextMenuItem `onSelect`.

Disponibilidad: ActionScript 1.0; Flash Player 7

Parámetros

`item``:Object` - Una referencia al objeto (clip de película, botón o campo de texto seleccionable) que estaba debajo del puntero del ratón cuando se activó el menú contextual de Flash Player y cuya propiedad `menu` está definida como un objeto ContextMenu válido.

`item_menu``:Object` - Una referencia al objeto ContextMenu asignado a la propiedad `menu` de `object`.

Ejemplo

El ejemplo siguiente determina sobre qué tipo de objeto se ha invocado el menú contextual.

```
my_cm:ContextMenu = new ContextMenu();
function menuHandler(obj:Object, menu:ContextMenu) {
  if(obj instanceof MovieClip) {
    trace("Movie clip: " + obj);
  }
  if(obj instanceof TextField) {
    trace("Text field: " + obj);
  }
  if(obj instanceof Button) {
    trace("Button: " + obj);
  }
}
```

```
my_cm.onSelect = menuHandler;
my_mc.menu = my_cm;
my_btn.menu = my_cm;
```

ContextMenuItem

```
Object
  |
  +-ContextMenuItem
```

```
public dynamic class ContextMenuItem
extends Object
```

La clase ContextMenuItem permite crear elementos de menú personalizados que deben aparecer en el menú contextual de Flash Player. Cada objeto ContextMenuItem tiene un texto que aparece en el menú contextual, así como un controlador callback (una función) que se invoca cuando se selecciona el elemento de menú. Para añadir un nuevo elemento a un menú contextual, añádalo a la matriz customItems de un objeto ContextMenu.

Puede activar o desactivar elementos de menú específicos, hacer que los elementos sean visibles o invisibles o cambiar el texto o el controlador callback asociado a un elemento de menú.

Los elementos de menú personalizados aparecen en la parte superior del menú contextual, por encima de los elementos de menú incorporados. Una barra separadora siempre divide los elementos de menú incorporados y los personalizados. Puede añadir un máximo de 15 elementos personalizados a un menú contextual. Cada elemento debe contener al menos un carácter visible; no se tendrán en cuenta los caracteres de control, las nuevas líneas y otros caracteres de espacio en blanco. Ningún elemento puede tener una longitud superior a 100 caracteres. Los elementos que sean idénticos a un elemento de menú incorporado o a otro elemento personalizado no se tendrán en cuenta, con independencia de si el elemento que es idéntico está o no visible. Los elementos de menú se comparan sin distinguir entre mayúsculas y minúsculas ni tener en cuenta la puntuación o los espacios en blanco.

Ninguna de las siguientes palabras puede aparecer en un elemento personalizado: *Macromedia*, *Flash Player* o *Configuración*.

Disponibilidad: ActionScript 1.0; Flash Player 7

Resumen de propiedades

Modificadores	Propiedad	Descripción
	`caption:String`	Cadena que especifica el texto del elemento de menú que se muestra en el menú contextual.
	`enabled:Boolean`	Valor booleano que indica si el elemento de menú especificado está activado o desactivado.
	`separatorBefore:Boolean`	Valor booleano que indica si debe aparecer una barra separadora por encima del elemento de menú especificado.
	`visible:Boolean`	Valor booleano que indica si el elemento de menú especificado estará visible cuando se muestre el menú contextual de Flash Player.

Propiedades heredadas de la clase Object

`constructor (propiedad Object.constructor), __proto__ (propiedad Object.__proto__), prototype (propiedad Object.prototype), __resolve (propiedad Object.__resolve)`

Resumen de eventos

Evento	Descripción
`onSelect = function(obj:Object, menuItem:Object) {}`	Se invoca cuando se selecciona el elemento de menú especificado del menú contextual de Flash Player.

Resumen de constructores

Firma	Descripción
`ContextMenuItem(caption:String, callbackFunction:Function, [separatorBefore:Boolean], [enabled:Boolean], [visible:Boolean])`	Crea un nuevo objeto ContextMenuItem que puede añadirse a la matriz `ContextMenu.customItems`.

Resumen de métodos

Modificadores	Firma	Descripción
	`copy() : ContextMenuItem`	Crea y devuelve una copia del objeto ContextMenuItem especificado.

Métodos heredados de la clase Object

`addProperty (método Object.addProperty), hasOwnProperty (método Object.hasOwnProperty), isPropertyEnumerable (método Object.isPropertyEnumerable), isPrototypeOf (método Object.isPrototypeOf), registerClass (método Object.registerClass), toString (método Object.toString), unwatch (método Object.unwatch), valueOf (método Object.valueOf), watch (método Object.watch)`

caption (propiedad ContextMenuItem.caption)

```
public caption : String
```

Cadena que especifica el texto del elemento de menú que se muestra en el menú contextual.

Disponibilidad: ActionScript 1.0; Flash Player 7

Ejemplo

El ejemplo siguiente muestra el texto del elemento de menú seleccionado (Pause Game) en el panel Salida:

```
var my_cm:ContextMenu = new ContextMenu();
var menuItem_cmi:ContextMenuItem = new ContextMenuItem("Pause Game",
  onPause);
my_cm.customItems.push(menuItem_cmi);
function onPause(obj, menuItem) {
  trace("You chose: " + menuItem.caption);
}
this.menu = my_cm;
```

Constructor ContextMenuItem

```
public ContextMenuItem(caption:String, callbackFunction:Function,
  [separatorBefore:Boolean], [enabled:Boolean], [visible:Boolean])
```

Crea un nuevo objeto ContextMenuItem que puede añadirse a la matriz `ContextMenu.customItems`.

Disponibilidad: ActionScript 1.0; Flash Player 7

Parámetros

caption:String - Una cadena que especifica el texto asociado con el elemento de menú.

callbackFunction:Function - Función definida a la que se llama cuando se selecciona el elemento de menú.

separatorBefore:Boolean [opcional] - Valor booleano que indica si debe aparecer una barra separadora por encima del elemento de menú en el menú contextual. El valor predeterminado es `false`.

enabled:Boolean [opcional] - Valor booleano que indica si el elemento de menú está activado o desactivado en el menú contextual. El valor predeterminado es `true`.

visible:Boolean [opcional] - Valor booleano que indica si el elemento de menú es visible o invisible. El valor predeterminado es `true`.

Ejemplo

Este ejemplo añade los elementos de menú Start y Stop, separados por una barra, al objeto ContextMenu `my_cm`. Se llama a la función `startHandler()` cuando se selecciona Start en el menú contextual, y a `stopHandler()` cuando se selecciona Stop. El objeto ContextMenu se aplica a la línea de tiempo actual.

```
var my_cm:ContextMenu = new ContextMenu();
my_cm.customItems.push(new ContextMenuItem("Start", startHandler));
my_cm.customItems.push(new ContextMenuItem("Stop", stopHandler, true));
function stopHandler(obj, item) {
  trace("Stopping...");
}
function startHandler(obj, item) {
  trace("Starting...");
}
this.menu = my_cm;
```

copy (método ContextMenuItem.copy)

```
public copy() : ContextMenuItem
```

Crea y devuelve una copia del objeto ContextMenuItem especificado. La copia incluye todas las propiedades del objeto original.

Disponibilidad: ActionScript 1.0; Flash Player 7

Valor devuelto

`ContextMenuItem` - Un objeto ContextMenuItem.

Ejemplo

Este ejemplo crea un nuevo objeto ContextMenuItem llamado `original_cmi` con el título Pause y un controlador callback establecido en la función `onPause`. A continuación, el ejemplo crea una copia del objeto ContextMenuItem y la asigna a la variable `copy_cmi`.

```
var original_cmi:ContextMenuItem = new ContextMenuItem("Pause", onPause);
function onPause(obj:Object, menu:ContextMenu) {
  trace("pause me");
}

var copy_cmi:ContextMenuItem = original_cmi.copy();

var my_cm:ContextMenu = new ContextMenu();
my_cm.customItems.push(original_cmi);
my_cm.customItems.push(copy_cmi);

my_mc.menu = my_cm;
```

enabled (propiedad ContextMenuItem.enabled)

```
public enabled : Boolean
```

Valor booleano que indica si el elemento de menú especificado está activado o desactivado. De forma predeterminada, esta propiedad es `true`.

Disponibilidad: ActionScript 1.0; Flash Player 7

Ejemplo

El ejemplo siguiente crea dos nuevos elementos de menú contextual: Start y Stop. Cuando el usuario selecciona Start se realiza un seguimiento del número de milisegundos transcurridos desde el momento en que se inició el archivo SWF. A continuación, se desactiva Start en el menú. Cuando se selecciona Stop se realiza un seguimiento del número de milisegundos transcurridos desde que se inició el archivo SWF. Se reactiva el elemento de menú Start y se desactiva el elemento Stop.

```
var my_cm:ContextMenu = new ContextMenu();
var startMenuItem:ContextMenuItem = new ContextMenuItem("Start",
  startHandler);
startMenuItem.enabled = true;
my_cm.customItems.push(startMenuItem);
var stopMenuItem:ContextMenuItem = new ContextMenuItem("Stop", stopHandler,
  true);
stopMenuItem.enabled = false;
my_cm.customItems.push(stopMenuItem);
function stopHandler(obj, item) {
  trace("Stopping... "+getTimer()+"ms");
  startMenuItem.enabled = true;
```

```
  stopMenuItem.enabled = false;
}
function startHandler(obj, item) {
  trace("Starting... "+getTimer()+"ms");
  startMenuItem.enabled = false;
  stopMenuItem.enabled = true;
}
this.menu = my_cm;
```

onSelect (controlador ContextMenuItem.onSelect)

```
onSelect = function(obj:Object, menuItem:Object) {}
```

Se invoca cuando se selecciona el elemento de menú especificado del menú contextual de Flash Player. El controlador callback especificado recibe dos parámetros: `obj`, una referencia al objeto situado bajo el ratón cuando el usuario invoca el menú contextual de Flash Player, y `item`, una referencia al objeto ContextMenuItem que representa al elemento de menú seleccionado.

Disponibilidad: ActionScript 1.0; Flash Player 7

Parámetros

`obj``:Object` - Una referencia al objeto (clip de película, línea de tiempo, botón o campo de texto seleccionable) en el que el usuario ha hecho clic con el botón derecho o la tecla Control presionada.

`menuItem``:Object` - Una referencia al objeto ContextMenuItem seleccionado.

Ejemplo

El ejemplo siguiente determina sobre qué tipo de objeto se ha invocado el menú contextual.

```
var my_cmi:ContextMenu = new ContextMenu();
var start_cmi:ContextMenuItem = new ContextMenuItem("Start");
start_cmi.onSelect = function(obj, item) {
  trace("You chose: "+item.caption);
};
my_cmi.customItems.push(start_cmi);
my_cmi.customItems.push(new ContextMenuItem("Stop", stopHandler, true));
function stopHandler(obj, item) {
  trace("Stopping...");
}
this.menu = my_cmi;
```

Véase también

onSelect (controlador ContextMenu.onSelect)

separatorBefore (propiedad ContextMenuItem.separatorBefore)

```
public separatorBefore : Boolean
```

Valor booleano que indica si debe aparecer una barra separadora por encima del elemento de menú especificado. De forma predeterminada, esta propiedad es `false`.

Nota: Siempre se muestra una barra separadora entre los elementos de menú personalizados y los elementos de menú incorporados.

Disponibilidad: ActionScript 1.0; Flash Player 7

Ejemplo

Este ejemplo crea tres elementos de menú, con las etiqueta Open, Save y Print. Una barra de separación divide los elementos Save y Print. A continuación, los elementos de menú se añaden a la matriz `customItems` del objeto ContextMenu. Por último, el menú se asocia a la línea de tiempo actual del archivo SWF.

```
var my_cm:ContextMenu = new ContextMenu();
var open_cmi:ContextMenuItem = new ContextMenuItem("Open", itemHandler);
var save_cmi:ContextMenuItem = new ContextMenuItem("Save", itemHandler);
var print_cmi:ContextMenuItem = new ContextMenuItem("Print", itemHandler);
print_cmi.separatorBefore = true;
my_cm.customItems.push(open_cmi, save_cmi, print_cmi);
function itemHandler(obj, menuItem) {
  trace("You chose: " + menuItem.caption);
};
this.menu = my_cm;
```

Véase también

`onSelect (controlador ContextMenu.onSelect)`

visible (propiedad ContextMenuItem.visible)

```
public visible : Boolean
```

Valor booleano que indica si el elemento de menú especificado estará visible cuando se muestre el menú contextual de Flash Player. De forma predeterminada, esta propiedad es `true`.

Disponibilidad: ActionScript 1.0; Flash Player 7

Ejemplo

El ejemplo siguiente crea dos nuevos elementos de menú contextual: Start y Stop. Cuando el usuario selecciona Start se muestra el número de milisegundos transcurridos desde el momento en que se inició el archivo SWF. A continuación, Start se hace invisible en el menú. Cuando se selecciona Stop se muestra el número de milisegundos transcurridos desde que se inició el archivo SWF. El elemento de menú Start se hace visible y el elemento Stop se hace invisible.

```
var my_cm:ContextMenu = new ContextMenu();
var startMenuItem:ContextMenuItem = new ContextMenuItem("Start",
  startHandler);
startMenuItem.visible = true;
my_cm.customItems.push(startMenuItem);
var stopMenuItem:ContextMenuItem = new ContextMenuItem("Stop", stopHandler,
  true);
stopMenuItem.visible = false;
my_cm.customItems.push(stopMenuItem);
function stopHandler(obj, item) {
  trace("Stopping... "+getTimer()+"ms");
  startMenuItem.visible = true;
  stopMenuItem.visible = false;
}
function startHandler(obj, item) {
  trace("Starting... "+getTimer()+"ms");
  startMenuItem.visible = false;
  stopMenuItem.visible = true;
}
this.menu = my_cm;
```

ConvolutionFilter (flash.filters.ConvolutionFilter)

```
Object
  |
  +-flash.filters.BitmapFilter
    |
    +-flash.filters.ConvolutionFilter
```

```
public class ConvolutionFilter
extends BitmapFilter
```

La clase ConvolutionFilter aplica un efecto de filtro de convolución de matrices. Una convolución combina píxeles en la imagen de entrada con píxeles colindantes para producir una imagen. Las convoluciones permiten conseguir una amplia variedad de operaciones de imagen, como desenfoque, detección de bordes, aumento de la nitidez, relieve y biselado. Este efecto se puede aplicar a mapas de bits e instancias de clip de película.

El uso de filtros depende del objeto al que se aplique el filtro:

- Para aplicar filtros a clips de película en tiempo de ejecución, utilice la propiedad `filters`. Establecer la propiedad `filters` de un objeto no modifica el objeto y se puede deshacer borrando la propiedad `filters`.
- Para aplicar filtros a instancias de BitmapData, utilice el método `BitmapData.applyFilter()`. Cuando se llama a `applyFilter()` en un objeto BitmapData, se genera una imagen filtrada al procesar el objeto BitmapData de origen y el objeto de filtro.

También puede aplicar efectos de filtro a imágenes y vídeo en tiempo de edición. Para más información, consulte la documentación de edición.

Si aplica un filtro a un clip de película o a un botón, la propiedad `cacheAsBitmap` del clip de película o del botón se establecerá como `true`. Si borra todos los filtros, se restaurará el valor original de `cacheAsBitmap`.

Una convolución de matriz se basa en una matriz *n* x *m*, que describe cómo el valor de un píxel dado en la imagen de entrada se combina los valores de los píxeles colindantes para producir un valor de píxel resultante. Cada píxel resultante se determina aplicando la matriz al píxel de origen correspondiente y a los píxeles colindantes.

Para una convolución de matriz 3 x 3 se emplea la fórmula siguiente para cada canal de color independiente:

```
dst (x, y) = ((src (x-1, y-1) * a0 + src(x, y-1) * a1....
src(x, y+1) * a7 + src (x+1,y+1) * a8) / divisor) + bias
```

Algunas especificaciones de filtro se ejecutan más rápidamente cuando se utiliza un procesador con SSE (Streaming SIMD Extensions).

- El filtro debe ser de tipo 3 x 3.
- Todos los términos del filtro deben ser enteros entre -127 y +127.
- El valor absoluto de la suma de todos los términos del filtro no debe ser mayor de 127.
- Si alguno de los términos del filtro es negativo, el divisor debe estar entre 2,00001 y 256.
- Si todos los términos del filtro son positivos, el divisor debe estar entre 1,1 y 256.
- El sesgo debe ser un entero.

Un filtro no se aplica si la imagen resultante supera los 2880 píxeles de anchura o altura. Por ejemplo, si amplía un clip de película grande al que se le ha aplicado un filtro, éste se desactiva si la imagen resultante alcanza el límite de 2880 píxeles.

Disponibilidad: ActionScript 1.0; Flash Player 8

Véase también

`applyFilter (método BitmapData.applyFilter)`, `filters (propiedad MovieClip.filters)`, `cacheAsBitmap (propiedad MovieClip.cacheAsBitmap)`

Resumen de propiedades

Modificadores	Propiedad	Descripción
	`alpha:Number`	Valor de transparencia alfa del color opcional.
	`bias:Number`	Sesgo que debe añadirse al resultado de la transformación de matriz.
	`clamp:Boolean`	Indica si la imagen debe fijarse.
	`color:Number`	Color hexadecimal para reemplazar los píxeles que quedan fuera de la imagen de origen.
	`divisor:Number`	Divisor que se utiliza durante la transformación de matriz.
	`matrix:Array`	Una matriz de valores utilizados para la transformación de matriz; devuelve una copia.
	`matrixX:Number`	La dimensión *x* de la matriz (el número de columnas de la matriz).
	`matrixY:Number`	La dimensión *y* de la matriz (el número de filas de la matriz).
	`preserveAlpha:Boolean`	Indica a qué se aplica la convolución.

Propiedades heredadas de la clase Object

`constructor (propiedad Object.constructor)`, `__proto__ (propiedad Object.__proto__)`, `prototype (propiedad Object.prototype)`, `__resolve (propiedad Object.__resolve)`

Resumen de constructores

Firma	Descripción
`ConvolutionFilter(matrixX:Number, matrixY:Number, matrix:Array, [divisor:Number], [bias:Number], [preserveAlpha:Boolean], [clamp:Boolean], [color:Number], [alpha:Number])`	Inicializa una instancia ConvolutionFilter con los parámetros especificados.

Resumen de métodos

Modificadores	Firma	Descripción
	`clone() : ConvolutionFilter`	Devuelve una copia de este objeto de filtro.

Métodos heredados de la clase BitmapFilter

`clone (método BitmapFilter.clone)`

Métodos heredados de la clase Object

`addProperty (método Object.addProperty), hasOwnProperty (método Object.hasOwnProperty), isPropertyEnumerable (método Object.isPropertyEnumerable), isPrototypeOf (método Object.isPrototypeOf), registerClass (método Object.registerClass), toString (método Object.toString), unwatch (método Object.unwatch), valueOf (método Object.valueOf), watch (método Object.watch)`

alpha (propiedad ConvolutionFilter.alpha)

`public alpha : Number`

Valor de transparencia alfa del color opcional. Los valores válidos van de 0 a 1,0. El valor predeterminado es 0. Por ejemplo, 0,25 establece un valor de transparencia del 25%. El valor predeterminado es 1,0.

Disponibilidad: ActionScript 1.0; Flash Player 8

Ejemplo

El ejemplo siguiente cambia la propiedad `alpha` de `filter` de su valor predeterminado 1 a 0,35.

```
import flash.filters.ConvolutionFilter;
import flash.display.BitmapData;
import flash.geom.Rectangle;
import flash.geom.Point;

var alpha:Number = .35;
var filter:ConvolutionFilter = new ConvolutionFilter(3, 3, [1, 1, 1, 1, 1,
  1, 1, 1, 1], 9, 0, true, false, 0x0000FF, alpha);

var myBitmapData:BitmapData = new BitmapData(100, 80, true, 0xCCFF0000);

var mc:MovieClip = this.createEmptyMovieClip("mc",
  this.getNextHighestDepth());
mc.attachBitmap(myBitmapData, this.getNextHighestDepth());
myBitmapData.noise(128, 0, 255, 1 | 2 | 4 | 8, false);

mc.onPress = function() {
  myBitmapData.applyFilter(myBitmapData, new Rectangle(0, 0, 98, 78), new
  Point(2, 2), filter);
}
```

bias (propiedad ConvolutionFilter.bias)

```
public bias : Number
```

Sesgo que debe añadirse al resultado de la transformación de matriz. El valor predeterminado es 0.

Disponibilidad: ActionScript 1.0; Flash Player 8

Ejemplo

El ejemplo siguiente cambia la propiedad `bias` de `filter` de su valor predeterminado 0 a 50.

```
import flash.filters.ConvolutionFilter;
import flash.display.BitmapData;

var bias:Number = 50;
var filter:ConvolutionFilter = new ConvolutionFilter(3, 3, [1, 1, 1, 1, 1,
  1, 1, 1, 1], 9, bias);

var myBitmapData:BitmapData = new BitmapData(100, 80, false, 0x00FF0000);

var mc:MovieClip = this.createEmptyMovieClip("mc",
  this.getNextHighestDepth());
mc.attachBitmap(myBitmapData, this.getNextHighestDepth());
```

```
myBitmapData.noise(128);

mc.onPress = function() {
  myBitmapData.applyFilter(myBitmapData, myBitmapData.rectangle, new
  Point(0, 0), filter);
}
```

clamp (propiedad ConvolutionFilter.clamp)

```
public clamp : Boolean
```

Indica si la imagen debe fijarse. Para los píxeles que quedan fuera de la imagen de origen, un valor de `true` indica que la imagen de entrada se amplía a lo largo de cada uno de sus bordes cuanto sea necesario duplicando los valores de color en cada borde de la imagen de entrada. Un valor de `false` indica que se utilizará otro color, según se haya especificado en las propiedades `color` y `alpha`. El valor predeterminado es `true`.

Disponibilidad: ActionScript 1.0; Flash Player 8

Ejemplo

El ejemplo siguiente cambia la propiedad `clamp` de `filter` de su valor predeterminado `true` a `false`.

```
import flash.filters.ConvolutionFilter;
import flash.display.BitmapData;
import flash.geom.Rectangle;
import flash.geom.Point;

var clamp:Boolean = false;
var filter:ConvolutionFilter = new ConvolutionFilter(3, 3, [1, 1, 1, 1, 1,
  1, 1, 1, 1], 9, 0, true, clamp, 0x00FF00, 1);

var myBitmapData:BitmapData = new BitmapData(100, 80, true, 0xCCFF0000);

var mc:MovieClip = this.createEmptyMovieClip("mc",
  this.getNextHighestDepth());
mc.attachBitmap(myBitmapData, this.getNextHighestDepth());
myBitmapData.noise(128, 0, 255, 1 | 2 | 4 | 8, false);

mc.onPress = function() {
  myBitmapData.applyFilter(myBitmapData, new Rectangle(0, 0, 98, 78), new
  Point(2, -2), filter);
}
```

clone (método ConvolutionFilter.clone)

```
public clone() : ConvolutionFilter
```

Devuelve una copia de este objeto de filtro.

Disponibilidad: ActionScript 1.0; Flash Player 8

Valor devuelto

`flash.filters.ConvolutionFilter` - Una instancia ConvolutionFilter nueva con las mismas propiedades que la original.

Ejemplo

El ejemplo siguiente crea tres objetos ConvolutionFilter y los compara: `filter_1` se crea utilizando el constructor `ConvolutionFilter`; `filter_2` se crea con el valor `filter_1` y `clonedFilter` se crea clonando `filter_1`. Observe que aunque `filter_2` da como resultado `filter_1`, `clonedFilter`, aunque contiene los mismos valores que `filter_1`, no es igual.

```
import flash.filters.ConvolutionFilter;

var filter_1:ConvolutionFilter = new ConvolutionFilter(3, 3, [1, 1, 1, 1, 1,
  1, 1, 1, 1], 9);
var filter_2:ConvolutionFilter = filter_1;
var clonedFilter:ConvolutionFilter = filter_1.clone();

trace(filter_1 == filter_2); // true
trace(filter_1 == clonedFilter); // false

for(var i in filter_1) {
  trace(">> " + i + ": " + filter_1[i]);
  // >> clone: [type Function]
  // >> alpha: 0
  // >> color: 0
  // >> clamp: true
  // >> preserveAlpha: true
  // >> bias: 0
  // >> divisor: 9
  // >> matrix: 1,1,1,1,1,1,1,1,1
  // >> matrixY: 3
  // >> matrixX: 3
}

for(var i in clonedFilter) {
  trace(">> " + i + ": " + clonedFilter[i]);
  // >> clone: [type Function]
  // >> alpha: 0
  // >> color: 0
  // >> clamp: true
```

```
  // >> preserveAlpha: true
  // >> bias: 0
  // >> divisor: 9
  // >> matrix: 1,1,1,1,1,1,1,1,1
  // >> matrixY: 3
  // >> matrixX: 3
}
```

Para mostrar más detalladamente las relaciones entre `filter_1`, `filter_2` y `clonedFilter`, el ejemplo siguiente modifica la propiedad `bias` de `filter_1`. Al modificar `bias`, el método `clone()` crea una nueva instancia basada en los valores de `filter_1`, en lugar de utilizarlos en la referencia.

```
import flash.filters.ConvolutionFilter;

var filter_1:ConvolutionFilter = new ConvolutionFilter(3, 3, [1, 1, 1, 1, 1,
  1, 1, 1, 1], 9);
var filter_2:ConvolutionFilter = filter_1;
var clonedFilter:ConvolutionFilter = filter_1.clone();
trace(filter_1.bias); // 0
trace(filter_2.bias); // 0
trace(clonedFilter.bias); // 0

filter_1.bias = 20;

trace(filter_1.bias); // 20
trace(filter_2.bias); // 20
trace(clonedFilter.bias); // 0
```

color (propiedad ConvolutionFilter.color)

```
public color : Number
```

Color hexadecimal para reemplazar los píxeles que quedan fuera de la imagen de origen. Es un valor RGB sin componente alfa. El valor predeterminado es 0.

Disponibilidad: ActionScript 1.0; Flash Player 8

Ejemplo

El ejemplo siguiente cambia la propiedad `color` de `filter` de su valor predeterminado 0 a 0xFF0000.

```
import flash.filters.ConvolutionFilter;
import flash.display.BitmapData;
import flash.geom.Rectangle;
import flash.geom.Point;

var color:Number = 0x0000FF;
```

```
var filter:ConvolutionFilter = new ConvolutionFilter(3, 3, [1, 1, 1, 1, 1,
  1, 1, 1, 1], 9, 0, true, false, color, 1);

var myBitmapData:BitmapData = new BitmapData(100, 80, true, 0xCCFF0000);

var mc:MovieClip = this.createEmptyMovieClip("mc",
  this.getNextHighestDepth());
mc.attachBitmap(myBitmapData, this.getNextHighestDepth());
myBitmapData.noise(128, 0, 255, 1 | 2 | 4 | 8, false);

var height:Number = 100;
var width:Number = 80;
mc.onPress = function() {
  height -= 2;
  width -= 2;
  myBitmapData.applyFilter(myBitmapData, new Rectangle(0, 0, height,
  width), new Point(2, 2), filter);
}
```

Constructor ConvolutionFilter

```
public ConvolutionFilter(matrixX:Number, matrixY:Number, matrix:Array,
  [divisor:Number], [bias:Number], [preserveAlpha:Boolean],
  [clamp:Boolean], [color:Number], [alpha:Number])
```

Inicializa una instancia ConvolutionFilter con los parámetros especificados.

Disponibilidad: ActionScript 1.0; Flash Player 8

Parámetros

matrixX:Number - La dimensión *x* de la matriz (el número de columnas de la matriz). El valor predeterminado es 0.

matrixY:Number - La dimensión *y* de la matriz (el número de filas de la matriz). El valor predeterminado es 0.

matrix:Array - Matriz de valores utilizados para la transformación de matriz; devuelve una copia. El número de elementos de la matriz debe ser igual a `matrixX*matrixY`.

divisor:Number [opcional] - Divisor que se utiliza durante la transformación de matriz. El valor predeterminado es 1. Un divisor que es la suma de todos los valores de matriz compensa la intensidad de color general del resultado. Se ignora un valor de 0 y en su lugar se utiliza el valor predeterminado.

bias:Number [opcional] - Sesgo que debe añadirse al resultado de la transformación de matriz. El valor predeterminado es 0.

preserveAlpha:Boolean [opcional] - Un valor de `false` indica que la convolución se aplica a todos los canales, incluido el canal alfa. Un valor de `true` indica que la convolución sólo se aplica a los canales de color. El valor predeterminado es `true`.

clamp:Boolean [opcional] - Para los píxeles que quedan fuera de la imagen de origen, un valor de `true` indica que la imagen de entrada se amplía a lo largo de cada uno de sus bordes cuanto sea necesario duplicando los valores de color en cada borde de la imagen de entrada. Un valor de `false` indica que se utilizará otro color, según se haya especificado en las propiedades `color` y `alpha`. El valor predeterminado es `true`.

color:Number [opcional] - Color hexadecimal para reemplazar los píxeles que quedan fuera de la imagen de origen.

alpha:Number [opcional] - Alfa del color opcional.

Ejemplo

El código siguiente crea un filtro de convolución 3 x 3 con un divisor de 9. El filtro haría que una imagen apareciera desenfocada:

```
var myArray:Array = [1, 1, 1, 1, 1, 1, 1, 1, 1];
var myFilter:ConvolutionFilter = new flash.filters.ConvolutionFilter (3,
3, myArray, 9);
```

El ejemplo siguiente crea un objeto ConvolutionFilter con los cuatro parámetros necesarios `matrixX`, `matrixY` `matrix` y `divisor`.

```
import flash.filters.ConvolutionFilter;
import flash.display.BitmapData;

var matrixX:Number = 3;
var matrixY:Number = 3;
var matrix:Array = [1, 1, 1, 1, 1, 1, 1, 1, 1];
var divisor:Number = 9;

var filter:ConvolutionFilter = new ConvolutionFilter(matrixX, matrixY,
  matrix, divisor);

var myBitmapData:BitmapData = new BitmapData(100, 80, false, 0x00FF0000);

var mc:MovieClip = this.createEmptyMovieClip("mc",
  this.getNextHighestDepth());
mc.attachBitmap(myBitmapData, this.getNextHighestDepth());
myBitmapData.noise(128);

mc.onPress = function() {
  myBitmapData.applyFilter(myBitmapData, myBitmapData.rectangle, new
  Point(0, 0), filter);
}
```

divisor (propiedad ConvolutionFilter.divisor)

```
public divisor : Number
```

Divisor que se utiliza durante la transformación de matriz. El valor predeterminado es 1. Un divisor que es la suma de todos los valores de matriz compensa la intensidad de color general del resultado. Se ignora un valor de 0 y en su lugar se utiliza el valor predeterminado.

Disponibilidad: ActionScript 1.0; Flash Player 8

Ejemplo

El ejemplo siguiente cambia la propiedad `divisor` de `filter` a 6.

```
import flash.filters.ConvolutionFilter;
import flash.display.BitmapData;

var filter:ConvolutionFilter = new ConvolutionFilter(3, 3, [1, 1, 1, 1, 1,
  1, 1, 1, 1], 9);

var myBitmapData:BitmapData = new BitmapData(100, 80, false, 0x00FF0000);

var mc:MovieClip = this.createEmptyMovieClip("mc",
  this.getNextHighestDepth());
mc.attachBitmap(myBitmapData, this.getNextHighestDepth());
myBitmapData.noise(128);

mc.onPress = function() {
  var newDivisor:Number = 6;
  filter.divisor = newDivisor;
    myBitmapData.applyFilter(myBitmapData, myBitmapData.rectangle, new
  Point(0, 0), filter);
}
```

matrix (propiedad ConvolutionFilter.matrix)

```
public matrix : Array
```

Una matriz de valores utilizados para la transformación de matriz; devuelve una copia. El número de elementos de la matriz debe ser igual a `matrixX*matrixY`.

La propiedad `matrix` no puede cambiarse directamente modificando los valores (por ejemplo, `myFilter.matrix[2] = 1;`). Es necesario obtener una matriz, realizar el cambio en la referencia y restablecer el valor mediante `filter.matrix = newMatrix;`, como se muestra en el ejemplo siguiente.

Disponibilidad: ActionScript 1.0; Flash Player 8

Ejemplo

El ejemplo siguiente cambia la propiedad `matrix` de `filter` de un valor que desenfoca un mapa de bits a uno que lo perfila.

```
import flash.filters.ConvolutionFilter;
import flash.display.BitmapData;

var filter:ConvolutionFilter = new ConvolutionFilter(3, 3, [1, 1, 1, 1, 1,
  1, 1, 1, 1], 9);

var myBitmapData:BitmapData = new BitmapData(100, 80, false, 0x00FF0000);

var mc:MovieClip = this.createEmptyMovieClip("mc",
  this.getNextHighestDepth());
mc.attachBitmap(myBitmapData, this.getNextHighestDepth());
myBitmapData.noise(128);

mc.onPress = function() {
  var newMatrix:Array = [0, -1, 0, -1, 8, -1, 0, -1, 0];
  filter.matrix = newMatrix;
  myBitmapData.applyFilter(myBitmapData, myBitmapData.rectangle, new
  Point(0, 0), filter);
}
```

matrixX (propiedad ConvolutionFilter.matrixX)

```
public matrixX : Number
```

La dimensión *x* de la matriz (el número de columnas de la matriz). El valor predeterminado es 0.

Disponibilidad: ActionScript 1.0; Flash Player 8

Ejemplo

El ejemplo siguiente muestra la propiedad `matrixX` de `filter`.

```
import flash.filters.ConvolutionFilter;

var filter:ConvolutionFilter = new ConvolutionFilter(2, 3, [1, 0, 0, 1, 0,
  0], 6);
trace(filter.matrixX); // 2
```

matrixY (propiedad ConvolutionFilter.matrixY)

```
public matrixY : Number
```

La dimensión *y* de la matriz (el número de filas de la matriz). El valor predeterminado es 0.

Disponibilidad: ActionScript 1.0; Flash Player 8

Ejemplo

El ejemplo siguiente muestra la propiedad `matrixY` de `filter`.

```
import flash.filters.ConvolutionFilter;

var filter:ConvolutionFilter = new ConvolutionFilter(2, 3, [1, 0, 0, 1, 0,
  0], 6);
trace(filter.matrixY); // 3
```

preserveAlpha (propiedad ConvolutionFilter.preserveAlpha)

```
public preserveAlpha : Boolean
```

Indica a qué se aplica la convolución. Un valor de `false` indica que la convolución se aplica a todos los canales, incluido el canal alfa. Un valor de `true` indica que la convolución sólo se aplica a los canales de color. El valor predeterminado es `true`.

Disponibilidad: ActionScript 1.0; Flash Player 8

Ejemplo

El ejemplo siguiente cambia la propiedad `preserveAlpha` de `filter` de su valor predeterminado `true` a `false`.

```
import flash.filters.ConvolutionFilter;
import flash.display.BitmapData;

var preserveAlpha:Boolean = false;
var filter:ConvolutionFilter = new ConvolutionFilter(3, 3, [1, 1, 1, 1, 1,
  1, 1, 1, 1], 9, 0, preserveAlpha);

var myBitmapData:BitmapData = new BitmapData(100, 80, true, 0xCCFF0000);

var mc:MovieClip = this.createEmptyMovieClip("mc",
  this.getNextHighestDepth());
mc.attachBitmap(myBitmapData, this.getNextHighestDepth());
myBitmapData.noise(128, 0, 255, 1 | 2 | 4 | 8, false);

mc.onPress = function() {
  myBitmapData.applyFilter(myBitmapData, myBitmapData.rectangle, new
  Point(0, 0), filter);
}
```

CustomActions

```
Object
  |
  +-CustomActions
```

```
public class CustomActions
extends Object
```

Los métodos de la clase CustomActions permiten a un archivo SWF que se está reproduciendo en la herramienta de edición de Flash administrar las acciones personalizadas que se hayan registrado en la herramienta de edición. Un archivo SWF puede instalar y desinstalar acciones personalizadas, recuperar la definición XML de una acción personalizada y recuperar la lista de acciones personalizadas registradas.

Puede utilizar estos métodos para crear archivos SWF que sean extensiones de la herramienta de edición de Flash. Una extensión de este tipo podría, por ejemplo, utilizar el protocolo de aplicación de Flash para desplazarse por un repositorio UDDI y descargar servicios Web a la caja de herramientas Acciones.

Disponibilidad: ActionScript 1.0; Flash Player 6

Resumen de propiedades

Propiedades heredadas de la clase Object

`constructor (propiedad Object.constructor), __proto__ (propiedad Object.__proto__), prototype (propiedad Object.prototype), __resolve (propiedad Object.__resolve)`

Resumen de métodos

Modificadores	Firma	Descripción
`static`	`get(name:String) : String`	Lee el contenido del archivo de definición XML de la acción personalizada denominado `name`.
`static`	`install(name:String, data:String) : Boolean`	Instala el nuevo archivo de definición XML de la acción personalizada indicado por el parámetro `name`.
`static`	`list() : Array`	Devuelve un objeto Array que contiene los nombres de todas las acciones personalizadas registradas en la herramienta de edición de Flash.
`static`	`uninstall(name:String) : Boolean`	Elimina el archivo de definición XML de acciones personalizadas denominado `name`.

Métodos heredados de la clase Object

addProperty (método Object.addProperty), hasOwnProperty (método Object.hasOwnProperty), isPropertyEnumerable (método Object.isPropertyEnumerable), isPrototypeOf (método Object.isPrototypeOf), registerClass (método Object.registerClass), toString (método Object.toString), unwatch (método Object.unwatch), valueOf (método Object.valueOf), watch (método Object.watch)

get (método CustomActions.get)

```
public static get(name:String) : Cadena
```

Lee el contenido del archivo de definición XML de la acción personalizada denominado `name`.

El nombre del archivo de definición debe ser un nombre de archivo simple, sin la extensión de archivo .xml y sin ningún separador de directorio (':', '/' o '\').

Si no se encuentra el archivo de definición especificado por `name`, se devuelve el valor `undefined` (no definido). Si se localiza la definición XML de acción personalizada especificada por el parámetro `name`, se lee íntegramente y se devuelve en forma de cadena.

Disponibilidad: ActionScript 1.0; Flash Player 6

Parámetros

`name``:String` - Nombre de la definición de acción personalizada que se va a recuperar.

Valor devuelto

`String` - Si se localiza la definición XML de acción personalizada, devuelve una cadena; en caso contrario, devuelve `undefined`.

Ejemplo

El ejemplo siguiente muestra las acciones personalizadas en una instancia ComboBox y obtiene la acción personalizada cuando se hace clic en una instancia Button. Arrastre una instancia de ComboBox, Button y TextArea sobre el escenario. Asigne al ComboBox el nombre de instancia `customActionName_cb`, al TextArea el nombre de instancia `customActionXml_ta` y al Button el nombre de instancia `view_button`. Introduzca el código ActionScript siguiente al fotograma 1 de la línea de tiempo:

```
import mx.controls.*;

var customActionName_cb:ComboBox;
var customActionXml_ta:TextArea;
var view_button:Button;
```

```
customActionName_cb.dataProvider = CustomActions.list();

customActionXml_ta.editable = false;

var viewListener:Object = new Object();
viewListener.click = function(evt:Object) {
  var caName:String = String(customActionName_cb.selectedItem);
  customActionXml_ta.text = CustomActions.get(caName);
};
view_button.addEventListener("click", viewListener);
```

install (método CustomActions.install)

```
public static install(name:String, data:String) : Booleano
```

Instala el nuevo archivo de definición XML de la acción personalizada indicado por el parámetro `name`. El contenido del archivo se especifica mediante la cadena *`customXML`* .

El nombre del archivo de definición debe ser un nombre de archivo simple, sin la extensión de archivo .xml y sin ningún separador de directorio (':', '/' o '\').

Si ya hay un archivo de acciones personalizadas con el nombre `name`, se sobrescribirá.

Si el directorio Configuration/ActionsPanel/CustomActions no existe al invocar este método, se creará dicho directorio.

Disponibilidad: ActionScript 1.0; Flash Player 6

Parámetros

`name``:String` - Nombre de la definición de acción personalizada que se va a instalar.

`data``:String` - Texto de la definición XML que se va a instalar.

Valor devuelto

`Boolean` - El valor booleano `false` si se produce un error durante la instalación; en caso contrario, se devuelve el valor `true` para indicar que la acción personalizada se ha instalado correctamente.

Ejemplo

El ejemplo siguiente instala información en el panel Acciones desde un archivo XML. Abra un editor de texto y guarde un documento nuevo llamado dogclass.xml. Introduzca el código siguiente:

```
<?xml version="1.0"?>
<customactions>
  <actionspanel>
```

```
  <folder version="7" id="DogClass" index="true" name="Dog" tiptext="Dog
  Class">
    <string version="7" id="getFleas" name="getFleas" tiptext="gets number
  of fleas" text=".getFleas(% fleas %)" />
  </folder>
  </actionspanel>
  <colorsyntax>
    <identifier text=".getFleas" />
  </colorsyntax>
  <codehints>
    <typeinfo pattern=" _dog" object="Dog"/>
  </codehints>
</customactions>
```

A continuación, abra un archivo FLA nuevo en el mismo directorio y seleccione el fotograma 1 de la línea de tiempo. Introduzca los códigos siguientes en el panel Acciones:

```
var my_xml:XML = new XML();
my_xml.ignoreWhite = true;
my_xml.onLoad = function(success:Boolean) {
  trace(success);
  CustomActions.install("dogclass", this.firstChild);
  trace(CustomActions.list());
};
my_xml.load("dogclass.xml");
```

Seleccione Control > Probar película, y si el XML se carga correctamente, aparecerá `true`, y una matriz con los nombres de todas las acciones personalizadas que se han registrado con la herramienta de edición de Flash en el panel Salida. Cierre el archivo SWF y abra el panel Acciones. Aparecerá un elemento nuevo llamado Dog en la caja de herramientas Acciones y dentro de esa carpeta aparecerá getFleas.

list (método CustomActions.list)

```
public static list() : Array
```

Devuelve un objeto Array que contiene los nombres de todas las acciones personalizadas registradas en la herramienta de edición de Flash. Los elementos de la matriz son nombres simples, sin la extensión de archivo .xml y sin ningún separador de directorio (por ejemplo, ":", "/" o "\"). Si no hay acciones personalizadas registradas, `list()` devuelve una matriz de longitud cero. Si se produce un error, `list()` devuelve el valor `undefined`.

Disponibilidad: ActionScript 1.0; Flash Player 6

Valor devuelto

`Array` - Una matriz.

Ejemplo

El ejemplo siguiente muestra las acciones personalizadas en una instancia ComboBox y obtiene la acción personalizada cuando se hace clic en una instancia Button. Arrastre una instancia de ComboBox, Button y TextArea sobre el escenario. Asigne al ComboBox el nombre de instancia `customActionName_cb`, al TextArea el nombre de instancia `customActionXml_ta` y al Button el nombre de instancia `view_button`. Introduzca el código ActionScript siguiente al fotograma 1 de la línea de tiempo:

```
import mx.controls.*;

var customActionName_cb:ComboBox;
var customActionXml_ta:TextArea;
var view_button:Button;

customActionName_cb.dataProvider = CustomActions.list();

customActionXml_ta.editable = false;

var viewListener:Object = new Object();
viewListener.click = function(evt:Object) {
  var caName:String = String(customActionName_cb.selectedItem);
  customActionXml_ta.text = CustomActions.get(caName);
};
view_button.addEventListener("click", viewListener);
```

uninstall (método CustomActions.uninstall)

```
public static uninstall(name:String) : Booleano
```

Elimina el archivo de definición XML de acciones personalizadas denominado `name`.

El nombre del archivo de definición debe ser un nombre de archivo simple, sin la extensión de archivo .xml y sin ningún separador de directorio (':', '/' o '\').

Disponibilidad: ActionScript 1.0; Flash Player 6

Parámetros

`name``:String` - Nombre de la definición de acción personalizada que se va a desinstalar.

Valor devuelto

`Boolean` - El valor booleano `false` si no se encuentran acciones personalizadas con el nombre `name`. Si se han eliminado correctamente las acciones personalizadas, se devolverá el valor `true`.

Ejemplo

El ejemplo siguiente instala una acción personalizada nueva y muestra una matriz con los nombres de todas las acciones personalizadas que se han registrado con la herramienta de edición de Flash en el panel Salida. Cuando se hace clic en uninstall_btn se desinstala la acción personalizada. Se muestra una matriz que contiene los nombres de las acciones personalizadas y, a continuación, debe quitarse dogclass de la matriz. Cree un botón llamado `uninstall_btn` e introduzca el código ActionScript siguiente en el fotograma 1 de la línea de tiempo:

```
var my_xml:XML = new XML();
my_xml.ignoreWhite = true;
my_xml.onLoad = function(success:Boolean) {
  trace(success);
  CustomActions.install("dogclass", this.firstChild);
  trace(CustomActions.list());
};
my_xml.load("dogclass.xml");

uninstall_btn.onRelease = function() {
  CustomActions.uninstall("dogclass");
  trace(CustomActions.list());
};
```

Para obtener información sobre la creación de dogclass.xml, consulte `CustomActions.install()`.

Véase también

`install (método CustomActions.install)`

Date

```
Object
  |
  +-Date
```

`public class` ***Date***
`extends Object`

La clase Date permite recuperar los valores de fecha y hora relativos a la hora universal (hora de Greenwich, ahora conocida como hora universal o UTC) o al sistema operativo en el que se ejecuta Flash Player. Los métodos de la clase Date no son estáticos, pero afectan solamente al objeto Date concreto especificado al llamar al método. El método `Date.UTC()` es una excepción; es un método estático.

La clase Date controla el horario de verano de forma distinta en función del sistema operativo y la versión de Flash Player. Flash Player 6 y posteriores controlan el horario de verano en los siguientes sistemas operativos de estas formas:

- Windows: el objeto Date ajusta automáticamente su salida para el horario de verano. El objeto Date detecta si se emplea el horario de verano en la ubicación actual y, si es así, detecta la fecha y la hora en la que se produce el cambio del horario estándar al horario de verano. No obstante, las fechas de transición en vigor actualmente se aplican a fechas pasadas y futuras, por lo que el horario de verano podría calcularse erróneamente en el caso de fechas pasadas si en la ubicación en cuestión las fechas de transición eran diferentes.
- Mac OS X: el objeto Date ajusta automáticamente su salida para el horario de verano. La base de datos de información de zona horaria de Mac OS X se utiliza para determinar si debe aplicarse el cambio de hora a alguna fecha u hora actual o pasada.
- Mac OS 9: el sistema operativo sólo proporciona información suficiente para determinar si debe aplicarse el cambio de hora a la fecha y la hora actuales. Por consiguiente, el objeto de fecha da por hecho que el cambio de hora actual afecta tanto a fechas y horas futuras como pasadas.

Flash Player 5 controla el cambio de hora en los siguientes sistemas operativos como se indica a continuación:

- Windows: se aplican siempre las normas estadounidenses de cambio de hora, lo que lleva a transiciones incorrectas en Europa y otras zonas que emplean el cambio de hora pero con transiciones diferentes a las de EE. UU. Flash detecta correctamente si se utiliza el cambio de hora en la ubicación actual.

Para llamar a los métodos de la clase Date, primero debe crear un objeto Date utilizando el constructor para la clase Date, descrito posteriormente en esta sección.

Disponibilidad: ActionScript 1.0; Flash Player 5

Resumen de propiedades

Propiedades heredadas de la clase Object

```
constructor (propiedad Object.constructor), __proto__ (propiedad
Object.__proto__), prototype (propiedad Object.prototype), __resolve
(propiedad Object.__resolve)
```

Resumen de constructores

Firma	Descripción
`Date([yearOrTimevalue:Number], [month:Number], [date:Number], [hour:Number], [minute:Number], [second:Number], [millisecond:Number])`	Construye un objeto new Date que contiene la fecha y la hora especificadas.

Resumen de métodos

Modificadores	Firma	Descripción
	`getDate() : Number`	Devuelve el día del mes (un entero del 1 al 31) del objeto Date especificado de acuerdo con la hora local.
	`getDay() : Number`	Devuelve el día de la semana (0 para domingo, 1 para lunes, etc.) del objeto Date especificado de acuerdo con la hora local.
	`getFullYear() : Number`	Devuelve el año completo (un número de cuatro dígitos, como, por ejemplo, 2000) del objeto Date especificado de acuerdo con la hora local.
	`getHours() : Number`	Devuelve la hora (un entero del 0 al 23) del objeto Date especificado de acuerdo con la hora local.
	`getMilliseconds() : Number`	Devuelve los milisegundos (un entero del 0 al 999) del objeto Date especificado de acuerdo con la hora local.
	`getMinutes() : Number`	Devuelve los minutos (un entero del 0 al 59) del objeto Date especificado de acuerdo con la hora local.
	`getMonth() : Number`	Devuelve el mes (0 para enero, 1 para febrero, etc.) del objeto Date especificado de acuerdo con la hora local.
	`getSeconds() : Number`	Devuelve los segundos (un entero del 0 al 59) del objeto Date especificado de acuerdo con la hora local.
	`getTime() : Number`	Devuelve el número de milisegundos desde la media noche del 1 de enero de 1970, hora universal, para el objeto Date especificado.
	`getTimezoneOffset() : Number`	Devuelve la diferencia, en minutos, entre la hora local del equipo y la hora universal.

Modificadores	Firma	Descripción
	`getUTCDate() : Number`	Devuelve el día del mes (un entero del 1 al 31) del objeto Date especificado de acuerdo con la hora universal.
	`getUTCDay() : Number`	Devuelve el día de la semana (0 para domingo, 1 para lunes, etc.) del objeto Date especificado de acuerdo con la hora universal.
	`getUTCFullYear() : Number`	Devuelve los cuatro dígitos del año del objeto Date especificado de acuerdo con la hora universal.
	`getUTCHours() : Number`	Devuelve la hora (un entero del 0 al 23) del objeto Date especificado de acuerdo con la hora universal.
	`getUTCMilliseconds() : Number`	Devuelve los milisegundos (un entero del 0 al 999) del objeto Date especificado de acuerdo con la hora universal.
	`getUTCMinutes() : Number`	Devuelve los minutos (un entero del 0 al 59) del objeto Date especificado de acuerdo con la hora universal.
	`getUTCMonth() : Number`	Devuelve el mes (de 0 [enero] a 11 [diciembre]) del objeto Date especificado de acuerdo con la hora universal.
	`getUTCSeconds() : Number`	Devuelve los segundos (un entero del 0 al 59) del objeto Date especificado de acuerdo con la hora universal.
	`getUTCYear() : Number`	Devuelve el año de este objeto `Date` según la hora universal (UTC).
	`getYear() : Number`	Devuelve el año del objeto Date especificado de acuerdo con la hora local.
	`setDate(date:Number) : Number`	Establece el día del mes para el objeto Date especificado de acuerdo con la hora local y devuelve la nueva hora en milisegundos.
	`setFullYear(year:Number, [month:Number], [date:Number]) : Number`	Establece el año del objeto Date especificado de acuerdo con la hora local y devuelve la nueva hora en milisegundos.
	`setHours(hour:Number) : Number`	Establece las horas del objeto Date especificado de acuerdo con la hora local y devuelve la nueva hora en milisegundos.

Modificadores	Firma	Descripción
	`setMilliseconds(millisecond:Number) : Number`	Establece los milisegundos del objeto Date especificado de acuerdo con la hora local y devuelve la nueva hora en milisegundos.
	`setMinutes(minute:Number) : Number`	Establece los minutos del objeto Date especificado de acuerdo con la hora local y devuelve la nueva hora en milisegundos.
	`setMonth(month:Number, [date:Number]) : Number`	Establece el mes del objeto Date especificado en la hora local y devuelve la nueva hora en milisegundos.
	`setSeconds(second:Number) : Number`	Establece los segundos del objeto Date especificado en la hora local y devuelve la nueva hora en milisegundos.
	`setTime(millisecond:Number) : Number`	Establece la fecha del objeto Date especificado en milisegundos desde la media noche del 1 de enero de 1970 y devuelve la nueva hora en milisegundos.
	`setUTCDate(date:Number) : Number`	Establece la fecha del objeto Date especificado en la hora universal y devuelve la nueva hora en milisegundos.
	`setUTCFullYear(year:Number, [month:Number], [date:Number]) : Number`	Establece el año del objeto Date especificado (*`my_date`*) en la hora universal y devuelve la nueva hora en milisegundos.
	`setUTCHours(hour:Number, [minute:Number], [second:Number], [millisecond:Number]) : Number`	Establece la hora del objeto Date especificado en la hora universal y devuelve la nueva hora en milisegundos.
	`setUTCMilliseconds(millisecond:Number) : Number`	Establece los milisegundos del objeto Date especificado en la hora universal y devuelve la nueva hora en milisegundos.
	`setUTCMinutes(minute:Number, [second:Number], [millisecond:Number]) : Number`	Establece el minuto del objeto Date especificado en la hora universal y devuelve la nueva hora en milisegundos.

Modificadores	Firma	Descripción
	`setUTCMonth(month:Number, [date:Number]) : Number`	Establece el mes y, opcionalmente, el día del objeto Date especificado en la hora universal y devuelve la nueva hora en milisegundos.
	`setUTCSeconds(second:Number, [millisecond:Number]) : Number`	Establece los segundos del objeto Date especificado en la hora universal y devuelve la nueva hora en milisegundos.
	`setYear(year:Number) : Number`	Establece el año del objeto Date especificado en la hora local y devuelve la nueva hora en milisegundos.
	`toString() : String`	Devuelve un valor de cadena para el objeto de fecha especificado en un formato legible.
`static`	`UTC(year:Number, month:Number, [date:Number], [hour:Number], [minute:Number], [second:Number], [millisecond:Number]) : Number`	Devuelve el número de milisegundos entre la media noche del 1 de enero de 1970, hora universal, y la hora especificada en los parámetros.
	`valueOf() : Number`	Devuelve el número de milisegundos desde la media noche del 1 de enero de 1970, hora universal, para este objeto `Date`.

Métodos heredados de la clase Object

`addProperty (método Object.addProperty), hasOwnProperty (método Object.hasOwnProperty), isPropertyEnumerable (método Object.isPropertyEnumerable), isPrototypeOf (método Object.isPrototypeOf), registerClass (método Object.registerClass), toString (método Object.toString), unwatch (método Object.unwatch), valueOf (método Object.valueOf), watch (método Object.watch)`

Constructor Date

```
public Date([yearOrTimevalue:Number], [month:Number], [date:Number],
  [hour:Number], [minute:Number], [second:Number], [millisecond:Number])
```

Construye un objeto new Date que contiene la fecha y la hora especificadas.

El constructor `Date()` toma hasta siete parámetros (year, month, ..., millisecond) para especificar una fecha y una hora en milisegundos. También es posible pasar un valor único al constructor `Date()` que indique un valor de hora basado en el número de milisegundos desde el 1 de enero de 1970 a las 0:00:000 GMT. También es posible no especificar parámetros y asignar al objeto de fecha `Date()` la fecha y la hora actuales.

Por ejemplo, este código muestra varias formas distintas de crear un objeto Date:

```
var d1:Date = new Date();
var d3:Date = new Date(2000, 0, 1);
var d4:Date = new Date(65, 2, 6, 9, 30, 15, 0);
var d5:Date = new Date(-14159025000);
```

En la primera línea de código, se establece un objeto Date con la hora en que se ejecutó la sentencia de asignación.

En la segunda línea se crea un objeto Date con los parámetros year, month y date que se le han pasado, lo que da como resultado la hora 0:00:00 GMT del 1 de enero de 2000.

En la tercera línea, se crea un objeto Date al que se han pasado los parámetros year, month y date, lo que da como resultado la hora 09:30:15 GMT (+ 0 milisegundos) 6 de marzo, 1965. El parámetro year se especifica como un entero de dos dígitos, por lo que se interpreta como 1965.

En la cuarta línea sólo se pasa un parámetro, que es un valor de hora que representa el número de milisegundos antes o después de las 0:00:00 GMT del 1 de enero de 1970; dado que el valor es negativo, representa una hora *anterior a* las 0:00:00 GMT del 1 de enero de 1970 y, en este caso, la hora es 02:56:15 GMT del 21 de julio de 1969.

Disponibilidad: ActionScript 1.0; Flash Player 5

Parámetros

yearOrTimevalue`:Number` [opcional] - Si se especifican otros parámetros, este número representa un año (por ejemplo, 1965); en caso contrario, representa un valor de hora. Si el número representa un año, un valor de 0 a 99 indica de 1900 a 1999; en caso contrario deben especificarse los cuatro dígitos del año. Si el número representa un valor de hora (no se especifican otros parámetros), es el número de milisegundos antes o después de las 0:00:00 GMT del 1 de enero de 1970; un valor negativo representa una hora *anterior a* las 0:00:00 GMT del 1 de enero de 1970, y un valor positivo representa una hora posterior.

month`:Number` [opcional] - Un entero de 0 (enero) a 11 (diciembre).

date:Number [opcional] - Entero de 1 a 31.

hour:Number [opcional] - Entero de 0 (medianoche) a 23 (11 de la noche).

minute:Number [opcional] - Entero de 0 a 59.

second:Number [opcional] - Entero de 0 a 59.

millisecond:Number [opcional] - Entero de 0 a 999 milisegundos.

Ejemplo

El ejemplo siguiente recupera la fecha y la hora actuales:

```
var now_date:Date = new Date();
```

El ejemplo siguiente crea un objeto new Date para el cumpleaños de Mary, el 12 de agosto de 1974 (dado que el parámetro month se basa en ceros, el ejemplo utiliza 7 para el mes, no 8):

```
var maryBirthday:Date = new Date (74, 7, 12);
```

El ejemplo siguiente crea un objeto new Date y concatena los valores devueltos de Date.getMonth(), Date.getDate() y Date.getFullYear():

```
var today_date:Date = new Date();
var date_str:String = ((today_date.getMonth()+1)+"/
  "+today_date.getDate()+"/"+today_date.getFullYear());
trace(date_str); // displays current date in United States date format
```

Véase también

getMonth (método Date.getMonth), getDate (método Date.getDate), getFullYear (método Date.getFullYear)

getDate (método Date.getDate)

```
public getDate() : Number
```

Devuelve el día del mes (un entero del 1 al 31) del objeto Date especificado de acuerdo con la hora local. La hora local la determina el sistema operativo en el que se ejecuta Flash Player.

Disponibilidad: ActionScript 1.0; Flash Player 5

Valor devuelto

Number - Un entero.

Ejemplo

El ejemplo siguiente crea un objeto new Date y concatena los valores devueltos de Date.getMonth(), Date.getDate() y Date.getFullYear():

```
var today_date:Date = new Date();
```

```
var date_str:String = (today_date.getDate()+"/
  "+(today_date.getMonth()+1)+"/"+today_date.getFullYear());
trace(date_str); // displays current date in United States date format
```

Véase también

getMonth (método Date.getMonth), getFullYear (método Date.getFullYear)

getDay (método Date.getDay)

```
public getDay() : Number
```

Devuelve el día de la semana (0 para domingo, 1 para lunes, etc.) del objeto Date especificado de acuerdo con la hora local. La hora local la determina el sistema operativo en el que se ejecuta Flash Player.

Disponibilidad: ActionScript 1.0; Flash Player 5

Valor devuelto

`Number` - Un entero que representa el día de la semana.

Ejemplo

El ejemplo siguiente crea un objeto new Date y utiliza `getDay()` para determinar el día actual de la semana:

```
var dayOfWeek_array:Array = new Array("Sunday", "Monday", "Tuesday",
  "Wednesday", "Thursday", "Friday", "Saturday");
var today_date:Date = new Date();
var day_str:String = dayOfWeek_array[today_date.getDay()];
trace("Today is "+day_str);
```

getFullYear (método Date.getFullYear)

```
public getFullYear() : Number
```

Devuelve el año completo (un número de cuatro dígitos, como, por ejemplo, 2000) del objeto Date especificado de acuerdo con la hora local. La hora local la determina el sistema operativo en el que se ejecuta Flash Player.

Disponibilidad: ActionScript 1.0; Flash Player 5

Valor devuelto

`Number` - Entero que representa el año.

Ejemplo

El ejemplo siguiente utiliza el constructor para crear un objeto Date. La sentencia trace muestra el valor devuelto por el método `getFullYear()`.

```
var my_date:Date = new Date();
trace(my_date.getYear()); // displays 104
trace(my_date.getFullYear()); // displays current year
```

getHours (método Date.getHours)

```
public getHours() : Number
```

Devuelve la hora (un entero del 0 al 23) del objeto Date especificado de acuerdo con la hora local. La hora local la determina el sistema operativo en el que se ejecuta Flash Player.

Disponibilidad: ActionScript 1.0; Flash Player 5

Valor devuelto

`Number` - Un entero.

Ejemplo

El ejemplo siguiente utiliza el constructor para crear un objeto Date basado en la hora actual y utiliza el método `getHours()` para mostrar valores de hora de ese objeto:

```
var my_date:Date = new Date();
trace(my_date.getHours());

var my_date:Date = new Date();
var hourObj:Object = getHoursAmPm(my_date.getHours());
trace(hourObj.hours);
trace(hourObj.ampm);

function getHoursAmPm(hour24:Number):Object {
  var returnObj:Object = new Object();
  returnObj.ampm = (hour24<12) ? "AM" : "PM";
  var hour12:Number = hour24%12;
  if (hour12 == 0) {
  hour12 = 12;
  }
  returnObj.hours = hour12;
  return returnObj;
}
```

getMilliseconds (método Date.getMilliseconds)

```
public getMilliseconds() : Number
```

Devuelve los milisegundos (un entero del 0 al 999) del objeto Date especificado de acuerdo con la hora local. La hora local la determina el sistema operativo en el que se ejecuta Flash Player.

Disponibilidad: ActionScript 1.0; Flash Player 5

Valor devuelto

`Number` - Un entero.

Ejemplo

El ejemplo siguiente utiliza el constructor para crear un objeto Date basado en la hora actual y utiliza el método `getMilliseconds()` para devolver el valor en milisegundos de ese objeto:

```
var my_date:Date = new Date();
trace(my_date.getMilliseconds());
```

getMinutes (método Date.getMinutes)

```
public getMinutes() : Number
```

Devuelve los minutos (un entero del 0 al 59) del objeto Date especificado de acuerdo con la hora local. La hora local la determina el sistema operativo en el que se ejecuta Flash Player.

Disponibilidad: ActionScript 1.0; Flash Player 5

Valor devuelto

`Number` - Un entero.

Ejemplo

El ejemplo siguiente utiliza el constructor para crear un objeto Date basado en la hora actual y utiliza el método `getMinutes()` para devolver el valor en minutos de ese objeto:

```
var my_date:Date = new Date();
trace(my_date.getMinutes());
```

getMonth (método Date.getMonth)

```
public getMonth() : Number
```

Devuelve el mes (0 para enero, 1 para febrero, etc.) del objeto Date especificado de acuerdo con la hora local. La hora local la determina el sistema operativo en el que se ejecuta Flash Player.

Disponibilidad: ActionScript 1.0; Flash Player 5

Valor devuelto

`Number` - Un entero.

Ejemplo

El ejemplo siguiente utiliza el constructor para crear un objeto Date basado en la hora actual y utiliza el método `getMonth()` para devolver el valor del mes de ese objeto:

```
var my_date:Date = new Date();
trace(my_date.getMonth());
```

El ejemplo siguiente utiliza el constructor para crear un objeto Date basado en la hora actual y utiliza el método `getMonth()` para mostrar el mes actual como un valor numérico y mostrar el nombre del mes.

```
var my_date:Date = new Date();
trace(my_date.getMonth());
trace(getMonthAsString(my_date.getMonth()));
function getMonthAsString(month:Number):String {
  var monthNames_array:Array = new Array("January", "February", "March",
  "April", "May", "June", "July", "August", "September", "October",
  "November", "December");
  return monthNames_array[month];
}
```

getSeconds (método Date.getSeconds)

```
public getSeconds() : Number
```

Devuelve los segundos (un entero del 0 al 59) del objeto Date especificado de acuerdo con la hora local. La hora local la determina el sistema operativo en el que se ejecuta Flash Player.

Disponibilidad: ActionScript 1.0; Flash Player 5

Valor devuelto

`Number` - Un entero.

Ejemplo

El ejemplo siguiente utiliza el constructor para crear un objeto Date basado en la hora actual y utiliza el método `getSeconds()` para devolver el valor en segundos de ese objeto:

```
var my_date:Date = new Date();
trace(my_date.getSeconds());
```

getTime (método Date.getTime)

```
public getTime() : Number
```

Devuelve el número de milisegundos desde la media noche del 1 de enero de 1970, hora universal, para el objeto Date especificado. Utilice este método para representar un instante concreto en el tiempo al comparar dos o más objetos Date.

Disponibilidad: ActionScript 1.0; Flash Player 5

Valor devuelto

`Number` - Un entero.

Ejemplo

El ejemplo siguiente utiliza el constructor para crear un objeto Date basado en la hora actual y utiliza el método `getTime()` para devolver el número de milisegundos desde la medianoche del 1 de enero de 1970:

```
var my_date:Date = new Date();
trace(my_date.getTime());
```

getTimezoneOffset (método Date.getTimezoneOffset)

```
public getTimezoneOffset() : Number
```

Devuelve la diferencia, en minutos, entre la hora local del equipo y la hora universal.

Disponibilidad: ActionScript 1.0; Flash Player 5

Valor devuelto

`Number` - Un entero.

Ejemplo

El ejemplo siguiente devuelve la diferencia entre la hora de verano local de San Francisco y la hora universal. La hora de verano sólo se tiene en cuenta en el resultado si la fecha definida en el objeto Date se encuentra dentro del horario de verano. El resultado de este ejemplo es 420 minutos y aparece en el panel Salida (7 horas * 60 minutos/hora = 420 minutos). Este ejemplo utiliza el horario de verano de la costa del Pacífico de EE.UU. (PDT, GMT-0700). El resultado varía según la ubicación y el momento del año.

```
var my_date:Date = new Date();
trace(my_date.getTimezoneOffset());
```

getUTCDate (método Date.getUTCDate)

```
public getUTCDate() : Number
```

Devuelve el día del mes (un entero del 1 al 31) del objeto Date especificado de acuerdo con la hora universal.

Disponibilidad: ActionScript 1.0; Flash Player 5

Valor devuelto

`Number` - Un entero.

Ejemplo

El ejemplo siguiente crea un nuevo objeto Date y emplea `Date.getUTCDate()` y `Date.getDate()`. El valor que devuelve `Date.getUTCDate()` puede ser distinto del valor que devuelve `Date.getDate()`, , según la relación existente entre la zona horaria local y la hora universal.

```
var my_date:Date = new Date(2004,8,25);
trace(my_date.getUTCDate()); // output: 25
```

Véase también

`getDate (método Date.getDate)`

getUTCDay (método Date.getUTCDay)

```
public getUTCDay() : Number
```

Devuelve el día de la semana (0 para domingo, 1 para lunes, etc.) del objeto Date especificado de acuerdo con la hora universal.

Disponibilidad: ActionScript 1.0; Flash Player 5

Valor devuelto

`Number` - Un entero.

Ejemplo

El ejemplo siguiente crea un nuevo objeto Date y emplea `Date.getUTCDay()` y `Date.getDay()`. El valor que devuelve `Date.getUTCDay()` puede ser distinto del valor que devuelve `Date.getDay()`, , según la relación existente entre la zona horaria local y la hora universal.

```
var today_date:Date = new Date();
trace(today_date.getDay()); // output will be based on local timezone
trace(today_date.getUTCDay()); // output will equal getDay() plus or minus
  one
```

Véase también

`getDay (método Date.getDay)`

getUTCFullYear (método Date.getUTCFullYear)

```
public getUTCFullYear() : Number
```

Devuelve los cuatro dígitos del año del objeto Date especificado de acuerdo con la hora universal.

Disponibilidad: ActionScript 1.0; Flash Player 5

Valor devuelto

`Number` - Un entero.

Ejemplo

El ejemplo siguiente crea un nuevo objeto Date y emplea `Date.getUTCFullYear()` y `Date.getFullYear()`. El valor que devuelve `Date.getUTCFullYear()` puede ser distinto del valor que devuelve `Date.getFullYear()` si la fecha del día es 31 de diciembre o 1 de enero, según la relación existente entre la zona horaria local y la hora universal.

```
var today_date:Date = new Date();
trace(today_date.getFullYear()); // display based on local timezone
trace(today_date.getUTCFullYear()); // displays getYear() plus or minus 1
```

Véase también

`getFullYear (método Date.getFullYear)`

getUTCHours (método Date.getUTCHours)

```
public getUTCHours() : Number
```

Devuelve la hora (un entero del 0 al 23) del objeto Date especificado de acuerdo con la hora universal.

Disponibilidad: ActionScript 1.0; Flash Player 5

Valor devuelto

`Number` - Un entero.

Ejemplo

El ejemplo siguiente crea un nuevo objeto Date y emplea `Date.getUTCHours()` y `Date.getHours()`. El valor que devuelve `Date.getUTCHours()` puede ser distinto del valor que devuelve `Date.getHours()`,, según la relación existente entre la zona horaria local y la hora universal.

```
var today_date:Date = new Date();
trace(today_date.getHours()); // display based on local timezone
trace(today_date.getUTCHours()); // display equals getHours() plus or minus
  12
```

Véase también

`getHours (método Date.getHours)`

getUTCMilliseconds (método Date.getUTCMilliseconds)

```
public getUTCMilliseconds() : Number
```

Devuelve los milisegundos (un entero del 0 al 999) del objeto Date especificado de acuerdo con la hora universal.

Disponibilidad: ActionScript 1.0; Flash Player 5

Valor devuelto

`Number` - Un entero.

Ejemplo

El ejemplo siguiente crea un objeto new Date y utiliza `getUTCMilliseconds()` para devolver el valor en milisegundos del objeto Date.

```
var today_date:Date = new Date();
trace(today_date.getUTCMilliseconds());
```

getUTCMinutes (método Date.getUTCMinutes)

```
public getUTCMinutes() : Number
```

Devuelve los minutos (un entero del 0 al 59) del objeto Date especificado, de acuerdo con la hora universal.

Disponibilidad: ActionScript 1.0; Flash Player 5

Valor devuelto

`Number` - Un entero.

Ejemplo

El ejemplo siguiente crea un objeto new Date y utiliza `getUTCMinutes()` para devolver el valor en minutos del objeto Date:

```
var today_date:Date = new Date();
trace(today_date.getUTCMinutes());
```

getUTCMonth (método Date.getUTCMonth)

```
public getUTCMonth() : Number
```

Devuelve el mes (de 0 [enero] a 11 [diciembre]) del objeto Date especificado de acuerdo con la hora universal.

Disponibilidad: ActionScript 1.0; Flash Player 5

Valor devuelto

`Number` - Un entero.

Ejemplo

El ejemplo siguiente crea un nuevo objeto Date y emplea `Date.getUTCMonth()` y `Date.getMonth()`. El valor que devuelve `Date.getUTCMonth()` puede ser distinto del valor que devuelve `Date.getMonth()` si la fecha del día es el primer o el última día del mes, según la relación existente entre la zona horaria local y la hora universal.

```
var today_date:Date = new Date();
trace(today_date.getMonth()); // output based on local timezone
trace(today_date.getUTCMonth()); // output equals getMonth() plus or minus
  1
```

Véase también

`getMonth (método Date.getMonth)`

getUTCSeconds (método Date.getUTCSeconds)

```
public getUTCSeconds() : Number
```

Devuelve los segundos (un entero del 0 al 59) del objeto Date especificado de acuerdo con la hora universal.

Disponibilidad: ActionScript 1.0; Flash Player 5

Valor devuelto

`Number` - Un entero.

Ejemplo

El ejemplo siguiente crea un objeto new Date y utiliza `getUTCSeconds()` para devolver el valor en segundos del objeto Date:

```
var today_date:Date = new Date();
trace(today_date.getUTCSeconds());
```

getUTCYear (método Date.getUTCYear)

```
public getUTCYear() : Number
```

Devuelve el año de este objeto `Date` según la hora universal (UTC). El año es el año completo menos 1900. Por ejemplo, el año 2000 se representa como 100.

Disponibilidad: ActionScript 1.0; Flash Player 8

Valor devuelto

`Number` - Un entero.

Ejemplo

El ejemplo siguiente crea un nuevo objeto Date y emplea `Date.getUTCFullYear()` y `Date.getFullYear()`. El valor que devuelve `Date.getUTCFullYear()` puede ser distinto del valor que devuelve `Date.getFullYear()` si la fecha del día es 31 de diciembre o 1 de enero, según la relación existente entre la zona horaria local y la hora universal.

```
var today_date:Date = new Date();

trace(today_date.getFullYear()); // display based on local timezone

trace(today_date.getUTCFullYear()); // displays getYear() plus or minus 1
```

getYear (método Date.getYear)

```
public getYear() : Number
```

Devuelve el año del objeto Date especificado de acuerdo con la hora local. La hora local la determina el sistema operativo en el que se ejecuta Flash Player. El año es el año completo menos 1900. Por ejemplo, el año 2000 se representa como 100.

Disponibilidad: ActionScript 1.0; Flash Player 5

Valor devuelto

`Number` - Un entero.

Ejemplo

El ejemplo siguiente crea un objeto Date con el mes y el año como mayo de 2004. El método `Date.getYear()` devuelve 104 y `Date.getFullYear()` devuelve 2004:

```
var today_date:Date = new Date(2004,4);
trace(today_date.getYear()); // output: 104
trace(today_date.getFullYear()); // output: 2004
```

Véase también

`getFullYear (método Date.getFullYear)`

setDate (método Date.setDate)

```
public setDate(date:Number) : Number
```

Establece el día del mes para el objeto Date especificado de acuerdo con la hora local y devuelve la nueva hora en milisegundos. La hora local la determina el sistema operativo en el que se ejecuta Flash Player.

Disponibilidad: ActionScript 1.0; Flash Player 5

Parámetros

`date``:Number` - Entero de 1 a 31.

Valor devuelto

`Number` - Un entero.

Ejemplo

El ejemplo siguiente crea inicialmente un objeto new Date, que establece la fecha como 15 de mayo de 2004 y utiliza `Date.setDate()` para cambiar la fecha a 25 de mayo de 2004:

```
var today_date:Date = new Date(2004,4,15);
trace(today_date.getDate()); //displays 15
today_date.setDate(25);
trace(today_date.getDate()); //displays 25
```

setFullYear (método Date.setFullYear)

```
public setFullYear(year:Number, [month:Number], [date:Number]) : Number
```

Establece el año del objeto Date especificado de acuerdo con la hora local y devuelve la nueva hora en milisegundos. Si se especifican los parámetros `month` y `date`, éstos se establecen con la hora local. La hora local la determina el sistema operativo en el que se ejecuta Flash Player.

La llamada a este método no modifica los demás campos del objeto Date especificado, aunque `Date.getUTCDay()` y `Date.getDay()` pueden indicar un nuevo valor si cambia el día de la semana como resultado de la llamada a este método.

Disponibilidad: ActionScript 1.0; Flash Player 5

Parámetros

`year``:Number` - Número de cuatro dígitos que especifica un año. Los números de dos dígitos no representan años de cuatro dígitos; por ejemplo, 99 no es el año 1999, sino el año 99.

`month``:Number` [opcional] - Un entero de 0 (enero) a 11 (diciembre). Si omite este parámetro, no se modificará el campo month del objeto Date especificado.

`date``:Number` [opcional] - Un número del 1 al 31. Si omite este parámetro, no se modificará el campo date del objeto Date especificado.

Valor devuelto

`Number` - Un entero.

Ejemplo

El ejemplo siguiente crea inicialmente un objeto new Date, que establece la fecha como 15 de mayo de 2004 y utiliza `Date.setFullYear()` para cambiar la fecha a 15 de mayo de 2002.

```
var my_date:Date = new Date(2004,4,15);
trace(my_date.getFullYear()); //output: 2004
my_date.setFullYear(2002);
trace(my_date.getFullYear()); //output: 2002
```

Véase también

`getUTCDay (método Date.getUTCDay)`, `getDay (método Date.getDay)`

setHours (método Date.setHours)

```
public setHours(hour:Number) : Number
```

Establece las horas del objeto Date especificado de acuerdo con la hora local y devuelve la nueva hora en milisegundos. La hora local la determina el sistema operativo en el que se ejecuta Flash Player.

Disponibilidad: ActionScript 1.0; Flash Player 5

Parámetros

`hour``:Number` - Entero de 0 (medianoche) a 23 (11 de la noche).

Valor devuelto

`Number` - Un entero.

Ejemplo

El ejemplo siguiente crea inicialmente un objeto new Date, que establece la hora y la fecha como 8:00 de la mañana del 15 de mayo de 2004, y utiliza `Date.setHours()` para cambiar la hora a las 4:00 de la tarde:

```
var my_date:Date = new Date(2004,4,15,8);
trace(my_date.getHours()); // output: 8
my_date.setHours(16);
trace(my_date.getHours()); // output: 16
```

setMilliseconds (método Date.setMilliseconds)

```
public setMilliseconds(millisecond:Number) : Number
```

Establece los milisegundos del objeto Date especificado de acuerdo con la hora local y devuelve la nueva hora en milisegundos. La hora local la determina el sistema operativo en el que se ejecuta Flash Player.

Disponibilidad: ActionScript 1.0; Flash Player 5

Parámetros

`millisecond``:Number` - Entero de 0 a 999.

Valor devuelto

`Number` - Un entero.

Ejemplo

El ejemplo siguiente crea inicialmente un objeto new Date, que establece la fecha como las 8:30 de la mañana del 15 de mayo de 2004 con el valor de milisegundos como 250 y, a continuación, utiliza `Date.setMilliseconds()` para cambiar el valor de milisegundos a 575:

```
var my_date:Date = new Date(2004,4,15,8,30,0,250);
trace(my_date.getMilliseconds()); // output: 250
my_date.setMilliseconds(575);
trace(my_date.getMilliseconds()); // output: 575
```

setMinutes (método Date.setMinutes)

```
public setMinutes(minute:Number) : Number
```

Establece los minutos del objeto Date especificado de acuerdo con la hora local y devuelve la nueva hora en milisegundos. La hora local la determina el sistema operativo en el que se ejecuta Flash Player.

Disponibilidad: ActionScript 1.0; Flash Player 5

Parámetros

`minute``:Number` - Entero de 0 a 59.

Valor devuelto

`Number` - Un entero.

Ejemplo

El ejemplo siguiente crea inicialmente un objeto new Date, que establece la hora y la fecha como las 8:00 de la mañana del 15 de mayo de 2004 y, a continuación, utiliza `Date.setMinutes()` para cambiar la hora a las 8:30 de la mañana:

```
var my_date:Date = new Date(2004,4,15,8,0);
trace(my_date.getMinutes()); // output: 0
my_date.setMinutes(30);
trace(my_date.getMinutes()); // output: 30
```

setMonth (método Date.setMonth)

```
public setMonth(month:Number, [date:Number]) : Number
```

Establece el mes del objeto Date especificado en la hora local y devuelve la nueva hora en milisegundos. La hora local la determina el sistema operativo en el que se ejecuta Flash Player.

Disponibilidad: ActionScript 1.0; Flash Player 5

Parámetros

month:Number - Un entero de 0 (enero) a 11 (diciembre).

date:Number [opcional] - Un número del 1 al 31. Si omite este parámetro, no se modificará el campo date del objeto Date especificado.

Valor devuelto

Number - Un entero.

Ejemplo

El ejemplo siguiente crea inicialmente un objeto new Date, que establece la fecha como 15 de mayo de 2004 y utiliza Date.setMonth() para cambiar la fecha a 15 de junio de 2004.

```
var my_date:Date = new Date(2004,4,15);
trace(my_date.getMonth()); //output: 4
my_date.setMonth(5);
trace(my_date.getMonth()); //output: 5
```

setSeconds (método Date.setSeconds)

```
public setSeconds(second:Number) : Number
```

Establece los segundos del objeto Date especificado en la hora local y devuelve la nueva hora en milisegundos. La hora local la determina el sistema operativo en el que se ejecuta Flash Player.

Disponibilidad: ActionScript 1.0; Flash Player 5

Parámetros

second:Number - Entero de 0 a 59.

Valor devuelto

Number - Un entero.

Ejemplo

El ejemplo siguiente crea inicialmente un objeto new Date, que establece la hora y la fecha como 8:00:00 de la mañana del 15 de mayo de 2004, y utiliza `Date.setSeconds()` para cambiar la hora a las 8:00:45 de la mañana:

```
var my_date:Date = new Date(2004,4,15,8,0,0);
trace(my_date.getSeconds()); // output: 0
my_date.setSeconds(45);
trace(my_date.getSeconds()); // output: 45
```

setTime (método Date.setTime)

```
public setTime(millisecond:Number) : Number
```

Establece la fecha del objeto Date especificado en milisegundos desde la media noche del 1 de enero de 1970 y devuelve la nueva hora en milisegundos.

Disponibilidad: ActionScript 1.0; Flash Player 5

Parámetros

`millisecond``:Number` - Un número; un valor de entero, donde 0 es la medianoche del 1 de enero, hora universal.

Valor devuelto

`Number` - Un entero.

Ejemplo

El ejemplo siguiente crea inicialmente un objeto new Date, que establece la hora y la fecha como 8:00 de la mañana del 15 de mayo de 2004 y utiliza `Date.setTime()` para cambiar la hora a las 8:30 de la mañana:

```
var my_date:Date = new Date(2004,4,15,8,0,0);
var myDate_num:Number = my_date.getTime(); // convert my_date to
  milliseconds
myDate_num += 30 * 60 * 1000; // add 30 minutes in milliseconds
my_date.setTime(myDate_num); // set my_date Date object 30 minutes forward
trace(my_date.getFullYear()); // output: 2004
trace(my_date.getMonth()); // output: 4
trace(my_date.getDate()); // output: 15
trace(my_date.getHours()); // output: 8
trace(my_date.getMinutes()); // output: 30
```

setUTCDate (método Date.setUTCDate)

```
public setUTCDate(date:Number) : Number
```

Establece la fecha del objeto Date especificado en la hora universal y devuelve la nueva hora en milisegundos. La llamada a este método no modifica los demás campos del objeto Date especificado, aunque `Date.getUTCDay()` y `Date.getDay()` pueden indicar un nuevo valor si cambia el día de la semana como resultado de la llamada a este método.

Disponibilidad: ActionScript 1.0; Flash Player 5

Parámetros

`date``:Number` - Un número; un entero de 1 a 31.

Valor devuelto

`Number` - Un entero.

Ejemplo

El ejemplo siguiente crea inicialmente un objeto new Date con la fecha del día, utiliza `Date.setUTCDate()` para cambiar el valor de fecha a 10 y lo vuelve a cambiar a 25:

```
var my_date:Date = new Date();
my_date.setUTCDate(10);
trace(my_date.getUTCDate()); // output: 10
my_date.setUTCDate(25);
trace(my_date.getUTCDate()); // output: 25
```

Véase también

`getUTCDay (método Date.getUTCDay)`, `getDay (método Date.getDay)`

getUTCFullYear (método Date.getUTCFullYear)

```
public setUTCFullYear(year:Number, [month:Number], [date:Number]) : Number
```

Establece el año del objeto Date especificado (*`my_date`*) en la hora universal y devuelve la nueva hora en milisegundos.

Opcionalmente, este método también puede establecer el mes y la fecha representados por el objeto Date especificado. La llamada a este método no modifica los demás campos del objeto Date especificado, aunque `Date.getUTCDay()` y `Date.getDay()` pueden indicar un nuevo valor si cambia el día de la semana como resultado de la llamada a este método.

Disponibilidad: ActionScript 1.0; Flash Player 5

Parámetros

year:Number - Un entero que representa el año especificado como año completo de cuatro dígitos, como 2000.

month:Number [opcional] - Un entero de 0 (enero) a 11 (diciembre). Si omite este parámetro, no se modificará el campo month del objeto Date especificado.

date:Number [opcional] - Un número del 1 al 31. Si omite este parámetro, no se modificará el campo date del objeto Date especificado.

Valor devuelto

Number - Un entero.

Ejemplo

El ejemplo siguiente crea inicialmente un objeto new Date con la fecha del día, utiliza Date.setUTCFullYear() para cambiar el valor de año a 2001 y cambia la fecha a 25 de mayo de 1995:

```
var my_date:Date = new Date();
my_date.setUTCFullYear(2001);
trace(my_date.getUTCFullYear()); // output: 2001
my_date.setUTCFullYear(1995, 4, 25);
trace(my_date.getUTCFullYear()); // output: 1995
trace(my_date.getUTCMonth()); // output: 4
trace(my_date.getUTCDate()); // output: 25
```

Véase también

getUTCDay (método Date.getUTCDay), getDay (método Date.getDay)

setUTCHours (método Date.setUTCHours)

```
public setUTCHours(hour:Number, [minute:Number], [second:Number],
  [millisecond:Number]) : Number
```

Establece la hora del objeto Date especificado en la hora universal y devuelve la nueva hora en milisegundos.

Disponibilidad: ActionScript 1.0; Flash Player 5

Parámetros

hour:Number - Un número; un entero desde 0 (medianoche) a 23 (11 de la noche).

minute:Number [opcional] - Un número del 0 al 59. Si omite este parámetro, no se modificará el campo minutes del objeto Date especificado.

second:Number [opcional] - Un número del 0 al 59. Si omite este parámetro, no se modificará el campo seconds del objeto Date especificado.

millisecond:Number [opcional] - Un número del 0 al 999. Si omite este parámetro, no se modificará el campo milliseconds del objeto Date especificado.

Valor devuelto

Number - Un entero.

Ejemplo

El ejemplo siguiente crea inicialmente un objeto new Date con la fecha del día, utiliza Date.setUTCHours() para cambiar la hora a 8:30 de la mañana y vuelve a cambiar la hora a 5:30:47 de la tarde:

```
var my_date:Date = new Date();
my_date.setUTCHours(8,30);
trace(my_date.getUTCHours()); // output: 8
trace(my_date.getUTCMinutes()); // output: 30
my_date.setUTCHours(17,30,47);
trace(my_date.getUTCHours()); // output: 17
trace(my_date.getUTCMinutes()); // output: 30
trace(my_date.getUTCSeconds()); // output: 47
```

setUTCMilliseconds (método Date.setUTCMilliseconds)

```
public setUTCMilliseconds(millisecond:Number) : Number
```

Establece los milisegundos del objeto Date especificado en la hora universal y devuelve la nueva hora en milisegundos.

Disponibilidad: ActionScript 1.0; Flash Player 5

Parámetros

millisecond:Number - Entero de 0 a 999.

Valor devuelto

Number - Un entero.

Ejemplo

El ejemplo siguiente crea inicialmente un objeto new Date, que establece la fecha como las 8:30 de la mañana del 15 de mayo de 2004 con el valor de milisegundos como 250 y, a continuación, utiliza `Date.setUTCMilliseconds()` para cambiar el valor de milisegundos a 575:

```
var my_date:Date = new Date(2004,4,15,8,30,0,250);
trace(my_date.getUTCMilliseconds()); // output: 250
my_date.setUTCMilliseconds(575);
trace(my_date.getUTCMilliseconds()); // output: 575
```

setUTCMinutes (método Date.setUTCMinutes)

```
public setUTCMinutes(minute:Number, [second:Number], [millisecond:Number])
  : Number
```

Establece el minuto del objeto Date especificado en la hora universal y devuelve la nueva hora en milisegundos.

Disponibilidad: ActionScript 1.0; Flash Player 5

Parámetros

`minute``:Number` - Entero de 0 a 59.

`second``:Number` [opcional] - Un número del 0 al 59. Si omite este parámetro, no se modificará el campo seconds del objeto Date especificado.

`millisecond``:Number` [opcional] - Un número del 0 al 999. Si omite este parámetro, no se modificará el campo milliseconds del objeto Date especificado.

Valor devuelto

`Number` - Un entero.

Ejemplo

El ejemplo siguiente crea inicialmente un objeto new Date, que establece la hora y la fecha como 8:00 de la mañana del 15 de mayo de 2004 y utiliza `Date.setUTCMinutes()` para cambiar la hora a las 8:30 de la mañana:

```
var my_date:Date = new Date(2004,4,15,8,0);
trace(my_date.getUTCMinutes()); // output: 0
my_date.setUTCMinutes(30);
trace(my_date.getUTCMinutes()); // output: 30
```

setUTCMonth (método Date.setUTCMonth)

```
public setUTCMonth(month:Number, [date:Number]) : Number
```

Establece el mes y, opcionalmente, el día del objeto Date especificado en la hora universal y devuelve la nueva hora en milisegundos. La llamada a este método no modifica los demás campos del objeto Date especificado, aunque `Date.getUTCDay()` y `Date.getDay()` podrían indicar un nuevo valor si cambia el día de la semana como resultado de especificar un valor para el parámetro `date`.

Disponibilidad: ActionScript 1.0; Flash Player 5

Parámetros

`month``:Number` - Un entero de 0 (enero) a 11 (diciembre).

`date``:Number` [opcional] - Un número del 1 al 31. Si omite este parámetro, no se modificará el campo date del objeto Date especificado.

Valor devuelto

`Number` - Un entero.

Ejemplo

El ejemplo siguiente crea inicialmente un objeto new Date, que establece la fecha como 15 de mayo de 2004 y utiliza `Date.setMonth()` para cambiar la fecha a 15 de junio de 2004.

```
var today_date:Date = new Date(2004,4,15);
trace(today_date.getUTCMonth()); // output: 4
today_date.setUTCMonth(5);
trace(today_date.getUTCMonth()); // output: 5
```

Véase también

`getUTCDay (método Date.getUTCDay)`, `getDay (método Date.getDay)`

setUTCSeconds (método Date.setUTCSeconds)

```
public setUTCSeconds(second:Number, [millisecond:Number]) : Number
```

Establece los segundos del objeto Date especificado en la hora universal y devuelve la nueva hora en milisegundos.

Disponibilidad: ActionScript 1.0; Flash Player 5

Parámetros

second:Number - Entero de 0 a 59.

millisecond:Number [opcional] - Un número del 0 al 999. Si omite este parámetro, no se modificará el campo milliseconds del objeto Date especificado.

Valor devuelto

Number - Un entero.

Ejemplo

El ejemplo siguiente crea inicialmente un objeto new Date, que establece la hora y la fecha como 8:00:00 de la mañana del 15 de mayo de 2004, y utiliza Date.setSeconds() para cambiar la hora a las 8:30:45 de la mañana:

```
var my_date:Date = new Date(2004,4,15,8,0,0);
trace(my_date.getUTCSeconds()); // output: 0
my_date.setUTCSeconds(45);
trace(my_date.getUTCSeconds()); // output: 45
```

setYear (método Date.setYear)

```
public setYear(year:Number) : Number
```

Establece el año del objeto Date especificado en la hora local y devuelve la nueva hora en milisegundos. La hora local la determina el sistema operativo en el que se ejecuta Flash Player.

Disponibilidad: ActionScript 1.0; Flash Player 5

Parámetros

year:Number - Un número que representa el año. Si year es un entero entre 0 y 99, setYear define el año como 1900 + year; en caso contrario, el año es el valor del parámetro year.

Valor devuelto

Number - Un entero.

Ejemplo

El ejemplo siguiente crea un objeto new Date en el que se establece la fecha como 25 de mayo de 2004, utiliza setYear() para cambiar el año a 1999 y cambia el año a 2003:

```
var my_date:Date = new Date(2004,4,25);
trace(my_date.getYear()); // output: 104
trace(my_date.getFullYear()); // output: 2004
my_date.setYear(99);
```

```
trace(my_date.getYear()); // output: 99
trace(my_date.getFullYear()); // output: 1999
my_date.setYear(2003);
trace(my_date.getYear()); // output: 103
trace(my_date.getFullYear()); // output: 2003
```

toString (método Date.toString)

```
public toString() : Cadena
```

Devuelve un valor de cadena para el objeto de fecha especificado en un formato legible.

Disponibilidad: ActionScript 1.0; Flash Player 5

Valor devuelto

`String` - Una cadena.

Ejemplo

El ejemplo siguiente devuelve la información del objeto Date `dateOfBirth_date` como una cadena. El resultado de las sentencias trace están en hora local y varían según corresponda. En el caso del horario de verano de la costa del Pacífico de EE.UU, el resultado es siete horas anterior a la hora universal: lunes, 12 de agosto 18:15:00 GMT-0700 de 1974.

```
var dateOfBirth_date:Date = new Date(74, 7, 12, 18, 15);
trace (dateOfBirth_date);
trace (dateOfBirth_date.toString());
```

UTC (método Date.UTC)

```
public static UTC(year:Number, month:Number, [date:Number], [hour:Number],
  [minute:Number], [second:Number], [millisecond:Number]) : Number
```

Devuelve el número de milisegundos entre la media noche del 1 de enero de 1970, hora universal, y la hora especificada en los parámetros. Este es un método estático que se invoca a través del constructor del objeto Date, no a través de un objeto Date específico. Este método le permite crear un objeto Date que da por hecho la aplicación de la hora universal, mientras que el constructor de Date da por hecho la aplicación de la hora local.

Disponibilidad: ActionScript 1.0; Flash Player 5

Parámetros

`year``:Number` - Un entero de cuatro dígitos que representa el año (por ejemplo, 2000).

`month``:Number` - Un entero de 0 (enero) a 11 (diciembre).

`date``:Number` [opcional] - Entero de 1 a 31.

hour:Number [opcional] - Entero de 0 (medianoche) a 23 (11 de la noche).

minute:Number [opcional] - Entero de 0 a 59.

second:Number [opcional] - Entero de 0 a 59.

millisecond:Number [opcional] - Entero de 0 a 999.

Valor devuelto

Number - Un entero.

Ejemplo

El ejemplo siguiente crea un nuevo objeto Date maryBirthday_date definido en hora universal. Esta es la variación en hora universal del ejemplo empleado con el método del constructor new Date. El resultado se expresa en hora local y varía según corresponda. En el caso del horario de verano de la costa del Pacífico de EE.UU, el resultado es siete horas anterior a la hora universal: domingo, 11 de agosto 17:00:00 GMT-0700 de 1974.

```
var maryBirthday_date:Date = new Date(Date.UTC(1974, 7, 12));
trace(maryBirthday_date);
```

valueOf (método Date.valueOf)

```
public valueOf() : Number
```

Devuelve el número de milisegundos desde la media noche del 1 de enero de 1970, hora universal, para este objeto Date.

Disponibilidad: ActionScript 1.0; Flash Player 5

Valor devuelto

Number - El número de milisegundos.

DisplacementMapFilter (flash.filters.DisplacementMapFilter)

```
Object
  |
  +-flash.filters.BitmapFilter
    |
    +-flash.filters.DisplacementMapFilter
```

```
public class DisplacementMapFilter
extends BitmapFilter
```

La clase DisplacementMapFilter utiliza los valores de píxel del objeto BitmapData especificado (llamado *imagen del mapa de desplazamiento*) para realizar un desplazamiento de un objeto en el escenario, como una instancia MovieClip. Puede utilizar este filtro para conseguir un efecto combado o moteado en una instancia BitmapData o MovieClip.

El uso de filtros depende del objeto al que se aplique el filtro.

Para aplicar filtros a clips de película en tiempo de ejecución, utilice la propiedad `filters`. Establecer la propiedad `filters` de un objeto no modifica el objeto y se puede deshacer borrando la propiedad `filters`.

Para aplicar filtros a instancias de BitmapData, utilice el método `BitmapData.applyFilter()`. La llamada a `applyFilter()` en un objeto BitmapData modifica ese objeto BitmapData y no se puede deshacer.

También puede aplicar efectos de filtro a imágenes y vídeo en tiempo de edición. Para más información, consulte la documentación de edición.

Si aplica un filtro a un clip de película o a un botón, la propiedad `cacheAsBitmap` del clip de película o del botón se establecerá como `true`. Si borra todos los filtros, se restaurará el valor original de `cacheAsBitmap`.

El filtro utiliza la fórmula siguiente:

```
dstPixel[x, y] = srcPixel[x + ((componentX(x, y) - 128) * scaleX) / 256, y +
  ((componentY(x, y) - 128) * scaleY) / 256]
```

donde `componentX(x, y)` obtiene el valor de color de la propiedad `componentX` `mapBitmap` en `(x - mapPoint.x ,y - mapPoint.y)`.

La imagen del mapa que utiliza el filtro se modifica para coincidir con el ajuste de escala del escenario. No se modifica de ninguna forma cuando el propio objeto está a escala.

Este filtro admite la aplicación de escala en el escenario, pero no el ajuste de escala, rotación o sesgo general. Si se ajusta la escala del propio objeto (si la escala-*x* y la escala-*y* no son 100%), el efecto de filtro no se modifica. Sólo se ajusta su escala cuando se acerca el escenario.

Así funciona la clase DisplacementMapFilter. Para cada píxel (*x*,*y*) en el mapa de bits *de destino*, la clase DisplacementMapFilter hace lo siguiente:

- Obtiene el color de (*x*,*y*) en el mapa de bits *map*
- Calcula un desplazamiento basado en ese color
- Comprueba esa ubicación de desplazamiento (*x*+*dx*,*y*+*dy*) en el mapa de bits *de origen*
- Escribe ese píxel en el destino (*x*,*y*), si lo permiten los límites

Un filtro no se aplica si la imagen resultante supera los 2880 píxeles de anchura o altura. Por ejemplo, si amplía un clip de película grande al que se le ha aplicado un filtro, éste se desactiva si la imagen resultante alcanza el límite de 2880 píxeles.

Disponibilidad: ActionScript 1.0; Flash Player 8

Véase también

`applyFilter (método BitmapData.applyFilter)`, `filters (propiedad MovieClip.filters)`, `cacheAsBitmap (propiedad MovieClip.cacheAsBitmap)`

Resumen de propiedades

Modificadores	Propiedad	Descripción
	`alpha:Number`	Especifica el valor de transparencia alfa que debe utilizarse para desplazamientos fuera de los límites.
	`color:Number`	Especifica el color que debe utilizarse para desplazamientos fuera de los límites.
	`componentX:Number`	Describe qué canal de color se utilizará en la imagen del mapa para desplazar el resultado *x*.
	`componentY:Number`	Describe qué canal de color se utilizará en la imagen del mapa para desplazar el resultado *y*.
	`mapBitmap:BitmapData`	Objeto BitmapData que contiene los datos del mapa de desplazamiento.
	`mapPoint:Point`	Un valor `flash.geom.Point` que contiene el desplazamiento de la esquina superior izquierda del clip de película de destino desde la esquina superior izquierda de la imagen del mapa.
	`mode:String`	Modo del filtro.
	`scaleX:Number`	Multiplicador que debe utilizarse para ajustar el tamaño del resultado del desplazamiento de *x* con respecto al cálculo del mapa.
	`scaleY:Number`	Multiplicador que debe utilizarse para ajustar el tamaño del resultado del desplazamiento de *y* con respecto al cálculo del mapa.

Propiedades heredadas de la clase Object

`constructor (propiedad Object.constructor)`, `__proto__ (propiedad Object.__proto__)`, `prototype (propiedad Object.prototype)`, `__resolve (propiedad Object.__resolve)`

Resumen de constructores

Firma	Descripción
`DisplacementMapFilter(mapBitmap:BitmapData, mapPoint:Point, componentX:Number, componentY:Number, scaleX:Number, scaleY:Number, [mode:String], [color:Number], [alpha:Number])`	Inicializa una instancia DisplacementMapFilter con los parámetros especificados.

Resumen de métodos

Modificadores	Firma	Descripción
	`clone() : DisplacementMapFilter`	Devuelve una copia de este objeto de filtro.

Métodos heredados de la clase BitmapFilter

`clone (método BitmapFilter.clone)`

Métodos heredados de la clase Object

`addProperty (método Object.addProperty), hasOwnProperty (método Object.hasOwnProperty), isPropertyEnumerable (método Object.isPropertyEnumerable), isPrototypeOf (método Object.isPrototypeOf), registerClass (método Object.registerClass), toString (método Object.toString), unwatch (método Object.unwatch), valueOf (método Object.valueOf), watch (método Object.watch)`

alpha (propiedad DisplacementMapFilter.alpha)

`public alpha : Number`

Especifica el valor de transparencia alfa que debe utilizarse para desplazamientos fuera de los límites. Se especifica como un valor normalizado de 0,0 a 1,0. Por ejemplo, 0,25 establece un valor de transparencia del 25%. El valor predeterminado es 0. Utilice esta propiedad si la propiedad `mode` es `3, COLOR`.

Disponibilidad: ActionScript 1.0; Flash Player 8

Ejemplo

El ejemplo siguiente modifica la propiedad `alpha` fuera de rango en el clip de película existente `filteredMc` por `0x00FF00` cuando un usuario hace clic en ella.

```
import flash.filters.DisplacementMapFilter;
import flash.display.BitmapData;
import flash.geom.Point;
import flash.geom.Matrix;
import flash.geom.ColorTransform;

var filteredMc:MovieClip = createDisplacementMapRectangle();

filteredMc.onPress = function() {
  var filter:DisplacementMapFilter = this.filters[0];
  filter.scaleY = 25;
  filter.mode = "color";
  filter.alpha = .25;
  this.filters = new Array(filter);
}

function createDisplacementMapRectangle():MovieClip {
  var mapBitmap:BitmapData = createGradientBitmap(300, 80, 0xFF000000,
  "radial");
  var filter:DisplacementMapFilter = new DisplacementMapFilter(mapBitmap,
  new Point(-30, -30), 1, 1, 10, 10, "wrap", 0x000000, 0x000000);

  var txtBlock:MovieClip = createTextBlock();
  txtBlock._x = 30;
  txtBlock._y = 30;

  txtBlock.filters = new Array(filter);

  return txtBlock;
}

function createGradientBitmap(w:Number, h:Number, bgColor:Number,
  type:String, hide:Boolean):BitmapData {
  var mc:MovieClip = this.createEmptyMovieClip("mc", 1);
  var matrix:Matrix = new Matrix();
  matrix.createGradientBox(w, h, 0, 0, 0);

  mc.beginGradientFill(type, [0xFF0000, 0x0000FF], [100, 100], [0x55,
  0x99], matrix, "pad");
  mc.lineTo(w, 0);
  mc.lineTo(w, h);
  mc.lineTo(0, h);
  mc.lineTo(0, 0);
  mc.endFill();
  (hide == true) ? mc._alpha = 0 : mc._alpha = 100;
```

```
  var bmp:BitmapData = new BitmapData(w, h, true, bgColor);
    bmp.draw(mc, new Matrix(), new ColorTransform(), "normal",
  bmp.rectangle, true);
  mc.attachBitmap(bmp, this.getNextHighestDepth());

  return bmp;
}

function createTextBlock():MovieClip {
  var txtBlock:MovieClip = this.createEmptyMovieClip("txtBlock",
  this.getNextHighestDepth());
  txtBlock.createTextField("txt", this.getNextHighestDepth(), 0, 0, 300,
  80);
  txtBlock.txt.text = "watch the text bend with the displacement map";
  return txtBlock;
}
```

clone (método DisplacementMapFilter.clone)

```
public clone() : DisplacementMapFilter
```

Devuelve una copia de este objeto de filtro.

Disponibilidad: ActionScript 1.0; Flash Player 8

Valor devuelto

`flash.filters.DisplacementMapFilter` - Una nueva instancia de DisplacementMapFilter con las mismas propiedades que la original.

Ejemplo

El ejemplo siguiente crea tres objetos DisplacementMapFilter y los compara: `filter_1` se crea utilizando el constructor `DisplacementMapFilter`, `filter_2` se crea con el mismo valor que `filter_1` y `clonedFilter` se crea clonando `filter_1`. Observe que aunque `filter_2` da como resultado `filter_1`, `clonedFilter`, aunque contiene los mismos valores que `filter_1`, no es igual.

```
import flash.filters.DisplacementMapFilter;
import flash.display.BitmapData;
import flash.geom.Point;
import flash.geom.Matrix;
import flash.geom.ColorTransform;

var mapBitmap:BitmapData = createGradientBitmap(300, 80, 0xFF000000,
  "radial", true);

var filter_1:DisplacementMapFilter = new DisplacementMapFilter(mapBitmap,
  new Point(-30, -30), 1, 1, 10, 10, "wrap", 0x000000, 0x000000);
```

```
var filter_2:DisplacementMapFilter = filter_1;
var clonedFilter:DisplacementMapFilter = filter_1.clone();

trace(filter_1 == filter_2); // true
trace(filter_1 == clonedFilter); // false

for(var i in filter_1) {
  trace(">> " + i + ": " + filter_1[i]);
  // >> clone: [type Function]
  // >> alpha: 0
  // >> color: 0
  // >> mode: wrap
  // >> scaleY: 10
  // >> scaleX: 10
  // >> componentY: 1
  // >> componentX: 1
  // >> mapPoint: (-30, -30)
  // >> mapBitmap: [object Object]
}

for(var i in clonedFilter) {
  trace(">> " + i + ": " + clonedFilter[i]);
  // >> clone: [type Function]
  // >> alpha: 0
  // >> color: 0
  // >> mode: wrap
  // >> scaleY: 10
  // >> scaleX: 10
  // >> componentY: 1
  // >> componentX: 1
  // >> mapPoint: (-30, -30)
  // >> mapBitmap: [object Object]
}

function createGradientBitmap(w:Number, h:Number, bgColor:Number,
  type:String, hide:Boolean):BitmapData {
  var mc:MovieClip = this.createEmptyMovieClip("mc", 1);
  var matrix:Matrix = new Matrix();
  matrix.createGradientBox(w, h, 0, 0, 0);

  mc.beginGradientFill(type, [0xFF0000, 0x0000FF], [100, 100], [0x55,
  0x99], matrix, "pad");
  mc.lineTo(w, 0);
  mc.lineTo(w, h);
  mc.lineTo(0, h);
  mc.lineTo(0, 0);
  mc.endFill();
  (hide == true) ? mc._alpha = 0 : mc._alpha = 100;

  var bmp:BitmapData = new BitmapData(w, h, true, bgColor);
```

```
    bmp.draw(mc, new Matrix(), new ColorTransform(), "normal",
  bmp.rectangle, true);
  mc.attachBitmap(bmp, this.getNextHighestDepth());

  return bmp;
}
```

Para mostrar más detalladamente las relaciones entre `filter_1`, `filter_2` y `clonedFilter`, el ejemplo siguiente modifica la propiedad `mode` de `filter_1`. Al modificar `mode`, el método `clone()` crea una nueva instancia basada en los valores de `filter_1` en lugar de utilizarlos en la referencia.

```
import flash.filters.DisplacementMapFilter;
import flash.display.BitmapData;
import flash.geom.Point;
import flash.geom.Matrix;
import flash.geom.ColorTransform;

var mapBitmap:BitmapData = createGradientBitmap(300, 80, 0xFF000000,
  "radial", true);

var filter_1:DisplacementMapFilter = new DisplacementMapFilter(mapBitmap,
  new Point(-30, -30), 1, 1, 10, 10, "wrap", 0x000000, 0x000000);
var filter_2:DisplacementMapFilter = filter_1;
var clonedFilter:DisplacementMapFilter = filter_1.clone();

trace(filter_1.mode); // wrap
trace(filter_2.mode); // wrap
trace(clonedFilter.mode); // wrap

filter_1.mode = "ignore";

trace(filter_1.mode); // ignore
trace(filter_2.mode); // ignore
trace(clonedFilter.mode); // wrap

function createGradientBitmap(w:Number, h:Number, bgColor:Number,
  type:String, hide:Boolean):BitmapData {
  var mc:MovieClip = this.createEmptyMovieClip("mc", 1);
  var matrix:Matrix = new Matrix();
  matrix.createGradientBox(w, h, 0, 0, 0);

  mc.beginGradientFill(type, [0xFF0000, 0x0000FF], [100, 100], [0x55,
  0x99], matrix, "pad");
  mc.lineTo(w, 0);
  mc.lineTo(w, h);
  mc.lineTo(0, h);
  mc.lineTo(0, 0);
  mc.endFill();
  (hide == true) ? mc._alpha = 0 : mc._alpha = 100;
```

```
  var bmp:BitmapData = new BitmapData(w, h, true, bgColor);
    bmp.draw(mc, new Matrix(), new ColorTransform(), "normal",
  bmp.rectangle, true);
  mc.attachBitmap(bmp, this.getNextHighestDepth());

  return bmp;
}
```

color (propiedad DisplacementMapFilter.color)

```
public color : Number
```

Especifica el color que debe utilizarse para desplazamientos fuera de los límites. El rango de desplazamiento válido va de 0,0 a 1,0. Los valores tienen formato hexadecimal. El valor predeterminado de `color` es 0. Utilice esta propiedad si la propiedad `mode` es `3, COLOR`.

Disponibilidad: ActionScript 1.0; Flash Player 8

Ejemplo

El ejemplo siguiente modifica la propiedad `color` fuera de rango en el clip de película existente `filteredMc` por `0x00FF00` cuando un usuario hace clic en ella.

```
import flash.filters.DisplacementMapFilter;
import flash.display.BitmapData;
import flash.geom.Point;
import flash.geom.Matrix;
import flash.geom.ColorTransform;

var filteredMc:MovieClip = createDisplacementMapRectangle();

filteredMc.onPress = function() {
  var filter:DisplacementMapFilter = this.filters[0];
  filter.scaleY = 25;
  filter.mode = "color";
  filter.alpha = .25;
  filter.color = 0x00FF00;
  this.filters = new Array(filter);
}

function createDisplacementMapRectangle():MovieClip {
  var mapBitmap:BitmapData = createGradientBitmap(300, 80, 0xFF000000,
  "radial");
  var filter:DisplacementMapFilter = new DisplacementMapFilter(mapBitmap,
  new Point(-30, -30), 1, 1, 10, 10, "wrap", 0x000000, 0x000000);

  var txtBlock:MovieClip = createTextBlock();
  txtBlock._x = 30;
  txtBlock._y = 30;
```

```
  txtBlock.filters = new Array(filter);

  return txtBlock;
}

function createGradientBitmap(w:Number, h:Number, bgColor:Number,
  type:String, hide:Boolean):BitmapData {
  var mc:MovieClip = this.createEmptyMovieClip("mc", 1);
  var matrix:Matrix = new Matrix();
  matrix.createGradientBox(w, h, 0, 0, 0);

  mc.beginGradientFill(type, [0xFF0000, 0x0000FF], [100, 100], [0x55,
  0x99], matrix, "pad");
  mc.lineTo(w, 0);
  mc.lineTo(w, h);
  mc.lineTo(0, h);
  mc.lineTo(0, 0);
  mc.endFill();
  (hide == true) ? mc._alpha = 0 : mc._alpha = 100;

  var bmp:BitmapData = new BitmapData(w, h, true, bgColor);
    bmp.draw(mc, new Matrix(), new ColorTransform(), "normal",
  bmp.rectangle, true);
  mc.attachBitmap(bmp, this.getNextHighestDepth());

  return bmp;
}

function createTextBlock():MovieClip {
  var txtBlock:MovieClip = this.createEmptyMovieClip("txtBlock",
  this.getNextHighestDepth());
  txtBlock.createTextField("txt", this.getNextHighestDepth(), 0, 0, 300,
  80);
  txtBlock.txt.text = "watch the text bend with the displacement map";
  return txtBlock;
}
```

componentX (propiedad DisplacementMapFilter.componentX)

```
public componentX : Number
```

Describe qué canal de color se utilizará en la imagen del mapa para desplazar el resultado *x*. Los valores posibles son 1 (rojo), 2 (verde), 4 (azul) y 8 (alfa).

Disponibilidad: ActionScript 1.0; Flash Player 8

Ejemplo

El ejemplo siguiente cambia la propiedad `componentX` del clip de película `filteredMc` existente cuando un usuario hace clic en ella. El valor cambia de 1 a 4, que cambia el canal de color de rojo a azul.

```
import flash.filters.DisplacementMapFilter;
import flash.display.BitmapData;
import flash.geom.Point;
import flash.geom.Matrix;
import flash.geom.ColorTransform;

var filteredMc:MovieClip = createDisplacementMapRectangle();

filteredMc.onPress = function() {
  var filter:DisplacementMapFilter = this.filters[0];
  filter.componentX = 4;
  this.filters = new Array(filter);
}


function createDisplacementMapRectangle():MovieClip {
  var mapBitmap:BitmapData = createGradientBitmap(300, 80, 0xFF000000,
  "radial");
  var filter:DisplacementMapFilter = new DisplacementMapFilter(mapBitmap,
  new Point(-30, -30), 1, 1, 10, 10, "wrap", 0x000000, 0x000000);

  var txtBlock:MovieClip = createTextBlock();
  txtBlock._x = 30;
  txtBlock._y = 30;

  txtBlock.filters = new Array(filter);

  return txtBlock;
}

function createGradientBitmap(w:Number, h:Number, bgColor:Number,
  type:String, hide:Boolean):BitmapData {
  var mc:MovieClip = this.createEmptyMovieClip("mc", 1);
  var matrix:Matrix = new Matrix();
  matrix.createGradientBox(w, h, 0, 0, 0);

  mc.beginGradientFill(type, [0xFF0000, 0x0000FF], [100, 100], [0x55,
  0x99], matrix, "pad");
  mc.lineTo(w, 0);
  mc.lineTo(w, h);
  mc.lineTo(0, h);
  mc.lineTo(0, 0);
  mc.endFill();
  (hide == true) ? mc._alpha = 0 : mc._alpha = 100;
```

```
  var bmp:BitmapData = new BitmapData(w, h, true, bgColor);
    bmp.draw(mc, new Matrix(), new ColorTransform(), "normal",
  bmp.rectangle, true);
  mc.attachBitmap(bmp, this.getNextHighestDepth());

  return bmp;
}

function createTextBlock():MovieClip {
  var txtBlock:MovieClip = this.createEmptyMovieClip("txtBlock",
  this.getNextHighestDepth());
  txtBlock.createTextField("txt", this.getNextHighestDepth(), 0, 0, 300,
  80);
  txtBlock.txt.text = "watch the text bend with the displacement map";
  return txtBlock;
}
```

Véase también

`BitmapData (flash.display.BitmapData)`

componentY (propiedad DisplacementMapFilter.componentY)

```
public componentY : Number
```

Describe qué canal de color se utilizará en la imagen del mapa para desplazar el resultado *y*. Los valores posibles son 1 (rojo), 2 (verde), 4 (azul) y 8 (alfa).

Disponibilidad: ActionScript 1.0; Flash Player 8

Ejemplo

El ejemplo siguiente cambia la propiedad `componentY` del clip de película `filteredMc` existente cuando un usuario hace clic en ella. El valor cambia de 1 a 4, que cambia el canal de color de rojo a azul.

```
import flash.filters.DisplacementMapFilter;
import flash.display.BitmapData;
import flash.geom.Point;
import flash.geom.Matrix;
import flash.geom.ColorTransform;

var filteredMc:MovieClip = createDisplacementMapRectangle();

filteredMc.onPress = function() {
  var filter:DisplacementMapFilter = this.filters[0];
  filter.componentY = 4;
```

```
  this.filters = new Array(filter);
}

function createDisplacementMapRectangle():MovieClip {
  var mapBitmap:BitmapData = createGradientBitmap(300, 80, 0xFF000000,
  "radial");
  var filter:DisplacementMapFilter = new DisplacementMapFilter(mapBitmap,
  new Point(-30, -30), 1, 1, 10, 10, "wrap", 0x000000, 0x000000);

  var txtBlock:MovieClip = createTextBlock();
  txtBlock._x = 30;
  txtBlock._y = 30;

  txtBlock.filters = new Array(filter);

  return txtBlock;
}

function createGradientBitmap(w:Number, h:Number, bgColor:Number,
  type:String, hide:Boolean):BitmapData {
  var mc:MovieClip = this.createEmptyMovieClip("mc", 1);
  var matrix:Matrix = new Matrix();
  matrix.createGradientBox(w, h, 0, 0, 0);

  mc.beginGradientFill(type, [0xFF0000, 0x0000FF], [100, 100], [0x55,
  0x99], matrix, "pad");
  mc.lineTo(w, 0);
  mc.lineTo(w, h);
  mc.lineTo(0, h);
  mc.lineTo(0, 0);
  mc.endFill();
  (hide == true) ? mc._alpha = 0 : mc._alpha = 100;

  var bmp:BitmapData = new BitmapData(w, h, true, bgColor);
    bmp.draw(mc, new Matrix(), new ColorTransform(), "normal",
  bmp.rectangle, true);
  mc.attachBitmap(bmp, this.getNextHighestDepth());

  return bmp;
}

function createTextBlock():MovieClip {
  var txtBlock:MovieClip = this.createEmptyMovieClip("txtBlock",
  this.getNextHighestDepth());
  txtBlock.createTextField("txt", this.getNextHighestDepth(), 0, 0, 300,
  80);
  txtBlock.txt.text = "watch the text bend with the displacement map";
  return txtBlock;
}
```

Véase también

`BitmapData (flash.display.BitmapData)`

Constructor DisplacementMapFilter

```
public DisplacementMapFilter(mapBitmap:BitmapData, mapPoint:Point,
  componentX:Number, componentY:Number, scaleX:Number, scaleY:Number,
  [mode:String], [color:Number], [alpha:Number])
```

Inicializa una instancia DisplacementMapFilter con los parámetros especificados.

Disponibilidad: ActionScript 1.0; Flash Player 8

Parámetros

`mapBitmap``:flash.display.BitmapData` - Objeto BitmapData que contiene los datos del mapa de desplazamiento.

`mapPoint``:flash.geom.Point` - Un valor `flash.geom.Point` que contiene el desplazamiento de la esquina superior izquierda del clip de película de destino desde la esquina superior izquierda de la imagen del mapa.

`componentX``:Number` - Describe qué canal de color se utilizará en la imagen del mapa para desplazar el resultado *x*. Los valores son los siguientes:

- `1` (rojo)
- `2` (verde)
- `4` (azul)
- `8` (alfa)

`componentY``:Number` - Describe qué canal de color se utilizará en la imagen del mapa para desplazar el resultado *y*. Los valores son los siguientes:

- `1` (rojo)
- `2` (verde)
- `4` (azul)
- `8` (alfa)

`scaleX``:Number` - Multiplicador que debe utilizarse para ajustar la escala del resultado del desplazamiento de *x* con respecto al cálculo del mapa.

`scaleY``:Number` - Multiplicador que debe utilizarse para ajustar la escala del resultado del desplazamiento de *y* con respecto al cálculo del mapa.

mode:String [opcional] - Modo del filtro. Los valores son los siguientes:

- "wrap" - Ajusta el valor de desplazamiento al otro lado de la imagen de origen.
- "clamp" - Fija el valor de desplazamiento en el borde de la imagen de origen.
- "ignore" - Si el valor de desplazamiento está fuera de rango, omite el desplazamiento y utiliza el píxel de origen.
- "color" - Si el valor de desplazamiento está fuera del rango, sustituye un valor de píxel compuesto de las propiedades alpha y color del filtro.

color:Number [opcional] - Especifica el color que debe utilizarse para desplazamientos fuera de los límites. El rango de desplazamiento válido va de 0,0 a 1,0. Utilice este parámetro si el mode se define como "color".

alpha:Number [opcional] - Especifica el valor alfa que debe utilizarse para desplazamientos fuera de los límites. Se especifica como un valor normalizado de 0,0 a 1,0. Por ejemplo, 0,25 establece un valor de transparencia del 25%. El valor predeterminado es 1,0. Utilice este parámetro si el mode se define como "color".

Ejemplo

La función constructora siguiente crea una nueva instancia del filtro:

```
myFilter = new flash.filters.DisplacementMapFilter (mapBitmap, mapPoint,
  componentX, componentY, scale, [mode], [color], [alpha])
```

El ejemplo siguiente crea una instancia del nuevo DisplacementMapFilter con un mapa de bits de degradado radial y la aplica al texto que contiene el objeto MovieClip txtBlock.

```
import flash.filters.DisplacementMapFilter;
import flash.display.BitmapData;
import flash.geom.Point;
import flash.geom.Matrix;
import flash.geom.ColorTransform;

var mapBitmap:BitmapData = createGradientBitmap(300, 80, 0xFF000000,
  "radial");

var mapPoint:Point = new Point(-30, -30);
var componentX:Number = 1;
var componentY:Number = 1;
var scaleX:Number = 10;
var scaleY:Number = 10;
var mode:String = "wrap";
var color:Number = 0x000000;
var alpha:Number = 0x000000;

var filter:DisplacementMapFilter = new DisplacementMapFilter(mapBitmap,
  mapPoint, componentX, componentY, scaleX, scaleY, mode, color, alpha);
```

```
var txtBlock:MovieClip = createTextBlock();
txtBlock._x = 30;
txtBlock._y = 30;

txtBlock.filters = new Array(filter);

function createGradientBitmap(w:Number, h:Number, bgColor:Number,
  type:String, hide:Boolean):BitmapData {
  var mc:MovieClip = this.createEmptyMovieClip("mc", 1);
  var matrix:Matrix = new Matrix();
  matrix.createGradientBox(w, h, 0, 0, 0);

  mc.beginGradientFill(type, [0xFF0000, 0x0000FF], [100, 100], [0x55,
  0x99], matrix, "pad");
  mc.lineTo(w, 0);
  mc.lineTo(w, h);
  mc.lineTo(0, h);
  mc.lineTo(0, 0);
  mc.endFill();
  (hide == true) ? mc._alpha = 0 : mc._alpha = 100;

  var bmp:BitmapData = new BitmapData(w, h, true, bgColor);
    bmp.draw(mc, new Matrix(), new ColorTransform(), "normal",
  bmp.rectangle, true);
  mc.attachBitmap(bmp, this.getNextHighestDepth());

  return bmp;
}

function createTextBlock():MovieClip {
  var txtBlock:MovieClip = this.createEmptyMovieClip("txtBlock",
  this.getNextHighestDepth());
    txtBlock.createTextField("txt", this.getNextHighestDepth(), 0, 0, 300,
  80);
  txtBlock.txt.text = "watch the text bend with the displacement map";
  return txtBlock;
}
```

mapBitmap (propiedad DisplacementMapFilter.mapBitmap)

```
public mapBitmap : BitmapData
```

Objeto BitmapData que contiene los datos del mapa de desplazamiento.

La propiedad `mapBitmap` no puede cambiarse directamente mediante la modificación de este valor. Debe obtener una referencia a `mapBitmap`, realizar el cambio en la referencia y definir `mapBitmap` como la referencia.

Disponibilidad: ActionScript 1.0; Flash Player 8

Ejemplo

El ejemplo siguiente cambia la propiedad `mapBitmap` del clip de película `filteredMc` existente cuando un usuario hace clic en ella.

```
import flash.filters.DisplacementMapFilter;
import flash.display.BitmapData;
import flash.geom.Point;
import flash.geom.Matrix;
import flash.geom.ColorTransform;

var filteredMc:MovieClip = createDisplacementMapRectangle();
var scope:Object = this;

filteredMc.onPress = function() {
  var filter:DisplacementMapFilter = this.filters[0];
  filter.mapBitmap = scope.createGradientBitmap(300, 80, 0xFF000000,
  "linear");
  this.filters = new Array(filter);
}

function createDisplacementMapRectangle():MovieClip {
  var mapBitmap:BitmapData = createGradientBitmap(300, 80, 0xFF000000,
  "radial");
  var filter:DisplacementMapFilter = new DisplacementMapFilter(mapBitmap,
  new Point(-30, -30), 1, 1, 10, 10, "wrap", 0x000000, 0x000000);

  var txtBlock:MovieClip = createTextBlock();
  txtBlock._x = 30;
  txtBlock._y = 30;

  txtBlock.filters = new Array(filter);

  return txtBlock;
}

function createGradientBitmap(w:Number, h:Number, bgColor:Number,
  type:String, hide:Boolean):BitmapData {
  var mc:MovieClip = this.createEmptyMovieClip("mc", 1);
  var matrix:Matrix = new Matrix();
  matrix.createGradientBox(w, h, 0, 0, 0);

  mc.beginGradientFill(type, [0xFF0000, 0x0000FF], [100, 100], [0x55,
  0x99], matrix, "pad");
  mc.lineTo(w, 0);
  mc.lineTo(w, h);
  mc.lineTo(0, h);
  mc.lineTo(0, 0);
```

```
  mc.endFill();
  (hide == true) ? mc._alpha = 0 : mc._alpha = 100;

  var bmp:BitmapData = new BitmapData(w, h, true, bgColor);
    bmp.draw(mc, new Matrix(), new ColorTransform(), "normal",
  bmp.rectangle, true);
  mc.attachBitmap(bmp, this.getNextHighestDepth());

  return bmp;
}

function createTextBlock():MovieClip {
  var txtBlock:MovieClip = this.createEmptyMovieClip("txtBlock",
  this.getNextHighestDepth());
  txtBlock.createTextField("txt", this.getNextHighestDepth(), 0, 0, 300,
  80);
  txtBlock.txt.text = "watch the text bend with the displacement map";
  return txtBlock;
}
```

Véase también

BitmapData (flash.display.BitmapData)

mapPoint (propiedad DisplacementMapFilter.mapPoint)

```
public mapPoint : Point
```

Un valor `flash.geom.Point` que contiene el desplazamiento de la esquina superior izquierda del clip de película de destino desde la esquina superior izquierda de la imagen del mapa.

La propiedad `mapPoint` no puede cambiarse directamente mediante la modificación de este valor. Debe obtener una referencia a `mapPoint`, realizar el cambio en la referencia y definir `mapPoint` como la referencia.

Disponibilidad: ActionScript 1.0; Flash Player 8

Ejemplo

El ejemplo siguiente cambia la propiedad `mapPoint` del clip de película `filteredMc` existente cuando un usuario hace clic en ella.

```
import flash.filters.DisplacementMapFilter;
import flash.display.BitmapData;
import flash.geom.Point;
import flash.geom.Matrix;
import flash.geom.ColorTransform;
```

```
var filteredMc:MovieClip = createDisplacementMapRectangle();

filteredMc.onPress = function() {
  var filter:DisplacementMapFilter = this.filters[0];
  filter.mapPoint = new Point(-30, -40);
  this.filters = new Array(filter);
  this._x = 30;
  this._y = 40;
}

function createDisplacementMapRectangle():MovieClip {
  var mapBitmap:BitmapData = createGradientBitmap(300, 80, 0xFF000000,
  "radial");
  var filter:DisplacementMapFilter = new DisplacementMapFilter(mapBitmap,
  new Point(-30, -30), 1, 1, 10, 10, "wrap", 0x000000, 0x000000);

  var txtBlock:MovieClip = createTextBlock();
  txtBlock._x = 30;
  txtBlock._y = 30;

  txtBlock.filters = new Array(filter);

  return txtBlock;
}

function createGradientBitmap(w:Number, h:Number, bgColor:Number,
  type:String, hide:Boolean):BitmapData {
  var mc:MovieClip = this.createEmptyMovieClip("mc", 1);
  var matrix:Matrix = new Matrix();
  matrix.createGradientBox(w, h, 0, 0, 0);

  mc.beginGradientFill(type, [0xFF0000, 0x0000FF], [100, 100], [0x55,
  0x99], matrix, "pad");
  mc.lineTo(w, 0);
  mc.lineTo(w, h);
  mc.lineTo(0, h);
  mc.lineTo(0, 0);
  mc.endFill();
  (hide == true) ? mc._alpha = 0 : mc._alpha = 100;

  var bmp:BitmapData = new BitmapData(w, h, true, bgColor);
    bmp.draw(mc, new Matrix(), new ColorTransform(), "normal",
  bmp.rectangle, true);
  mc.attachBitmap(bmp, this.getNextHighestDepth());

  return bmp;
}

function createTextBlock():MovieClip {
```

```
  var txtBlock:MovieClip = this.createEmptyMovieClip("txtBlock",
  this.getNextHighestDepth());
  txtBlock.createTextField("txt", this.getNextHighestDepth(), 0, 0, 300,
  80);
  txtBlock.txt.text = "watch the text bend with the displacement map";
  return txtBlock;
}
```

Véase también

Point (flash.geom.Point)

mode (propiedad DisplacementMapFilter.mode)

```
public mode : String
```

Modo del filtro. Los valores son los siguientes:

- `"wrap"` - Ajusta el valor de desplazamiento al otro lado de la imagen de origen. Éste es el valor predeterminado.
- `"clamp"` - Fija el valor de desplazamiento en el borde de la imagen de origen.
- `"ignore"` - Si el valor de desplazamiento está fuera de rango, omite el desplazamiento y utiliza el píxel de origen.
- `"color"` - Si el valor de desplazamiento está fuera del rango, sustituye un valor de píxel compuesto de las propiedades `alpha` y `color` del filtro.

Disponibilidad: ActionScript 1.0; Flash Player 8

Ejemplo

El ejemplo siguiente modifica `scaleY` para crear un valor de desplazamiento que está fuera de rango y, a continuación, cambia la propiedad `mode` del clip de película existente `filteredMc` a `ignore` cuando el usuario hace clic en ella.

```
import flash.filters.DisplacementMapFilter;
import flash.display.BitmapData;
import flash.geom.Point;
import flash.geom.Matrix;
import flash.geom.ColorTransform;

var filteredMc:MovieClip = createDisplacementMapRectangle();

filteredMc.onPress = function() {
  var filter:DisplacementMapFilter = this.filters[0];
  filter.scaleY = 25;
  filter.mode = "ignore";
  this.filters = new Array(filter);
}
```

```
function createDisplacementMapRectangle():MovieClip {
  var mapBitmap:BitmapData = createGradientBitmap(300, 80, 0xFF000000,
  "radial");
  var filter:DisplacementMapFilter = new DisplacementMapFilter(mapBitmap,
  new Point(-30, -30), 1, 1, 10, 10, "wrap", 0x000000, 0x000000);

  var txtBlock:MovieClip = createTextBlock();
  txtBlock._x = 30;
  txtBlock._y = 30;

  txtBlock.filters = new Array(filter);

  return txtBlock;
}

function createGradientBitmap(w:Number, h:Number, bgColor:Number,
  type:String, hide:Boolean):BitmapData {
  var mc:MovieClip = this.createEmptyMovieClip("mc", 1);
  var matrix:Matrix = new Matrix();
  matrix.createGradientBox(w, h, 0, 0, 0);

  mc.beginGradientFill(type, [0xFF0000, 0x0000FF], [100, 100], [0x55,
  0x99], matrix, "pad");
  mc.lineTo(w, 0);
  mc.lineTo(w, h);
  mc.lineTo(0, h);
  mc.lineTo(0, 0);
  mc.endFill();
  (hide == true) ? mc._alpha = 0 : mc._alpha = 100;

  var bmp:BitmapData = new BitmapData(w, h, true, bgColor);
    bmp.draw(mc, new Matrix(), new ColorTransform(), "normal",
  bmp.rectangle, true);
  mc.attachBitmap(bmp, this.getNextHighestDepth());

  return bmp;
}

function createTextBlock():MovieClip {
  var txtBlock:MovieClip = this.createEmptyMovieClip("txtBlock",
  this.getNextHighestDepth());
  txtBlock.createTextField("txt", this.getNextHighestDepth(), 0, 0, 300,
  80);
  txtBlock.txt.text = "watch the text bend with the displacement map";
  return txtBlock;
}
```

scaleX (propiedad DisplacementMapFilter.scaleX)

```
public scaleX : Number
```

Multiplicador que debe utilizarse para ajustar el tamaño del resultado del desplazamiento de *x* con respecto al cálculo del mapa.

Disponibilidad: ActionScript 1.0; Flash Player 8

Ejemplo

El ejemplo siguiente cambia la propiedad `scaleX` del clip de película `filteredMc` existente cuando un usuario hace clic en ella.

```
import flash.filters.DisplacementMapFilter;
import flash.display.BitmapData;
import flash.geom.Point;
import flash.geom.Matrix;
import flash.geom.ColorTransform;

var filteredMc:MovieClip = createDisplacementMapRectangle();

filteredMc.onPress = function() {
  var filter:DisplacementMapFilter = this.filters[0];
  filter.scaleX = 5;
  this.filters = new Array(filter);
}

function createDisplacementMapRectangle():MovieClip {
  var mapBitmap:BitmapData = createGradientBitmap(300, 80, 0xFF000000,
  "radial");
  var filter:DisplacementMapFilter = new DisplacementMapFilter(mapBitmap,
  new Point(-30, -30), 1, 1, 10, 10, "wrap", 0x000000, 0x000000);

  var txtBlock:MovieClip = createTextBlock();
  txtBlock._x = 30;
  txtBlock._y = 30;

  txtBlock.filters = new Array(filter);

  return txtBlock;
}

function createGradientBitmap(w:Number, h:Number, bgColor:Number,
  type:String, hide:Boolean):BitmapData {
  var mc:MovieClip = this.createEmptyMovieClip("mc", 1);
  var matrix:Matrix = new Matrix();
  matrix.createGradientBox(w, h, 0, 0, 0);

  mc.beginGradientFill(type, [0xFF0000, 0x0000FF], [100, 100], [0x55,
  0x99], matrix, "pad");
```

```
  mc.lineTo(w, 0);
  mc.lineTo(w, h);
  mc.lineTo(0, h);
  mc.lineTo(0, 0);
  mc.endFill();
  (hide == true) ? mc._alpha = 0 : mc._alpha = 100;

  var bmp:BitmapData = new BitmapData(w, h, true, bgColor);
    bmp.draw(mc, new Matrix(), new ColorTransform(), "normal",
  bmp.rectangle, true);
  mc.attachBitmap(bmp, this.getNextHighestDepth());

  return bmp;
}

function createTextBlock():MovieClip {
  var txtBlock:MovieClip = this.createEmptyMovieClip("txtBlock",
  this.getNextHighestDepth());
  txtBlock.createTextField("txt", this.getNextHighestDepth(), 0, 0, 300,
  80);
  txtBlock.txt.text = "watch the text bend with the displacement map";
  return txtBlock;
}
```

scaleY (propiedad DisplacementMapFilter.scaleY)

```
public scaleY : Number
```

Multiplicador que debe utilizarse para ajustar el tamaño del resultado del desplazamiento de *y* con respecto al cálculo del mapa.

Disponibilidad: ActionScript 1.0; Flash Player 8

Ejemplo

El ejemplo siguiente cambia la propiedad `scaleY` del clip de película `filteredMc` existente cuando un usuario hace clic en ella.

```
import flash.filters.DisplacementMapFilter;
import flash.display.BitmapData;
import flash.geom.Point;
import flash.geom.Matrix;
import flash.geom.ColorTransform;

var filteredMc:MovieClip = createDisplacementMapRectangle();

filteredMc.onPress = function() {
  var filter:DisplacementMapFilter = this.filters[0];
  filter.scaleY = 5;
  this.filters = new Array(filter);
```

```
}

function createDisplacementMapRectangle():MovieClip {
  var mapBitmap:BitmapData = createGradientBitmap(300, 80, 0xFF000000,
  "radial");
  var filter:DisplacementMapFilter = new DisplacementMapFilter(mapBitmap,
  new Point(-30, -30), 1, 1, 10, 10, "wrap", 0x000000, 0x000000);

  var txtBlock:MovieClip = createTextBlock();
  txtBlock._x = 30;
  txtBlock._y = 30;

  txtBlock.filters = new Array(filter);

  return txtBlock;
}

function createGradientBitmap(w:Number, h:Number, bgColor:Number,
  type:String, hide:Boolean):BitmapData {
  var mc:MovieClip = this.createEmptyMovieClip("mc", 1);
  var matrix:Matrix = new Matrix();
  matrix.createGradientBox(w, h, 0, 0, 0);

  mc.beginGradientFill(type, [0xFF0000, 0x0000FF], [100, 100], [0x55,
  0x99], matrix, "pad");
  mc.lineTo(w, 0);
  mc.lineTo(w, h);
  mc.lineTo(0, h);
  mc.lineTo(0, 0);
  mc.endFill();
  (hide == true) ? mc._alpha = 0 : mc._alpha = 100;

  var bmp:BitmapData = new BitmapData(w, h, true, bgColor);
    bmp.draw(mc, new Matrix(), new ColorTransform(), "normal",
  bmp.rectangle, true);
  mc.attachBitmap(bmp, this.getNextHighestDepth());

  return bmp;
}

function createTextBlock():MovieClip {
  var txtBlock:MovieClip = this.createEmptyMovieClip("txtBlock",
  this.getNextHighestDepth());
  txtBlock.createTextField("txt", this.getNextHighestDepth(), 0, 0, 300,
  80);
  txtBlock.txt.text = "watch the text bend with the displacement map";
  return txtBlock;
}
```

DropShadowFilter (flash.filters.DropShadowFilter)

```
Object
  |
  +-flash.filters.BitmapFilter
    |
    +-flash.filters.DropShadowFilter
```

```
public class DropShadowFilter
extends BitmapFilter
```

La clase DropShadowFilter permite añadir una sombra a diversos objetos de Flash. Cuenta con varias opciones para el estilo de la sombra, incluida sombra interior o exterior y modo de extractor.

El uso de filtros depende del objeto al que se aplique el filtro:

- Para aplicar filtros a clips de película, campos de texto y botones en tiempo de ejecución, utilice la propiedad `filters`. Establecer la propiedad `filters` de un objeto no modifica el objeto y se puede deshacer borrando la propiedad `filters`.
- Para aplicar filtros a instancias de BitmapData, utilice el método `BitmapData.applyFilter()`. Cuando se llama a `applyFilter()` en un objeto BitmapData, se genera una imagen filtrada al procesar el objeto BitmapData de origen y el objeto de filtro.

También puede aplicar efectos de filtro a imágenes y vídeo en tiempo de edición. Para más información, consulte la documentación de edición.

Si aplica un filtro a un clip de película o a un botón, la propiedad `cacheAsBitmap` del clip de película o del botón se establecerá como `true`. Si borra todos los filtros, se restaurará el valor original de `cacheAsBitmap`.

Este filtro admite la aplicación de escala en el escenario. Sin embargo, no se puede ajustar la escala, rotación y sesgo general. Si el objeto está a escala (si `_xscale` y `_yscale` no son 100%), el efecto de filtro no se modifica. Sólo se ajusta su escala cuando se acerca el escenario.

Un filtro no se aplica si la imagen resultante supera los 2880 píxeles de anchura o altura. Por ejemplo, si amplía un clip de película grande al que se le ha aplicado un filtro, éste se desactiva si la imagen resultante supera el límite de 2880 píxeles.

Disponibilidad: ActionScript 1.0; Flash Player 8

Véase también

filters (propiedad MovieClip.filters), cacheAsBitmap (propiedad MovieClip.cacheAsBitmap), filters (propiedad Button.filters), cacheAsBitmap (propiedad Button.cacheAsBitmap), filters (propiedad TextField.filters), applyFilter (método BitmapData.applyFilter)

Resumen de propiedades

Modificadores	Propiedad	Descripción
	alpha:Number	Valor de transparencia alfa del color de la sombra.
	angle:Number	Ángulo de la sombra.
	blurX:Number	Cantidad de desenfoque horizontal.
	blurY:Number	Cantidad de desenfoque vertical.
	color:Number	Color de la sombra.
	distance:Number	Distancia de desplazamiento de la sombra, en píxeles.
	hideObject:Boolean	Indica si el objeto está oculto.
	inner:Boolean	Indica si la sombra es una sombra interior.
	knockout:Boolean	Aplica un efecto de extractor (true), que hace que el relleno del objeto sea transparente y muestra el color de fondo del documento.
	quality:Number	Número de veces que debe aplicarse el filtro.
	strength:Number	Intensidad de la impresión o extensión.

Propiedades heredadas de la clase Object

constructor (propiedad Object.constructor), __proto__ (propiedad Object.__proto__), prototype (propiedad Object.prototype), __resolve (propiedad Object.__resolve)

Resumen de constructores

Firma	Descripción
`DropShadowFilter([distance:Number], [angle:Number], [color:Number], [alpha:Number], [blurX:Number], [blurY:Number], [strength:Number], [quality:Number], [inner:Boolean], [knockout:Boolean], [hideObject:Boolean])`	Crea una instancia DropShadowFilter nueva con los parámetros especificados.

Resumen de métodos

Modificadores	Firma	Descripción
	`clone() : DropShadowFilter`	Devuelve una copia de este objeto de filtro.

Métodos heredados de la clase BitmapFilter

`clone (método BitmapFilter.clone)`

Métodos heredados de la clase Object

`addProperty (método Object.addProperty), hasOwnProperty (método Object.hasOwnProperty), isPropertyEnumerable (método Object.isPropertyEnumerable), isPrototypeOf (método Object.isPrototypeOf), registerClass (método Object.registerClass), toString (método Object.toString), unwatch (método Object.unwatch), valueOf (método Object.valueOf), watch (método Object.watch)`

alpha (propiedad DropShadowFilter.alpha)

`public alpha : Number`

Valor de transparencia alfa del color de la sombra. Los valores válidos van de 0 a 1. Por ejemplo, 0,25 define un valor de transparencia de 25%. El valor predeterminado es 1.

Disponibilidad: ActionScript 1.0; Flash Player 8

Ejemplo

El ejemplo siguiente cambia la propiedad `alpha` de un clip de película existente cuando un usuario hace clic en ella.

```
import flash.filters.DropShadowFilter;
var mc:MovieClip = createDropShadowRectangle("DropShadowAlpha");
mc.onRelease = function() {
  var filter:DropShadowFilter = this.filters[0];
  filter.alpha = .4;
  this.filters = new Array(filter);
}

function createDropShadowRectangle(name:String):MovieClip {
  var art:MovieClip = this.createEmptyMovieClip(name,
  this.getNextHighestDepth());
  var w:Number = 100;
  var h:Number = 100;
  art.beginFill(0x003366);
  art.lineTo(w, 0);
  art.lineTo(w, h);
  art.lineTo(0, h);
  art.lineTo(0, 0);
  art._x = 20;
  art._y = 20;

  var filter:DropShadowFilter = new DropShadowFilter(15, 45, 0x000000, .8,
  16, 16, 1, 3, false, false, false);
  var filterArray:Array = new Array();
  filterArray.push(filter);
  art.filters = filterArray;
  return art;
}
```

angle (propiedad DropShadowFilter.angle)

```
public angle : Number
```

Ángulo de la sombra. Los valores válidos van de 0 a 360° (coma flotante). El valor predeterminado es 45.

El valor de ángulo representa el ángulo de la fuente de luz que en teoría cae sobre el objeto y determina la ubicación del efecto en relación al efecto. Si la distancia es 0, el efecto no se desplaza del objeto y, por lo tanto, la propiedad angle no causa ningún efecto.

Disponibilidad: ActionScript 1.0; Flash Player 8

Ejemplo

El ejemplo siguiente cambia la propiedad `angle` de un clip de película existente cuando un usuario hace clic en ella.

```
import flash.filters.DropShadowFilter;
var mc:MovieClip = createDropShadowRectangle("DropShadowAngle");
mc.onRelease = function() {
  var filter:DropShadowFilter = this.filters[0];
  filter.angle = 135;
  this.filters = new Array(filter);
}

function createDropShadowRectangle(name:String):MovieClip {
  var art:MovieClip = this.createEmptyMovieClip(name,
  this.getNextHighestDepth());
  var w:Number = 100;
  var h:Number = 100;
  art.beginFill(0x003366);
  art.lineTo(w, 0);
  art.lineTo(w, h);
  art.lineTo(0, h);
  art.lineTo(0, 0);
  art._x = 20;
  art._y = 20;

  var filter:DropShadowFilter = new DropShadowFilter(15, 45, 0x000000, .8,
  16, 16, 1, 3, false, false, false);
  var filterArray:Array = new Array();
  filterArray.push(filter);
  art.filters = filterArray;
  return art;
}
```

blurX (propiedad DropShadowFilter.blurX)

```
public blurX : Number
```

Cantidad de desenfoque horizontal. Los valores válidos van de 0 a 255 (coma flotante). El valor predeterminado es 4. Los valores que son una potencia de 2 (como 2, 4, 8, 16 y 32) se optimizan para representarse más rápidamente que otros valores.

Disponibilidad: ActionScript 1.0; Flash Player 8

Ejemplo

El ejemplo siguiente cambia la propiedad `blurX` de un clip de película existente cuando un usuario hace clic en ella.

```
import flash.filters.DropShadowFilter;
var mc:MovieClip = createDropShadowRectangle("DropShadowBlurX");
```

```
mc.onRelease = function() {
  var filter:DropShadowFilter = this.filters[0];
  filter.blurX = 40;
  this.filters = new Array(filter);
}

function createDropShadowRectangle(name:String):MovieClip {
  var art:MovieClip = this.createEmptyMovieClip(name,
  this.getNextHighestDepth());
  var w:Number = 100;
  var h:Number = 100;
  art.beginFill(0x003366);
  art.lineTo(w, 0);
  art.lineTo(w, h);
  art.lineTo(0, h);
  art.lineTo(0, 0);
  art._x = 20;
  art._y = 20;

  var filter:DropShadowFilter = new DropShadowFilter(15, 45, 0x000000, .8,
  16, 16, 1, 3, false, false, false);
  var filterArray:Array = new Array();
  filterArray.push(filter);
  art.filters = filterArray;
  return art;
}
```

blurY (propiedad DropShadowFilter.blurY)

```
public blurY : Number
```

Cantidad de desenfoque vertical. Los valores válidos van de 0 a 255 (coma flotante). El valor predeterminado es 4. Los valores que son una potencia de 2 (como 2, 4, 8, 16 y 32) se optimizan para representarse más rápidamente que otros valores.

Disponibilidad: ActionScript 1.0; Flash Player 8

Ejemplo

El ejemplo siguiente cambia la propiedad `blurY` de un clip de película existente cuando un usuario hace clic en ella.

```
import flash.filters.DropShadowFilter;
var mc:MovieClip = createDropShadowRectangle("DropShadowBlurY");
mc.onRelease = function() {
  var filter:DropShadowFilter = this.filters[0];
  filter.blurY = 40;
  this.filters = new Array(filter);
}
```

```
function createDropShadowRectangle(name:String):MovieClip {
  var art:MovieClip = this.createEmptyMovieClip(name,
  this.getNextHighestDepth());
  var w:Number = 100;
  var h:Number = 100;
  art.beginFill(0x003366);
  art.lineTo(w, 0);
  art.lineTo(w, h);
  art.lineTo(0, h);
  art.lineTo(0, 0);
  art._x = 20;
  art._y = 20;

  var filter:DropShadowFilter = new DropShadowFilter(15, 45, 0x000000, .8,
  16, 16, 1, 3, false, false, false);
  var filterArray:Array = new Array();
  filterArray.push(filter);
  art.filters = filterArray;
  return art;
}
```

clone (método DropShadowFilter.clone)

```
public clone() : DropShadowFilter
```

Devuelve una copia de este objeto de filtro.

Disponibilidad: ActionScript 1.0; Flash Player 8

Valor devuelto

`flash.filters.DropShadowFilter` - Una instancia DropShadowFilter nueva con las mismas propiedades que la original.

Ejemplo

El ejemplo siguiente crea tres objetos DropShadowFilter y los compara; `filter_1` se crea utilizando el constructor DropShadowFilter; `filter_2` se crea con el valor de `filter_1` y `clonedFilter` se crea clonando `filter_1`. Observe que aunque `filter_2` da como resultado `filter_1`, `clonedFilter`, aunque contiene los mismos valores que `filter_1`, no es igual.

```
import flash.filters.DropShadowFilter;

var filter_1:DropShadowFilter = new DropShadowFilter(15, 45, 0x000000, .8,
  16, 16, 1, 3, false, false, false);
var filter_2:DropShadowFilter = filter_1;
var clonedFilter:DropShadowFilter = filter_1.clone();

trace(filter_1 == filter_2); // true
trace(filter_1 == clonedFilter); // false
```

```
for(var i in filter_1) {
  trace(">> " + i + ": " + filter_1[i]);
  // >> clone: [type Function]
  // >> hideObject: false
  // >> strength: 1
  // >> blurY: 16
  // >> blurX: 16
  // >> knockout: false
  // >> inner: false
  // >> quality: 3
  // >> alpha: 0.8
  // >> color: 0
  // >> angle: 45
  // >> distance: 15
}

for(var i in clonedFilter) {
  trace(">> " + i + ": " + clonedFilter[i]);
  // >> clone: [type Function]
  // >> hideObject: false
  // >> strength: 1
  // >> blurY: 16
  // >> blurX: 16
  // >> knockout: false
  // >> inner: false
  // >> quality: 3
  // >> alpha: 0.8
  // >> color: 0
  // >> angle: 45
  // >> distance: 15
}
```

Para mostrar más detalladamente las relaciones entre `filter_1`, `filter_2` y `clonedFilter`, el ejemplo siguiente modifica la propiedad `knockout` de `filter_1`. Al modificar `knockout`, el método `clone()` crea una nueva instancia basada en los valores de `filter_1` en lugar de utilizarlos en la referencia.

```
import flash.filters.DropShadowFilter;

var filter_1:DropShadowFilter = new DropShadowFilter(15, 45, 0x000000, .8,
  16, 16, 1, 3, false, false, false);
var filter_2:DropShadowFilter = filter_1;
var clonedFilter:DropShadowFilter = filter_1.clone();

trace(filter_1.knockout); // false
trace(filter_2.knockout); // false
trace(clonedFilter.knockout); // false

filter_1.knockout = true;
```

```
trace(filter_1.knockout); // true
trace(filter_2.knockout); // true
trace(clonedFilter.knockout); // false
```

color (propiedad DropShadowFilter.color)

```
public color : Number
```

Color de la sombra. Los valores válidos tienen formato hexadecimal *0x00RRGGBB*. El valor predeterminado es 0x000000.

Disponibilidad: ActionScript 1.0; Flash Player 8

Ejemplo

El ejemplo siguiente cambia la propiedad `color` de un clip de película existente cuando un usuario hace clic en ella.

```
import flash.filters.DropShadowFilter;
var mc:MovieClip = createDropShadowRectangle("DropShadowColor");
mc.onRelease = function() {
  var filter:DropShadowFilter = this.filters[0];
  filter.color = 0xFF0000;
  this.filters = new Array(filter);
}

function createDropShadowRectangle(name:String):MovieClip {
  var art:MovieClip = this.createEmptyMovieClip(name,
  this.getNextHighestDepth());
  var w:Number = 100;
  var h:Number = 100;
  art.beginFill(0x003366);
  art.lineTo(w, 0);
  art.lineTo(w, h);
  art.lineTo(0, h);
  art.lineTo(0, 0);
  art._x = 20;
  art._y = 20;

  var filter:DropShadowFilter = new DropShadowFilter(15, 45, 0x000000, .8,
  16, 16, 1, 3, false, false, false);
  var filterArray:Array = new Array();
  filterArray.push(filter);
  art.filters = filterArray;
  return art;
}
```

distance (propiedad DropShadowFilter.distance)

```
public distance : Number
```

Distancia de desplazamiento de la sombra, en píxeles. El valor predeterminado es 4 (coma flotante).

Disponibilidad: ActionScript 1.0; Flash Player 8

Ejemplo

El ejemplo siguiente cambia la propiedad `distance` de un clip de película existente cuando un usuario hace clic en ella.

```
import flash.filters.DropShadowFilter;
var mc:MovieClip = createDropShadowRectangle("DropShadowDistance");
mc.onRelease = function() {
  var filter:DropShadowFilter = this.filters[0];
  filter.distance = 40;
  this.filters = new Array(filter);
}

function createDropShadowRectangle(name:String):MovieClip {
  var art:MovieClip = this.createEmptyMovieClip(name,
  this.getNextHighestDepth());
  var w:Number = 100;
  var h:Number = 100;
  art.beginFill(0x003366);
  art.lineTo(w, 0);
  art.lineTo(w, h);
  art.lineTo(0, h);
  art.lineTo(0, 0);
  art._x = 20;
  art._y = 20;

  var filter:DropShadowFilter = new DropShadowFilter(15, 45, 0x000000, .8,
  16, 16, 1, 3, false, false, false);
  var filterArray:Array = new Array();
  filterArray.push(filter);
  art.filters = filterArray;
  return art;
}
```

Constructor DropShadowFilter

```
public DropShadowFilter([distance:Number], [angle:Number], [color:Number],
  [alpha:Number], [blurX:Number], [blurY:Number], [strength:Number],
  [quality:Number], [inner:Boolean], [knockout:Boolean],
  [hideObject:Boolean])
```

Crea una instancia DropShadowFilter nueva con los parámetros especificados.

Disponibilidad: ActionScript 1.0; Flash Player 8

Parámetros

distance:Number [opcional] - Distancia de desplazamiento de la sombra, en píxeles. El valor predeterminado es 4 (coma flotante).

angle:Number [opcional] - El ángulo de la sombra, de 0 a 360° (coma flotante). El valor predeterminado es 45.

color:Number [opcional] - Color de la sombra, expresado en formato hexadecimal *0xRRGGBB*. El valor predeterminado es 0x000000.

alpha:Number [opcional] - Valor de transparencia alfa del color de la sombra. Los valores válidos van de 0 a 1. Por ejemplo, 0,25 define un valor de transparencia de 25%. El valor predeterminado es 1.

blurX:Number [opcional] - Cantidad de desenfoque horizontal. Los valores válidos van de 0 a 255 (coma flotante). El valor predeterminado es 4. Los valores que son una potencia de 2 (como 2, 4, 8, 16 y 32) se optimizan para representarse más rápidamente que otros valores.

blurY:Number [opcional] - Cantidad de desenfoque vertical. Los valores válidos van de 0 a 255 (coma flotante). El valor predeterminado es 4. Los valores que son una potencia de 2 (como 2, 4, 8, 16 y 32) se optimizan para representarse más rápidamente que otros valores.

strength:Number [opcional] Intensidad de la impresión o extensión. Cuanto más alto sea el valor, más color se imprimirá y mayor será el contraste entre la sombra y el fondo. Los valores válidos van de 0 a 255. El valor predeterminado es 1.

quality:Number [opcional] - Número de veces que debe aplicarse el filtro. Los valores válidos van de 0 a 15. El valor predeterminado es 1, que equivale a una calidad baja. Un valor de 2 equivale a una calidad media y un valor de 3 a una calidad alta.

inner:Boolean [opcional] Indica si la sombra es una sombra interior. Un valor de `true` especifica una sombra interior. El valor predeterminado es `false`, una sombra exterior (una sombra alrededor de los bordes exteriores del objeto).

knockout:Boolean [opcional] - Aplica un efecto de extractor (`true`), que hace que el relleno del objeto sea transparente y muestra el color de fondo del documento. El valor predeterminado es `false` (sin extractor).

hideObject:Boolean [opcional] - Indica si el objeto está oculto. Un valor de `true` indica que no se dibuja el objeto propiamente dicho; sólo es visible la sombra. El valor predeterminado es `false` (mostrar el objeto).

Ejemplo

El ejemplo siguiente crea una instancia de un nuevo DropShadowFilter y la aplica a una forma plana, rectangular.

```
import flash.filters.DropShadowFilter;
var art:MovieClip = createRectangle(100, 100, 0x003366,
  "gradientGlowFilterExample");
var distance:Number = 20;
var angleInDegrees:Number = 45;
var color:Number = 0x000000;
var alpha:Number = .8;
var blurX:Number = 16;
var blurY:Number = 16;
var strength:Number = 1;
var quality:Number = 3;
var inner:Boolean = false;
var knockout:Boolean = false;
var hideObject:Boolean = false;

var filter:DropShadowFilter = new DropShadowFilter(distance,
                              angleInDegrees,
                              color,
                              alpha,
                              blurX,
                              blurY,
                              strength,
                              quality,
                              inner,
                              knockout,
                              hideObject);
var filterArray:Array = new Array();
filterArray.push(filter);
art.filters = filterArray;

function createRectangle(w:Number, h:Number, bgColor:Number,
  name:String):MovieClip {
  var mc:MovieClip = this.createEmptyMovieClip(name,
  this.getNextHighestDepth());
  mc.beginFill(bgColor);
  mc.lineTo(w, 0);
  mc.lineTo(w, h);
  mc.lineTo(0, h);
  mc.lineTo(0, 0);
  mc._x = 20;
  mc._y = 20;
  return mc;
}
```

hideObject (propiedad DropShadowFilter.hideObject)

```
public hideObject : Boolean
```

Indica si el objeto está oculto. El valor `true` indica que no se dibuja el objeto propiamente dicho; sólo es visible la sombra. El valor predeterminado es `false` (mostrar el objeto).

Disponibilidad: ActionScript 1.0; Flash Player 8

Ejemplo

El ejemplo siguiente cambia la propiedad `hideObject` de un clip de película existente cuando un usuario hace clic en ella.

```
import flash.filters.DropShadowFilter;
var mc:MovieClip = createDropShadowRectangle("DropShadowHideObject");
mc.onRelease = function() {
  var filter:DropShadowFilter = this.filters[0];
  filter.hideObject = true;
  this.filters = new Array(filter);
}

function createDropShadowRectangle(name:String):MovieClip {
  var art:MovieClip = this.createEmptyMovieClip(name,
  this.getNextHighestDepth());
  var w:Number = 100;
  var h:Number = 100;
  art.beginFill(0x003366);
  art.lineTo(w, 0);
  art.lineTo(w, h);
  art.lineTo(0, h);
  art.lineTo(0, 0);
  art._x = 20;
  art._y = 20;

  var filter:DropShadowFilter = new DropShadowFilter(15, 45, 0x000000, .8,
  16, 16, 1, 3, false, false, false);
  var filterArray:Array = new Array();
  filterArray.push(filter);
  art.filters = filterArray;
  return art;
}
```

inner (propiedad DropShadowFilter.inner)

```
public inner : Boolean
```

Indica si la sombra es una sombra interior. El valor `true` indica una sombra interior. El valor predeterminado es `false`, una sombra exterior (una sombra alrededor de los bordes exteriores del objeto).

Disponibilidad: ActionScript 1.0; Flash Player 8

Ejemplo

El ejemplo siguiente cambia la propiedad `inner` de un clip de película existente cuando un usuario hace clic en ella.

```
import flash.filters.DropShadowFilter;
var mc:MovieClip = createDropShadowRectangle("DropShadowInner");
mc.onRelease = function() {
  var filter:DropShadowFilter = this.filters[0];
  filter.inner = true;
  this.filters = new Array(filter);
}

function createDropShadowRectangle(name:String):MovieClip {
  var art:MovieClip = this.createEmptyMovieClip(name,
  this.getNextHighestDepth());
  var w:Number = 100;
  var h:Number = 100;
  art.beginFill(0x003366);
  art.lineTo(w, 0);
  art.lineTo(w, h);
  art.lineTo(0, h);
  art.lineTo(0, 0);
  art._x = 20;
  art._y = 20;

  var filter:DropShadowFilter = new DropShadowFilter(15, 45, 0x000000, .8,
  16, 16, 1, 3, false, false, false);
  var filterArray:Array = new Array();
  filterArray.push(filter);
  art.filters = filterArray;
  return art;
}
```

knockout (propiedad DropShadowFilter.knockout)

```
public knockout : Boolean
```

Aplica un efecto de extractor (`true`), que hace que el relleno del objeto sea transparente y muestra el color de fondo del documento. El valor predeterminado es `false` (sin extractor).

Disponibilidad: ActionScript 1.0; Flash Player 8

Ejemplo

El ejemplo siguiente cambia la propiedad `knockout` de un clip de película existente cuando un usuario hace clic en ella.

```
import flash.filters.DropShadowFilter;
var mc:MovieClip = createDropShadowRectangle("DropShadowKnockout");
mc.onRelease = function() {
  var filter:DropShadowFilter = this.filters[0];
  filter.knockout = true;
  this.filters = new Array(filter);
}

function createDropShadowRectangle(name:String):MovieClip {
  var art:MovieClip = this.createEmptyMovieClip(name,
  this.getNextHighestDepth());
  var w:Number = 100;
  var h:Number = 100;
  art.beginFill(0x003366);
  art.lineTo(w, 0);
  art.lineTo(w, h);
  art.lineTo(0, h);
  art.lineTo(0, 0);
  art._x = 20;
  art._y = 20;

  var filter:DropShadowFilter = new DropShadowFilter(15, 45, 0x000000, .8,
  16, 16, 1, 3, false, false, false);
  var filterArray:Array = new Array();
  filterArray.push(filter);
  art.filters = filterArray;
  return art;
}
```

quality (propiedad DropShadowFilter.quality)

```
public quality : Number
```

Número de veces que debe aplicarse el filtro. Los valores válidos van de 0 a 15. El valor predeterminado es 1, que equivale a una calidad baja. Un valor 2 equivale a calidad media y un valor 3 a alta. Los filtros con valores más bajos se representan con mayor rapidez.

Para la mayoría de las aplicaciones, es suficiente un valor `quality` de 1, 2 o 3. Aunque puede utilizar valores numéricos hasta 15 para conseguir efectos distintos, los valores más altos se representan más lentamente. En lugar de aumentar el valor de `quality`, habitualmente se consigue un efecto similar, y con una representación más rápida, sencillamente aumentando los valores de `blurX` y `blurY`.

Disponibilidad: ActionScript 1.0; Flash Player 8

Ejemplo

El ejemplo siguiente cambia la propiedad `quality` de un clip de película existente cuando un usuario hace clic en ella.

```
import flash.filters.DropShadowFilter;
var mc:MovieClip = createDropShadowRectangle("DropShadowQuality");
mc.onRelease = function() {
  var filter:DropShadowFilter = this.filters[0];
  filter.quality = 0;
  this.filters = new Array(filter);
}

function createDropShadowRectangle(name:String):MovieClip {
  var art:MovieClip = this.createEmptyMovieClip(name,
  this.getNextHighestDepth());
  var w:Number = 100;
  var h:Number = 100;
  art.beginFill(0x003366);
  art.lineTo(w, 0);
  art.lineTo(w, h);
  art.lineTo(0, h);
  art.lineTo(0, 0);
  art._x = 20;
  art._y = 20;

  var filter:DropShadowFilter = new DropShadowFilter(15, 45, 0x000000, .8,
  16, 16, 1, 3, false, false, false);
  var filterArray:Array = new Array();
  filterArray.push(filter);
  art.filters = filterArray;
  return art;
}
```

strength (propiedad DropShadowFilter.strength)

```
public strength : Number
```

Intensidad de la impresión o extensión. Cuanto más alto sea el valor, más color se imprimirá y mayor será el contraste entre la sombra y el fondo. Los valores válidos van de 0 a 255. El valor predeterminado es 1.

Disponibilidad: ActionScript 1.0; Flash Player 8

Ejemplo

El ejemplo siguiente cambia la propiedad `strength` de un clip de película existente cuando un usuario hace clic en ella.

```
import flash.filters.DropShadowFilter;
var mc:MovieClip = createDropShadowRectangle("DropShadowStrength");
mc.onRelease = function() {
  var filter:DropShadowFilter = this.filters[0];
  filter.strength = .6;
  this.filters = new Array(filter);
}

function createDropShadowRectangle(name:String):MovieClip {
  var art:MovieClip = this.createEmptyMovieClip(name,
  this.getNextHighestDepth());
  var w:Number = 100;
  var h:Number = 100;
  art.beginFill(0x003366);
  art.lineTo(w, 0);
  art.lineTo(w, h);
  art.lineTo(0, h);
  art.lineTo(0, 0);
  art._x = 20;
  art._y = 20;

  var filter:DropShadowFilter = new DropShadowFilter(15, 45, 0x000000, .8,
  16, 16, 1, 3, false, false, false);
  var filterArray:Array = new Array();
  filterArray.push(filter);
  art.filters = filterArray;
  return art;
}
```

Error

```
Object
  |
  +-Error
```

`public class` ***Error***
`extends Object`

Contiene información sobre un error que se ha producido en un script. Puede crear un objeto Error utilizando la función constructora `Error`. Normalmente emitirá (`throw`) un nuevo objeto Error desde dentro de un bloque de código `try` que posteriormente será detectado por un bloque de código `catch` o `finally`.

También puede crear una subclase de la clase Error y emitir instancias de dicha subclase.

Disponibilidad: ActionScript 1.0; Flash Player 7

Resumen de propiedades

Modificadores	Propiedad	Descripción
	`message:String`	Contiene el mensaje asociado al objeto Error.
	`name:String`	Contiene el nombre del objeto Error.

Propiedades heredadas de la clase Object

`constructor (propiedad Object.constructor), __proto__ (propiedad Object.__proto__), prototype (propiedad Object.prototype), __resolve (propiedad Object.__resolve)`

Resumen de constructores

Firma	Descripción
`Error([message:String])`	Crea un nuevo objeto Error.

Resumen de métodos

Modificadores	Firma	Descripción
	`toString() : String`	Devuelve la cadena "Error" de manera predeterminada o el valor contenido en Error.message, en el caso de que se haya definido.

addProperty (método Object.addProperty), hasOwnProperty (método Object.hasOwnProperty), isPropertyEnumerable (método Object.isPropertyEnumerable), isPrototypeOf (método Object.isPrototypeOf), registerClass (método Object.registerClass), toString (método Object.toString), unwatch (método Object.unwatch), valueOf (método Object.valueOf), watch (método Object.watch)

Constructor Error

```
public Error([message:String])
```

Crea un nuevo objeto Error. Si se especifica *message*, su valor se asigna a la propiedad `Error.message` del objeto.

Disponibilidad: ActionScript 1.0; Flash Player 7

Parámetros

message`:String` [opcional] - Una cadena asociada al objeto Error.

Ejemplo

En el ejemplo siguiente, una función emite un error (con un mensaje especificado) si las dos cadenas que se pasan no son idénticas:

```
function compareStrings(str1_str:String, str2_str:String):Void {
  if (str1_str != str2_str) {
  throw new Error("Strings do not match.");
  }
}
try {
  compareStrings("Dog", "dog");
  // output: Strings do not match.
} catch (e_err:Error) {
  trace(e_err.toString());
}
```

Véase también

Sentencia throw, Sentencia try..catch..finally

message (propiedad Error.message)

```
public message : String
```

Contiene el mensaje asociado al objeto Error. De manera predeterminada, el valor de esta propiedad es "Error". Puede especificar una propiedad message cuando cree un objeto Error pasando la cadena de error a la función constructora Error.

Disponibilidad: ActionScript 1.0; Flash Player 7

Ejemplo

En el ejemplo siguiente, una función emite un mensaje especificado según los parámetros introducidos en theNum. Si pueden dividirse dos números, se mostrarán SUCCESS y el número. Se mostrarán errores específicos sin intenta dividir entre 0 o sólo introduce un parámetro:

```
function divideNum(num1:Number, num2:Number):Number {
  if (isNaN(num1) || isNaN(num2)) {
  throw new Error("divideNum function requires two numeric parameters.");
  } else if (num2 == 0) {
  throw new Error("cannot divide by zero.");
  }
  return num1/num2;
}
try {
  var theNum:Number = divideNum(1, 0);
  trace("SUCCESS! "+theNum);
} catch (e_err:Error) {
  trace("ERROR! "+e_err.message);
  trace("\t"+e_err.name);
}
```

Si comprueba este código ActionScript sin ninguna modificación en los números que divide, aparecerá un error en el panel Salida porque está intentando dividir entre 0.

Véase también

Sentencia throw, Sentencia try..catch..finally

name (propiedad Error.name)

```
public name : String
```

Contiene el nombre del objeto Error. De manera predeterminada, el valor de esta propiedad es `"Error"`.

Disponibilidad: ActionScript 1.0; Flash Player 7

Ejemplo

En el ejemplo siguiente, una función emite un error especificado según los dos números que se intentan dividir. Añada el siguiente código ActionScript al fotograma 1 de la línea de tiempo:

```
function divideNumber(numerator:Number, denominator:Number):Number {
  if (isNaN(numerator) || isNaN(denominator)) {
  throw new Error("divideNum function requires two numeric parameters.");
  } else if (denominator == 0) {
  throw new DivideByZeroError();
  }
  return numerator/denominator;
}
try {
  var theNum:Number = divideNumber(1, 0);
  trace("SUCCESS! "+theNum);
  // output: DivideByZeroError -> Unable to divide by zero.
} catch (e_err:DivideByZeroError) {
  // divide by zero error occurred
  trace(e_err.name+" -> "+e_err.toString());
} catch (e_err:Error) {
  // generic error occurred
  trace(e_err.name+" -> "+e_err.toString());
}
```

Para añadir un error personalizado, añada el código siguiente a un archivo .AS llamado DivideByZeroError.as y guarde el archivo de clase en el mismo directorio que el documento FLA.

```
class DivideByZeroError extends Error {
  var name:String = "DivideByZeroError";
  var message:String = "Unable to divide by zero.";
}
```

Véase también

`Sentencia throw`, `Sentencia try..catch..finally`

toString (método Error.toString)

```
public toString() : String
```

Devuelve la cadena "Error" de manera predeterminada o el valor contenido en Error.message, en el caso de que se haya definido.

Disponibilidad: ActionScript 1.0; Flash Player 7

Valor devuelto

`String` - Una cadena.

Ejemplo

En el ejemplo siguiente, una función emite un error (con un mensaje especificado) si las dos cadenas que se pasan no son idénticas:

```
function compareStrings(str1_str:String, str2_str:String):Void {
if (str1_str != str2_str) {
  throw new Error("Strings do not match.");
  }
}
try {
  compareStrings("Dog", "dog");
  // output: Strings do not match.
} catch (e_err:Error) {
  trace(e_err.toString());
}
```

Véase también

`message (propiedad Error.message)`, `Sentencia throw`, `Sentencia try..catch..finally`

ExternalInterface (flash.external.ExternalInterface)

```
Object
  |
  +-flash.external.ExternalInterface
```

```
public class ExternalInterface
extends Object
```

La clase ExternalInterface es la API externa, una nueva interfaz de programación de aplicaciones que permite la comunicación directa entre ActionScript y el contenedor de Flash Player; por ejemplo, una página HTML con JavaScript o una aplicación de escritorio que incorpore Flash Player.

La funcionalidad de ExternalInterface es similar a la de los métodos `fscommand()`, `CallFrame()` y `CallLabel()`, aunque es más flexible y se puede aplicar de forma más general. Se recomienda el uso de ExternalInterface para la comunicación entre JavaScript y ActionScript.

Desde ActionScript, puede llamar a cualquier función JavaScript de la página HTML, pasando cualquier número de argumentos de cualquier tipo de datos y recibir el valor devuelto.

Desde JavaScript en la página HTML, puede llamar a una función ActionScript en Flash Player. La función ActionScript puede devolver un valor y JavaScript la recibe inmediatamente como el valor devuelto por la llamada.

ExternalInterface se admite en las siguientes combinaciones de navegador y sistema operativo:

Navegador	**Sistema operativo**	
Internet Explorer 5.0 y superior	Windows	
Netscape 8.0 y superior	Windows	Macintosh
Mozilla 1.7.5 y superior	Windows	Macintosh
Firefox 1.0 y superior	Windows	Macintosh
Safari 1.3 y superior		Macintosh

ExternalInterface necesita que el navegador Web del usuario admita ActiveX o la API NPRuntime que muestran algunos navegadores para creación de scripts de plug-in. Véase http://www.mozilla.org/projects/plugins/npruntime.html.

Disponibilidad: ActionScript 1.0; Flash Player 8

Resumen de propiedades

Modificadores	Propiedad	Descripción
`static`	`available:Boolean` [read-only]	Indica si este reproductor se encuentra en un contenedor que ofrece una interfaz externa.

Propiedades heredadas de la clase Object

`constructor (propiedad Object.constructor), __proto__ (propiedad Object.__proto__), prototype (propiedad Object.prototype), __resolve (propiedad Object.__resolve)`

Resumen de métodos

Modificadores	Firma	Descripción
`static`	`addCallback(methodName:String, instance:Object, method:Function) : Boolean`	Registra un método de ActionScript como método que puede llamarse desde el contenedor.
`static`	`call(methodName:String, [parameter1:Object]) : Object`	Llama a una función expuesta por el contenedor de Flash Player, pasando 0 o más argumentos.

Métodos heredados de la clase Object

`addProperty (método Object.addProperty), hasOwnProperty (método Object.hasOwnProperty), isPropertyEnumerable (método Object.isPropertyEnumerable), isPrototypeOf (método Object.isPrototypeOf), registerClass (método Object.registerClass), toString (método Object.toString), unwatch (método Object.unwatch), valueOf (método Object.valueOf), watch (método Object.watch)`

addCallback (método ExternalInterface.addCallback)

```
public static addCallback(methodName:String, instance:Object,
  method:Function) : Boolean
```

Registra un método de ActionScript como método que puede llamarse desde el contenedor. Después de invocar correctamente a `addCallBack()`, el código JavaScript o ActiveX del contenedor puede llamar a la función registrada en Flash Player.

Disponibilidad: ActionScript 1.0; Flash Player 8

Parámetros

methodName:String - El nombre por el que el código JavaScript puede llamar a la función de ActionScript. No es necesario que este nombre coincida con el nombre real del método ActionScript.

instance:Object - El objeto que da como resultado this en el método. No tiene por qué ser el objeto en el que se encuentra el método. Puede especificar cualquier objeto (o null) aquí.

method:Function - El método ActionScript que se llamará desde JavaScript.

Valor devuelto

Boolean - Si la llamada tiene éxito, devuelve true. Será false si ha fallado porque la instancia no estaba disponible, se ha encontrado una restricción de seguridad, no existía el objeto de la función, se ha producido una recursión o algo similar.

Un valor false también puede significar que el entorno contenedor pertenece a un entorno limitado de seguridad al que no tiene acceso el código que realiza la llamada. Puede solucionar este problema definiendo un valor adecuado para la etiqueta allowScriptAccess OBJECT o EMBED en el HTML del entorno contenedor.

Ejemplo

El ejemplo siguiente registra la función goToMacromedia() como que puede llamarse desde el contenedor con el nombre goHome.

```
import flash.external.*;

var methodName:String = "goHome";
var instance:Object = null;
var method:Function = goToMacromedia;
var wasSuccessful:Boolean = ExternalInterface.addCallback(methodName,
  instance, method);

var txtField:TextField = this.createTextField("txtField",
  this.getNextHighestDepth(), 0, 0, 200, 50);
txtField.border = true;
txtField.text = wasSuccessful.toString();

function goToMacromedia() {
  txtField.text = "http://www.macromedia.com";
  getURL("http://www.macromedia.com", "_self");
}
```

Para que el ejemplo anterior funcione correctamente, debe copiar y pegar el código siguiente en la página HTML. Este código utiliza el atributo `id` de la etiqueta `OBJECT` y el atributo `name` de la etiqueta `EMBED` para tener el valor `externalInterfaceExample`. La función `thisMovie` devuelve la sintaxis adecuada según el navegador, ya que Internet Explorer y Netscape se refieren a la película de forma diferente. A menos que la página HTML se aloje en un servidor, su navegador puede avisarle con una advertencia de seguridad.

Nota: Evite utilizar otros métodos para acceder al objeto plug-in, como `document.getElementById("pluginName")` o `document.all.pluginName`, porque estos otros métodos no funcionan de la misma forma en todos los navegadores.

```
<form>
  <input type="button" onclick="callExternalInterface()" value="Call
  ExternalInterface" />
</form>
<script>
function callExternalInterface() {
  thisMovie("externalInterfaceExample").goHome();
}

function thisMovie(movieName) {
  if (navigator.appName.indexOf("Microsoft") != -1) {
    return window[movieName]
  }
  else {
    return document[movieName]
  }
}
</script>
```

available (propiedad ExternalInterface.available)

```
public static available : Boolean [read-only]
```

Indica si este reproductor se encuentra en un contenedor que ofrece una interfaz externa. Si la interfaz externa está disponible, esta propiedad es `true`; en caso contrario es `false`.

Disponibilidad: ActionScript 1.0; Flash Player 8

Ejemplo

El ejemplo siguiente utiliza `ExternalInterface.available` para determinar si el reproductor está en un contenedor que ofrece una interfaz externa.

```
import flash.external.*;

var isAvailable:Boolean = ExternalInterface.available;
trace(isAvailable);
```

call (método ExternalInterface.call)

```
public static call(methodName:String, [parameter1:Object]) : Object
```

Llama a una función expuesta por el contenedor de Flash Player, pasando 0 o más argumentos. Si la función deseada no está disponible, la llamada devuelve `null`; en caso contrario, devuelve el valor suministrado por la función. No se permite la recursión; una llamada recursiva produce una respuesta `null`.

Si el contenedor es una página HTML, este método invoca una función JavaScript en un elemento `<script>`.

Si el contenedor es algún otro contenedor ActiveX, este método difunde un evento con el nombre especificado y el contenedor procesa el evento.

Si el contenedor aloja el plug-in de Netscape, puede escribir un soporte personalizado para la nueva interfaz NPRuntime o incorporar un control HTML y Flash Player en el control HTML. Si incorpora un control HTML, puede comunicarse con Flash Player a través de una interfaz de JavaScript que se comunica con la aplicación contenedora nativa.

Disponibilidad: ActionScript 1.0; Flash Player 8

Parámetros

`methodName``:String` - El nombre de la función que se va a llamar en el contenedor. Si la función acepta parámetros, deberán aparecer después del parámetro `methodName`.

`parameter1``:Object` [opcional] - Los parámetros que se pasan a la función. Puede especificar varios parámetros, separados por comas, o no especificar ningún parámetro. Los parámetros pueden ser cualquier tipo de datos ActionScript. Cuando la llamada se dirige a una función de JavaScript, los tipos de ActionScript se reúnen automáticamente en tipos de JavaScript. Cuando la llamada se realiza a algún otro contenedor ActiveX, los parámetros se codifican en el mensaje de solicitud.

Valor devuelto

`Object` - La respuesta recibida del contenedor. Si la llamada falla (por ejemplo, porque no existe esa función en el contenedor, la interfaz no está disponible, se produce una recursión o hay un problema de seguridad), el valor devuelto es `null`.

Ejemplo

El ejemplo siguiente llama a la función de JavaScript `sayHello()` en la página HTML que contiene el archivo SWF. La llamada se realiza mediante el método `ExternalInterface.call()`.

```
import flash.external.*;

var greeting:String;
var btn:MovieClip = createButton(100, 30, 0xCCCCCC);
btn.onPress = function() {
  greeting = String(ExternalInterface.call("sayHello", "browser"));
  this.mcTxt.text = greeting; // >> Hi Flash.
}

function createButton(width:Number, height:Number, color:Number):MovieClip
  {
  var depth:Number = this.getNextHighestDepth();
  var mc:MovieClip = this.createEmptyMovieClip("mc_" + depth, depth);
  var mcFmt:TextFormat;

  mc.beginFill(color);
  mc.lineTo(0, height);
  mc.lineTo(width, height);
  mc.lineTo(width, 0);
  mc.lineTo(0, 0);

  mcFmt = new TextFormat();
  mcFmt.align = "center";
  mcFmt.bold = true;

  mc.createTextField("mcTxt", depth, 0, 0, width, height);
  mc.mcTxt.text = "Call JS Function";
  mc.mcTxt.setTextFormat(mcFmt);

  return mc;
}
```

Para que el ejemplo anterior funcione correctamente, debe copiar y pegar el código siguiente en la página HTML. A menos que la página HTML se aloje en un servidor, su navegador puede avisarle con una advertencia de seguridad.

```
<script>
  function sayHello(name) {
    alert(">> Hello " + name + ".");
    return ">> Hi Flash.";
  }
</script>
```

FileReference (flash.net.FileReference)

```
Object
  |
  +-flash.net.FileReference
```

`public class` ***FileReference***
`extends Object`

La clase FileReference proporciona un medio para cargar y descargar archivos entre el equipo de un usuario y un servidor. El sistema operativo pide al usuario mediante un cuadro de diálogo que seleccione un archivo para cargar o una ubicación para descargar. Cada objeto FileReference hace referencia a un archivo único en el disco duro del usuario y tiene propiedades que contienen información sobre tamaño, tipo, nombre, fecha de creación, fecha de modificación y tipo de creador (sólo Macintosh) del archivo.

Las instancias FileReference se crean de dos formas:

- Cuando utiliza el operador `new` con el constructor FileReference: `var myFileReference = new FileReference();`
- Cuando se llama a `FileReferenceList.browse()`, que crea una matriz de objetos FileReference.

Durante una operación de carga, todas las propiedades de un objeto FileReference se llenan con llamadas a `FileReference.browse()` o `FileReferenceList.browse()`. Durante una operación de descarga, la propiedad `name` se llena cuando se ha llamado a `onSelect`; las demás propiedades se llenan cuando se ha llamado a `onComplete`.

El método `browse()` abre un cuadro de diálogo del sistema operativo que pide al usuario que seleccione un archivo local para la carga. El método `FileReference.browse()` permite al usuario seleccionar un solo archivo; el método `FileReferenceList.browse()` permite al usuario seleccionar varios archivos. Después de llamar con éxito a `browse()`, llame al método `FileReference.upload()` para cargar un archivo cada vez. El método `FileReference.download()` pide al usuario que seleccione una ubicación para guardar el archivo e inicia la descarga desde una URL remota.

Las clases FileReference y FileReferenceList no permiten definir la ubicación predeterminada del archivo para el cuadro de diálogo generado por las llamadas `browse()` y `download()`. La ubicación predeterminada que se muestra en el cuadro de diálogo es la última carpeta a la que se ha accedido, si puede determinarse esa ubicación, o bien el escritorio. Las clases no permiten leer ni escribir en el archivo transferido. No permiten al archivo SWF que inició la carga o descarga acceder al archivo cargado o descargado, ni a la ubicación del archivo en el disco del usuario.

Las clases FileReference y FileReferenceList no proporcionan tampoco métodos para la autenticación. Cuando los servidores requieren autenticación, puede descargar archivos con el plug-in del navegador de Flash Player, pero no se pueden realizar cargas (en todos los reproductores) ni descargas (en el reproductor externo o autónomo). Utilice detectores de eventos FileReference para comprobar si las operaciones se han realizado correctamente y para solucionar los errores.

Para las operaciones de carga y descarga, un archivo SWF sólo puede acceder a archivos de su propio dominio, incluidos los dominios especificados en un archivo de política de varios dominios. Si el archivo SWF que inicia la carga o descarga no procede del mismo dominio que el servidor de archivos, debe situar un archivo de política en el servidor de archivos.

Mientras se ejecutan las llamadas a los métodos `FileReference.browse()`, `FileReferenceList.browse()` o `FileReference.download()`, la reproducción del archivo SWF se interrumpe en las siguientes plataformas: el plug-in de Flash Player para Mac OS X, Flash Player externo para Macintosh y el reproductor autónomo para Mac OS X 10.1 y versiones anteriores. El archivo SWF sigue ejecutándose en todos los reproductores para Windows y en el reproductor autónomo para Macintosh en Mac OS X 10.2 y versiones posteriores.

Disponibilidad: ActionScript 1.0; Flash Player 8

Ejemplo

El ejemplo siguiente crea un objeto FileReference que pide al usuario que seleccione una imagen o archivo de texto para descargarlo. También detecta posibles eventos.

```
import flash.net.FileReference;

var allTypes:Array = new Array();
var imageTypes:Object = new Object();
imageTypes.description = "Images (*.jpg, *.jpeg, *.gif, *.png)";
imageTypes.extension = "*.jpg; *.jpeg; *.gif; *.png";
allTypes.push(imageTypes);

var textTypes:Object = new Object();
textTypes.description = "Text Files (*.txt, *.rtf)";
textTypes.extension = "*.txt;*.rtf";
allTypes.push(textTypes);

var listener:Object = new Object();

listener.onSelect = function(file:FileReference):Void {
  trace("onSelect: " + file.name);
  if(!file.upload("http://www.yourdomain.com/
  yourUploadHandlerScript.cfm")) {
    trace("Upload dialog failed to open.");
```

```
    }
}

listener.onCancel = function(file:FileReference):Void {
    trace("onCancel");
}

listener.onOpen = function(file:FileReference):Void {
    trace("onOpen: " + file.name);
}

listener.onProgress = function(file:FileReference, bytesLoaded:Number,
    bytesTotal:Number):Void {
    trace("onProgress with bytesLoaded: " + bytesLoaded + " bytesTotal: " +
    bytesTotal);
}

listener.onComplete = function(file:FileReference):Void {
    trace("onComplete: " + file.name);
}

listener.onHTTPError = function(file:FileReference):Void {
    trace("onHTTPError: " + file.name);
}

listener.onIOError = function(file:FileReference):Void {
    trace("onIOError: " + file.name);
}

listener.onSecurityError = function(file:FileReference,
    errorString:String):Void {
    trace("onSecurityError: " + file.name + " errorString: " + errorString);
}

var fileRef:FileReference = new FileReference();
fileRef.addListener(listener);
fileRef.browse(allTypes);
```

Véase también

FileReferenceList (flash.net.FileReferenceList)

Resumen de propiedades

Modificadores	Propiedad	Descripción
	`creationDate:Date` [read-only]	La fecha de creación del archivo en el disco local.
	`creator:String` [read-only]	El tipo de creador del archivo en Macintosh.
	`modificationDate:Date` [read-only]	La última fecha de modificación del archivo en el disco local.
	`name:String` [read-only]	El nombre del archivo en el disco local.
	`size:Number` [read-only]	El tamaño del archivo en el disco local, expresado en bytes.
	`type:String` [read-only]	El tipo de archivo.

Propiedades heredadas de la clase Object

`constructor (propiedad Object.constructor), __proto__ (propiedad Object.__proto__), prototype (propiedad Object.prototype), __resolve (propiedad Object.__resolve)`

Resumen de eventos

Evento	Descripción
`onCancel = function(fileRef:FileReference) {}`	Se invoca cuando el usuario cierra el cuadro de diálogo de búsqueda de archivos.
`onComplete = function(fileRef:FileReference) {}`	Se invoca cuando la operación de carga o descarga de archivos se realiza correctamente.
`onHTTPError = function(fileRef:FileReference, httpError:Number) {}`	Se invoca cuando falla una carga debido a un error HTTP.

Evento	Descripción
`onIOError = function(fileRef:FileReference) {}`	Se invoca cuando se produce un error de entrada/salida.
`onOpen = function(fileRef:FileReference) {}`	Se invoca cuando se inicia una operación de carga o descarga.
`onProgress = function(fileRef:FileReference, bytesLoaded:Number, bytesTotal:Number) {}`	Se invoca periódicamente durante la operación de carga o descarga de archivos.
`onSecurityError = function(fileRef:FileReference, errorString:String) {}`	Se invoca cuando falla la carga o descarga debido a un error de seguridad.
`onSelect = function(fileRef:FileReference) {}`	Se invoca cuando el usuario selecciona un archivo para la carga o descarga en el cuadro de diálogo de búsqueda de archivos.

Resumen de constructores

Firma	Descripción
`FileReference()`	Crea un nuevo objeto FileReference.

Resumen de métodos

Modificadores	Firma	Descripción
	`addListener(listener:Object) : Void`	Registra un objeto para recibir una notificación cuando se invoca un detector de eventos FileReference.
	`browse([typelist:Array]) : Boolean`	Muestra un cuadro de diálogo de búsqueda de archivos en el que el usuario puede seleccionar un archivo local para la carga.

Modificadores	Firma	Descripción
	`cancel() : Void`	Cancela la operación de carga o descarga en curso en este objeto FileReference.
	`download(url:String, [defaultFileName:String]) : Boolean`	Invoca un cuadro de diálogo en el que el usuario puede descargar un archivo desde un servidor remoto.
	`removeListener(listener:Object) : Boolean`	Quita un objeto de la lista de objetos que reciben mensajes de notificación de eventos.
	`upload(url:String) : Boolean`	Inicia la carga de un archivo seleccionado por un usuario en un servidor remoto.

Métodos heredados de la clase Object

`addProperty (método Object.addProperty), hasOwnProperty (método Object.hasOwnProperty), isPropertyEnumerable (método Object.isPropertyEnumerable), isPrototypeOf (método Object.isPrototypeOf), registerClass (método Object.registerClass), toString (método Object.toString), unwatch (método Object.unwatch), valueOf (método Object.valueOf), watch (método Object.watch)`

addListener (método FileReference.addListener)

`public addListener(listener:Object) : Void`

Registra un objeto para recibir una notificación cuando se invoca un detector de eventos FileReference.

Disponibilidad: ActionScript 1.0; Flash Player 8

Parámetros

`listener``:Object` - Un objeto que detecta una notificación callback de los detectores de eventos FileReference.

Ejemplo

El ejemplo siguiente añade un detector a una instancia de FileReference.

```
import flash.net.FileReference;

var listener:Object = new Object();

listener.onProgress = function(file:FileReference, bytesLoaded:Number,
  bytesTotal:Number):Void {
  trace("onProgress with bytesLoaded: " + bytesLoaded + " bytesTotal: " +
  bytesTotal);
}

listener.onComplete = function(file:FileReference):Void {
  trace("onComplete: " + file.name);
}

var fileRef:FileReference = new FileReference();
fileRef.addListener(listener);
var url:String = "http://www.macromedia.com/platform/whitepapers/
  platform_overview.pdf";
fileRef.download(url, "FlashPlatform.pdf");
```

browse (método FileReference.browse)

```
public browse([typelist:Array]) : Boolean
```

Muestra un cuadro de diálogo de búsqueda de archivos en el que el usuario puede seleccionar un archivo local para la carga. El cuadro de diálogo depende del sistema operativo del usuario. Cuando se llama a este método y el usuario selecciona correctamente un archivo, las propiedades de este objeto FileReference se llenan con las propiedades de dicho archivo. Posteriormente, cada vez que se llama a `FileReference.browse()`, las propiedades del objeto FileReference se restablecen según el archivo seleccionado por el usuario en el cuadro de diálogo.

Sólo se puede realizar una sesión `browse()` o `download()` cada vez (porque no se puede mostrar más de un cuadro de diálogo de forma simultánea).

Puede pasar una matriz de tipos de archivos para determinar qué archivos muestra el cuadro de diálogo.

Disponibilidad: ActionScript 1.0; Flash Player 8

Parámetros

typelist:Array [opcional] - Una matriz de tipos de archivo para filtrar los archivos que se muestran en el cuadro de diálogo. Si omite este parámetro, se mostrarán todos los archivos. Si incluye este parámetro, la matriz deberá contener uno o más elementos entre llaves { }. Puede utilizar uno de estos dos formatos para la matriz:

- Una lista de descripciones de tipos de archivo seguida de las extensiones de archivo de Windows solamente. Cada elemento de la matriz debe contener una cadena que describa el tipo de archivo y una lista de extensiones de archivos de Windows delimitadas por punto y coma, con un comodín (*) delante de cada extensión La sintaxis del elemento es la siguiente: `[{description: "string describing the first set of file types", extension: "semicolon-delimited list of file extensions"}]` Ejemplo: `[{description: "Images", extension: "*.jpg;*.gif;*.png"}, {description: "Flash Movies", extension: "*.swf"}, {description: "Documents", extension: "*.doc;*.pdf"}]`
- Una lista de descripciones de tipos de archivo seguida de sus extensiones de archivo de Windows y sus tipos de archivo de Macintosh. Cada elemento de la matriz debe contener una cadena que describa el tipo de archivo; una lista de extensiones de archivos de Windows delimitadas por punto y coma, con un comodín (*) delante de cada extensión y una lista de extensiones de archivos de Macintosh delimitadas por punto y coma, con un comodín (*) delante de cada extensión. La sintaxis del elemento es la siguiente:
  `[{description: "string describing the first set of file types", extension: "semicolon-delimited list of Windows file extensions", macType: "semicolon-delimited list of Macintosh file types"}]`
  Ejemplo:
  `[{description: "Image files", extension: "*.jpg;*.gif;*.png", macType: "JPEG;jp2_;GIFF"}, {description: "Flash Movies", extension: "*.swf", macType: "SWFL"}]`

Los dos formatos no son intercambiables en una sola llamada a `browse()`. Deberá utilizar uno u otro.

La lista de extensiones se emplea para filtrar los archivos en Windows, de acuerdo con el archivo seleccionado por el usuario. No aparece en el cuadro de diálogo. Para que los usuarios vean los tipos de archivos, debe mostrar los tipos de archivo en la cadena de descripción, así como la lista de extensiones. La cadena de descripción aparece en el cuadro de diálogo en Windows. (No se utiliza en Macintosh.) En Macintosh, si se suministra una lista de tipos de archivos de Macintosh, se utiliza ésta para filtrar los archivos. En caso contrario, se utiliza la lista de extensiones de Windows.

Valor devuelto

`Boolean` - Devuelve `true` si los parámetros son válidos y aparece el cuadro de diálogo de búsqueda de archivos. Devuelve `false` si el cuadro de diálogo no se ha mostrado, si hay otra sesión de búsqueda de archivos en curso o si ha utilizado el parámetro `typelist` pero no ha proporcionado una descripción o extensión en ningún elemento de la matriz.

Eventos

onCancel

Se invoca cuando el usuario cancela el cuadro de diálogo haciendo clic en Cancelar o cerrándolo.

onSelect

Se invoca cuando el usuario selecciona en el cuadro de diálogo un elemento para la carga.

Ejemplo

El ejemplo siguiente muestra un cuadro de diálogo en el que el usuario puede seleccionar un archivo de imagen para cargarlo.

```
import flash.net.FileReference;

var listener:Object = new Object();
listener.onSelect = function(file:FileReference):Void {
  trace("Opened " + file.name);
}

listener.onCancel = function(file:FileReference):Void {
  trace("User cancelled");
}

var fileRef:FileReference = new FileReference();
fileRef.addListener(listener);
fileRef.browse();
```

Véase también

onSelect (detector de eventos FileReferenceList.onSelect), onCancel (detector de eventos FileReference.onCancel), download (método FileReference.download), browse (método FileReferenceList.browse)

cancel (método FileReference.cancel)

```
public cancel() : Void
```

Cancela la operación de carga o descarga en curso en este objeto FileReference.

Disponibilidad: ActionScript 1.0; Flash Player 8

Ejemplo

El ejemplo siguiente descarga aproximadamente la mitad del archivo solicitado y, a continuación, cancela la descarga. Obviamente no se trata de una sintaxis típica. Posiblemente utilice este método con más frecuencia para permitir a los usuarios hacer clic en Cancelar en el cuadro de diálogo de estado de descarga.

```
import flash.net.FileReference;

var listener:Object = new Object();

listener.onProgress = function(file:FileReference, bytesLoaded:Number,
  bytesTotal:Number):Void {
  trace("onProgress with bytesLoaded: " + bytesLoaded + " bytesTotal: " +
  bytesTotal);
  if(bytesLoaded >= (bytesTotal / 2)) {
    file.cancel();
  }
}

var fileRef:FileReference = new FileReference();
fileRef.addListener(listener);
var url:String = "http://www.macromedia.com/platform/whitepapers/
  platform_overview.pdf";
fileRef.download(url, "FlashPlatform.pdf");
```

creationDate (propiedad FileReference.creationDate)

```
public creationDate : Date [read-only]
```

La fecha de creación del archivo en el disco local. Si aún no se ha llenado el objeto FileReference, una llamada para obtener el valor de esta propiedad devolverá `null`.

Disponibilidad: ActionScript 1.0; Flash Player 8

Ejemplo

El ejemplo siguiente recupera la fecha de creación de un archivo seleccionado por el usuario.

```
import flash.net.FileReference;

var listener:Object = new Object();
listener.onSelect = function(file:FileReference):Void {
  trace("creationDate: " + file.creationDate);
}

var fileRef:FileReference = new FileReference();
fileRef.addListener(listener);
fileRef.browse();
```

Véase también

browse (método FileReference.browse)

creator (propiedad FileReference.creator)

```
public creator : String [read-only]
```

El tipo de creador del archivo en Macintosh. En Windows esta propiedad es `null`. Si aún no se ha llenado el objeto FileReference, una llamada para obtener el valor de esta propiedad devolverá `null`.

Disponibilidad: ActionScript 1.0; Flash Player 8

Ejemplo

El ejemplo siguiente recupera el tipo de creador de Macintosh o un archivo seleccionado por el usuario.

```
import flash.net.FileReference;

var listener:Object = new Object();
listener.onSelect = function(file:FileReference):Void {
  trace("creator: " + file.creator);
}

var fileRef:FileReference = new FileReference();
fileRef.addListener(listener);
fileRef.browse();
```

Véase también

browse (método FileReference.browse)

download (método FileReference.download)

```
public download(url:String, [defaultFileName:String]) : Boolean
```

Invoca un cuadro de diálogo en el que el usuario puede descargar un archivo desde un servidor remoto. Flash Player puede descargar archivos de hasta 100 MB.

Este método abre en primer lugar un cuadro de diálogo del sistema operativo que pide al usuario que introduzca un nombre de archivo y seleccione una ubicación en el equipo local para guardar el archivo. Cuando el usuario selecciona una ubicación y confirma la operación de descarga (por ejemplo, haciendo clic en Guardar), comienza la descarga del servidor remoto. Los detectores reciben eventos para indicar el progreso, el éxito o un error en la descarga. Para comprobar el estado del cuadro de diálogo y la operación de descarga después de llamar a `download()`, el código ActionScript deberá detectar eventos mediante detectores como `onCancel`, `onOpen`, `onProgress` y `onComplete`.

Una vez descargado el archivo correctamente, las propiedades del objeto FileReference se llenan con las propiedades del archivo local y se invoca el detector `onComplete`.

Sólo se puede realizar una sesión `browse()` o `download()` cada vez (porque sólo no se puede mostrar más de un cuadro de diálogo de forma simultánea).

Este método admite la descarga de cualquier tipo de archivo, con HTTP o HTTPS. También puede enviar datos al servidor con la llamada `download()` añadiendo parámetros a la URL, para que se analice el script de servidor.

Nota: Si el servidor requiere autenticación del usuario, sólo los archivos que se ejecutan en un navegador, es decir que utilizan el plug-in de navegador o controles ActiveX, pueden mostrar un cuadro de diálogo para pedir al usuario un nombre de usuario y una contraseña para la autenticación, y sólo para las descargas. La transferencia de archivos fallará en las cargas que utilicen el plug-in o el control ActiveX, y en las cargas y descargas que utilicen el reproductor autónomo o externo.

Cuando utilice este método, puede ser conveniente usar el modelo de seguridad de Flash Player:

- No admitido si el archivo SWF que realiza la llamada está en un entorno limitado local que no es de confianza.
- El valor predeterminado es denegar el acceso entre entornos limitados. Un sitio Web puede activar el acceso a un recurso utilizando un archivo de política de varios dominios.

Para más información, consulte lo siguiente:

- Capítulo 17, "Aspectos básicos de la seguridad," en *Aprendizaje de ActionScript 2.0 en Flash*
- El documento técnico sobre la seguridad de Flash Player 8 en http://www.macromedia.com/go/fp8_security

- El documento técnico sobre interfaces API relativas a seguridad de Flash Player 8 en http://www.macromedia.com/go/fp8_security_apis

Disponibilidad: ActionScript 1.0; Flash Player 8

Parámetros

url:String - La URL del archivo que se va a descargar en el equipo local. Puede enviar datos al servidor con la llamada `download()` añadiendo parámetros a la URL, para que se analice el script de servidor. Por ejemplo: `http://www.myserver.com/picture.jpg?userID=jdoe`

En algunos navegadores, las cadenas URL tienen un límite de longitud. Las cadenas con una longitud superior a los 256 caracteres pueden producir un error en algunos navegadores o servidores.

defaultFileName:String [opcional] - El nombre de archivo predeterminado que se muestra en el cuadro de diálogo para el archivo que se va a descargar. Esta cadena no puede contener los siguientes caracteres: / \ : * ? " < > | %

Si omite este parámetro, se analizará el nombre de archivo de la URL remota y se utilizará como predeterminado.

Valor devuelto

`Boolean` - Un valor `true` si se muestra el cuadro de diálogo en el que el usuario puede seleccionar un archivo. Si el cuadro de diálogo no se muestra, el método devuelve `false`. Es posible que el cuadro de diálogo no aparezca por cualquiera de los siguientes motivos:

- No ha pasado el valor para el parámetro `url`.
- El tipo o el formato de los parámetros pasados es incorrecto.
- El parámetro `url` tiene una longitud 0.
- Se ha producido una infracción de seguridad, es decir, su archivo SWF ha intentado acceder a un archivo de un servidor que está fuera del entorno limitado de seguridad de su archivo SWF.
- Hay otra sesión de búsqueda de archivos en curso. Pueden iniciar una sesión de búsqueda de archivos `FileReference.browse()`, `FileReferenceList.browse()` o `FileReference.download()`.
- El protocolo no es HTTP o HTTPS.

Eventos

onCancel

Se invoca cuando el usuario cierra el cuadro de diálogo.

onComplete

Se invoca cuando la operación de descarga de archivos se realiza completamente.

onIOError

Se invoca por alguna de los motivos siguientes:

- Se produce un error de entrada/salida mientras se lee o transmite el archivo.
- El archivo SWF intenta descargar un archivo desde un servidor que requiere autenticación, en el reproductor autónomo o externo. Durante la descarga, los reproductores autónomos y externos no proporcionan un medio para que los usuarios introduzcan contraseñas. Si un archivo SWF en estos reproductores intenta descargar un archivo desde un servidor que requiere autenticación, la descarga produce un error. La descarga de archivos sólo podrá realizarse correctamente en el control ActiveX y los reproductores plug-in del navegador.

onOpen

Se invoca cuando se inicia una operación de descarga.

onProgress

Se invoca periódicamente durante la operación de descarga de archivos.

onSecurityError

Se invoca cuando falla la descarga debido a un error de seguridad.

onSelect

Se invoca cuando el usuario selecciona un archivo del cuadro de diálogo para la descarga.

Ejemplo

El ejemplo siguiente intenta descargar un archivo mediante el método `download`. Observe que hay detectores para todos los eventos.

```
import flash.net.FileReference;

var listener:Object = new Object();

listener.onSelect = function(file:FileReference):Void {
  trace("onSelect: " + file.name);
}

listener.onCancel = function(file:FileReference):Void {
  trace("onCancel");
}

listener.onOpen = function(file:FileReference):Void {
  trace("onOpen: " + file.name);
}

listener.onProgress = function(file:FileReference, bytesLoaded:Number,
  bytesTotal:Number):Void {
  trace("onProgress with bytesLoaded: " + bytesLoaded + " bytesTotal: " +
  bytesTotal);
}

listener.onComplete = function(file:FileReference):Void {
  trace("onComplete: " + file.name);
}

listener.onIOError = function(file:FileReference):Void {
  trace("onIOError: " + file.name);
}

var fileRef:FileReference = new FileReference();
fileRef.addListener(listener);
var url:String = "http://www.macromedia.com/platform/whitepapers/
  platform_overview.pdf";
if(!fileRef.download(url, "FlashPlatform.pdf")) {
  trace("dialog box failed to open.");
}
```

Véase también

browse (método FileReference.browse), browse (método FileReferenceList.browse), upload (método FileReference.upload)

Constructor FileReference

```
public FileReference()
```

Crea un nuevo objeto FileReference. Cuando está lleno, un objeto FileReference representa un archivo en el disco local del usuario.

Disponibilidad: ActionScript 1.0; Flash Player 8

Ejemplo

El ejemplo siguiente crea un nuevo objeto FileReference e inicia la descarga de un archivo PDF.

```
import flash.net.FileReference;

var listener:Object = new Object();
listener.onComplete = function(file:FileReference) {
  trace("onComplete : " + file.name);
}

var url:String = "http://www.macromedia.com/platform/whitepapers/
  platform_overview.pdf";
var fileRef:FileReference = new FileReference();
fileRef.addListener(listener);
fileRef.download(url, "FlashPlatform.pdf");
```

Véase también

browse (método FileReference.browse)

modificationDate (propiedad FileReference.modificationDate)

```
public modificationDate : Date [read-only]
```

La última fecha de modificación del archivo en el disco local. Si aún no se ha llenado el objeto FileReference, una llamada para obtener el valor de esta propiedad devolverá `null`.

Disponibilidad: ActionScript 1.0; Flash Player 8

Ejemplo

El ejemplo siguiente recupera `modificationDate` de un archivo seleccionado por el usuario.

```
import flash.net.FileReference;

var listener:Object = new Object();
listener.onSelect = function(file:FileReference):Void {
  trace("modificationDate: " + file.modificationDate);
}

var fileRef:FileReference = new FileReference();
fileRef.addListener(listener);
fileRef.browse();
```

Véase también

`browse (método FileReference.browse)`

name (propiedad FileReference.name)

```
public name : String [read-only]
```

El nombre del archivo en el disco local. Si aún no se ha llenado el objeto FileReference, una llamada para obtener el valor de esta propiedad devolverá `null`.

Al llamar a `browse()` se llenan todas las propiedades de un objeto FileReference. A diferencia de otras propiedades FileReference, si llama a `download()`, la propiedad `name` se llena cuando se invoca `onSelect`.

Disponibilidad: ActionScript 1.0; Flash Player 8

Ejemplo

El ejemplo siguiente recupera el nombre de un archivo seleccionado por el usuario.

```
import flash.net.FileReference;

var listener:Object = new Object();
listener.onSelect = function(file:FileReference):Void {
  trace("name: " + file.name);
}

var fileRef:FileReference = new FileReference();
fileRef.addListener(listener);
fileRef.browse();
```

Véase también

`browse (método FileReference.browse)`

onCancel (detector de eventos FileReference.onCancel)

```
onCancel = function(fileRef:FileReference) {}
```

Se invoca cuando el usuario cierra el cuadro de diálogo de búsqueda de archivos. Este cuadro de diálogo aparece cuando llama a `FileReference.browse()`, `FileReferenceList.browse()` o `FileReference.download()`.

Disponibilidad: ActionScript 1.0; Flash Player 8

Parámetros

`fileRef``:flash.net.FileReference` - El objeto FileReference que inició la operación.

Ejemplo

El ejemplo siguiente muestra un mensaje si el usuario cierra el cuadro de diálogo de búsqueda de archivos. Este método sólo se activa si el usuario hace clic en Cancelar o presiona la tecla Esc una vez que se ha mostrado el cuadro de diálogo.

```
import flash.net.FileReference;

var listener:Object = new Object();

listener.onCancel = function(file:FileReference):Void {
  trace("onCancel");
}

var fileRef:FileReference = new FileReference();
fileRef.addListener(listener);
var url:String = "http://www.macromedia.com/platform/whitepapers/
  platform_overview.pdf";
if(!fileRef.download(url, "FlashPlatform.pdf")) {
  trace("dialog box failed to open.");
}
```

onComplete (detector de eventos FileReference.onComplete)

```
onComplete = function(fileRef:FileReference) {}
```

Se invoca cuando la operación de carga o descarga de archivos se realiza correctamente. Realizarse correctamente significa que se ha cargado o descargado el archivo completo.

Disponibilidad: ActionScript 1.0; Flash Player 8

Parámetros

fileRef:flash.net.FileReference - El objeto FileReference que inició la operación.

Ejemplo

El ejemplo siguiente muestra un mensaje cuando se activa el evento onComplete.

```
import flash.net.FileReference;

var listener:Object = new Object();

listener.onComplete = function(file:FileReference):Void {
  trace("onComplete: " + file.name);
}

var fileRef:FileReference = new FileReference();
fileRef.addListener(listener);
var url:String = "http://www.macromedia.com/platform/whitepapers/
  platform_overview.pdf";
fileRef.download(url, "FlashPlatform.pdf");
```

onHTTPError (detector de eventos FileReference.onHTTPError)

```
onHTTPError = function(fileRef:FileReference, httpError:Number) {}
```

Se invoca cuando falla una carga debido a un error HTTP.

Debido a la forma en que Flash Player se basa en la pila del navegador durante la descarga de archivos, este error no es aplicable a fallos de descarga. Si falla una descarga debido a un error HTTP, el error se transmite como error de E/S.

Disponibilidad: ActionScript 1.0; Flash Player 8

Parámetros

fileRef:flash.net.FileReference - El objeto FileReference que inició la operación.

httpError:Number El error HTTP que ha provocado el fallo de esta carga. Por ejemplo, un httpError de 404 indica que no se encuentra una página. Los valores de error HTTP se encuentran en las secciones 10.4 y 10.5 de la especificación HTTP en ftp://ftp.isi.edu/in-notes/rfc2616.txt.

Ejemplo

El ejemplo siguiente crea un objeto FileReference con un detector para cada evento posible incluido `onHttpError`. Este detector se activa solamente si falla la carga debido a un error HTTP.

```
import flash.net.FileReference;

var listener:Object = new Object();

listener.onSelect = function(file:FileReference):Void {
  trace("onSelect: " + file.name);
  if(!file.upload("http://www.yourdomain.com/
  yourUploadHandlerScript.cfm")) {
    trace("Upload dialog failed to open.");
  }
}

listener.onCancel = function(file:FileReference):Void {
  trace("onCancel");
}

listener.onOpen = function(file:FileReference):Void {
  trace("onOpen: " + file.name);
}

listener.onProgress = function(file:FileReference, bytesLoaded:Number,
  bytesTotal:Number):Void {
  trace("onProgress with bytesLoaded: " + bytesLoaded + " bytesTotal: " +
  bytesTotal);
}

listener.onComplete = function(file:FileReference):Void {
  trace("onComplete: " + file.name);
}

listener.onHTTPError = function(file:FileReference):Void {
  trace("onHTTPError: " + file.name);
}

listener.onIOError = function(file:FileReference):Void {
  trace("onIOError: " + file.name);
}

listener.onSecurityError = function(file:FileReference,
  errorString:String):Void {
  trace("onSecurityError: " + file.name + " errorString: " + errorString);
}
```

```
var fileRef:FileReference = new FileReference();
fileRef.addListener(listener);
fileRef.browse();
```

onIOError (detector de eventos FileReference.onIOError)

```
onIOError = function(fileRef:FileReference) {}
```

Se invoca cuando se produce un error de entrada/salida.

Este detector se invoca cuando falla la carga o descarga por cualquiera de las razones siguientes:

- Se produce un error de entrada/salida durante la lectura, escritura o transmisión del archivo.
- El SWF intenta cargar un archivo en un servidor que requiere autenticación, como un nombre de usuario y una contraseña. Durante la carga, Flash Player no proporciona un medio para que los usuarios introduzcan contraseñas. Si un SWF intenta cargar un archivo en un servidor que requiere autenticación, la carga produce un fallo.
- El archivo SWF intenta descargar un archivo desde un servidor que requiere autenticación, en el reproductor autónomo o externo. Durante la descarga, los reproductores autónomos y externos no proporcionan un medio para que los usuarios introduzcan contraseñas. Si un archivo SWF en estos reproductores intenta descargar un archivo desde un servidor que requiere autenticación, la descarga produce un error. La descarga de archivos sólo podrá realizarse correctamente en el control ActiveX y los reproductores plug-in del navegador.
- El valor pasado al parámetro `url` en `upload()` contiene un protocolo no válido. Los protocolos válidos son HTTP y HTTPS.

Importante: sólo las aplicaciones Flash que se ejecutan en un navegador, es decir, que emplean el plug-in de navegador o un control ActiveX, pueden mostrar un cuadro de diálogo para pedir al usuario que introduzca un nombre de usuario y una contraseña para la autenticación y, además, sólo para descargas. La transferencia de archivos fallará en las cargas que utilicen el plug-in o el control ActiveX, y en las cargas y descargas que utilicen reproductores autónomos o externos.

Disponibilidad: ActionScript 1.0; Flash Player 8

Parámetros

`fileRef``:flash.net.FileReference` - El objeto FileReference que inició la operación.

Ejemplo

El ejemplo siguiente muestra un mensaje cuando se activa el evento `onIOError`. Para que resulte más fácil, no se incluyen ninguno de los otros detectores de eventos en este ejemplo.

```
import flash.net.FileReference;

var listener:Object = new Object();

listener.onIOError = function(file:FileReference):Void {
  trace("onIOError");
}

var fileRef:FileReference = new FileReference();
fileRef.addListener(listener);
fileRef.download("http://www.macromedia.com/NonExistentFile.pdf",
  "NonExistentFile.pdf");
```

onOpen (detector de eventos FileReference.onOpen)

```
onOpen = function(fileRef:FileReference) {}
```

Se invoca cuando se inicia una operación de carga o descarga.

Disponibilidad: ActionScript 1.0; Flash Player 8

Parámetros

`fileRef``:flash.net.FileReference` - El objeto FileReference que inició la operación.

Ejemplo

El ejemplo siguiente muestra un mensaje cuando se activa el evento `onOpen`

```
import flash.net.FileReference;

var listener:Object = new Object();

listener.onOpen = function(file:FileReference):Void {
  trace("onOpen: " + file.name);
}

var fileRef:FileReference = new FileReference();
fileRef.addListener(listener);
var url:String = "http://www.macromedia.com/platform/whitepapers/
  platform_overview.pdf";
fileRef.download(url, "FlashPlatform.pdf");
```

onProgress (detector de eventos FileReference.onProgress)

```
onProgress = function(fileRef:FileReference, bytesLoaded:Number,
  bytesTotal:Number) {}
```

Se invoca periódicamente durante la operación de carga o descarga de archivos. El detector `onProgress` se invoca mientras Flash Player transmite bytes a un servidor y se invoca de forma periódica durante la transmisión, aunque no finalice correctamente. Para determinar si la transmisión es correcta y completa, utilice `onComplete`.

En algunos casos no se invocan los detectores `onProgress`, por ejemplo, cuando el archivo que se está transmitiendo es muy pequeño o la carga o descarga se realiza con gran rapidez.

El progreso de carga de archivos no puede determinarse en plataformas Macintosh anteriores a OS X 10.3. Durante la operación de carga se llama al evento `onProgress`, pero el valor del parámetro `bytesLoaded` es -1, que indica que no se puede determinar el progreso.

Disponibilidad: ActionScript 1.0; Flash Player 8

Parámetros

`fileRef``:flash.net.FileReference` - El objeto FileReference que inició la operación.

`bytesLoaded``:Number` - El número de bytes transmitidos hasta ahora.

`bytesTotal``:Number` - El tamaño total del archivo que se va a transmitir, en bytes. Si no se puede determinar el tamaño, el valor es -1.

Ejemplo

El ejemplo siguiente devuelve el progreso de una descarga utilizando el detector de eventos `onProgress`.

```
import flash.net.FileReference;

var listener:Object = new Object();

listener.onProgress = function(file:FileReference, bytesLoaded:Number,
  bytesTotal:Number):Void {
  trace("onProgress: " + file.name + " with bytesLoaded: " + bytesLoaded +
  " bytesTotal: " + bytesTotal);
}

var fileRef:FileReference = new FileReference();
fileRef.addListener(listener);
var url:String = "http://www.macromedia.com/platform/whitepapers/
  platform_overview.pdf";
fileRef.download(url, "FlashPlatform.pdf");
```

Véase también

onSecurityError (detector de eventos FileReference.onSecurityError)

```
onSecurityError = function(fileRef:FileReference, errorString:String) {}
```

Se invoca cuando falla la carga o descarga debido a un error de seguridad. Es posible que el archivo SWF que realiza la llamada haya intentado acceder a un archivo SWF fuera de su dominio y no tenga permiso para hacerlo. Puede intentar solucionar este error utilizando un archivo de política de varios dominios.

Disponibilidad: ActionScript 1.0; Flash Player 8

Parámetros

fileRef:flash.net.FileReference - El objeto FileReference que inició la operación.

errorString:String - Describe el error que ha provocado la llamada a onSecurityError. El valor es "securitySandboxError".

Ejemplo

El ejemplo siguiente crea un objeto FileReference con un detector para cada evento posible incluido onSecurityError. El detector onSecurityError se activa solamente si falla la carga debido a un error de seguridad.

```
import flash.net.FileReference;

var listener:Object = new Object();

listener.onSelect = function(file:FileReference):Void {
  trace("onSelect: " + file.name);
  if(!file.upload("http://www.yourdomain.com/
  yourUploadHandlerScript.cfm")) {
    trace("Upload dialog failed to open.");
  }
}

listener.onCancel = function(file:FileReference):Void {
  trace("onCancel");
}

listener.onOpen = function(file:FileReference):Void {
  trace("onOpen: " + file.name);
}
```

```
listener.onProgress = function(file:FileReference, bytesLoaded:Number,
  bytesTotal:Number):Void {
  trace("onProgress with bytesLoaded: " + bytesLoaded + " bytesTotal: " +
  bytesTotal);
}

listener.onComplete = function(file:FileReference):Void {
  trace("onComplete: " + file.name);
}

listener.onHTTPError = function(file:FileReference):Void {
  trace("onHTTPError: " + file.name);
}

listener.onIOError = function(file:FileReference):Void {
  trace("onIOError: " + file.name);
}

listener.onSecurityError = function(file:FileReference,
  errorString:String):Void {
  trace("onSecurityError: " + file.name + " errorString: " + errorString);
}

var fileRef:FileReference = new FileReference();
fileRef.addListener(listener);
fileRef.browse();
```

onSelect (detector de eventos FileReference.onSelect)

```
onSelect = function(fileRef:FileReference) {}
```

Se invoca cuando el usuario selecciona un archivo para la carga o descarga en el cuadro de diálogo de búsqueda de archivos. (Este cuadro de diálogo aparece cuando llama a `FileReference.browse()`, `FileReferenceList.browse()` o `FileReference.download()`.) Cuando el usuario selecciona un archivo y confirma la operación (por ejemplo, haciendo clic en Aceptar), se llenan las propiedades del objeto FileReference.

El detector `onSelect` funciona de forma algo diferente según el método que lo invoque. Cuando se invoca `onSelect` después de una llamada a `browse()`, Flash Player puede leer todas las propiedades del objeto FileReference, ya que el archivo seleccionado por el usuario está en el sistema de archivos local. Cuando se invoca `onSelect` después de una llamada a `download()`, Flash Player sólo puede leer la propiedad `name`, debido a que el archivo no se ha descargado todavía en el sistema local de archivos en el momento de invocar `onSelect`. Una vez que se haya descargado el archivo y se haya invocado `onComplete`, Flash Player puede leer todas las demás propiedades del objeto FileReference.

Disponibilidad: ActionScript 1.0; Flash Player 8

Parámetros

`fileRef``:flash.net.FileReference` - El objeto FileReference que inició la operación.

Ejemplo

El ejemplo siguiente muestra un mensaje dentro del detector de eventos `onSelect`.

```
import flash.net.FileReference;

var listener:Object = new Object();
listener.onSelect = function(file:FileReference):Void {
  trace("onSelect: " + file.name);
  if(!file.upload("http://www.yourdomain.com/
  yourUploadHandlerScript.cfm")) {
    trace("Upload dialog failed to open.");
  }
}

var fileRef:FileReference = new FileReference();
fileRef.addListener(listener);
fileRef.browse();
```

removeListener (método FileReference.removeListener)

```
public removeListener(listener:Object) : Boolean
```

Quita un objeto de la lista de objetos que reciben mensajes de notificación de eventos.

Disponibilidad: ActionScript 1.0; Flash Player 8

Parámetros

`listener``:Object` - Un objeto que detecta una notificación callback de los detectores de eventos FileReference.

Valor devuelto

`Boolean` - Devuelve `true` si el objeto especificado en el parámetro `listener` se ha eliminado correctamente. En caso contrario, este método devuelve `false`.

Ejemplo

El ejemplo siguiente elimina un detector de eventos mediante el método `removeListener`. Si un usuario cancela la descarga, el detector se elimina para que deje de recibir eventos de ese objeto FileReference.

```
import flash.net.FileReference;

var listener:Object = new Object();

listener.onCancel = function(file:FileReference):Void {
  trace(file.removeListener(this)); // true
}

var fileRef:FileReference = new FileReference();
fileRef.addListener(listener);
var url:String = "http://www.macromedia.com/platform/whitepapers/
  platform_overview.pdf";
fileRef.download(url, "FlashPlatform.pdf");
```

size (propiedad FileReference.size)

```
public size : Number [read-only]
```

El tamaño del archivo en el disco local, expresado en bytes. Si aún no se ha llenado el objeto FileReference, una llamada para obtener el valor de esta propiedad devolverá `null`.

Disponibilidad: ActionScript 1.0; Flash Player 8

Ejemplo

El ejemplo siguiente recupera el tamaño de un archivo seleccionado por el usuario.

```
import flash.net.FileReference;

var listener:Object = new Object();
listener.onSelect = function(file:FileReference):Void {
  trace("size: " + file.size + " bytes");
}

var fileRef:FileReference = new FileReference();
fileRef.addListener(listener);
fileRef.browse();
```

Véase también

browse (método FileReference.browse)

type (propiedad FileReference.type)

```
public type : String [read-only]
```

El tipo de archivo. En Windows esta propiedad es la extensión de archivo. En Macintosh, esta propiedad es el tipo de archivo de cuatro caracteres. Si aún no se ha llenado el objeto FileReference, una llamada para obtener el valor de esta propiedad devolverá `null`.

Disponibilidad: ActionScript 1.0; Flash Player 8

Ejemplo

El ejemplo siguiente recupera el tipo de un archivo seleccionado por el usuario.

```
import flash.net.FileReference;

var listener:Object = new Object();
listener.onSelect = function(file:FileReference):Void {
  trace("type: " + file.type);
}

var fileRef:FileReference = new FileReference();
fileRef.addListener(listener);
fileRef.browse();
```

Véase también

browse (método FileReference.browse)

upload (método FileReference.upload)

```
public upload(url:String) : Boolean
```

Inicia la carga de un archivo seleccionado por un usuario en un servidor remoto. Flash Player puede cargar archivos de hasta 100 MB. Debe llamar a `FileReference.browse()` o `FileReferenceList.browse()` antes de llamar a este método.

Los detectores reciben eventos para indicar el progreso, éxito o error en la carga. Aunque puede utilizar el objeto FileReferenceList para permitir a los usuarios seleccionar varios archivos para la carga, debe cargar los archivos uno a uno. Para ello, repita la selección por la matriz `FileReferenceList.fileList` de objetos FileReference.

El archivo se carga en la URL pasada en el parámetro `url`. La URL debe ser un script de servidor configurado para aceptar cargas. Flash Player carga archivos mediante el método HTTP `POST`. El script de servidor que gestiona la carga espera una petición `POST` con los siguientes elementos:

- Un elemento `Content-Type` de `multipart/form-data`
- Un elemento `Content-Disposition` con un atributo `name` definido como `"Filedata"` y un atributo `filename` definido como el nombre del archivo original
- El contenido binario del archivo

Esta es una petición `POST` de muestra:

```
Content-Type: multipart/form-data; boundary=AaB03x
--AaB03x
Content-Disposition: form-data; name="Filedata"; filename="example.jpg"
Content-Type: application/octet-stream
... contents of example.jpg ...
--AaB03x--
```

Puede enviar datos al servidor con la llamada `upload()` añadiendo parámetros a la URL.

Nota: Si el servidor requiere autenticación del usuario, sólo los archivos que se ejecutan en un navegador, es decir que utilizan el plug-in de navegador o controles ActiveX, pueden mostrar un cuadro de diálogo para pedir al usuario un nombre de usuario y una contraseña para la autenticación, y sólo para las descargas. La transferencia de archivos fallará en las cargas que utilicen el plug-in o el control ActiveX, y en las cargas y descargas que utilicen el reproductor autónomo o externo.

Cuando utilice este método, puede ser conveniente usar el modelo de seguridad de Flash Player:

- No admitido si el archivo SWF que realiza la llamada está en un entorno limitado local que no es de confianza.
- El valor predeterminado es denegar el acceso entre entornos limitados. Un sitio Web puede activar el acceso a un recurso utilizando un archivo de política de varios dominios.

Para más información, consulte lo siguiente:

- Capítulo 17, "Aspectos básicos de la seguridad," en *Aprendizaje de ActionScript 2.0 en Flash*
- El documento técnico sobre la seguridad de Flash Player 8 en http://www.macromedia.com/go/fp8_security
- El documento técnico sobre interfaces API relativas a seguridad de Flash Player 8 en http://www.macromedia.com/go/fp8_security_apis

Disponibilidad: ActionScript 1.0; Flash Player 8

Parámetros

url:String - La URL del script de servidor configurada para gestionar la carga mediante llamadas HTTP POST. La URL puede ser HTTP o, en cargas seguras, HTTPS.

Puede enviar datos al servidor con la llamada upload() añadiendo parámetros a la URL; por ejemplo, http://www.myserver.com/upload.cgi?userID=jdoe.

En algunos navegadores, las cadenas URL tienen un límite de longitud. Las cadenas con una longitud superior a los 256 caracteres pueden producir un error en algunos navegadores o servidores.

Valor devuelto

Boolean - Un valor de false en cualquiera de las siguientes situaciones:

- Aún no se ha llamado a FileReference.browse() correctamente en este objeto o a FileReferenceList.browse() con este objeto en su matriz filelist.
- El protocolo no es HTTP o HTTPS.
- Se ha producido una infracción de seguridad, es decir, su archivo SWF ha intentado acceder a un archivo de un servidor que está fuera del entorno limitado de seguridad de su archivo SWF.
- El tipo o el formato del parámetro url es incorrecto.
- La llamada no tiene el número correcto de parámetros.

Eventos

onCancel

Se invoca cuando el usuario cierra el cuadro de diálogo.

onComplete

Se invoca cuando la operación de carga de archivos se realiza completamente.

onHTTPError

Se invoca cuando falla una carga debido a un error HTTP.

onIOError

Se invoca en cualquiera de las situaciones siguientes:

- La carga falla debido a un error de entrada/salida mientras Flash Player lee, escribe o transmite el archivo.
- La carga falla debido a que el SWF intenta cargar un archivo en un servidor que requiere autenticación, como un nombre de usuario y una contraseña. Durante la carga, Flash Player no proporciona un medio para que los usuarios introduzcan contraseñas.
- La carga falla debido a que el parámetro `url` contiene un protocolo no válido. `FileReference.upload()` debe utilizar HTTP o HTTPS.

onOpen

Se invoca cuando se inicia una operación de carga.

onProgress

Se invoca periódicamente durante la operación de carga de archivos.

onSecurityError

Se invoca cuando falla la carga debido a una infracción de la seguridad.

Ejemplo

El ejemplo siguiente muestra una implementación del método `upload()` que en primer lugar solicita al usuario que seleccione un archivo para la carga, a continuación, gestiona los detectores `onSelect` y `onCancel`, y por último, gestiona el resultado de la carga del archivo.

```
import flash.net.FileReference;

var allTypes:Array = new Array();
var imageTypes:Object = new Object();
imageTypes.description = "Images (*.jpg, *.jpeg, *.gif, *.png)";
imageTypes.extension = "*.jpg; *.jpeg; *.gif; *.png";
allTypes.push(imageTypes);

var listener:Object = new Object();

listener.onSelect = function(file:FileReference):Void {
  trace("onSelect: " + file.name);
  if(!file.upload("http://www.yourdomain.com/
  yourUploadHandlerScript.cfm")) {
    trace("Upload dialog failed to open.");
  }
}

listener.onCancel = function(file:FileReference):Void {
```

```
  trace("onCancel");
}

listener.onOpen = function(file:FileReference):Void {
  trace("onOpen: " + file.name);
}

listener.onProgress = function(file:FileReference, bytesLoaded:Number,
  bytesTotal:Number):Void {
  trace("onProgress with bytesLoaded: " + bytesLoaded + " bytesTotal: " +
  bytesTotal);
}

listener.onComplete = function(file:FileReference):Void {
  trace("onComplete: " + file.name);
}

listener.onHTTPError = function(file:FileReference):Void {
  trace("onHTTPError: " + file.name);
}

listener.onIOError = function(file:FileReference):Void {
  trace("onIOError: " + file.name);
}

listener.onSecurityError = function(file:FileReference,
  errorString:String):Void {
  trace("onSecurityError: " + file.name + " errorString: " + errorString);
}

var fileRef:FileReference = new FileReference();
fileRef.addListener(listener);
fileRef.browse(allTypes);
```

Véase también

browse (método FileReference.browse), browse (método FileReferenceList.browse), download (método FileReference.download), fileList (propiedad FileReferenceList.fileList)

FileReferenceList (flash.net.FileReferenceList)

```
Object
  |
  +-flash.net.FileReferenceList
```

```
public class FileReferenceList
extends Object
```

La clase FileReferenceList proporciona un medio para permitir a los usuarios seleccionar uno o varios archivos para la carga. Un objeto FileReferenceList representa un grupo de uno o varios archivos locales en el disco del usuario como una matriz de objetos FileReference. Para obtener información detallada y consideraciones importantes sobre los objetos FileReference y la clase FileReference, que se utiliza con FileReferenceList, consulte la clase FileReference.

Para trabajar con la clase FileReferenceList:

- Cree una instancia de la clase: `var myFileRef = new FileReferenceList();`
- Llame a `FileReferenceList.browse()`, para mostrar un cuadro de diálogo en el que el usuario puede seleccionar uno o varios archivos para la carga: `myFileRef.browse();`
- Después de llamar correctamente a `browse()`, la propiedad `fileList` del objeto FileReferenceList se llena con una matriz de objetos FileReference.
- Llame a `FileReference.upload()` en cada elemento de la matriz `fileList`.

La clase FileReferenceList incluye un método `browse()` y una propiedad `fileList` para trabajar con varios archivos.

Disponibilidad: ActionScript 1.0; Flash Player 8

Ejemplo

El ejemplo siguiente permite al usuario seleccionar varios archivos y, a continuación, los carga en un servidor:

```
import flash.net.FileReferenceList;
import flash.net.FileReference;

var listener:Object = new Object();

listener.onSelect = function(fileRefList:FileReferenceList) {
  trace("onSelect");
  var list:Array = fileRefList.fileList;
  var item:FileReference;
  for(var i:Number = 0; i < list.length; i++) {
    item = list[i];
    trace("name: " + item.name);
```

```
    trace(item.addListener(this));
    item.upload("http://www.yourdomain.com/");
  }
}

listener.onCancel = function():Void {
  trace("onCancel");
}

listener.onOpen = function(file:FileReference):Void {
  trace("onOpen: " + file.name);
}

listener.onProgress = function(file:FileReference, bytesLoaded:Number,
  bytesTotal:Number):Void {
  trace("onProgress with bytesLoaded: " + bytesLoaded + " bytesTotal: " +
  bytesTotal);
}

listener.onComplete = function(file:FileReference):Void {
  trace("onComplete: " + file.name);
}

listener.onHTTPError = function(file:FileReference, httpError:Number):Void
  {
  trace("onHTTPError: " + file.name + " httpError: " + httpError);
}

listener.onIOError = function(file:FileReference):Void {
  trace("onIOError: " + file.name);
}

listener.onSecurityError = function(file:FileReference,
  errorString:String):Void {
  trace("onSecurityError: " + file.name + " errorString: " + errorString);
}

var fileRef:FileReferenceList = new FileReferenceList();
fileRef.addListener(listener);
fileRef.browse();
```

Véase también

FileReference (flash.net.FileReference)

Resumen de propiedades

Modificadores	Propiedad	Descripción
	`fileList:Array`	Conjunto de objetos FileReference.

Propiedades heredadas de la clase Object

constructor (propiedad Object.constructor), __proto__ (propiedad Object.__proto__), prototype (propiedad Object.prototype), __resolve (propiedad Object.__resolve)

Resumen de eventos

Evento	Descripción
`onCancel = function(fileRefList:FileReferenceList) {}`	Se invoca cuando el usuario cierra el cuadro de diálogo de búsqueda de archivos.
`onSelect = function(fileRefList:FileReferenceList) {}`	Se invoca cuando el usuario selecciona uno o varios archivos para la carga en el cuadro de diálogo de búsqueda de archivos.

Resumen de constructores

Firma	Descripción
`FileReferenceList()`	Crea un nuevo objeto FileReferenceList.

Resumen de métodos

Modificadores	Firma	Descripción
	`addListener(listener:Object) : Void`	Registra un objeto para recibir una notificación cuando se invoca un detector de eventos FileReferenceList.
	`browse([typelist:Array]) : Boolean`	Muestra un cuadro de diálogo de búsqueda de archivos en el que el usuario puede seleccionar uno o varios archivos locales para la carga.
	`removeListener(listener:Object) : Boolean`	Quita un objeto de la lista de objetos que reciben mensajes de notificación de eventos.

Métodos heredados de la clase Object

addProperty (método Object.addProperty), hasOwnProperty (método Object.hasOwnProperty), isPropertyEnumerable (método Object.isPropertyEnumerable), isPrototypeOf (método Object.isPrototypeOf), registerClass (método Object.registerClass), toString (método Object.toString), unwatch (método Object.unwatch), valueOf (método Object.valueOf), watch (método Object.watch)

addListener (método FileReferenceList.addListener)

```
public addListener(listener:Object) : Void
```

Registra un objeto para recibir una notificación cuando se invoca un detector de eventos FileReferenceList.

Disponibilidad: ActionScript 1.0; Flash Player 8

Parámetros

listener:Object - Un objeto que detecta una notificación callback de los detectores de eventos FileReferenceList.

Ejemplo

En el ejemplo siguiente muestra el método `addListener()`.

```
import flash.net.FileReferenceList;

var listener:Object = new Object();
listener.onCancel = function(fileRefList:FileReferenceList) {
  trace("onCancel");
}

listener.onSelect = function(fileRefList:FileReferenceList) {
  trace("onSelect: " + fileRefList.fileList.length);
}

var fileRef:FileReferenceList = new FileReferenceList();
fileRef.addListener(listener);
fileRef.browse();
```

browse (método FileReferenceList.browse)

```
public browse([typelist:Array]) : Boolean
```

Muestra un cuadro de diálogo de búsqueda de archivos en el que el usuario puede seleccionar uno o varios archivos locales para la carga. El cuadro de diálogo depende del sistema operativo del usuario. Cuando se llama a este método y el usuario selecciona archivos correctamente, la propiedad `fileList` de este objeto FileReferenceList se llena con una matriz de objetos FileReference, uno para cada archivo seleccionado por el usuario. Posteriormente, cada vez que se llama a `FileReferenceList.browse()`, la propiedad FileReferenceList.fileList se restablece según el archivo o archivos seleccionados por el usuario en el cuadro de diálogo.

Puede pasar una matriz de tipos de archivos para determinar qué archivos muestra el cuadro de diálogo.

Sólo se puede realizar una sesión `browse()` o `download()` cada vez en un objeto FileReferenceList (porque no se puede mostrar más de un cuadro de diálogo de forma simultánea).

Disponibilidad: ActionScript 1.0; Flash Player 8

Parámetros

`typelist``:Array` [opcional] - Una matriz de tipos de archivo para filtrar los archivos que se muestran en el cuadro de diálogo. Si omite este parámetro, se mostrarán todos los archivos. Si incluye este parámetro, la matriz deberá contener uno o más elementos entre llaves { }. Puede utilizar uno de estos dos formatos para la matriz:

- Una lista de descripciones de tipos de archivo seguida de las extensiones de archivo solamente. Cada elemento de la matriz debe contener una cadena que describa el tipo de archivo y una lista de extensiones de archivos de Windows delimitadas por punto y coma, con un carácter comodín (*) delante de cada extensión La sintaxis del elemento es la siguiente: `[{description: "`*`string describing the first set of file types`*`", extension: "`*`semicolon-delimited list of file extensions`*`"}]` Ejemplo: `[{description: "Images", extension: "*.jpg;*.gif;*.png"}, {description: "Flash Movies", extension: "*.swf"}, {description: "Documents", extension: "*.doc;*.pdf"}]`

- Una lista de descripciones de tipos de archivo seguida de las extensiones de archivo de Windows y los tipos de archivo de Macintosh. Cada elemento de la matriz debe contener una cadena que describa el tipo de archivo; una lista de extensiones de archivos de Windows delimitadas por punto y coma, con un comodín (*) delante de cada extensión, y una lista de extensiones de archivos de Macintosh delimitadas por punto y coma, con un carácter comodín (*) delante de cada extensión. La sintaxis del elemento es la siguiente:
  ```
  [{description: "string describing the first set of file types",
  extension: "semicolon-delimited list of Windows file extensions",
  macType: "semicolon-delimited list of Macintosh file types"}]
  ```
  Ejemplo:
  ```
  [{description: "Image files", extension: "*.jpg;*.gif;*.png", macType:
  "JPEG;jp2_;GIFF"}, {description: "Flash Movies", extension: "*.swf",
  macType: "SWFL"}]
  ```

Los dos formatos no son intercambiables en una sola llamada a `browse()`. Deberá utilizar uno u otro.

La lista de extensiones se emplea para filtrar los archivos en Windows, de acuerdo con el tipo de archivo que seleccione el usuario. No aparece en el cuadro de diálogo. Para que los usuarios vean los tipos de archivos, debe mostrar los tipos de archivo en la cadena de descripción, así como la lista de extensiones. La cadena de descripción aparece en el cuadro de diálogo en Windows. (No se utiliza en Macintosh.) En Macintosh, si se suministra una lista de tipos de archivos de Macintosh, se utiliza ésta para filtrar los archivos. En caso contrario, se utiliza la lista de extensiones de Windows.

Valor devuelto

`Boolean` - Devuelve `true` si los parámetros son válidos y aparece el cuadro de diálogo de búsqueda de archivos. Devuelve `false` si el cuadro de diálogo no se ha mostrado, si hay otra sesión de búsqueda de archivos en curso o si ha utilizado el parámetro `typelist` pero no ha proporcionado una descripción o extensión en ningún elemento de la matriz.

Eventos

onCancel

Se invoca cuando el usuario cancela el cuadro de diálogo haciendo clic en Cancelar o cerrándolo.

onSelect

Se invoca cuando el usuario selecciona en el cuadro de diálogo un elemento para la carga.

Ejemplo

En el ejemplo siguiente muestra el método `browse()`.

```
import flash.net.FileReferenceList;

var allTypes:Array = new Array();
var imageTypes:Object = new Object();
imageTypes.description = "Images (*.JPG;*.JPEG;*.JPE;*.GIF;*.PNG;)";
imageTypes.extension = "*.jpg; *.jpeg; *.jpe; *.gif; *.png;";
allTypes.push(imageTypes);

var textTypes:Object = new Object();
textTypes.description = "Text Files (*.TXT;*.RTF;)";
textTypes.extension = "*.txt; *.rtf";
allTypes.push(textTypes);

var fileRef:FileReferenceList = new FileReferenceList();
fileRef.browse(allTypes);
```

Véase también

`browse (método FileReference.browse)`, `FileReference (flash.net.FileReference)`

fileList (propiedad FileReferenceList.fileList)

```
public fileList : Array
```

Conjunto de objetos FileReference.

Cuando se ha llamado al método `FileReferenceList.browse()` y el usuario ha seleccionado uno o varios archivos del cuadro de diálogo que ha abierto `browse()`, esta propiedad se llena con una matriz de objetos FileReference. Cada uno representa un archivo seleccionado por el usuario. A continuación puede utilizar esta matriz para cargar los archivos con `FileReference.upload()`. Debe cargar los archivos de uno en uno.

La propiedad `fileList` se llena cada vez que se llama a `browse()` en ese objeto FileReferenceList.

Las propiedades de los objetos FileReference se describen en la documentación de la clase FileReference.

Disponibilidad: ActionScript 1.0; Flash Player 8

Ejemplo

En el ejemplo siguiente muestra la propiedad `fileList`.

```
import flash.net.FileReferenceList;
import flash.net.FileReference;
```

```
var listener:Object = new Object();
listener.onSelect = function(fileRefList:FileReferenceList) {
  trace("onSelect");
  var list:Array = fileRefList.fileList;
  var item:FileReference;
  for(var i:Number = 0; i < list.length; i++) {
    item = list[i];
    trace("name: " + item.name);
  }
}

var fileRef:FileReferenceList = new FileReferenceList();
fileRef.addListener(listener);
fileRef.browse();
```

Véase también

FileReference (flash.net.FileReference), upload (método FileReference.upload), browse (método FileReferenceList.browse)

Constructor FileReferenceList

```
public FileReferenceList()
```

Crea un nuevo objeto FileReferenceList. Este objeto no contiene nada hasta que se llama a `browse()`. Cuando llama a `browse()` en el objeto FileReference, la propiedad `fileList` del objeto se llena con una matriz de objetos FileReference.

Disponibilidad: ActionScript 1.0; Flash Player 8

Ejemplo

El ejemplo siguiente crea un nuevo objeto `FileReferenceList`, se repite en cada archivo seleccionado y da como resultado sus nombres.

```
import flash.net.FileReferenceList;

var listener:Object = new Object();
listener.onSelect = function(fileRefList:FileReferenceList) {
  trace("onSelect");
  var arr:Array = fileRefList.fileList;
  for(var i:Number = 0; i < arr.length; i++) {
    trace("name: " + arr[i].name);
  }
}

var fileRef:FileReferenceList = new FileReferenceList();
fileRef.addListener(listener);
fileRef.browse();
```

Véase también

FileReference (flash.net.FileReference), browse (método FileReferenceList.browse)

onCancel (detector de eventos FileReferenceList.onCancel)

```
onCancel = function(fileRefList:FileReferenceList) {}
```

Se invoca cuando el usuario cierra el cuadro de diálogo de búsqueda de archivos. (Este cuadro de diálogo aparece cuando llama a los métodos `FileReferenceList.browse()`, `FileReference.browse()` o `FileReference.download()`.)

Disponibilidad: ActionScript 1.0; Flash Player 8

Parámetros

`fileRefList``:flash.net.FileReferenceList` - El objeto FileReferenceList que inició la operación.

Ejemplo

En el ejemplo siguiente muestra el detector `onCancel`.

```
import flash.net.FileReferenceList;

var listener:Object = new Object();
listener.onCancel = function(fileRefList:FileReferenceList) {
  trace("onCancel");
}

var fileRef:FileReferenceList = new FileReferenceList();
fileRef.addListener(listener);
fileRef.browse();
```

Véase también

browse (método FileReferenceList.browse)

onSelect (detector de eventos FileReferenceList.onSelect)

```
onSelect = function(fileRefList:FileReferenceList) {}
```

Se invoca cuando el usuario selecciona uno o varios archivos para la carga en el cuadro de diálogo de búsqueda de archivos. (Este cuadro de diálogo aparece cuando llama a los métodos `FileReferenceList.browse()`, `FileReference.browse()` o `FileReference.download()`.) Cuando un usuario selecciona un archivo y confirma la operación (por ejemplo, haciendo clic en Save), el objeto FileReferenceList se llena con objetos FileReference que representan los archivos seleccionados por el usuario.

Disponibilidad: ActionScript 1.0; Flash Player 8

Parámetros

`fileRefList``:flash.net.FileReferenceList` - El objeto FileReferenceList que inició la operación.

Ejemplo

En el ejemplo siguiente muestra el detector `onSelect`.

```
import flash.net.FileReferenceList;
import flash.net.FileReference;

var listener:Object = new Object();

listener.onSelect = function(fileRefList:FileReferenceList) {
  trace("onSelect");
  var list:Array = fileRefList.fileList;
  var item:FileReference;
  for(var i:Number = 0; i < list.length; i++) {
    item = list[i];
    trace("name: " + item.name);
    trace(item.addListener(this));
    item.upload("http://www.yourdomain.com/");
  }
}

listener.onComplete = function(file:FileReference):Void {
  trace("onComplete: " + file.name);
}

var fileRef:FileReferenceList = new FileReferenceList();
fileRef.addListener(listener);
fileRef.browse();
```

Véase también

browse (método FileReferenceList.browse)

removeListener (método FileReferenceList.removeListener)

```
public removeListener(listener:Object) : Boolean
```

Quita un objeto de la lista de objetos que reciben mensajes de notificación de eventos.

Disponibilidad: ActionScript 1.0; Flash Player 8

Parámetros

listener:Object - Un objeto que detecta una notificación callback de los detectores de eventos FileReferenceList.

Valor devuelto

Boolean - Si el objeto detector se ha eliminado, devuelve `true`. En caso contrario, este método devuelve `false`.

Ejemplo

En el ejemplo siguiente muestra el método `removeListener`.

```
import flash.net.FileReferenceList;

var listener:Object = new Object();
listener.onCancel = function(fileRefList:FileReferenceList) {
  trace("onCancel");
  trace(fileRefList.removeListener(this)); // true
}

listener.onSelect = function(fileRefList:FileReferenceList) {
  trace("onSelect: " + fileRefList.fileList.length);
}

var fileRef:FileReferenceList = new FileReferenceList();
fileRef.addListener(listener);
fileRef.browse();
```

Function

```
Object
  |
  +-Function
```

```
public dynamic class Function
extends Object
```

Tanto las funciones definidas por el usuario como las funciones incorporadas de ActionScript se representan mediante objetos Function, que son instancias de la función Function.

Disponibilidad: ActionScript 1.0; Flash Player 6

Resumen de propiedades

Propiedades heredadas de la clase Object

constructor (propiedad Object.constructor), __proto__ (propiedad Object.__proto__), prototype (propiedad Object.prototype), __resolve (propiedad Object.__resolve)

Resumen de métodos

Modificadores	Firma	Descripción
	apply(thisObject:Object, [argArray:Array]) : Void	Especifica el valor de thisObject que debe utilizarse dentro de cualquier función a la que llame ActionScript.
	call(thisObject:Object, [parameter1:Object]) : Object	Invoca la función representada por un objeto Function.

Métodos heredados de la clase Object

addProperty (método Object.addProperty), hasOwnProperty (método Object.hasOwnProperty), isPropertyEnumerable (método Object.isPropertyEnumerable), isPrototypeOf (método Object.isPrototypeOf), registerClass (método Object.registerClass), toString (método Object.toString), unwatch (método Object.unwatch), valueOf (método Object.valueOf), watch (método Object.watch)

apply (método Function.apply)

```
public apply(thisObject:Object, [argArray:Array]) : Void
```

Especifica el valor de `thisObject` que debe utilizarse dentro de cualquier función a la que llame ActionScript. Este método también especifica parámetros que deben pasarse a la función llamada. Dado que `apply()` es un método de la clase Function, es también un método de todos los objetos Function de ActionScript.

Los parámetros se especifican como objeto Array, a diferencia de `Function.call()`, que especifica los parámetros como una lista separada por comas. Suele ser útil cuando no se conoce el número de parámetros hasta que se ejecuta el script.

Devuelve el valor que la función llamada especifica como valor de devolución.

Disponibilidad: ActionScript 1.0; Flash Player 6

Parámetros

`thisObject``:Object` - Objeto al que se aplica myFunction.

`argArray``:Array` [opcional] - Matriz cuyos elementos se pasan a myFunction como parámetros.

Ejemplo

Las siguientes invocaciones de función son equivalentes:

```
Math.atan2(1, 0)
Math.atan2.apply(null, [1, 0])
```

El sencillo ejemplo siguiente muestra cómo `apply()` pasa una matriz de parámetros:

```
function theFunction() {
  trace(arguments);
}

// create a new array to pass as a parameter to apply()
var firstArray:Array = new Array(1,2,3);
theFunction.apply(null,firstArray);
// outputs: 1,2,3

// create a second array to pass as a parameter to apply()
var secondArray:Array = new Array("a", "b", "c");
theFunction.apply(null,secondArray);
// outputs a,b,c
```

El ejemplo siguiente muestra cómo `apply()` pasa una matriz de parámetros y especifica el valor de ésta:

```
// define a function
function theFunction() {
  trace("this == myObj? " + (this == myObj));
```

```
  trace("arguments: " + arguments);
}

// instantiate an object
var myObj:Object = new Object();

// create arrays to pass as a parameter to apply()
var firstArray:Array = new Array(1,2,3);
var secondArray:Array = new Array("a", "b", "c");

// use apply() to set the value of this to be myObj and send firstArray
theFunction.apply(myObj,firstArray);
// output:
// this == myObj? true
// arguments: 1,2,3

// use apply() to set the value of this to be myObj and send secondArray
theFunction.apply(myObj,secondArray);
// output:
// this == myObj? true
// arguments: a,b,c
```

Véase también

call (método Function.call)

call (método Function.call)

```
public call(thisObject:Object, [parameter1:Object]) : Object
```

Invoca la función representada por un objeto Function. Cada función de ActionScript se representa mediante un objeto Function, de modo que todas las funciones admiten este método.

En casi todos los casos puede utilizarse el operador de llamada de función (`()`) en lugar de este método. El operador de llamada de función hace que el código sea conciso y legible. Este método es de gran utilidad cuando debe controlarse explícitamente el parámetro `thisObject` de la llamada de función. Normalmente, si se invoca una función como método de un objeto, el parámetro `thisObject` se establece en `myObject` dentro del cuerpo de la función, como se muestra en el siguiente ejemplo:

```
myObject.myMethod(1, 2, 3);
```

En algunos casos, es posible que desee que `thisObject` haga referencia a otro elemento; por ejemplo, si debe invocarse una función como un método de un objeto, pero en realidad no se almacena como método de dicho objeto:

```
myObject.myMethod.call(myOtherObject, 1, 2, 3);
```

Puede pasar el valor null para el parámetro `thisObject` para invocar una función como función regular y no como un método de un objeto. Por ejemplo, las llamadas de función siguientes son equivalentes:

```
Math.sin(Math.PI / 4)
Math.sin.call(null, Math.PI / 4)
```

Devuelve el valor que la función llamada especifica como valor de devolución.

Disponibilidad: ActionScript 1.0; Flash Player 6

Parámetros

`thisObject``:Object` - Objeto que especifica el valor de `thisObject` en el cuerpo de la función.

`parameter1``:Object` [opcional] - Un parámetro que se pasa a myFunction. Puede especificar cero o más parámetros.

Valor devuelto

`Object` -

Ejemplo

El ejemplo siguiente utiliza `Function.call()` para hacer que una función se comporte como un método de otro objeto, sin almacenar la función en el objeto:

```
function myObject() {
}
function myMethod(obj) {
  trace("this == obj? " + (this == obj));
}
var obj:Object = new myObject();
myMethod.call(obj, obj);
```

La sentencia `trace()` muestra:

```
this == obj? true
```

Véase también

`apply (método Function.apply)`

GlowFilter (flash.filters.GlowFilter)

```
Object
  |
  +-flash.filters.BitmapFilter
    |
    +-flash.filters.GlowFilter
```

```
public class GlowFilter
extends BitmapFilter
```

La clase GlowFilter permite aplicar un efecto de iluminado a distintos objetos en Flash. Cuenta con varias opciones para el estilo del iluminado, incluidos iluminado interior o exterior y modo de extractor. El filtro de iluminado es muy similar al filtro de sombreado con las propiedades `distance` y `angle` de la sombra definidas como cero.

El uso de filtros depende del objeto al que se aplique el filtro:

- Para aplicar filtros a clips de película, campos de texto y botones en tiempo de ejecución, utilice la propiedad `filters`. Establecer la propiedad `filters` de un objeto no modifica el objeto y se puede deshacer borrando la propiedad `filters`.
- Para aplicar filtros a instancias de BitmapData, utilice el método `BitmapData.applyFilter()`. Cuando se llama a `applyFilter()` en un objeto BitmapData, se genera una imagen filtrada al procesar el objeto BitmapData de origen y el objeto de filtro.

También puede aplicar efectos de filtro a imágenes y vídeo durante la edición. Para más información, consulte la documentación de edición.

Si aplica un filtro a un clip de película o a un botón, la propiedad `cacheAsBitmap` del clip de película o del botón se establecerá como `true`. Si borra todos los filtros, se restaurará el valor original de `cacheAsBitmap`.

Este filtro admite la aplicación de escala en el escenario. Sin embargo, no se puede utilizar escala, rotación y sesgo general; si el propio objeto está a escala (si `_xscale` y `_yscale` no son 100%), no se aplica la escala al efecto de filtro. Sólo se ajusta a escala cuando se acerca el escenario.

El filtro no se aplica si la imagen resultante supera los 2880 píxeles de anchura o altura. Por ejemplo, si amplía un clip de película grande al que se le ha aplicado un filtro, éste se desactiva si la imagen resultante supera el límite de 2880 píxeles.

Disponibilidad: ActionScript 1.0; Flash Player 8

Véase también

applyFilter (método BitmapData.applyFilter), cacheAsBitmap (propiedad Button.cacheAsBitmap), filters (propiedad Button.filters), DropShadowFilter (flash.filters.DropShadowFilter), cacheAsBitmap (propiedad MovieClip.cacheAsBitmap), filters (propiedad MovieClip.filters), filters (propiedad TextField.filters)

Resumen de propiedades

Modificadores	Propiedad	Descripción
	alpha:Number	Valor de transparencia alfa del color.
	blurX:Number	Cantidad de desenfoque horizontal.
	blurY:Number	Cantidad de desenfoque vertical.
	color:Number	Color del iluminado.
	inner:Boolean	Especifica si el iluminado es interior.
	knockout:Boolean	Especifica si el objeto tiene un efecto extractor.
	quality:Number	Número de veces que debe aplicarse el filtro.
	strength:Number	Intensidad de la impresión o extensión.

Propiedades heredadas de la clase Object

constructor (propiedad Object.constructor), __proto__ (propiedad Object.__proto__), prototype (propiedad Object.prototype), __resolve (propiedad Object.__resolve)

Resumen de constructores

Firma	Descripción
GlowFilter([color:Number], [alpha:Number], [blurX:Number], [blurY:Number], [strength:Number], [quality:Number], [inner:Boolean], [knockout:Boolean])	Inicializa una instancia GlowFilter nueva con los parámetros especificados.

Resumen de métodos

Modificadores	Firma	Descripción
	`clone() : GlowFilter`	Devuelve una copia de este objeto de filtro.

Métodos heredados de la clase BitmapFilter

`clone (método BitmapFilter.clone)`

Métodos heredados de la clase Object

`addProperty (método Object.addProperty), hasOwnProperty (método Object.hasOwnProperty), isPropertyEnumerable (método Object.isPropertyEnumerable), isPrototypeOf (método Object.isPrototypeOf), registerClass (método Object.registerClass), toString (método Object.toString), unwatch (método Object.unwatch), valueOf (método Object.valueOf), watch (método Object.watch)`

alpha (propiedad GlowFilter.alpha)

`public alpha : Number`

Valor de transparencia alfa del color. Los valores válidos son de 0 a 1. Por ejemplo, .25 define un valor de transparencia de 25%. El valor predeterminado es 1.

Disponibilidad: ActionScript 1.0; Flash Player 8

Ejemplo

El ejemplo siguiente cambia la propiedad `alpha` de un clip de película existente cuando un usuario hace clic en ella.

```
import flash.filters.GlowFilter;

var mc:MovieClip = createGlowFilterRectangle("GlowFilterAlpha");
mc.onRelease = function() {
  var filter:GlowFilter = this.filters[0];
  filter.alpha = .4;
  this.filters = new Array(filter);
}

function createGlowFilterRectangle(name:String):MovieClip {
  var rect:MovieClip = this.createEmptyMovieClip(name,
  this.getNextHighestDepth());
  var w:Number = 100;
  var h:Number = 100;
  rect.beginFill(0x003366);
  rect.lineTo(w, 0);
```

```
  rect.lineTo(w, h);
  rect.lineTo(0, h);
  rect.lineTo(0, 0);
  rect._x = 20;
  rect._y = 20;

  var filter:GlowFilter = new GlowFilter(0x000000, .8, 16, 16, 1, 3, false,
  false);
  var filterArray:Array = new Array();
  filterArray.push(filter);
  rect.filters = filterArray;
  return rect;
}
```

blurX (propiedad GlowFilter.blurX)

```
public blurX : Number
```

Cantidad de desenfoque horizontal. Los valores válidos van de 0 a 255 (coma flotante). El valor predeterminado es 6. Los valores que son múltiplos de 2 (como 2, 4, 8, 16 y 32) se optimizan para representarse más rápidamente que otros valores.

Disponibilidad: ActionScript 1.0; Flash Player 8

Ejemplo

El ejemplo siguiente cambia la propiedad `blurX` de un clip de película existente cuando un usuario hace clic en ella.

```
import flash.filters.GlowFilter;

var mc:MovieClip = createGlowFilterRectangle("GlowFilterBlurX");
mc.onRelease = function() {
  var filter:GlowFilter = this.filters[0];
  filter.blurX = 20;
  this.filters = new Array(filter);
}

function createGlowFilterRectangle(name:String):MovieClip {
  var rect:MovieClip = this.createEmptyMovieClip(name,
  this.getNextHighestDepth());
  var w:Number = 100;
  var h:Number = 100;
  rect.beginFill(0x003366);
  rect.lineTo(w, 0);
  rect.lineTo(w, h);
  rect.lineTo(0, h);
  rect.lineTo(0, 0);
  rect._x = 20;
  rect._y = 20;
```

```
  var filter:GlowFilter = new GlowFilter(0x000000, .8, 16, 16, 1, 3, false,
  false);
  var filterArray:Array = new Array();
  filterArray.push(filter);
  rect.filters = filterArray;
  return rect;
}
```

blurY (propiedad GlowFilter.blurY)

```
public blurY : Number
```

Cantidad de desenfoque vertical. Los valores válidos van de 0 a 255 (coma flotante). El valor predeterminado es 6. Los valores que son múltiplos de 2 (como 2, 4, 8, 16 y 32) se optimizan para representarse más rápidamente que otros valores.

Disponibilidad: ActionScript 1.0; Flash Player 8

Ejemplo

El ejemplo siguiente cambia la propiedad `blurY` de un clip de película existente cuando un usuario hace clic en ella.

```
import flash.filters.GlowFilter;

var mc:MovieClip = createGlowFilterRectangle("GlowFilterBlurY");
mc.onRelease = function() {
  var filter:GlowFilter = this.filters[0];
  filter.blurY = 20;
  this.filters = new Array(filter);
}

function createGlowFilterRectangle(name:String):MovieClip {
  var rect:MovieClip = this.createEmptyMovieClip(name,
  this.getNextHighestDepth());
  var w:Number = 100;
  var h:Number = 100;
  rect.beginFill(0x003366);
  rect.lineTo(w, 0);
  rect.lineTo(w, h);
  rect.lineTo(0, h);
  rect.lineTo(0, 0);
  rect._x = 20;
  rect._y = 20;

  var filter:GlowFilter = new GlowFilter(0x000000, .8, 16, 16, 1, 3, false,
  false);
  var filterArray:Array = new Array();
  filterArray.push(filter);
```

```
  rect.filters = filterArray;
  return rect;
}
```

clone (método GlowFilter.clone)

```
public clone() : GlowFilter
```

Devuelve una copia de este objeto de filtro.

Disponibilidad: ActionScript 1.0; Flash Player 8

Valor devuelto

`flash.filters.GlowFilter` - Instancia GlowFilter nueva con todas las propiedades de la instancia GlowFilter original.

Ejemplo

El ejemplo siguiente crea tres objetos GlowFilter y los compara: `filter_1` se crea utilizando el constructor GlowFilter, `filter_2` se crea con el mismo valor que `filter_1` y `clonedFilter` se crea clonando `filter_1`. Observe que aunque `filter_2` da como resultado `filter_1`, `clonedFilter`, aunque contiene los mismos valores que `filter_1`, no es igual.

```
import flash.filters.GlowFilter;

var filter_1:GlowFilter = new GlowFilter(0x33CCFF, .8, 35, 35, 2, 3, false,
  false);
var filter_2:GlowFilter = filter_1;
var clonedFilter:GlowFilter = filter_1.clone();

trace(filter_1 == filter_2); // true
trace(filter_1 == clonedFilter); // false

for(var i in filter_1) {
  trace(">> " + i + ": " + filter_1[i]);
  // >> clone: [type Function]
  // >> strength: 2
  // >> blurY: 35
  // >> blurX: 35
  // >> knockout: false
  // >> inner: false
  // >> quality: 3
  // >> alpha: 0.8
  // >> color: 3394815
}

for(var i in clonedFilter) {
  trace(">> " + i + ": " + clonedFilter[i]);
  // >> clone: [type Function]
```

```
  // >> strength: 2
  // >> blurY: 35
  // >> blurX: 35
  // >> knockout: false
  // >> inner: false
  // >> quality: 3
  // >> alpha: 0.8
  // >> color: 3394815
}
```

Para demostrar mejor la relación entre `filter_1`, `filter_2` y `clonedFilter`, en el ejemplo siguiente se modifica la propiedad `knockout` de `filter_1`. Al modificar `knockout`, el método `clone()` crea una nueva instancia basada en los valores de `filter_1` en lugar de utilizarlos en la referencia.

```
import flash.filters.GlowFilter;

var filter_1:GlowFilter = new GlowFilter(0x33CCFF, .8, 35, 35, 2, 3, false,
  false);
var filter_2:GlowFilter = filter_1;
var clonedFilter:GlowFilter = filter_1.clone();

trace(filter_1.knockout); // false
trace(filter_2.knockout); // false
trace(clonedFilter.knockout); // false

filter_1.knockout = true;

trace(filter_1.knockout); // true
trace(filter_2.knockout); // true
trace(clonedFilter.knockout); // false
```

color (propiedad GlowFilter.color)

```
public color : Number
```

Color del iluminado. Los valores válidos tienen formato hexadecimal 0x*RRGGBB*. El valor predeterminado es 0xFF0000.

Disponibilidad: ActionScript 1.0; Flash Player 8

Ejemplo

El ejemplo siguiente cambia la propiedad `color` de un clip de película existente cuando un usuario hace clic en ella.

```
import flash.filters.GlowFilter;

var mc:MovieClip = createGlowFilterRectangle("GlowFilterColor");
mc.onRelease = function() {
```

```
  var filter:GlowFilter = this.filters[0];
  filter.color = 0x00FF33;
  this.filters = new Array(filter);
}

function createGlowFilterRectangle(name:String):MovieClip {
  var rect:MovieClip = this.createEmptyMovieClip(name,
  this.getNextHighestDepth());
  var w:Number = 100;
  var h:Number = 100;
  rect.beginFill(0x003366);
  rect.lineTo(w, 0);
  rect.lineTo(w, h);
  rect.lineTo(0, h);
  rect.lineTo(0, 0);
  rect._x = 20;
  rect._y = 20;

  var filter:GlowFilter = new GlowFilter(0x000000, .8, 16, 16, 1, 3, false,
  false);
  var filterArray:Array = new Array();
  filterArray.push(filter);
  rect.filters = filterArray;
  return rect;
}
```

Constructor GlowFilter

```
public GlowFilter([color:Number], [alpha:Number], [blurX:Number],
  [blurY:Number], [strength:Number], [quality:Number], [inner:Boolean],
  [knockout:Boolean])
```

Inicializa una instancia GlowFilter nueva con los parámetros especificados.

Disponibilidad: ActionScript 1.0; Flash Player 8

Parámetros

color:Number [opcional] - Color del iluminado, expresado en formato hexadecimal *RRGGBB*. El valor predeterminado es 0xFF0000.

alpha:Number [opcional] - Valor de transparencia alfa del color. Los valores válidos son de 0 a 1. Por ejemplo, 0,25 define un valor de transparencia de 25%. El valor predeterminado es 1.

blurX:Number [opcional] - Cantidad de desenfoque horizontal. Los valores válidos van de 0 a 255 (coma flotante). El valor predeterminado es 6. Los valores que son múltiplos de 2 (como 2, 4, 8, 16 y 32) se optimizan para representarse más rápidamente que otros valores.

blurY:Number [opcional] - Cantidad de desenfoque vertical. Los valores válidos van de 0 a 255 (coma flotante). El valor predeterminado es 6. Los valores que son múltiplos de 2 (como 2, 4, 8, 16 y 32) se optimizan para representarse más rápidamente que otros valores.

strength:Number [opcional] - Intensidad de la impresión o extensión. Cuanto más alto sea el valor, más color se imprimirá y mayor será el contraste entre el iluminado y el fondo. Los valores válidos van de 0 a 255. El valor predeterminado es 2.

quality:Number [opcional] - Número de veces que debe aplicarse el filtro. Los valores válidos van de 0 a 15. El valor predeterminado es 1, que equivale a una calidad baja. Un valor 2 equivale a calidad media y un valor 3 a alta.

inner:Boolean [opcional] - Especifica si el iluminado es interior. El valor `true` indica un iluminado interior. El valor predeterminado es `false`, un iluminado exterior (un iluminado alrededor de los bordes exteriores del objeto).

knockout:Boolean [opcional] - Especifica si el objeto tiene un efecto extractor. Con el valor `true` el relleno del objeto es transparente y el color de fondo del documento queda visible. El valor predeterminado es `false` (sin efecto extractor).

Ejemplo

El ejemplo siguiente crea una instancia de un nuevo GlowFilter y la aplica a una forma plana, rectangular.

```
import flash.filters.GlowFilter;

var rect:MovieClip = createRectangle(100, 100, 0x003366,
  "gradientGlowFilterExample");

var color:Number = 0x33CCFF;
var alpha:Number = .8;
var blurX:Number = 35;
var blurY:Number = 35;
var strength:Number = 2;
var quality:Number = 3;
var inner:Boolean = false;
var knockout:Boolean = false;

var filter:GlowFilter = new GlowFilter(color,
                        alpha,
                        blurX,
                        blurY,
                        strength,
                        quality,
                        inner,
                        knockout);
var filterArray:Array = new Array();
filterArray.push(filter);
```

```
rect.filters = filterArray;

function createRectangle(w:Number, h:Number, bgColor:Number,
  name:String):MovieClip {
  var mc:MovieClip = this.createEmptyMovieClip(name,
  this.getNextHighestDepth());
  mc.beginFill(bgColor);
  mc.lineTo(w, 0);
  mc.lineTo(w, h);
  mc.lineTo(0, h);
  mc.lineTo(0, 0);
  mc._x = 20;
  mc._y = 20;
  return mc;
}
```

inner (propiedad GlowFilter.inner)

```
public inner : Boolean
```

Especifica si el iluminado es interior. El valor `true` indica un iluminado interior. El valor predeterminado es `false`, un iluminado exterior (un iluminado alrededor de los bordes exteriores del objeto).

Disponibilidad: ActionScript 1.0; Flash Player 8

Ejemplo

El ejemplo siguiente cambia la propiedad `inner` de un clip de película existente cuando un usuario hace clic en ella.

```
import flash.filters.GlowFilter;

var mc:MovieClip = createGlowFilterRectangle("GlowFilterInner");
mc.onRelease = function() {
  var filter:GlowFilter = this.filters[0];
  filter.inner = true;
  this.filters = new Array(filter);
}

function createGlowFilterRectangle(name:String):MovieClip {
  var rect:MovieClip = this.createEmptyMovieClip(name,
  this.getNextHighestDepth());
  var w:Number = 100;
  var h:Number = 100;
  rect.beginFill(0x003366);
  rect.lineTo(w, 0);
  rect.lineTo(w, h);
  rect.lineTo(0, h);
  rect.lineTo(0, 0);
```

```
  rect._x = 20;
  rect._y = 20;

  var filter:GlowFilter = new GlowFilter(0x000000, .8, 16, 16, 1, 3, false,
  false);
  var filterArray:Array = new Array();
  filterArray.push(filter);
  rect.filters = filterArray;
  return rect;
}
```

knockout (propiedad GlowFilter.knockout)

```
public knockout : Boolean
```

Especifica si el objeto tiene un efecto extractor. Con el valor `true` el relleno del objeto es transparente y el color de fondo del documento queda visible. El valor predeterminado es `false` (sin efecto extractor).

Disponibilidad: ActionScript 1.0; Flash Player 8

Ejemplo

El ejemplo siguiente cambia la propiedad `knockout` de un clip de película existente cuando un usuario hace clic en ella.

```
import flash.filters.GlowFilter;

var mc:MovieClip = createGlowFilterRectangle("GlowFilterKnockout");
mc.onRelease = function() {
  var filter:GlowFilter = this.filters[0];
  filter.knockout = true;
  this.filters = new Array(filter);
}

function createGlowFilterRectangle(name:String):MovieClip {
  var rect:MovieClip = this.createEmptyMovieClip(name,
  this.getNextHighestDepth());
  var w:Number = 100;
  var h:Number = 100;
  rect.beginFill(0x003366);
  rect.lineTo(w, 0);
  rect.lineTo(w, h);
  rect.lineTo(0, h);
  rect.lineTo(0, 0);
  rect._x = 20;
  rect._y = 20;

  var filter:GlowFilter = new GlowFilter(0x000000, .8, 16, 16, 1, 3, false,
  false);
```

```
  var filterArray:Array = new Array();
  filterArray.push(filter);
  rect.filters = filterArray;
  return rect;
}
```

quality (propiedad GlowFilter.quality)

```
public quality : Number
```

Número de veces que debe aplicarse el filtro. Los valores válidos van de 0 a 15. El valor predeterminado es 1, que equivale a una calidad baja. Un valor 2 equivale a calidad media y un valor 3 a alta. Los filtros con valores más bajos se representan con mayor rapidez.

Para la mayoría de las aplicaciones, es suficiente un valor `quality` de 1, 2 o 3. Aunque puede utilizar valores numéricos hasta 15 para conseguir efectos distintos, los valores más altos se representan más lentamente. En lugar de aumentar el valor de `quality`, habitualmente se consigue un efecto similar, y con una representación más rápida, sencillamente aumentando los valores de `blurX` y `blurY`.

Disponibilidad: ActionScript 1.0; Flash Player 8

Ejemplo

El ejemplo siguiente cambia la propiedad `quality` de un clip de película existente cuando un usuario hace clic en ella.

```
import flash.filters.GlowFilter;

var mc:MovieClip = createGlowFilterRectangle("GlowFilterQuality");
mc.onRelease = function() {
  var filter:GlowFilter = this.filters[0];
  filter.quality = 1;
  this.filters = new Array(filter);
}

function createGlowFilterRectangle(name:String):MovieClip {
  var rect:MovieClip = this.createEmptyMovieClip(name,
  this.getNextHighestDepth());
  var w:Number = 100;
  var h:Number = 100;
  rect.beginFill(0x003366);
  rect.lineTo(w, 0);
  rect.lineTo(w, h);
  rect.lineTo(0, h);
  rect.lineTo(0, 0);
  rect._x = 20;
  rect._y = 20;
```

```
  var filter:GlowFilter = new GlowFilter(0x000000, .8, 16, 16, 1, 3, false,
  false);
  var filterArray:Array = new Array();
  filterArray.push(filter);
  rect.filters = filterArray;
  return rect;
}
```

strength (propiedad GlowFilter.strength)

```
public strength : Number
```

Intensidad de la impresión o extensión. Cuanto más alto sea el valor, más color se imprimirá y mayor será el contraste entre el iluminado y el fondo. Los valores válidos van de 0 a 255. El valor predeterminado es 2.

Disponibilidad: ActionScript 1.0; Flash Player 8

Ejemplo

El ejemplo siguiente cambia la propiedad `strength` de un clip de película existente cuando un usuario hace clic en ella.

```
import flash.filters.GlowFilter;

var mc:MovieClip = createGlowFilterRectangle("GlowFilterStrength");
mc.onRelease = function() {
  var filter:GlowFilter = this.filters[0];
  filter.strength = .8;
  this.filters = new Array(filter);
}

function createGlowFilterRectangle(name:String):MovieClip {
  var rect:MovieClip = this.createEmptyMovieClip(name,
  this.getNextHighestDepth());
  var w:Number = 100;
  var h:Number = 100;
  rect.beginFill(0x003366);
  rect.lineTo(w, 0);
  rect.lineTo(w, h);
  rect.lineTo(0, h);
  rect.lineTo(0, 0);
  rect._x = 20;
  rect._y = 20;
```

```
  var filter:GlowFilter = new GlowFilter(0x000000, .8, 16, 16, 1, 3, false,
  false);
  var filterArray:Array = new Array();
  filterArray.push(filter);
  rect.filters = filterArray;
  return rect;
}
```

GradientBevelFilter (flash.filters.GradientBevelFilter)

```
Object
  |
  +-flash.filters.BitmapFilter
    |
    +-flash.filters.GradientBevelFilter
```

`public class` ***GradientBevelFilter***
`extends BitmapFilter`

La clase GradientBevelFilter permite aplicar un efecto de bisel degradado a distintos objetos de Flash. Un bisel degradado es un borde degradado, realzado con color degradado en el exterior, interior o la parte superior de un objeto. Los bordes biselados hacen que los objetos parezcan tridimensionales.

El uso de filtros depende del objeto al que se aplique el filtro:

- Para aplicar filtros a clips de película, campos de texto y botones en tiempo de ejecución, utilice la propiedad `filters`. La propiedad `filters` de un objeto no modifica el objeto y se puede deshacer el filtro borrando la propiedad `filters`.
- Para aplicar filtros a instancias de BitmapData, utilice el método `BitmapData.applyFilter()`. Cuando se llama a `applyFilter()` en un objeto BitmapData, se genera una imagen filtrada al procesar el objeto BitmapData de origen y el objeto de filtro.

También puede aplicar efectos de filtro a imágenes y vídeo durante la edición. Para más información, consulte la documentación de edición.

Si aplica un filtro a un clip de película o a un botón, la propiedad `cacheAsBitmap` del clip de película o del botón se establecerá como `true`. Si borra todos los filtros, se restaurará el valor original de `cacheAsBitmap`.

Este filtro admite la aplicación de escala en el escenario. Sin embargo, no se puede utilizar escala, rotación y sesgo general; si el propio objeto está a escala (si `_xscale` y `_yscale` no son 100%), no se aplica la escala al efecto de filtro. Sólo se ajusta a escala cuando se acerca el escenario.

El filtro no se aplica si la imagen resultante supera los 2880 píxeles de anchura o altura. Por ejemplo, si amplía un clip de película grande al que se le ha aplicado un filtro, éste se desactiva si la imagen resultante supera el límite de 2880 píxeles.

Disponibilidad: ActionScript 1.0; Flash Player 8

Véase también

`ratios (propiedad GradientBevelFilter.ratios)`, `applyFilter (método BitmapData.applyFilter)`, `BevelFilter (flash.filters.BevelFilter)`, `filters (propiedad Button.filters)`, `cacheAsBitmap (propiedad Button.cacheAsBitmap)`, `cacheAsBitmap (propiedad MovieClip.cacheAsBitmap)`, `filters (propiedad MovieClip.filters)`, `filters (propiedad TextField.filters)`

Resumen de propiedades

Modificadores	Propiedad	Descripción
	`alphas:Array`	Matriz de valores de transparencia alfa para los colores correspondientes de la matriz `colors`.
	`angle:Number`	Ángulo, expresado en grados.
	`blurX:Number`	Cantidad de desenfoque horizontal.
	`blurY:Number`	Cantidad de desenfoque vertical.
	`colors:Array`	Matriz de valores de colores hexadecimales RGB que se van a utilizar en el degradado.
	`distance:Number`	Distancia de desplazamiento.
	`knockout:Boolean`	Especifica si el objeto tiene un efecto extractor.
	`quality:Number`	Número de veces que debe aplicarse el filtro.
	`ratios:Array`	Matriz de proporciones de distribución de color para los colores correspondientes de la matriz `colors`.
	`strength:Number`	Intensidad de la impresión o extensión.
	`type:String`	Colocación del efecto de bisel.

Propiedades heredadas de la clase Object

```
constructor (propiedad Object.constructor), __proto__ (propiedad
Object.__proto__), prototype (propiedad Object.prototype), __resolve
(propiedad Object.__resolve)
```

Resumen de constructores

Firma	Descripción
`GradientBevelFilter( [distance:Number], [angle:Number], [colors:Array], [alphas:Array], [ratios:Array], [blurX:Number], [blurY:Number], [strength:Number], [quality:Number], [type:String], [knockout:Boolean])`	Inicializa el filtro con los parámetros especificados.

Resumen de métodos

Modificadores	Firma	Descripción
	`clone() : GradientBevelFilter`	Devuelve una copia de este objeto de filtro.

Métodos heredados de la clase BitmapFilter

```
clone (método BitmapFilter.clone)
```

Métodos heredados de la clase Object

```
addProperty (método Object.addProperty), hasOwnProperty (método
Object.hasOwnProperty), isPropertyEnumerable (método
Object.isPropertyEnumerable), isPrototypeOf (método Object.isPrototypeOf),
registerClass (método Object.registerClass), toString (método
Object.toString), unwatch (método Object.unwatch), valueOf (método
Object.valueOf), watch (método Object.watch)
```

alphas (propiedad GradientBevelFilter.alphas)

```
public alphas : Array
```

Matriz de valores de transparencia alfa para los colores correspondientes de la matriz `colors`. Los valores válidos para cada elemento de la matriz son de 0 a 1. Por ejemplo, 0,25 define un valor de transparencia de 25%.

La propiedad `alphas` no puede cambiarse directamente mediante la modificación de estos valores. Debe obtener una referencia a `alphas`, realizar el cambio en la referencia y definir `alphas` como la referencia.

Las propiedades `colors`, `alphas` y `ratios` están relacionadas. El primer elemento de la matriz `colors` corresponde al primer elemento de la matriz `alphas` en la matriz `ratios` y así sucesivamente.

Disponibilidad: ActionScript 1.0; Flash Player 8

Ejemplo

El ejemplo siguiente demuestra cómo establecer la propiedad `alphas` en una entidad existente.

```
import flash.filters.GradientBevelFilter;

var mc:MovieClip = setUpFilter("alphasExample");
mc.onPress = function() {
  var arr:Array = this.filters;
  var alphas:Array = [.2, 0, .2];
  arr[0].alphas = alphas;
  this.filters = arr;
}
mc.onRelease = function() {
  var arr:Array = this.filters;
  var alphas:Array = [1, 0, 1];
  arr[0].alphas = alphas;
  this.filters = arr;
}

function setUpFilter(name:String):MovieClip {
  var art:MovieClip = this.createEmptyMovieClip(name,
  this.getNextHighestDepth());
  var w:Number = 150;
  var h:Number = 150;
  art.beginFill(0xCCCCCC);
  art.lineTo(w, 0);
  art.lineTo(w, h);
  art.lineTo(0, h);
  art.lineTo(0, 0);
```

```
  var colors:Array = [0xFFFFFF, 0xCCCCCC, 0x000000];
  var alphas:Array = [1, 0, 1];
  var ratios:Array = [0, 128, 255];
  var filter:GradientBevelFilter = new GradientBevelFilter(5, 225, colors,
  alphas, ratios, 5, 5, 5, 2, "inner", false);

  art.filters = new Array(filter);
  return art;
}
```

Véase también

colors (propiedad GradientBevelFilter.colors), ratios (propiedad GradientBevelFilter.ratios)

angle (propiedad GradientBevelFilter.angle)

```
public angle : Number
```

Ángulo, expresado en grados. Los valores válidos van de 0 a 360. El valor predeterminado es 45.

El valor de angle representa el ángulo de la fuente de luz que en teoría cae sobre el objeto. El valor determina el ángulo en el que se aplican al objeto los colores de degradado: donde aparecen la luz y la sombra o donde aparece el primer color de la matriz. A continuación se aplican los colores en el orden en el que aparecen en la matriz.

Disponibilidad: ActionScript 1.0; Flash Player 8

Ejemplo

El ejemplo siguiente demuestra cómo establecer la propiedad `angle` en un objeto existente.

```
import flash.filters.GradientBevelFilter;

var mc:MovieClip = setUpFilter("angleExample");
mc.onRelease = function() {
  var arr:Array = this.filters;
  arr[0].angle = 45;
  this.filters = arr;
}

function setUpFilter(name:String):MovieClip {
  var art:MovieClip = this.createEmptyMovieClip(name,
  this.getNextHighestDepth());
  var w:Number = 150;
  var h:Number = 150;
  art.beginFill(0xCCCCCC);
  art.lineTo(w, 0);
  art.lineTo(w, h);
```

```
  art.lineTo(0, h);
  art.lineTo(0, 0);

  var colors:Array = [0xFFFFFF, 0xCCCCCC, 0x000000];
  var alphas:Array = [1, 0, 1];
  var ratios:Array = [0, 128, 255];
  var filter:GradientBevelFilter = new GradientBevelFilter(5, 225, colors,
  alphas, ratios, 5, 5, 5, 3, "inner", false);

  art.filters = new Array(filter);
  return art;
}
```

Véase también

ratios (propiedad GradientBevelFilter.ratios)

blurX (propiedad GradientBevelFilter.blurX)

```
public blurX : Number
```

Cantidad de desenfoque horizontal. Los valores válidos son de 0 a 255. Un desenfoque de 1 o inferior significa que la imagen original se copia tal cual. El valor predeterminado es 4. Los valores que son múltiplos de 2 (como 2, 4, 8, 16 y 32) se optimizan para representarse más rápidamente que otros valores.

Disponibilidad: ActionScript 1.0; Flash Player 8

Ejemplo

El ejemplo siguiente demuestra cómo establecer la propiedad `blurX` en un objeto existente.

```
import flash.filters.GradientBevelFilter;

var mc:MovieClip = setUpFilter("blurXExample");
mc.onRelease = function() {
  var arr:Array = this.filters;
  arr[0].blurX = 16;
  this.filters = arr;
}

function setUpFilter(name:String):MovieClip {
  var art:MovieClip = this.createEmptyMovieClip(name,
  this.getNextHighestDepth());
  var w:Number = 150;
  var h:Number = 150;
  art.beginFill(0xCCCCCC);
  art.lineTo(w, 0);
  art.lineTo(w, h);
  art.lineTo(0, h);
```

```
  art.lineTo(0, 0);

  var colors:Array = [0xFFFFFF, 0xCCCCCC, 0x000000];
  var alphas:Array = [1, 0, 1];
  var ratios:Array = [0, 128, 255];
  var filter:GradientBevelFilter = new GradientBevelFilter(5, 225, colors,
  alphas, ratios, 5, 5, 5, 3, "inner", false);

  art.filters = new Array(filter);
  return art;
}
```

blurY (propiedad GradientBevelFilter.blurY)

```
public blurY : Number
```

Cantidad de desenfoque vertical. Los valores válidos son de 0 a 255. Un desenfoque de 1 o inferior significa que la imagen original se copia tal cual. El valor predeterminado es 4. Los valores que son múltiplos de 2 (como 2, 4, 8, 16 y 32) se optimizan para representarse más rápidamente que otros valores.

Disponibilidad: ActionScript 1.0; Flash Player 8

Ejemplo

El ejemplo siguiente demuestra cómo establecer la propiedad `blurY` en un objeto existente.

```
import flash.filters.GradientBevelFilter;

var mc:MovieClip = setUpFilter("blurYExample");
mc.onRelease = function() {
  var arr:Array = this.filters;
  arr[0].blurY = 16;
  this.filters = arr;
}

function setUpFilter(name:String):MovieClip {
  var art:MovieClip = this.createEmptyMovieClip(name,
  this.getNextHighestDepth());
  var w:Number = 150;
  var h:Number = 150;
  art.beginFill(0xCCCCCC);
  art.lineTo(w, 0);
  art.lineTo(w, h);
  art.lineTo(0, h);
  art.lineTo(0, 0);

  var colors:Array = [0xFFFFFF, 0xCCCCCC, 0x000000];
  var alphas:Array = [1, 0, 1];
  var ratios:Array = [0, 128, 255];
```

```
  var filter:GradientBevelFilter = new GradientBevelFilter(5, 225, colors,
  alphas, ratios, 5, 5, 5, 3, "inner", false);

  art.filters = new Array(filter);
  return art;
}
```

clone (método GradientBevelFilter.clone)

```
public clone() : GradientBevelFilter
```

Devuelve una copia de este objeto de filtro.

Disponibilidad: ActionScript 1.0; Flash Player 8

Valor devuelto

`flash.filters.GradientBevelFilter` - Instancia GradientBevelFilter nueva con las mismas propiedades que la instancia GradientBevelFilter original.

Ejemplo

El ejemplo siguiente crea dos formas rectangulares. La primera, `sourceClip`, tiene un efecto de bisel. La segunda, `resultClip`, no tiene ningún efecto hasta que se hace clic en ella.

```
import flash.filters.GradientBevelFilter;

var sourceClip:MovieClip = setUpFlatRectangle(150, 150, 0xCCCCCC,
  "cloneSourceClip");
var resultClip:MovieClip = setUpFlatRectangle(150, 150, 0xCCCCCC,
  "cloneResultClip");

resultClip.source = sourceClip;

var sourceFilter:GradientBevelFilter = getNewFilter();
sourceClip.filters = new Array(sourceFilter);

resultClip._x = 180;
resultClip.onRelease = function() {
  this.filters = new Array(this.source.filters[0].clone());
}

function setUpFlatRectangle(w:Number, h:Number, bgColor:Number,
  name:String):MovieClip {
  var mc:MovieClip = this.createEmptyMovieClip(name,
  this.getNextHighestDepth());
  mc.beginFill(bgColor);
  mc.lineTo(w, 0);
  mc.lineTo(w, h);
  mc.lineTo(0, h);
```

```
  mc.lineTo(0, 0);
  return mc;
}

function getNewFilter():GradientBevelFilter {
  var colors:Array = [0xFFFFFF, 0xCCCCCC, 0x000000];
  var alphas:Array = [1, 0, 1];
  var ratios:Array = [0, 128, 255];
  return new GradientBevelFilter(5, 225, colors, alphas, ratios, 5, 5, 5,
  2, "inner", false);
}
```

colors (propiedad GradientBevelFilter.colors)

```
public colors : Array
```

Matriz de valores de colores hexadecimales RGB que se van a utilizar en el degradado. Por ejemplo, rojo es 0xFF0000, azul es 0x0000FF, y así sucesivamente.

La propiedad `colors` no puede cambiarse directamente mediante la modificación de estos valores. Debe obtener una referencia a `colors`, realizar el cambio en la referencia y definir `colors` como la referencia.

Las propiedades `colors`, `alphas` y `ratios` están relacionadas. El primer elemento de la matriz `colors` corresponde al primer elemento de la matriz `alphas` en la matriz `ratios` y así sucesivamente.

Disponibilidad: ActionScript 1.0; Flash Player 8

Ejemplo

El ejemplo siguiente demuestra cómo establecer la propiedad `colors` en una entidad existente.

```
import flash.filters.GradientBevelFilter;

var mc:MovieClip = setUpFilter("colorsExample");
mc.onPress = function() {
  var arr:Array = this.filters;
  var colors:Array = [0x000000, 0xCCCCCC, 0xFFFFFF];
  arr[0].colors = colors;
  this.filters = arr;
}
mc.onRelease = function() {
  var arr:Array = this.filters;
  var colors:Array = [0xFFFFFF, 0xCCCCCC, 0x000000];
  arr[0].colors = colors;
  this.filters = arr;
}
```

```
function setUpFilter(name:String):MovieClip {
  var art:MovieClip = this.createEmptyMovieClip(name,
  this.getNextHighestDepth());
  var w:Number = 150;
  var h:Number = 150;
  art.beginFill(0xCCCCCC);
  art.lineTo(w, 0);
  art.lineTo(w, h);
  art.lineTo(0, h);
  art.lineTo(0, 0);

  var colors:Array = [0xFFFFFF, 0xCCCCCC, 0x000000];
  var alphas:Array = [1, 0, 1];
  var ratios:Array = [0, 128, 255];
  var filter:GradientBevelFilter = new GradientBevelFilter(5, 225, colors,
  alphas, ratios, 5, 5, 5, 2, "inner", false);

  art.filters = new Array(filter);
  return art;
}
```

Véase también

`alphas (propiedad GradientBevelFilter.alphas)`, `ratios (propiedad GradientBevelFilter.ratios)`

distance (propiedad GradientBevelFilter.distance)

```
public distance : Number
```

Distancia de desplazamiento. El valor predeterminado es 4.

Disponibilidad: ActionScript 1.0; Flash Player 8

Ejemplo

El ejemplo siguiente demuestra cómo establecer la propiedad `distance` en un objeto existente.

```
import flash.filters.GradientBevelFilter;

var mc:MovieClip = setUpFilter("distanceExample");
mc.onRelease = function() {
  var arr:Array = this.filters;
  arr[0].distance = 1;
  this.filters = arr;
}

function setUpFilter(name:String):MovieClip {
```

```
  var art:MovieClip = this.createEmptyMovieClip(name,
  this.getNextHighestDepth());
  var w:Number = 150;
  var h:Number = 150;
  art.beginFill(0xCCCCCC);
  art.lineTo(w, 0);
  art.lineTo(w, h);
  art.lineTo(0, h);
  art.lineTo(0, 0);

  var colors:Array = [0xFFFFFF, 0xCCCCCC, 0x000000];
  var alphas:Array = [1, 0, 1];
  var ratios:Array = [0, 128, 255];
  var filter:GradientBevelFilter = new GradientBevelFilter(5, 225, colors,
  alphas, ratios, 5, 5, 5, 3, "inner", false);

  art.filters = new Array(filter);
  return art;
}
```

Constructor GradientBevelFilter

```
public GradientBevelFilter([distance:Number], [angle:Number],
  [colors:Array], [alphas:Array], [ratios:Array], [blurX:Number],
  [blurY:Number], [strength:Number], [quality:Number], [type:String],
  [knockout:Boolean])
```

Inicializa el filtro con los parámetros especificados.

Disponibilidad: ActionScript 1.0; Flash Player 8

Parámetros

distance:Number [opcional] - Distancia de desplazamiento. Los valores válidos son de 0 a 8. El valor predeterminado es 4.

angle:Number [opcional] - Ángulo, expresado en grados. Los valores válidos van de 0 a 360. El valor predeterminado es 45.

colors:Array [opcional] - Matriz de valores de colores hexadecimales RGB que se van a utilizar en el degradado. Por ejemplo, rojo es 0xFF0000, azul es 0x0000FF, y así sucesivamente.

alphas:Array [opcional] - Matriz de valores de transparencia alfa para los colores correspondientes de la matriz colors. Los valores válidos para cada elemento de la matriz son de 0 a 1. Por ejemplo, 0,25 define un valor de transparencia de 25%.

ratios:Array [opcional] - Una matriz de proporciones de distribución de color; los valores válidos van de 0 a 255.

blurX:Number [opcional] - Cantidad de desenfoque horizontal. Los valores válidos son de 0 a 255. Un desenfoque de 1 o inferior significa que la imagen original se copia tal cual. El valor predeterminado es 4. Los valores que son múltiplos de 2 (como 2, 4, 8, 16 y 32) se optimizan para representarse más rápidamente que otros valores.

blurY:Number [opcional] - Cantidad de desenfoque vertical. Los valores válidos son de 0 a 255. Un desenfoque de 1 o inferior significa que la imagen original se copia tal cual. El valor predeterminado es 4. Los valores que son múltiplos de 2 (como 2, 4, 8, 16 y 32) se optimizan para representarse más rápidamente que otros valores.

strength:Number [opcional] - Intensidad de la impresión o extensión. Cuanto más alto sea el valor, más color se imprimirá y mayor será el contraste entre el bisel y el fondo. Los valores válidos son de 0 a 255. Un valor de 0 significa que no se aplica el filtro. El valor predeterminado es 1.

quality:Number [opcional] - Calidad del filtro. Los valores válidos son de 0 a 15. El valor predeterminado es 1. En casi todos los casos, los valores más utilizados son 1 (calidad baja), 2 (calidad media) y 3 (calidad alta). Los filtros con valores más bajos se representan con mayor rapidez.

type:String [opcional] - Colocación del efecto de bisel. Los valores posibles son:

- "outer": Bisel en el lado exterior del objeto.
- "inner": Bisel en el lado interior del objeto.
- "full": Bisel en la parte superior del objeto.

El valor predeterminado es "inner".

knockout:Boolean [opcional] - Especifica si se aplica el efecto extractor. Con el valor true el relleno del objeto es transparente y el color de fondo del documento queda visible. El valor predeterminado es false (sin extractor).

Ejemplo

El ejemplo siguiente crea una nueva instancia GradientBevelFilter, asigna sus valores y la aplica a una imagen rectangular plana.

```
import flash.filters.GradientBevelFilter;
import flash.filters.BitmapFilter;
var art:MovieClip = setUpFlatRectangle(150, 150, 0xCCCCCC,
  "gradientBevelFilterExample");
var distance:Number = 5;
var angleInDegrees:Number = 225; // opposite 45 degrees
var colors:Array = [0xFFFFFF, 0xCCCCCC, 0x000000];
var alphas:Array = [1, 0, 1];
var ratios:Array = [0, 128, 255];
var blurX:Number = 8;
var blurY:Number = 8;
```

```
var strength:Number = 2;
var quality:Number = 3;
var type:String = "inner";
var knockout:Boolean = true;

var filter:GradientBevelFilter = new GradientBevelFilter(distance,
                                 angleInDegrees,
                                 colors,
                                 alphas,
                                 ratios,
                                 blurX,
                                 blurY,
                                 strength,
                                 quality,
                                 type,
                                 knockout);
var filterArray:Array = new Array();
filterArray.push(filter);
art.filters = filterArray;

function setUpFlatRectangle(w:Number, h:Number, bgColor:Number,
  name:String):MovieClip {
  var mc:MovieClip = this.createEmptyMovieClip(name,
  this.getNextHighestDepth());
  mc.beginFill(bgColor);
  mc.lineTo(w, 0);
  mc.lineTo(w, h);
  mc.lineTo(0, h);
  mc.lineTo(0, 0);
  return mc;
}
```

Véase también

`ratios (propiedad GradientBevelFilter.ratios)`

knockout (propiedad GradientBevelFilter.knockout)

`public knockout : Boolean`

Especifica si el objeto tiene un efecto extractor. El efecto extractor hace que el relleno del objeto sea transparente y el color de fondo del documento quede visible. El valor `true` especifica un efecto extractor; el valor predeterminado es `false` (sin efecto recorte).

Disponibilidad: ActionScript 1.0; Flash Player 8

Ejemplo

El ejemplo siguiente demuestra cómo establecer la propiedad `knockout` en un objeto existente.

```
import flash.filters.GradientBevelFilter;

var mc:MovieClip = setUpFilter("knockoutExample");
mc.onRelease = function() {
  var arr:Array = this.filters;
  arr[0].knockout = true;
  this.filters = arr;
}

function setUpFilter(name:String):MovieClip {
  var art:MovieClip = this.createEmptyMovieClip(name,
  this.getNextHighestDepth());
  var w:Number = 150;
  var h:Number = 150;
  art.beginFill(0xCCCCCC);
  art.lineTo(w, 0);
  art.lineTo(w, h);
  art.lineTo(0, h);
  art.lineTo(0, 0);

  var colors:Array = [0xFFFFFF, 0xCCCCCC, 0x000000];
  var alphas:Array = [1, 0, 1];
  var ratios:Array = [0, 128, 255];
  var filter:GradientBevelFilter = new GradientBevelFilter(5, 225, colors,
  alphas, ratios, 5, 5, 5, 3, "inner", false);

  art.filters = new Array(filter);
  return art;
}
```

quality (propiedad GradientBevelFilter.quality)

```
public quality : Number
```

Número de veces que debe aplicarse el filtro. Los valores válidos van de 0 a 15. El valor predeterminado es 1, que equivale a una calidad baja. Un valor 2 equivale a calidad media y un valor 3 a alta. Los filtros con valores más bajos se representan con mayor rapidez.

Para la mayoría de las aplicaciones, es suficiente un valor `quality` de 1, 2 o 3. Aunque puede utilizar valores numéricos hasta 15 para conseguir efectos distintos, los valores más altos se representan más lentamente. En lugar de aumentar el valor de `quality`, habitualmente se consigue un efecto similar, y con una representación más rápida, sencillamente aumentando los valores de `blurX` y `blurY`.

Disponibilidad: ActionScript 1.0; Flash Player 8

Ejemplo

El ejemplo siguiente demuestra cómo establecer la propiedad `quality` en un objeto existente.

```
import flash.filters.GradientBevelFilter;

var mc:MovieClip = setUpFilter("qualityExample");
mc.onRelease = function() {
  var arr:Array = this.filters;
  arr[0].quality = 1; // low quality
  this.filters = arr;
}

function setUpFilter(name:String):MovieClip {
  var art:MovieClip = this.createEmptyMovieClip(name,
  this.getNextHighestDepth());
  var w:Number = 150;
  var h:Number = 150;
  art.beginFill(0xCCCCCC);
  art.lineTo(w, 0);
  art.lineTo(w, h);
  art.lineTo(0, h);
  art.lineTo(0, 0);

  var colors:Array = [0xFFFFFF, 0xCCCCCC, 0x000000];
  var alphas:Array = [1, 0, 1];
  var ratios:Array = [0, 128, 255];
  var filter:GradientBevelFilter = new GradientBevelFilter(5, 225, colors,
  alphas, ratios, 5, 5, 5, 3, "inner", false);

  art.filters = new Array(filter);
  return art;
}
```

Véase también

`ratios (propiedad GradientBevelFilter.ratios)`

ratios (propiedad GradientBevelFilter.ratios)

```
public ratios : Array
```

Matriz de proporciones de distribución de color para los colores correspondientes de la matriz `colors`. Los valores válidos para cada elemento de la matriz van de 0 a 255.

La propiedad `ratios` no puede cambiarse directamente mediante la modificación de estos valores. Debe obtener una referencia a `ratios`, realizar el cambio en la referencia y definir `ratios` como la referencia.

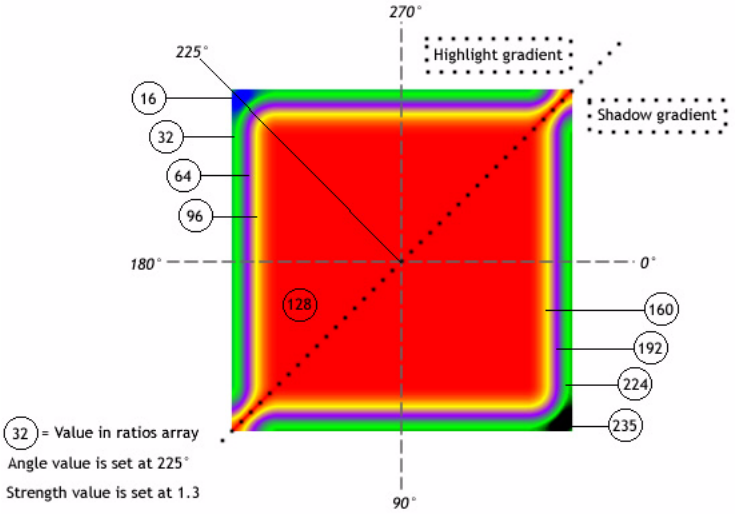
Las propiedades `colors`, `alphas` y `ratios` están relacionadas. El primer elemento de la matriz `colors` corresponde al primer elemento de la matriz `alphas` en la matriz `ratios` y así sucesivamente.

Para entender cómo se distribuyen los colores en un bisel degradado, piense primero en los colores que desea incluir en el bisel. Un bisel sencillo tiene un color de resaltado y un color de sombra; un bisel degradado tiene un degradado de resaltado y un degradado de sombra. Suponga que el resaltado aparece en la esquina superior izquierda y la sombra en la esquina inferior derecha. Un filtro podría tener cuatro colores en el resaltado y cuatro en la sombra. Además del resaltado y la sombra, el filtro utiliza un color de relleno base que aparece donde se encuentran los bordes del resaltado y de la sombra. Por lo tanto, el número total de colores es nueve y el número correspondiente de elementos en la matriz de proporciones es nueve.

En un degradado compuesto de franjas de varios colores combinados, el valor de proporción define la posición del color en el radio del degradado, donde 0 representa el punto en el límite exterior del degradado y 255 representa el punto más interior del degradado. En una situación típica, el valor intermedio es 128 y ése es el valor de relleno base. Para obtener el efecto de bisel que se muestra en la imagen, asigne los valores de proporciones siguientes, utilizando el ejemplo de nueve colores:

- Los primeros cuatro colores van del 0 al 127, de forma que cada valor es superior o igual que el anterior. Este es el borde biselado resaltado.
- El quinto color (el del centro) es el relleno de base, definido como 128. El valor de píxel 128 define el relleno de base, que aparece fuera de la forma (y alrededor de los bordes biselados) si el tipo es exterior; o dentro de la forma, cubriendo el relleno del propio objeto, si el tipo seleccionado es interior.
- Los últimos cuatro colores van del 129 al 255, de forma que cada valor es superior o igual que el anterior. Este es el borde biselado de sombra.

Si desea obtener una distribución uniforme de los colores en cada borde, utilice un número impar de colores, con el color central como relleno de base. Distribuya los valores entre 0 y 127, y 129 y 255 de forma uniforme entre sus colores, a continuación, ajuste el valor para cambiar la anchura de cada franja de color del degradado. Para obtener un bisel degradado de nueve colores, una matriz posible sería [16, 32, 64, 96, 128, 160, 192, 224, 235]. La imagen siguiente representa el bisel degradado descrito:



Tenga en cuenta que la extensión de los colores del degradado varía en función de los valores de las propiedades `blurX`, `blurY`, `strength` y `quality`, así como de los valores de `ratios`.

Disponibilidad: ActionScript 1.0; Flash Player 8

Ejemplo

El ejemplo siguiente muestra cómo establecer la propiedad `ratios` en una entidad existente.

```
import flash.filters.GradientBevelFilter;

var mc:MovieClip = setUpFilter("ratiosExample");
mc.onPress = function() {
  var arr:Array = this.filters;
  var ratios:Array = [127, 128, 129];
  arr[0].ratios = ratios;
  this.filters = arr;
}
mc.onRelease = function() {
  var arr:Array = this.filters;
  var ratios:Array = [0, 128, 255];
  arr[0].ratios = ratios;
```

```
  this.filters = arr;
}

function setUpFilter(name:String):MovieClip {
  var art:MovieClip = this.createEmptyMovieClip(name,
  this.getNextHighestDepth());
  var w:Number = 150;
  var h:Number = 150;
  art.beginFill(0xCCCCCC);
  art.lineTo(w, 0);
  art.lineTo(w, h);
  art.lineTo(0, h);
  art.lineTo(0, 0);

  var colors:Array = [0xFFFFFF, 0xCCCCCC, 0x000000];
  var alphas:Array = [1, 0, 1];
  var ratios:Array = [0, 128, 255];
  var filter:GradientBevelFilter = new GradientBevelFilter(5, 225, colors,
  alphas, ratios, 5, 5, 5, 2, "inner", false);

  art.filters = new Array(filter);
  return art;
}
```

Véase también

alphas (propiedad GradientBevelFilter.alphas), colors (propiedad GradientBevelFilter.colors), beginGradientFill (método MovieClip.beginGradientFill)

strength (propiedad GradientBevelFilter.strength)

```
public strength : Number
```

Intensidad de la impresión o extensión. Cuanto más alto sea el valor, más color se imprimirá y mayor será el contraste entre el bisel y el fondo. Los valores válidos son de 0 a 255. Un valor de 0 significa que no se aplica el filtro. El valor predeterminado es 1.

Disponibilidad: ActionScript 1.0; Flash Player 8

Ejemplo

El ejemplo siguiente muestra cómo establecer la propiedad `strength` en un objeto existente.

```
import flash.filters.GradientBevelFilter;

var mc:MovieClip = setUpFilter("strengthExample");
mc.onRelease = function() {
  var arr:Array = this.filters;
  arr[0].strength = 1;
```

```
  this.filters = arr;
}

function setUpFilter(name:String):MovieClip {
  var art:MovieClip = this.createEmptyMovieClip(name,
  this.getNextHighestDepth());
  var w:Number = 150;
  var h:Number = 150;
  art.beginFill(0xCCCCCC);
  art.lineTo(w, 0);
  art.lineTo(w, h);
  art.lineTo(0, h);
  art.lineTo(0, 0);

  var colors:Array = [0xFFFFFF, 0xCCCCCC, 0x000000];
  var alphas:Array = [1, 0, 1];
  var ratios:Array = [0, 128, 255];
  var filter:GradientBevelFilter = new GradientBevelFilter(5, 225, colors,
  alphas, ratios, 5, 5, 5, 3, "inner", false);

  art.filters = new Array(filter);
  return art;
}
```

Véase también

ratios (propiedad GradientBevelFilter.ratios)

type (propiedad GradientBevelFilter.type)

```
public type : String
```

Colocación del efecto de bisel. Los valores posibles son:

- `"outer"`: Bisel en el lado exterior del objeto.
- `"inner"`: Bisel en el lado interior del objeto.
- `"full"`: Bisel en la parte superior del objeto.

El valor predeterminado es `"inner"`.

Disponibilidad: ActionScript 1.0; Flash Player 8

Ejemplo

El ejemplo siguiente muestra cómo establecer la propiedad `type` en un objeto existente.

```
import flash.filters.GradientBevelFilter;

var mc:MovieClip = setUpFilter("typeExample");
mc.onRelease = function() {
  var arr:Array = this.filters;
```

```
  arr[0].type = "outer";
  this.filters = arr;
}

function setUpFilter(name:String):MovieClip {
  var art:MovieClip = this.createEmptyMovieClip(name,
  this.getNextHighestDepth());
  var w:Number = 150;
  var h:Number = 150;
  art.beginFill(0xCCCCCC);
  art.lineTo(w, 0);
  art.lineTo(w, h);
  art.lineTo(0, h);
  art.lineTo(0, 0);

  var colors:Array = [0xFFFFFF, 0xCCCCCC, 0x000000];
  var alphas:Array = [1, 0, 1];
  var ratios:Array = [0, 128, 255];
  var filter:GradientBevelFilter = new GradientBevelFilter(5, 225, colors,
  alphas, ratios, 5, 5, 5, 3, "inner", false);

  art.filters = new Array(filter);
  return art;
}
```

GradientGlowFilter (flash.filters.GradientGlowFilter)

```
Object
  |
  +-flash.filters.BitmapFilter
    |
    +-flash.filters.GradientGlowFilter
```

`public class` ***GradientGlowFilter***
`extends BitmapFilter`

La clase GradientGlowFilter permite aplicar un efecto de iluminado degradado en diversos objetos en Flash. Un iluminado degradado es un iluminado de aspecto realista con un degradado de color que se puede controlar. Puede aplicar un iluminado degradado alrededor del borde interior o exterior de un objeto o sobre un objeto.

El uso de filtros depende del objeto al que se aplique el filtro:

- Para aplicar filtros a clips de película, campos de texto y botones en tiempo de ejecución, utilice la propiedad `filters`. La propiedad `filters` de un objeto no modifica el objeto y se puede deshacer el filtro borrando la propiedad `filters`.

- Para aplicar filtros a instancias de BitmapData, utilice el método `BitmapData.applyFilter()`. Cuando se llama a `applyFilter()` en un objeto BitmapData, se genera una imagen filtrada al procesar el objeto BitmapData de origen y el objeto de filtro.

También puede aplicar efectos de filtro a imágenes y vídeo durante la edición. Para más información, consulte la documentación de edición.

Si aplica un filtro a un clip de película o a un botón, la propiedad `cacheAsBitmap` del clip de película o del botón se establecerá como `true`. Si borra todos los filtros, se restaurará el valor original de `cacheAsBitmap`.

Este filtro admite la aplicación de escala en el escenario. Sin embargo, no se puede utilizar escala, rotación y sesgo general; si el propio objeto está a escala (si `_xscale` y `_yscale` no son 100%), no se aplica la escala al efecto de filtro. Sólo se ajusta a escala cuando se acerca el escenario.

El filtro no se aplica si la imagen resultante supera los 2880 píxeles de anchura o altura. Por ejemplo, si amplía un clip de película grande al que se le ha aplicado un filtro, éste se desactiva si la imagen resultante supera el límite de 2880 píxeles.

Disponibilidad: ActionScript 1.0; Flash Player 8

Véase también

`ratios (propiedad GradientGlowFilter.ratios)`, `applyFilter (método BitmapData.applyFilter)`, `cacheAsBitmap (propiedad Button.cacheAsBitmap)`, `filters (propiedad Button.filters)`, `GlowFilter (flash.filters.GlowFilter)`, `cacheAsBitmap (propiedad MovieClip.cacheAsBitmap)`, `filters (propiedad MovieClip.filters)`, `filters (propiedad TextField.filters)`

Resumen de propiedades

Modificadores	Propiedad	Descripción
	`alphas:Array`	Matriz de valores de transparencia alfa para los colores correspondientes de la matriz `colors`.
	`angle:Number`	Ángulo, expresado en grados.
	`blurX:Number`	Cantidad de desenfoque horizontal.
	`blurY:Number`	Cantidad de desenfoque vertical.
	`colors:Array`	Conjunto de colores que define un degradado.
	`distance:Number`	Distancia de desplazamiento del iluminado.
	`knockout:Boolean`	Especifica si el objeto tiene un efecto extractor.

Modificadores	Propiedad	Descripción
	`quality:Number`	Número de veces que debe aplicarse el filtro.
	`ratios:Array`	Matriz de proporciones de distribución de color para los colores correspondientes de la matriz `colors`.
	`strength:Number`	Intensidad de la impresión o extensión.
	`type:String`	Colocación del efecto de filtro.

Propiedades heredadas de la clase Object

`constructor (propiedad Object.constructor)`, `__proto__ (propiedad Object.__proto__)`, `prototype (propiedad Object.prototype)`, `__resolve (propiedad Object.__resolve)`

Resumen de constructores

Firma	Descripción
`GradientGlowFilter([distance:Number], [angle:Number], [colors:Array], [alphas:Array], [ratios:Array], [blurX:Number], [blurY:Number], [strength:Number], [quality:Number], [type:String], [knockout:Boolean])`	Inicializa el filtro con los parámetros especificados.

Resumen de métodos

Modificadores	Firma	Descripción
	`clone() : GradientGlowFilter`	Devuelve una copia de este objeto de filtro.

Métodos heredados de la clase BitmapFilter

`clone (método BitmapFilter.clone)`

addProperty (método Object.addProperty), hasOwnProperty (método Object.hasOwnProperty), isPropertyEnumerable (método Object.isPropertyEnumerable), isPrototypeOf (método Object.isPrototypeOf), registerClass (método Object.registerClass), toString (método Object.toString), unwatch (método Object.unwatch), valueOf (método Object.valueOf), watch (método Object.watch)

alphas (propiedad GradientGlowFilter.alphas)

```
public alphas : Array
```

Matriz de valores de transparencia alfa para los colores correspondientes de la matriz `colors`. Los valores válidos para cada elemento de la matriz son de 0 a 1. Por ejemplo, 0,25 define el valor de transparencia alfa como 25%.

La propiedad `alphas` no puede cambiarse directamente mediante la modificación de estos valores. Debe obtener una referencia a `alphas`, realizar el cambio en la referencia y definir `alphas` como la referencia.

Las propiedades `colors`, `alphas` y `ratios` están relacionadas. El primer elemento de la matriz `colors` corresponde al primer elemento de la matriz `alphas` en la matriz `ratios` y así sucesivamente.

Disponibilidad: ActionScript 1.0; Flash Player 8

Ejemplo

El ejemplo siguiente cambia la propiedad `alphas` de un clip de película existente cuando un usuario hace clic en ella.

```
import flash.filters.GradientGlowFilter;
var mc:MovieClip = createGradientGlowRectangle("GlowAlphas");
mc.onRelease = function() {
  var filter:GradientGlowFilter = this.filters[0];
  var alphas:Array = filter.alphas;
  alphas.pop();
  alphas.pop();
  alphas.push(.3);
  alphas.push(1);
  filter.alphas = alphas;
  this.filters = new Array(filter);
}

function createGradientGlowRectangle(name:String):MovieClip {
  var art:MovieClip = this.createEmptyMovieClip(name,
  this.getNextHighestDepth());
```

```
  var w:Number = 100;
  var h:Number = 100;
  art.beginFill(0x003366);
  art.lineTo(w, 0);
  art.lineTo(w, h);
  art.lineTo(0, h);
  art.lineTo(0, 0);
  art._x = 20;
  art._y = 20;

  var colors:Array = [0xFFFFFF, 0xFF0000, 0xFFFF00, 0x00CCFF];
  var alphas:Array = [0, 1, 1, 1];
  var ratios:Array = [0, 63, 126, 255];
  var filter:GradientGlowFilter = new GradientGlowFilter(0, 45, colors,
  alphas, ratios, 55, 55, 2.5, 2, "outer", false);
  var filterArray:Array = new Array();
  filterArray.push(filter);
  art.filters = filterArray;
  return art;
}
```

Véase también

colors (propiedad GradientGlowFilter.colors), ratios (propiedad GradientGlowFilter.ratios)

angle (propiedad GradientGlowFilter.angle)

```
public angle : Number
```

Ángulo, expresado en grados. Los valores válidos van de 0 a 360. El valor predeterminado es 45.

El valor de angle representa el ángulo de la fuente de luz teórica que ilumina el objeto y determina la posición del efecto con respecto al objeto Si se define `distance` como 0, el efecto no se desplaza del objeto, por lo que la propiedad `angle` no tiene ningún efecto.

Disponibilidad: ActionScript 1.0; Flash Player 8

Ejemplo

El ejemplo siguiente cambia la propiedad `angle` de un clip de película existente cuando un usuario hace clic en ella.

```
import flash.filters.GradientGlowFilter;
var mc:MovieClip = createGradientGlowRectangle("GlowAngle");
mc.onRelease = function() {
  var filter:GradientGlowFilter = this.filters[0];
  filter.distance = 50;
```

```
  filter.angle = 90;
  this.filters = new Array(filter);
}

function createGradientGlowRectangle(name:String):MovieClip {
  var art:MovieClip = this.createEmptyMovieClip(name,
  this.getNextHighestDepth());
  var w:Number = 100;
  var h:Number = 100;
  art.beginFill(0x003366);
  art.lineTo(w, 0);
  art.lineTo(w, h);
  art.lineTo(0, h);
  art.lineTo(0, 0);
  art._x = 20;
  art._y = 20;

  var colors:Array = [0xFFFFFF, 0xFF0000, 0xFFFF00, 0x00CCFF];
  var alphas:Array = [0, 1, 1, 1];
  var ratios:Array = [0, 63, 126, 255];
  var filter:GradientGlowFilter = new GradientGlowFilter(0, 45, colors,
  alphas, ratios, 55, 55, 2.5, 2, "outer", false);
  var filterArray:Array = new Array();
  filterArray.push(filter);
  art.filters = filterArray;
  return art;
}
```

blurX (propiedad GradientGlowFilter.blurX)

```
public blurX : Number
```

Cantidad de desenfoque horizontal. Los valores válidos son de 0 a 255. Un desenfoque de 1 o inferior significa que la imagen original se copia tal cual. El valor predeterminado es 4. Los valores que son múltiplos de 2 (como 2, 4, 8, 16 y 32) se optimizan para representarse más rápidamente que otros valores.

Disponibilidad: ActionScript 1.0; Flash Player 8

Ejemplo

El ejemplo siguiente cambia la propiedad `blurX` de un clip de película existente cuando un usuario hace clic en ella.

```
import flash.filters.GradientGlowFilter;
var mc:MovieClip = createGradientGlowRectangle("GlowBlurX");
mc.onRelease = function() {
  var filter:GradientGlowFilter = this.filters[0];
  filter.blurX = 255;
  this.filters = new Array(filter);
```

```
}

function createGradientGlowRectangle(name:String):MovieClip {
  var art:MovieClip = this.createEmptyMovieClip(name,
  this.getNextHighestDepth());
  var w:Number = 100;
  var h:Number = 100;
  art.beginFill(0x003366);
  art.lineTo(w, 0);
  art.lineTo(w, h);
  art.lineTo(0, h);
  art.lineTo(0, 0);
  art._x = 20;
  art._y = 20;

  var colors:Array = [0xFFFFFF, 0xFF0000, 0xFFFF00, 0x00CCFF];
  var alphas:Array = [0, 1, 1, 1];
  var ratios:Array = [0, 63, 126, 255];
  var filter:GradientGlowFilter = new GradientGlowFilter(0, 45, colors,
  alphas, ratios, 55, 55, 2.5, 2, "outer", false);
  var filterArray:Array = new Array();
  filterArray.push(filter);
  art.filters = filterArray;
  return art;
}
```

blurY (propiedad GradientGlowFilter.blurY)

```
public blurY : Number
```

Cantidad de desenfoque vertical. Los valores válidos son de 0 a 255. Un desenfoque de 1 o inferior significa que la imagen original se copia tal cual. El valor predeterminado es 4. Los valores que son múltiplos de 2 (como 2, 4, 8, 16 y 32) se optimizan para representarse más rápidamente que otros valores.

Disponibilidad: ActionScript 1.0; Flash Player 8

Ejemplo

El ejemplo siguiente cambia la propiedad `blurY` de un clip de película existente cuando un usuario hace clic en ella.

```
import flash.filters.GradientGlowFilter;
var mc:MovieClip = createGradientGlowRectangle("GlowBlurY");
mc.onRelease = function() {
  var filter:GradientGlowFilter = this.filters[0];
  filter.blurY = 255;
  this.filters = new Array(filter);
}
```

```
function createGradientGlowRectangle(name:String):MovieClip {
  var art:MovieClip = this.createEmptyMovieClip(name,
  this.getNextHighestDepth());
  var w:Number = 100;
  var h:Number = 100;
  art.beginFill(0x003366);
  art.lineTo(w, 0);
  art.lineTo(w, h);
  art.lineTo(0, h);
  art.lineTo(0, 0);
  art._x = 20;
  art._y = 20;

  var colors:Array = [0xFFFFFF, 0xFF0000, 0xFFFF00, 0x00CCFF];
  var alphas:Array = [0, 1, 1, 1];
  var ratios:Array = [0, 63, 126, 255];
  var filter:GradientGlowFilter = new GradientGlowFilter(0, 45, colors,
  alphas, ratios, 55, 55, 2.5, 2, "outer", false);
  var filterArray:Array = new Array();
  filterArray.push(filter);
  art.filters = filterArray;
  return art;
}
```

clone (método GradientGlowFilter.clone)

```
public clone() : GradientGlowFilter
```

Devuelve una copia de este objeto de filtro.

Disponibilidad: ActionScript 1.0; Flash Player 8

Valor devuelto

`flash.filters.GradientGlowFilter` - Instancia GradientGlowFilter nueva con las mismas propiedades que la instancia GradientGlowFilter original.

Ejemplo

El ejemplo siguiente crea tres objetos GradientGlowFilter y los compara; `filter_1` se crea utilizando el constructor GradientGlowFilter; `filter_2` se crea con el valor de `filter_1` y `clonedFilter` se crea clonando `filter_1`. Observe que aunque `filter_2` da como resultado `filter_1`, `clonedFilter`, aunque contiene los mismos valores que `filter_1`, no es igual.

```
import flash.filters.GradientGlowFilter;

var colors:Array = [0xFFFFFF, 0xFF0000, 0xFFFF00, 0x00CCFF];
var alphas:Array = [0, 1, 1, 1];
```

```
var ratios:Array = [0, 63, 126, 255];
var filter_1:GradientGlowFilter = new GradientGlowFilter(0, 45, colors,
  alphas, ratios, 55, 55, 2.5, 2, "outer", false);
var filter_2:GradientGlowFilter = filter_1;
var clonedFilter:GradientGlowFilter = filter_1.clone();

trace(filter_1 == filter_2); // true
trace(filter_1 == clonedFilter); // false

for(var i in filter_1) {
  trace(">> " + i + ": " + filter_1[i]);
  // >> clone: [type Function]
  // >> type: outer
  // >> knockout: false
  // >> strength: 2.5
  // >> quality: 2
  // >> blurY: 55
  // >> blurX: 55
  // >> ratios: 0,63,126,255
  // >> alphas: 0,1,1,1
  // >> colors: 16777215,16711680,16776960,52479
  // >> angle: 45
  // >> distance: 0
}

for(var i in clonedFilter) {
  trace(">> " + i + ": " + clonedFilter[i]);
  // >> clone: [type Function]
  // >> type: outer
  // >> knockout: false
  // >> strength: 2.5
  // >> quality: 2
  // >> blurY: 55
  // >> blurX: 55
  // >> ratios: 0,63,126,255
  // >> alphas: 0,1,1,1
  // >> colors: 16777215,16711680,16776960,52479
  // >> angle: 45
  // >> distance: 0
}
```

Para demostrar mejor la relación entre `filter_1`, `filter_2` y `clonedFilter`, en el ejemplo siguiente se modifica la propiedad `knockout` de `filter_1`. Al modificar `knockout`, el método `clone()` crea una nueva instancia basada en los valores de `filter_1` en lugar de utilizarlos en la referencia.

```
import flash.filters.GradientGlowFilter;

var colors:Array = [0xFFFFFF, 0xFF0000, 0xFFFF00, 0x00CCFF];
var alphas:Array = [0, 1, 1, 1];
```

```
var ratios:Array = [0, 63, 126, 255];
var filter_1:GradientGlowFilter = new GradientGlowFilter(0, 45, colors,
  alphas, ratios, 55, 55, 2.5, 2, "outer", false);
var filter_2:GradientGlowFilter = filter_1;
var clonedFilter:GradientGlowFilter = filter_1.clone();

trace(filter_1.knockout); // false
trace(filter_2.knockout); // false
trace(clonedFilter.knockout); // false

filter_1.knockout = true;

trace(filter_1.knockout); // true
trace(filter_2.knockout); // true
trace(clonedFilter.knockout); // false
```

colors (propiedad GradientGlowFilter.colors)

```
public colors : Array
```

Conjunto de colores que define un degradado. Por ejemplo, rojo es 0xFF0000, azul es 0x0000FF, y así sucesivamente.

La propiedad `colors` no puede cambiarse directamente mediante la modificación de estos valores. Debe obtener una referencia a `colors`, realizar el cambio en la referencia y definir `colors` como la referencia.

Las propiedades `colors`, `alphas` y `ratios` están relacionadas. El primer elemento de la matriz `colors` corresponde al primer elemento de la matriz `alphas` en la matriz `ratios` y así sucesivamente.

Disponibilidad: ActionScript 1.0; Flash Player 8

Ejemplo

El ejemplo siguiente cambia la propiedad `colors` de un clip de película existente cuando un usuario hace clic en ella.

```
import flash.filters.GradientGlowFilter;
var mc:MovieClip = createGradientGlowRectangle("GlowColors");
mc.onRelease = function() {
  var filter:GradientGlowFilter = this.filters[0];
  var colors:Array = filter.colors;
  colors.pop();
  colors.push(0xFF00FF);
  filter.colors = colors;
  this.filters = new Array(filter);
}

function createGradientGlowRectangle(name:String):MovieClip {
```

```
  var art:MovieClip = this.createEmptyMovieClip(name,
  this.getNextHighestDepth());
  var w:Number = 100;
  var h:Number = 100;
  art.beginFill(0x003366);
  art.lineTo(w, 0);
  art.lineTo(w, h);
  art.lineTo(0, h);
  art.lineTo(0, 0);
  art._x = 20;
  art._y = 20;

  var colors:Array = [0xFFFFFF, 0xFF0000, 0xFFFF00, 0x00CCFF];
  var alphas:Array = [0, 1, 1, 1];
  var ratios:Array = [0, 63, 126, 255];
  var filter:GradientGlowFilter = new GradientGlowFilter(0, 45, colors,
  alphas, ratios, 55, 55, 2.5, 2, "outer", false);
  var filterArray:Array = new Array();
  filterArray.push(filter);
  art.filters = filterArray;
  return art;
}
```

Véase también

alphas (propiedad GradientGlowFilter.alphas), ratios (propiedad GradientGlowFilter.ratios)

distance (propiedad GradientGlowFilter.distance)

```
public distance : Number
```

Distancia de desplazamiento del iluminado. El valor predeterminado es 4.

Disponibilidad: ActionScript 1.0; Flash Player 8

Ejemplo

El ejemplo siguiente cambia la propiedad `distance` de un clip de película existente cuando un usuario hace clic en ella.

```
import flash.filters.GradientGlowFilter;
var mc:MovieClip = createGradientGlowRectangle("GlowDistance");
mc.onRelease = function() {
  var filter:GradientGlowFilter = this.filters[0];
  filter.distance = 20;
  this.filters = new Array(filter);
}
```

```
function createGradientGlowRectangle(name:String):MovieClip {
  var art:MovieClip = this.createEmptyMovieClip(name,
  this.getNextHighestDepth());
  var w:Number = 100;
  var h:Number = 100;
  art.beginFill(0x003366);
  art.lineTo(w, 0);
  art.lineTo(w, h);
  art.lineTo(0, h);
  art.lineTo(0, 0);
  art._x = 20;
  art._y = 20;

  var colors:Array = [0xFFFFFF, 0xFF0000, 0xFFFF00, 0x00CCFF];
  var alphas:Array = [0, 1, 1, 1];
  var ratios:Array = [0, 63, 126, 255];
  var filter:GradientGlowFilter = new GradientGlowFilter(0, 45, colors,
  alphas, ratios, 55, 55, 2.5, 2, "outer", false);
  var filterArray:Array = new Array();
  filterArray.push(filter);
  art.filters = filterArray;
  return art;
}
```

Constructor GradientGlowFilter

```
public GradientGlowFilter([distance:Number], [angle:Number],
  [colors:Array], [alphas:Array], [ratios:Array], [blurX:Number],
  [blurY:Number], [strength:Number], [quality:Number], [type:String],
  [knockout:Boolean])
```

Inicializa el filtro con los parámetros especificados.

Disponibilidad: ActionScript 1.0; Flash Player 8

Parámetros

distance:Number [opcional] - Distancia de desplazamiento del iluminado. El valor predeterminado es 4.

angle:Number [opcional] - Ángulo, expresado en grados. Los valores válidos van de 0 a 360. El valor predeterminado es 45.

colors:Array [opcional] - Matriz de colores que define un degradado. Por ejemplo, rojo es 0xFF0000, azul es 0x0000FF, y así sucesivamente.

alphas:Array [opcional] - Matriz de valores de transparencia alfa para los colores correspondientes de la matriz `colors`. Los valores válidos para cada elemento de la matriz van de 0 a 1. Por ejemplo, 0,25 define el valor de transparencia alfa como 25%.

ratios:Array [opcional] - Matriz de proporciones de distribución de colores. Los valores válidos van de 0 a 255. Este valor define el porcentaje de la anchura donde el color se muestrea al 100%.

blurX:Number [opcional] - Cantidad de desenfoque horizontal. Los valores válidos son de 0 a 255. Un desenfoque de 1 o inferior significa que la imagen original se copia tal cual. El valor predeterminado es 4. Los valores que son múltiplos de 2 (como 2, 4, 8, 16 y 32) se optimizan para representarse más rápidamente que otros valores.

blurY:Number [opcional] - Cantidad de desenfoque vertical. Los valores válidos son de 0 a 255. Un desenfoque de 1 o inferior significa que la imagen original se copia tal cual. El valor predeterminado es 4. Los valores que son múltiplos de 2 (como 2, 4, 8, 16 y 32) se optimizan para representarse más rápidamente que otros valores.

strength:Number [opcional] - Intensidad de la impresión o extensión. Cuanto más alto sea el valor, más color se imprimirá y mayor será el contraste entre el iluminado y el fondo. Los valores válidos van de 0 a 255. Cuanto mayor sea el valor, más intensa será la impresión. Un valor de 0 significa que el filtro no se aplica. El valor predeterminado es 1.

quality:Number [opcional] - Número de veces que debe aplicarse el filtro. Los valores válidos van de 0 a 15. El valor predeterminado es 1, que equivale a una calidad baja. Un valor 2 equivale a calidad media y un valor 3 a alta.

type:String [opcional] - Colocación del efecto de filtro. Los valores posibles son:

- `"outer"`: Iluminado en lado exterior del objeto.
- `"inner"`: Iluminado en lado interior del objeto; es el valor predeterminado.
- `"full"`: Iluminado en la parte superior del objeto.

El valor predeterminado es `"inner"`.

knockout:Boolean [opcional] - Especifica si el objeto tiene un efecto extractor. El efecto extractor hace que el relleno del objeto sea transparente y el color de fondo del documento quede visible. El valor `true` especifica un efecto extractor; el valor predeterminado es `false` (sin efecto recorte).

Ejemplo

El ejemplo siguiente crea un nuevo filtro de iluminado degradado, asigna sus valores y lo aplica a una imagen rectangular plana.

```
import flash.filters.GradientGlowFilter;
var art:MovieClip = createRectangle(100, 100, 0x003366,
  "gradientGlowFilterExample");
var distance:Number = 0;
var angleInDegrees:Number = 45;
var colors:Array = [0xFFFFFF, 0xFF0000, 0xFFFF00, 0x00CCFF];
var alphas:Array = [0, 1, 1, 1];
```

```
var ratios:Array = [0, 63, 126, 255];
var blurX:Number = 50;
var blurY:Number = 50;
var strength:Number = 2.5;
var quality:Number = 3;
var type:String = "outer";
var knockout:Boolean = false;

var filter:GradientGlowFilter = new GradientGlowFilter(distance,
                                angleInDegrees,
                                colors,
                                alphas,
                                ratios,
                                blurX,
                                blurY,
                                strength,
                                quality,
                                type,
                                knockout);
var filterArray:Array = new Array();
filterArray.push(filter);
art.filters = filterArray;

function createRectangle(w:Number, h:Number, bgColor:Number,
  name:String):MovieClip {
  var mc:MovieClip = this.createEmptyMovieClip(name,
  this.getNextHighestDepth());
  mc.beginFill(bgColor);
  mc.lineTo(w, 0);
  mc.lineTo(w, h);
  mc.lineTo(0, h);
  mc.lineTo(0, 0);
  mc._x = 20;
  mc._y = 20;
  return mc;
}
```

knockout (propiedad GradientGlowFilter.knockout)

```
public knockout : Boolean
```

Especifica si el objeto tiene un efecto extractor. El efecto extractor hace que el relleno del objeto sea transparente y el color de fondo del documento quede visible. El valor `true` especifica un efecto extractor; el valor predeterminado es `false` (sin efecto recorte).

Disponibilidad: ActionScript 1.0; Flash Player 8

Ejemplo

El ejemplo siguiente cambia la propiedad `knockout` de un clip de película existente cuando un usuario hace clic en ella.

```
import flash.filters.GradientGlowFilter;
var mc:MovieClip = createGradientGlowRectangle("GlowKnockout");
mc.onRelease = function() {
  var filter:GradientGlowFilter = this.filters[0];
  filter.knockout = true;
  this.filters = new Array(filter);
}

function createGradientGlowRectangle(name:String):MovieClip {
  var art:MovieClip = this.createEmptyMovieClip(name,
  this.getNextHighestDepth());
  var w:Number = 100;
  var h:Number = 100;
  art.beginFill(0x003366);
  art.lineTo(w, 0);
  art.lineTo(w, h);
  art.lineTo(0, h);
  art.lineTo(0, 0);
  art._x = 20;
  art._y = 20;

  var colors:Array = [0xFFFFFF, 0xFF0000, 0xFFFF00, 0x00CCFF];
  var alphas:Array = [0, 1, 1, 1];
  var ratios:Array = [0, 63, 126, 255];
  var filter:GradientGlowFilter = new GradientGlowFilter(0, 45, colors,
  alphas, ratios, 55, 55, 2.5, 2, "outer", false);
  var filterArray:Array = new Array();
  filterArray.push(filter);
  art.filters = filterArray;
  return art;
}
```

quality (propiedad GradientGlowFilter.quality)

```
public quality : Number
```

Número de veces que debe aplicarse el filtro. Los valores válidos van de 0 a 15. El valor predeterminado es 1, que equivale a una calidad baja. Un valor 2 equivale a calidad media y un valor 3 a alta. Los filtros con valores más bajos se representan con mayor rapidez.

Para la mayoría de las aplicaciones, es suficiente un valor `quality` de 1, 2 o 3. Aunque puede utilizar valores numéricos hasta 15 para conseguir efectos distintos, los valores más altos se representan más lentamente. En lugar de aumentar el valor de `quality`, habitualmente se consigue un efecto similar, y con una representación más rápida, sencillamente aumentando los valores de `blurX` y `blurY`.

Disponibilidad: ActionScript 1.0; Flash Player 8

Ejemplo

El ejemplo siguiente cambia la propiedad `quality` de un clip de película existente cuando un usuario hace clic en ella.

```
import flash.filters.GradientGlowFilter;
var mc:MovieClip = createGradientGlowRectangle("GlowQuality");
mc.onRelease = function() {
  var filter:GradientGlowFilter = this.filters[0];
  filter.quality = 3;
  this.filters = new Array(filter);
}

function createGradientGlowRectangle(name:String):MovieClip {
  var art:MovieClip = this.createEmptyMovieClip(name,
  this.getNextHighestDepth());
  var w:Number = 100;
  var h:Number = 100;
  art.beginFill(0x003366);
  art.lineTo(w, 0);
  art.lineTo(w, h);
  art.lineTo(0, h);
  art.lineTo(0, 0);
  art._x = 20;
  art._y = 20;

  var colors:Array = [0xFFFFFF, 0xFF0000, 0xFFFF00, 0x00CCFF];
  var alphas:Array = [0, 1, 1, 1];
  var ratios:Array = [0, 63, 126, 255];
  var filter:GradientGlowFilter = new GradientGlowFilter(0, 45, colors,
  alphas, ratios, 55, 55, 2.5, 2, "outer", false);
  var filterArray:Array = new Array();
  filterArray.push(filter);
  art.filters = filterArray;
  return art;
}
```

ratios (propiedad GradientGlowFilter.ratios)

```
public ratios : Array
```

Matriz de proporciones de distribución de color para los colores correspondientes de la matriz `colors`. Los valores válidos van de 0 a 255.

La propiedad `ratios` no puede cambiarse directamente mediante la modificación de estos valores. Debe obtener una referencia a `ratios`, realizar el cambio en la referencia y definir `ratios` como la referencia.

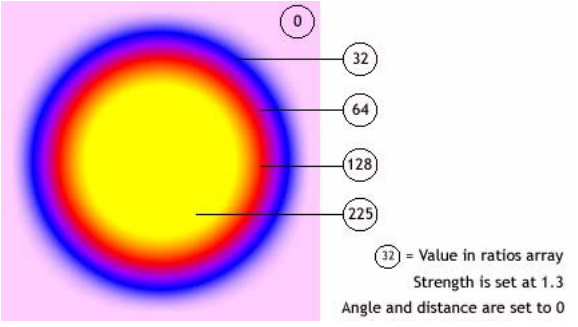
Las propiedades `colors`, `alphas` y `ratios` están relacionadas. El primer elemento de la matriz `colors` corresponde al primer elemento de la matriz `alphas` en la matriz `ratios` y así sucesivamente.

El filtro de iluminado degradado se puede considerar como un iluminado que surge del centro del objeto (si se ha definido el valor `distance` como 0), con degradados que son franjas de colores combinadas. El primer color de la matriz `colors` es el color exterior del iluminado. El último es el color interior del iluminado.

Cada valor de la matriz `ratios` define la posición del color en el radio del degradado, donde 0 representa el punto exterior del degradado y 255 representa el punto interior. Los valores de proporciones van de 0 a 255 píxeles, de forma ascendente; por ejemplo [0, 64, 128, 200, 255]. Los valores de 0 a 128 aparecen en los bordes exteriores del iluminado. Los valores del 129 al 255 aparecen en la zona interior del iluminado. Según los valores de proporciones de los colores y el valor `type` del filtro, el objeto al que se aplica el filtro puede ocultar sus colores.

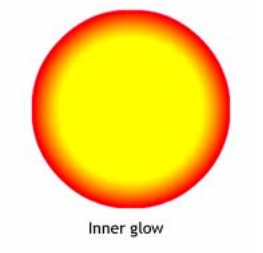
En el código y la imagen siguientes, se aplica un filtro a un clip de película circular negro, con el tipo definido como `"full"`. Para que este ejemplo resulte gráfico, el primer color de la matriz `colors`, rosa, tiene un valor de `alpha` 1, por lo que resalta con el fondo blanco del documento. (En la práctica, probablemente no elegiría el primer color con este valor). El último color de la matriz, amarillo, oculta el círculo negro al que se aplica el filtro:

```
var colors = [0xFFCCFF, 0x0000FF, 0x9900FF, 0xFF0000, 0xFFFF00]; var alphas =
[1, 1, 1, 1, 1]; var ratios = [0, 32, 64, 128, 225]; var myGGF = new
GradientGlowFilter(0, 0, colors, alphas, ratios, 50, 50, 1, 2, "full", false);
```



Para conseguir un efecto uniforme con el fondo del documento, cuando defina el valor de `type` como `"outer"` o `"full"`, defina el primer color de la matriz con el mismo color que el fondo del documento, o defina el valor alfa del primer color como 0. Con cualquiera de estas técnicas el filtro se mezcla con el fondo.

Si realiza dos pequeños cambios en el código, el efecto del iluminado puede ser muy distinto, incluso con las mismas matrices `ratios` y `colors`. Defina el valor de alfa del primer color de la matriz como 0, para que el filtro se mezcle con el fondo blanco del documento y la propiedad `type` como `"outer"` o `"inner"`. Observe el resultado, como muestran las siguientes imágenes.



Tenga en cuenta que la extensión de los colores del degradado varía en función de los valores de las propiedades `blurX`, `blurY`, `strength` y `quality`, así como de los valores de `ratios`.

Disponibilidad: ActionScript 1.0; Flash Player 8

Ejemplo

El ejemplo siguiente cambia la propiedad `ratios` de un clip de película existente cuando un usuario hace clic en ella.

```
import flash.filters.GradientGlowFilter;
var mc:MovieClip = createGradientGlowRectangle("GlowRatios");
mc.onRelease = function() {
  var filter:GradientGlowFilter = this.filters[0];
  var ratios:Array = filter.ratios;
  ratios.shift();
  ratios.unshift(40);
  filter.ratios = ratios;
  this.filters = new Array(filter);
}

function createGradientGlowRectangle(name:String):MovieClip {
  var art:MovieClip = this.createEmptyMovieClip(name,
  this.getNextHighestDepth());
  var w:Number = 100;
  var h:Number = 100;
  art.beginFill(0x003366);
  art.lineTo(w, 0);
  art.lineTo(w, h);
  art.lineTo(0, h);
  art.lineTo(0, 0);
  art._x = 20;
  art._y = 20;

  var colors:Array = [0xFFFFFF, 0xFF0000, 0xFFFF00, 0x00CCFF];
  var alphas:Array = [0, 1, 1, 1];
  var ratios:Array = [0, 63, 126, 255];
  var filter:GradientGlowFilter = new GradientGlowFilter(0, 45, colors,
  alphas, ratios, 55, 55, 2.5, 2, "outer", false);
  var filterArray:Array = new Array();
  filterArray.push(filter);
  art.filters = filterArray;
  return art;
}
```

Véase también

colors (propiedad GradientGlowFilter.colors), alphas (propiedad GradientGlowFilter.alphas), beginGradientFill (método MovieClip.beginGradientFill)

strength (propiedad GradientGlowFilter.strength)

```
public strength : Number
```

Intensidad de la impresión o extensión. Cuanto más alto sea el valor, más color se imprimirá y mayor será el contraste entre el iluminado y el fondo. Los valores válidos son de 0 a 255. Un valor de 0 significa que no se aplica el filtro. El valor predeterminado es 1.

Disponibilidad: ActionScript 1.0; Flash Player 8

Ejemplo

El ejemplo siguiente cambia la propiedad `strength` de un clip de película existente cuando un usuario hace clic en ella.

```
import flash.filters.GradientGlowFilter;
var mc:MovieClip = createGradientGlowRectangle("GlowStrength");
mc.onRelease = function() {
  var filter:GradientGlowFilter = this.filters[0];
  filter.strength = 1;
  this.filters = new Array(filter);
}

function createGradientGlowRectangle(name:String):MovieClip {
  var art:MovieClip = this.createEmptyMovieClip(name,
  this.getNextHighestDepth());
  var w:Number = 100;
  var h:Number = 100;
  art.beginFill(0x003366);
  art.lineTo(w, 0);
  art.lineTo(w, h);
  art.lineTo(0, h);
  art.lineTo(0, 0);
  art._x = 20;
  art._y = 20;

  var colors:Array = [0xFFFFFF, 0xFF0000, 0xFFFF00, 0x00CCFF];
  var alphas:Array = [0, 1, 1, 1];
  var ratios:Array = [0, 63, 126, 255];
  var filter:GradientGlowFilter = new GradientGlowFilter(0, 45, colors,
  alphas, ratios, 55, 55, 2.5, 2, "outer", false);
  var filterArray:Array = new Array();
  filterArray.push(filter);
  art.filters = filterArray;
  return art;
}
```

type (propiedad GradientGlowFilter.type)

```
public type : String
```

Colocación del efecto de filtro. Los valores posibles son:

- `"outer"`: Iluminado en lado exterior del objeto.
- `"inner"`: Iluminado en lado interior del objeto; es el valor predeterminado.
- `"full"`: Iluminado en la parte superior del objeto.

El valor predeterminado es `"inner"`.

Disponibilidad: ActionScript 1.0; Flash Player 8

Ejemplo

El ejemplo siguiente cambia la propiedad `type` de un clip de película existente cuando un usuario hace clic en ella.

```
import flash.filters.GradientGlowFilter;
var mc:MovieClip = createGradientGlowRectangle("GlowType");
mc.onRelease = function() {
  var filter:GradientGlowFilter = this.filters[0];
  filter.type = "inner";
  filter.strength = 1;
  this.filters = new Array(filter);
}

function createGradientGlowRectangle(name:String):MovieClip {
  var art:MovieClip = this.createEmptyMovieClip(name,
  this.getNextHighestDepth());
  var w:Number = 100;
  var h:Number = 100;
  art.beginFill(0x003366);
  art.lineTo(w, 0);
  art.lineTo(w, h);
  art.lineTo(0, h);
  art.lineTo(0, 0);
  art._x = 20;
  art._y = 20;

  var colors:Array = [0xFFFFFF, 0xFF0000, 0xFFFF00, 0x00CCFF];
  var alphas:Array = [0, 1, 1, 1];
  var ratios:Array = [0, 63, 126, 255];
  var filter:GradientGlowFilter = new GradientGlowFilter(0, 45, colors,
  alphas, ratios, 55, 55, 2.5, 2, "outer", false);
  var filterArray:Array = new Array();
  filterArray.push(filter);
  art.filters = filterArray;
  return art;
}
```

IME (System.IME)

```
Object
  |
  +-System.IME
```

```
public class IME
extends Object
```

La clase IME permite manipular directamente el editor de método de entrada (IME) del sistema operativo en la aplicación Flash Player que se ejecuta en un equipo cliente. Puede determinar si un IME está instalado, si está activado y qué IME está activado. Puede activar o desactivar el IME en la aplicación Flash Player y puede realizar otras funciones limitadas, según el sistema operativo.

Los editores de método de entrada permiten a los usuarios introducir caracteres de texto no ASCII en idiomas asiáticos, como chino, japonés o coreano. Para más información sobre los IME, consulte la documentación del sistema operativo correspondiente a la plataforma para la que desarrolla aplicaciones. A continuación se indican algunos recursos adicionales para obtener información sobre métodos de entrada:

- http://www.microsoft.com/globaldev/default.mspx
- http://developer.apple.com/documentation/
- http://java.sun.com

La tabla siguiente muestra la cobertura de plataforma para esta clase:

Función	Windows	Macintosh OSX	Macintosh Classic	Linux / Solaris XIM
Determinar si el IME está instalado `System.capabilities.hasIME`	Sí	Sí	Sí	Sí
Activar o desactivar IME `System.IME.setEnabled()`	Sí	Sí	Sí	Sí
Averiguar si el IME está activado o desactivado `System.IME.getEnabled()`	Sí	Sí	Sí	Sí

Función	Windows	Macintosh OSX	Macintosh Classic	Linux / Solaris XIM
Definir el modo de conversión de IME `System.IME.setConversionMode()`	Sí	Sí**	No	Sí
Obtener el modo de conversión de IME `System.IME.getConversionMode()`	Sí	Sí**	No	Sí
Enviar al IME la cadena que se va a convertir `System.IME.setCompositionString()`	Sí *	No	No	Sí
Obtener del IME la cadena original antes de la conversión `System.IME.addListener()` `listener.onIMEComposition()` `System.IME.removeListener()`	Sí *	No	No	Sí
Enviar petición para convertir al IME `System.IME.doConversion()`	Sí *	No	No	Sí

* No todos los IME de Windows admiten todas estas operaciones. Por el momento, el único IME que las admite todas es el japonés. Según el IME varían las llamadas del sistema operativo que se admiten.

** En Macintosh, estos métodos sólo se admiten para japonés y no se admiten para IME de otros proveedores.

Disponibilidad: ActionScript 1.0; Flash Player 8

Resumen de propiedades

Modificadores	Propiedad	Descripción
static	ALPHANUMERIC_FULL:String	Una cadena con el valor "ALPHANUMERIC_FULL" para uso con setConversionMode() y getConversionMode().
static	ALPHANUMERIC_HALF:String	Una cadena con el valor "ALPHANUMERIC_HALF" para uso con setConversionMode() y getConversionMode().
static	CHINESE:String	Una cadena con el valor "CHINESE" para uso con setConversionMode() y getConversionMode().
static	JAPANESE_HIRAGANA:String	Una cadena con el valor "JAPANESE_HIRAGANA" para uso con setConversionMode() y getConversionMode().
static	JAPANESE_KATAKANA_FULL:String	Una cadena con el valor "JAPANESE_KATAKANA_FULL" para uso con setConversionMode() y getConversionMode().
static	JAPANESE_KATAKANA_HALF:String	Una cadena con el valor "JAPANESE_KATAKANA_HALF" para uso con setConversionMode() y getConversionMode().
static	KOREAN:String	Una cadena con el valor "KOREAN" para uso con setConversionMode() y getConversionMode().
static	UNKNOWN:String	Una cadena con el valor "UNKNOWN" para uso con getConversionMode().

Propiedades heredadas de la clase Object

constructor (propiedad Object.constructor), __proto__ (propiedad Object.__proto__), prototype (propiedad Object.prototype), __resolve (propiedad Object.__resolve)

Resumen de eventos

Evento	Descripción
onIMEComposition = function([readingString:String]) {}	Recibe una notificación cuando se define la cadena de composición del IME.

Resumen de métodos

Modificadores	Firma	Descripción
static	addListener(listener:Object) : Void	Registra un objeto para recibir notificación cuando el evento onIMEComposition invoca un controlador de eventos del IME.
static	doConversion() : Boolean	Indica al IME que seleccione el primer candidato de la cadena de composición actual.
static	getConversionMode() : String	Indica el modo de conversión del IME actual.
static	getEnabled() : Boolean	Indica si está activado el IME del sistema.
static	removeListener(listener:Object) : Boolean	Suprime un objeto detector que se había registrado en una instancia IME con IME.addListener().
static	setCompositionString(composition:String) : Boolean	Establece la cadena de composición del IME.
static	setConversionMode(mode:String) : Boolean	Establece el modo de conversión del IME actual.
static	setEnabled(enabled:Boolean) : Boolean	Activa o desactiva el IME del sistema.

Métodos heredados de la clase Object

addProperty (método Object.addProperty), hasOwnProperty (método Object.hasOwnProperty), isPropertyEnumerable (método Object.isPropertyEnumerable), isPrototypeOf (método Object.isPrototypeOf), registerClass (método Object.registerClass), toString (método Object.toString), unwatch (método Object.unwatch), valueOf (método Object.valueOf), watch (método Object.watch)

addListener (método IME.addListener)

```
public static addListener(listener:Object) : Void
```

Registra un objeto para recibir notificación cuando el evento `onIMEComposition` invoca un controlador de eventos del IME.

Disponibilidad: ActionScript 1.0; Flash Player 8

Parámetros

`listener``:Object` - Un objeto, con un método `onIMEComposition` (readingString), que detecta una notificación de callback de los controladores de eventos del IME. La cadena de lectura pasada a este método está en el modo de composición del IME. Por ejemplo, si el usuario escribe en Hiragana y, a continuación, selecciona un candidato Kanji, la cadena de lectura será el Hiragana original.

Ejemplo

El ejemplo siguiente muestra cómo añadir un objeto detector a `System.IME` que envía una notificación cuando un usuario define la cadena de composición haciendo clic en el campo de texto.

```
var IMEListener:Object = new Object();
IMEListener.onIMEComposition = function(str:String) {
  trace(">> onIMEComposition: " + str);
}
System.IME.addListener(IMEListener);
trace(System.IME.length);

var mc:MovieClip = this.createEmptyMovieClip("mc",
  this.getNextHighestDepth());
mc.createTextField("txt", this.getNextHighestDepth(), 0, 0, 0, 0);
mc.txt.border = true;
mc.txt.background = true;
mc.txt.autoSize = "left";
mc.txt.text = "Click this text to add a listener.";

mc.onPress = function() {
  if(System.capabilities.hasIME) {
    Selection.setFocus(mc.txt);
    System.IME.setCompositionString(mc.txt.text);
  }
}
```

ALPHANUMERIC_FULL (propiedad IME.ALPHANUMERIC_FULL)

```
public static ALPHANUMERIC_FULL : String
```

Una cadena con el valor `"ALPHANUMERIC_FULL"` para uso con `setConversionMode()` y `getConversionMode()`. Esta constante se utiliza con todos los IME.

Disponibilidad: ActionScript 1.0; Flash Player 8

Ejemplo

El ejemplo siguiente define el IME como `ALPHANUMERIC_FULL` si el sistema tiene un editor de método de entrada (IME) instalado (`System.capabilities.hasIME`).

```
if(System.capabilities.hasIME) {
  trace(System.IME.getConversionMode());

  System.IME.setConversionMode(System.IME.ALPHANUMERIC_FULL);
  trace(System.IME.getConversionMode());
}
```

Véase también

`setConversionMode (método IME.setConversionMode)`, `getConversionMode (método IME.getConversionMode)`, `hasIME (propiedad capabilities.hasIME)`

ALPHANUMERIC_HALF (propiedad IME.ALPHANUMERIC_HALF)

```
public static ALPHANUMERIC_HALF : String
```

Una cadena con el valor `"ALPHANUMERIC_HALF"` para uso con `setConversionMode()` y `getConversionMode()`. Esta constante se utiliza con todos los IME.

Disponibilidad: ActionScript 1.0; Flash Player 8

Ejemplo

El ejemplo siguiente define el IME como `ALPHANUMERIC_HALF` si el sistema tiene un editor de método de entrada (IME) instalado (`System.capabilities.hasIME`).

```
if(System.capabilities.hasIME) {
  trace(System.IME.getConversionMode());

  System.IME.setConversionMode(System.IME.ALPHANUMERIC_HALF);
  trace(System.IME.getConversionMode());
}
```

Véase también

setConversionMode (método IME.setConversionMode), getConversionMode (método IME.getConversionMode)

CHINESE (propiedad IME.CHINESE)

```
public static CHINESE : String
```

Una cadena con el valor "CHINESE" para uso con setConversionMode() y getConversionMode(). Esta constante se utiliza con IME de chino simplificado y tradicional.

Disponibilidad: ActionScript 1.0; Flash Player 8

Ejemplo

El ejemplo siguiente define el IME como CHINESE si el sistema tiene un editor de método de entrada (IME) instalado (System.capabilities.hasIME).

```
if(System.capabilities.hasIME) {
  trace(System.IME.getConversionMode());

  System.IME.setConversionMode(System.IME.CHINESE);
  trace(System.IME.getConversionMode());
}
```

Véase también

setConversionMode (método IME.setConversionMode), getConversionMode (método IME.getConversionMode)

doConversion (método IME.doConversion)

```
public static doConversion() : Boolean
```

Indica al IME que seleccione el primer candidato de la cadena de composición actual.

Disponibilidad: ActionScript 1.0; Flash Player 8

Valor devuelto

Boolean - Devuelve true si la llamada se ha realizado correctamente; en caso contrario, false.

Ejemplo

El ejemplo siguiente muestra cómo seleccionar el primer candidato para la cadena de composición del IME. Si el sistema del usuario tiene IME, al hacer clic en el campo de texto se selecciona el candidato.

```
var mc:MovieClip = this.createEmptyMovieClip("mc",
  this.getNextHighestDepth());
mc.createTextField("txt", this.getNextHighestDepth(), 0, 0, 0, 0);
mc.txt.border = true;
mc.txt.background = true;
mc.txt.autoSize = "left";
mc.txt.text = "Set this text as the composition string and convert it.";

mc.onPress = function() {
  if(System.capabilities.hasIME) {
    Selection.setFocus(mc.txt);
    System.IME.setCompositionString(mc.txt.text);
    trace(System.IME.doConversion());
  }
}
```

getConversionMode (método IME.getConversionMode)

```
public static getConversionMode() : String
```

Indica el modo de conversión del IME actual.

Disponibilidad: ActionScript 1.0; Flash Player 8

Valor devuelto

`String` - El modo de conversión. Los valores posibles son constantes de cadena de modo del IME que indican el modo de conversión:

- `ALPHANUMERIC_FULL`
- `ALPHANUMERIC_HALF`
- `CHINESE`
- `JAPANESE_HIRAGANA`
- `JAPANESE_KATAKANA_FULL`
- `JAPANESE_KATAKANA_HALF`
- `KOREAN`
- `UNKNOWN`

Ejemplo

El ejemplo siguiente obtiene el IME si el sistema tiene un editor de método de entrada (IME) instalado (`System.capabilities.hasIME`).

```
var mode:String = System.IME.UNKNOWN;
if(System.capabilities.hasIME) {
  mode = System.IME.getConversionMode();
}
trace(mode);
```

Véase también

`ALPHANUMERIC_FULL (propiedad IME.ALPHANUMERIC_FULL)`, `ALPHANUMERIC_HALF (propiedad IME.ALPHANUMERIC_HALF)`, `CHINESE (propiedad IME.CHINESE)`, `JAPANESE_HIRAGANA (propiedad IME.JAPANESE_HIRAGANA)`, `JAPANESE_KATAKANA_FULL (propiedad IME.JAPANESE_KATAKANA_FULL)`, `JAPANESE_KATAKANA_HALF (propiedad IME.JAPANESE_KATAKANA_HALF)`, `KOREAN (propiedad IME.KOREAN)`, `UNKNOWN (propiedad IME.UNKNOWN)`

getEnabled (método IME.getEnabled)

```
public static getEnabled() : Boolean
```

Indica si está activado el IME del sistema. Un IME activado realiza entrada multibyte; un IME desactivado realiza entrada alfanumérica.

Disponibilidad: ActionScript 1.0; Flash Player 8

Valor devuelto

`Boolean` - Devuelve `true` si el IME del sistema está activado, `false` si está desactivado.

Ejemplo

El ejemplo siguiente comprueba si el IME está activado llamando al método `isEnabled()`.

```
if(System.capabilities.hasIME) {
  var isImeEnabled:Boolean = System.IME.getEnabled();
  trace(isImeEnabled);
}
```

JAPANESE_HIRAGANA (propiedad IME.JAPANESE_HIRAGANA)

```
public static JAPANESE_HIRAGANA : String
```

Una cadena con el valor "JAPANESE_HIRAGANA" para uso con `setConversionMode()` y `getConversionMode()`. Esta constante se utiliza con IME japoneses.

Disponibilidad: ActionScript 1.0; Flash Player 8

Ejemplo

El ejemplo siguiente define el IME como `JAPANESE_HIRAGANA` si el sistema tiene un editor de método de entrada (IME) instalado (`System.capabilities.hasIME`).

```
if(System.capabilities.hasIME) {
  trace(System.IME.getConversionMode());

  System.IME.setConversionMode(System.IME.JAPANESE_HIRAGANA);
  trace(System.IME.getConversionMode());
}
```

Véase también

`setConversionMode (método IME.setConversionMode)`, `getConversionMode (método IME.getConversionMode)`

JAPANESE_KATAKANA_FULL (propiedad IME.JAPANESE_KATAKANA_FULL)

```
public static JAPANESE_KATAKANA_FULL : String
```

Una cadena con el valor "JAPANESE_KATAKANA_FULL" para uso con `setConversionMode()` y `getConversionMode()`. Esta constante se utiliza con IME japoneses.

Disponibilidad: ActionScript 1.0; Flash Player 8

Ejemplo

El ejemplo siguiente define el IME como `JAPANESE_KATAKANA_FULL` si el sistema tiene un editor de método de entrada (IME) instalado (`System.capabilities.hasIME`).

```
if(System.capabilities.hasIME) {
  trace(System.IME.getConversionMode());

  System.IME.setConversionMode(System.IME.JAPANESE_KATAKANA_FULL);
  trace(System.IME.getConversionMode());
}
```

Véase también

setConversionMode (método IME.setConversionMode), getConversionMode (método IME.getConversionMode)

JAPANESE_KATAKANA_HALF (propiedad IME.JAPANESE_KATAKANA_HALF)

```
public static JAPANESE_KATAKANA_HALF : String
```

Una cadena con el valor "JAPANESE_KATAKANA_HALF" para uso con setConversionMode() y getConversionMode(). Esta constante se utiliza con IME japoneses.

Disponibilidad: ActionScript 1.0; Flash Player 8

Ejemplo

El ejemplo siguiente define el IME como JAPANESE_KATAKANA_HALF si el sistema tiene un editor de método de entrada (IME) instalado (System.capabilities.hasIME).

```
if(System.capabilities.hasIME) {
  trace(System.IME.getConversionMode());

  System.IME.setConversionMode(System.IME.JAPANESE_KATAKANA_HALF);
  trace(System.IME.getConversionMode());
}
```

Véase también

setConversionMode (método IME.setConversionMode), getConversionMode (método IME.getConversionMode)

KOREAN (propiedad IME.KOREAN)

```
public static KOREAN : String
```

Una cadena con el valor "KOREAN" para uso con setConversionMode() y getConversionMode(). Esta constante se utiliza con IME coreanos.

Disponibilidad: ActionScript 1.0; Flash Player 8

Ejemplo

El ejemplo siguiente define el IME como `KOREAN` si el sistema tiene un editor de método de entrada (IME) instalado (`System.capabilities.hasIME`).

```
if(System.capabilities.hasIME) {
  trace(System.IME.getConversionMode());

  System.IME.setConversionMode(System.IME.KOREAN);
  trace(System.IME.getConversionMode());
}
```

Véase también

`setConversionMode (método IME.setConversionMode)`, `getConversionMode (método IME.getConversionMode)`

onIMEComposition (detector de eventos IME.onIMEComposition)

```
onIMEComposition = function([readingString:String]) {}
```

Recibe una notificación cuando se define la cadena de composición del IME. Para utilizar este detector, debe crear un objeto detector. Posteriormente podrá definir una función para este detector y utilizar `IME.addListener()` para registrar el detector en el objeto IME, como se muestra en el siguiente código:

```
var someListener:Object = new Object();
someListener.onIMEComposition = function () {
  // statements
}
System.IME.addListener(someListener);
```

Los detectores permiten que diversas partes del código cooperen, ya que varios detectores pueden recibir notificación de un solo evento.

Disponibilidad: ActionScript 1.0; Flash Player 8

Parámetros

`readingString``:String` [opcional] - El texto original escrito en el IME antes de que el usuario empezara a elegir candidatos.

Ejemplo

El ejemplo siguiente muestra cómo añadir un objeto detector con el método callback `onIMEComposition()` a `System.IME`, que devuelve la notificación cuando un usuario define la cadena de composición haciendo clic en el campo de texto.

```
var IMEListener:Object = new Object();
IMEListener.onIMEComposition = function(str:String) {
  trace(">> onIMEComposition: " + str);
}
System.IME.addListener(IMEListener);
trace(System.IME.length);

var mc:MovieClip = this.createEmptyMovieClip("mc",
  this.getNextHighestDepth());
mc.createTextField("txt", this.getNextHighestDepth(), 0, 0, 0, 0);
mc.txt.border = true;
mc.txt.background = true;
mc.txt.autoSize = "left";
mc.txt.text = "Click this text to add a listener.";

mc.onPress = function() {
  if(System.capabilities.hasIME) {
    Selection.setFocus(mc.txt);
    System.IME.setCompositionString(mc.txt.text);
  }
}
```

Véase también

`addListener (método IME.addListener)`, `setCompositionString (método IME.setCompositionString)`

removeListener (método IME.removeListener)

```
public static removeListener(listener:Object) : Boolean
```

Suprime un objeto detector que se había registrado en una instancia `IME` con `IME.addListener()`.

Disponibilidad: ActionScript 1.0; Flash Player 8

Parámetros

`listener``:Object` - Objeto que no volverá a recibir notificación de callback de los controladores de eventos del IME.

Valor devuelto

`Boolean` - Si el objeto detector se ha eliminado, devuelve `true`, en caso contrario devuelve `false`.

Ejemplo

El ejemplo siguiente muestra cómo eliminar un objeto detector de `System.IME` cuando un usuario define la cadena de composición haciendo clic en el campo de texto.

```
var IMEListener:Object = new Object();
IMEListener.onIMEComposition = function(str:String) {
  trace(">> onIMEComposition: " + str);

  System.IME.removeListener(this);
  trace(System.IME.length); // 0
}
System.IME.addListener(IMEListener);
trace(System.IME.length); // 1

var mc:MovieClip = this.createEmptyMovieClip("mc",
  this.getNextHighestDepth());
mc.createTextField("txt", this.getNextHighestDepth(), 0, 0, 0, 0);
mc.txt.border = true;
mc.txt.background = true;
mc.txt.autoSize = "left";
mc.txt.text = "Click this text to add and remove a listener.";

mc.onPress = function() {
  if(System.capabilities.hasIME) {
    Selection.setFocus(mc.txt);
    System.IME.setCompositionString(mc.txt.text);
  }
}
```

setCompositionString (método IME.setCompositionString)

```
public static setCompositionString(composition:String) : Boolean
```

Establece la cadena de composición del IME. Una vez establecida, el usuario puede seleccionar candidatos del IME antes de enviar el resultado al campo de texto seleccionado actualmente.

Disponibilidad: ActionScript 1.0; Flash Player 8

Parámetros

`composition``:String` - Cadena a la que se envía el IME.

Valor devuelto

`Boolean` - Si la cadena de composición del IME se establece correctamente, será `true`. Este método falla y devuelve `false` si no hay ningún campo de texto seleccionado.

Ejemplo

El ejemplo siguiente muestra cómo definir la cadena de composición del IME Si el sistema del usuario tiene IME, al hacer clic en el campo de texto se muestran las opciones del IME.

```
var mc:MovieClip = this.createEmptyMovieClip("mc",
  this.getNextHighestDepth());
mc.createTextField("txt", this.getNextHighestDepth(), 0, 0, 0, 0);
mc.txt.border = true;
mc.txt.background = true;
mc.txt.autoSize = "left";
mc.txt.text = "Set this text as the composition string.";

mc.onPress = function() {
  if(System.capabilities.hasIME) {
    Selection.setFocus(mc.txt);
    trace(System.IME.setCompositionString(mc.txt.text));
  }
}
```

setConversionMode (método IME.setConversionMode)

```
public static setConversionMode(mode:String) : Boolean
```

Establece el modo de conversión del IME actual.

Disponibilidad: ActionScript 1.0; Flash Player 8

Parámetros

mode`:String` - Modo de conversión. Los valores posibles son constantes de cadena de modo del IME:

- `ALPHANUMERIC_FULL`
- `ALPHANUMERIC_HALF`
- `CHINESE`
- `JAPANESE_HIRAGANA`
- `JAPANESE_KATAKANA_FULL`
- `JAPANESE_KATAKANA_HALF`
- `KOREAN`

Valor devuelto

`Boolean` - Si el modo de conversión se ha definido correctamente, devuelve `true`, en caso contrario devuelve `false`.

Ejemplo

El ejemplo siguiente obtiene el IME si el sistema tiene un editor de método de entrada (IME) instalado (`System.capabilities.hasIME`) y define la variable `mode` con ese valor.

```
var mode:String = System.IME.UNKNOWN;
if(System.capabilities.hasIME) {
  mode = System.IME.getConversionMode();
}
System.IME.setConversionMode(mode);
trace(System.IME.getConversionMode());
```

Véase también

`ALPHANUMERIC_FULL (propiedad IME.ALPHANUMERIC_FULL)`, `ALPHANUMERIC_HALF (propiedad IME.ALPHANUMERIC_HALF)`, `CHINESE (propiedad IME.CHINESE)`, `JAPANESE_HIRAGANA (propiedad IME.JAPANESE_HIRAGANA)`, `JAPANESE_KATAKANA_FULL (propiedad IME.JAPANESE_KATAKANA_FULL)`, `JAPANESE_KATAKANA_HALF (propiedad IME.JAPANESE_KATAKANA_HALF)`, `KOREAN (propiedad IME.KOREAN)`

setEnabled (método IME.setEnabled)

```
public static setEnabled(enabled:Boolean) : Boolean
```

Activa o desactiva el IME del sistema. Un IME activado realiza entrada multibyte; un IME desactivado realiza entrada alfanumérica.

Disponibilidad: ActionScript 1.0; Flash Player 8

Parámetros

`enabled``:Boolean` - Se establece como `true` para activar el IME del sistema, y como `false` para desactivarlo.

Valor devuelto

`Boolean` - Si el intento de activar el IME del sistema se realiza correctamente, será `true`; en caso contrario, será `false`.

Ejemplo

El ejemplo siguiente comprueba si el IME está activado llamando al método `isEnabled()` y, a continuación, cambia su estado de activación al valor opuesto mediante el método `setEnabled()`.

```
if(System.capabilities.hasIME) {
  var isImeEnabled:Boolean = System.IME.getEnabled();
  trace(isImeEnabled);

  if(isImeEnabled) {
    System.IME.setEnabled(false);
  }
  else {
    System.IME.setEnabled(true);
  }

  var isImeEnabled:Boolean = System.IME.getEnabled();
  trace(isImeEnabled);
}
```

UNKNOWN (propiedad IME.UNKNOWN)

```
public static UNKNOWN : String
```

Una cadena con el valor `"UNKNOWN"` para uso con `getConversionMode()`. Esta constante se utiliza con todos los IME.

Disponibilidad: ActionScript 1.0; Flash Player 8

Ejemplo

El ejemplo siguiente define el IME como `UNKNOWN` si el sistema tiene un editor de método de entrada (IME) instalado (`System.capabilities.hasIME`).

```
if(System.capabilities.hasIME) {
  trace(System.IME.getConversionMode());

  System.IME.setConversionMode(System.IME.UNKNOWN);
  trace(System.IME.getConversionMode());
}
```

Véase también

`getConversionMode (método IME.getConversionMode)`

Key

```
Object
  |
  +-Key
```

```
public class Key
extends Object
```

La clase Key es una clase de nivel superior cuyos métodos y propiedades se pueden utilizar sin emplear un constructor. Utilice los métodos de la clase Key para crear una interfaz que un usuario pueda controlar con un teclado estándar. Las propiedades de la clase Key son constantes que representan las teclas más utilizadas en las aplicaciones de control, como las teclas de flecha AvPág y RePág.

Una aplicación Flash sólo puede controlar los eventos de teclado que se produzcan en la propia aplicación. Una aplicación Flash no puede detectar eventos de teclado de otra aplicación.

Disponibilidad: ActionScript 1.0; Flash Player 6

Resumen de propiedades

Modificadores	Propiedad	Descripción
static	BACKSPACE:Number	El valor de código de tecla asociado a la tecla Retroceso (8).
static	CAPSLOCK:Number	El valor de código de tecla asociado a la tecla Bloq Mayús (20).
static	CONTROL:Number	El valor de código de tecla asociado a la tecla Control (17).
static	DELETEKEY:Number	El valor de código de tecla asociado a la tecla Supr (46).
static	DOWN:Number	El valor de código de tecla asociado a la tecla de flecha abajo (40).
static	END:Number	El valor de código de tecla asociado a la tecla Fin (35).
static	ENTER:Number	El valor de código de tecla asociado a la tecla Intro (13).
static	ESCAPE:Number	El valor de código de tecla asociado a la tecla Esc (27).
static	HOME:Number	El valor de código de tecla asociado a la tecla Inicio (36).

Modificadores	Propiedad	Descripción
static	INSERT:Number	El valor de código de tecla asociado a la tecla Insert (45).
static	LEFT:Number	El valor de código de tecla asociado a la tecla de flecha izquierda (37).
static	_listeners:Array [sólo lectura]	Una lista de las referencias a todos los objetos detectores registrados con el objeto Key.
static	PGDN:Number	El valor de código de tecla asociado a la tecla AvPág (34).
static	PGUP:Number	El valor de código de tecla asociado a la tecla RePág (33).
static	RIGHT:Number	El valor de código de tecla asociado a la tecla de flecha derecha (39).
static	SHIFT:Number	El valor de código de tecla asociado a la tecla Mayús (16).
static	SPACE:Number	El valor de código de tecla asociado a la tecla Barra espaciadora (32).
static	TAB:Number	El valor de código de tecla asociado a la tecla Tabulador (9).
static	UP:Number	El valor de código de tecla asociado a la tecla de flecha arriba (38).

Propiedades heredadas de la clase Object

```
constructor (propiedad Object.constructor), __proto__ (propiedad
Object.__proto__), prototype (propiedad Object.prototype), __resolve
(propiedad Object.__resolve)
```

Resumen de eventos

Evento	Descripción
onKeyDown = function() {}	Se notifica cuando se presiona una tecla.
onKeyUp = function() {}	Se notifica cuando se suelta una tecla.

Resumen de métodos

Modificadores	Firma	Descripción
static	addListener(listener:Object) : Void	Registra un objeto para recibir notificación de onKeyDown y onKeyUp.
static	getAscii() : Number	Devuelve el código ASCII de la última tecla presionada o soltada.
static	getCode() : Number	Devuelve el valor de código de la última tecla presionada.
static	isAccessible() : Boolean	Devuelve un valor booleano que indica, según las restricciones de seguridad, si otros archivos SWF pueden acceder a la última tecla presionada.
static	isDown(code:Number) : Boolean	Devuelve true si está presionada la tecla especificada en *keycode*; false en caso contrario.
static	isToggled(code:Number) : Boolean	Devuelve true si está activada la tecla Bloq Mayús o Bloq Num (en estado de activación); false en caso contrario.
static	removeListener(listener:Object) : Boolean	Elimina un objeto registrado previamente con Key.addListener().

Métodos heredados de la clase Object

addProperty (método Object.addProperty), hasOwnProperty (método Object.hasOwnProperty), isPropertyEnumerable (método Object.isPropertyEnumerable), isPrototypeOf (método Object.isPrototypeOf), registerClass (método Object.registerClass), toString (método Object.toString), unwatch (método Object.unwatch), valueOf (método Object.valueOf), watch (método Object.watch)

addListener (método Key.addListener)

```
public static addListener(listener:Object) : Void
```

Registra un objeto para recibir notificación de onKeyDown y onKeyUp. Cuando se presiona o se suelta una tecla, con independencia de donde se encuentre la selección de entrada del teclado, se invocará el método onKeyUp u onKeyDown de todos los objetos que estén a la escucha registrados con addListener(). Puede haber varios objetos a la escucha de notificaciones de teclado. Si el *newListener* ya se ha registrado, no se producirá ningún cambio.

Una aplicación Flash sólo puede controlar los eventos de teclado que se produzcan en la propia aplicación. Una aplicación Flash no puede detectar eventos de teclado de otra aplicación.

Disponibilidad: ActionScript 1.0; Flash Player 6

Parámetros

listener:Object - Un objeto con métodos onKeyDown y onKeyDown.

Ejemplo

En el ejemplo siguiente se crea un nuevo objeto detector y se define una función para onKeyDown y onKeyUp. En la última línea se utiliza addListener() para registrar el detector con el objeto Key de modo que se pueda recibir notificación de los eventos que se producen al presionar y soltar teclas.

```
var myListener:Object = new Object();
myListener.onKeyDown = function () {
  trace ("You pressed a key.");
}
myListener.onKeyUp = function () {
  trace ("You released a key.");
}
Key.addListener(myListener);
```

En el ejemplo siguiente se asigna el método abreviado de teclado Control+7 a un botón cuyo nombre de instancia es my_btn y se pone a disposición de los lectores de pantalla la información sobre el método abreviado (consulte _accProps). En este ejemplo, cuando se presiona Control+7, la función myOnPress muestra el texto hello en el panel Salida.

```
function myOnPress() {
  trace("hello");
}
function myOnKeyDown() {
  // 55 is key code for 7
  if (Key.isDown(Key.CONTROL) && Key.getCode() == 55) {
  Selection.setFocus(my_btn);
  my_btn.onPress();
  }
}
var myListener:Object = new Object();
myListener.onKeyDown = myOnKeyDown;
Key.addListener(myListener);
my_btn.onPress = myOnPress;
my_btn._accProps.shortcut = "Ctrl+7";
Accessibility.updateProperties();
```

Véase también

getCode (método Key.getCode), isDown (método Key.isDown), onKeyDown (detector de eventos Key.onKeyDown), onKeyDown (detector de eventos Key.onKeyUp), removeListener (método Key.removeListener)

BACKSPACE (propiedad Key.BACKSPACE)

```
public static BACKSPACE : Number
```

El valor de código de tecla asociado a la tecla Retroceso (8).

Disponibilidad: ActionScript 1.0; Flash Player 5

Ejemplo

En el ejemplo siguiente se crea un nuevo objeto detector y se define una función para `onKeyDown`. En la última línea se utiliza `addListener()` para registrar el detector con el objeto Key de modo que se pueda recibir notificación de los eventos que se producen al presionar teclas.

```
var keyListener:Object = new Object();
keyListener.onKeyDown = function() {
  if (Key.isDown(Key.BACKSPACE)) {
  trace("you pressed the Backspace key.");
  } else {
  trace("you DIDN'T press the Backspace key.");
  }
};
Key.addListener(keyListener);
```

Cuando utilice este ejemplo, asegúrese de seleccionar Control > Deshabilitar métodos abreviados de teclado en el entorno de prueba.

CAPSLOCK (propiedad Key.CAPSLOCK)

```
public static CAPSLOCK : Number
```

El valor de código de tecla asociado a la tecla Bloq Mayús (20).

Disponibilidad: ActionScript 1.0; Flash Player 5

Ejemplo

En el ejemplo siguiente se crea un nuevo objeto detector y se define una función para `onKeyDown`. En la última línea se utiliza `addListener()` para registrar el detector con el objeto Key de modo que se pueda recibir notificación de los eventos que se producen al presionar teclas.

```
var keyListener:Object = new Object();
keyListener.onKeyDown = function() {
  if (Key.isDown(Key.CAPSLOCK)) {
  trace("you pressed the Caps Lock key.");
  trace("\tCaps Lock == "+Key.isToggled(Key.CAPSLOCK));
  }
};
Key.addListener(keyListener);
```

Aparece información en el panel Salida al presionar la tecla Bloq Mayús. El panel Salida muestra `true` o `false`, en función de si se activa la tecla Bloq Mayús mediante el método isToggled.

CONTROL (propiedad Key.CONTROL)

```
public static CONTROL : Number
```

El valor de código de tecla asociado a la tecla Control (17).

Disponibilidad: ActionScript 1.0; Flash Player 5

Ejemplo

En el ejemplo siguiente se asigna el método abreviado de teclado Control+7 a un botón cuyo nombre de instancia es `my_btn` y se pone a disposición de los lectores de pantalla la información sobre el método abreviado (consulte `_accProps`). En este ejemplo, cuando se presiona Control+7, la función `myOnPress` muestra el texto `hello` en el panel Salida.

```
function myOnPress() {
  trace("hello");
}
function myOnKeyDown() {
  // 55 is key code for 7
  if (Key.isDown(Key.CONTROL) && Key.getCode() == 55) {
  Selection.setFocus(my_btn);
  my_btn.onPress();
  }
}
var myListener:Object = new Object();
myListener.onKeyDown = myOnKeyDown;
Key.addListener(myListener);
my_btn.onPress = myOnPress;
my_btn._accProps.shortcut = "Ctrl+7";
Accessibility.updateProperties();
```

DELETEKEY (propiedad Key.DELETEKEY)

```
public static DELETEKEY : Number
```

El valor de código de tecla asociado a la tecla Supr (46).

Disponibilidad: ActionScript 1.0; Flash Player 5

Ejemplo

El ejemplo siguiente permite dibujar líneas con el puntero del ratón utilizando la interfaz API de dibujo y objetos detectores. Presione la tecla Retroceso o Suprimir para eliminar la líneas que haya dibujado.

```
this.createEmptyMovieClip("canvas_mc", this.getNextHighestDepth());
var mouseListener:Object = new Object();
mouseListener.onMouseDown = function() {
  this.drawing = true;
  canvas_mc.moveTo(_xmouse, _ymouse);
  canvas_mc.lineStyle(3, 0x99CC00, 100);
};
mouseListener.onMouseUp = function() {
  this.drawing = false;
};
mouseListener.onMouseMove = function() {
  if (this.drawing) {
  canvas_mc.lineTo(_xmouse, _ymouse);
  }
  updateAfterEvent();
};
Mouse.addListener(mouseListener);
//
var keyListener:Object = new Object();
keyListener.onKeyDown = function() {
  if (Key.isDown(Key.DELETEKEY) || Key.isDown(Key.BACKSPACE)) {
  canvas_mc.clear();
  }
};
Key.addListener(keyListener);
```

Cuando utilice este ejemplo, asegúrese de seleccionar Control > Deshabilitar métodos abreviados de teclado en el entorno de prueba.

El método `MovieClip.getNextHighestDepth()` utilizado en este ejemplo necesita Flash Player 7 o una versión posterior. Si el archivo SWF incluye un componente de la versión 2, utilice la clase DepthManager de los componentes de la versión 2 en lugar del método `MovieClip.getNextHighestDepth()`.

DOWN (propiedad Key.DOWN)

```
public static DOWN : Number
```

El valor de código de tecla asociado a la tecla de flecha abajo (40).

Disponibilidad: ActionScript 1.0; Flash Player 5

Ejemplo

El ejemplo siguiente mueve un clip de película denominado `car_mc` hasta una distancia constante (10) al presionar las teclas de flecha. Se reproduce un sonido cuando se presiona la Barra espaciadora. Para este ejemplo, asigne a un sonido de la biblioteca el identificador de vinculación `horn_id`.

```
var DISTANCE:Number = 10;
var horn_sound:Sound = new Sound();
horn_sound.attachSound("horn_id");
var keyListener_obj:Object = new Object();
keyListener_obj.onKeyDown = function() {
  switch (Key.getCode()) {
  case Key.SPACE :
  horn_sound.start();
  break;
  case Key.LEFT :
  car_mc._x -= DISTANCE;
  break;
  case Key.UP :
  car_mc._y -= DISTANCE;
  break;
  case Key.RIGHT :
  car_mc._x += DISTANCE;
  break;
  case Key.DOWN :
  car_mc._y += DISTANCE;
  break;
  }
};
Key.addListener(keyListener_obj);
```

END (propiedad Key.END)

```
public static END : Number
```

El valor de código de tecla asociado a la tecla Fin (35).

Disponibilidad: ActionScript 1.0; Flash Player 5

END (propiedad Key.ENTER)

```
public static ENTER : Number
```

El valor de código de tecla asociado a la tecla Intro (13).

Disponibilidad: ActionScript 1.0; Flash Player 5

Ejemplo

El ejemplo siguiente mueve un clip de película denominado `car_mc` hasta una distancia constante (10) al presionar las teclas de flecha. La instancia `car_mc` se detiene al presionar Intro y eliminar el evento `onEnterFrame`.

```
var DISTANCE:Number = 5;
var keyListener:Object = new Object();
keyListener.onKeyDown = function() {
  switch (Key.getCode()) {
  case Key.LEFT :
  car_mc.onEnterFrame = function() {
    this._x -= DISTANCE;
  };
  break;
  case Key.UP :
  car_mc.onEnterFrame = function() {
    this._y -= DISTANCE;
  };
  break;
  case Key.RIGHT :
  car_mc.onEnterFrame = function() {
    this._x += DISTANCE;
  };
  break;
  case Key.DOWN :
  car_mc.onEnterFrame = function() {
    this._y += DISTANCE;
  };
  break;
  case Key.ENTER :
  delete car_mc.onEnterFrame;
  break;
  }
};
Key.addListener(keyListener);
```

Cuando utilice este ejemplo, asegúrese de seleccionar Control > Deshabilitar métodos abreviados de teclado en el entorno de prueba.

ESCAPE (propiedad Key.ESCAPE)

```
public static ESCAPE : Number
```

El valor de código de tecla asociado a la tecla Esc (27).

Disponibilidad: ActionScript 1.0; Flash Player 5

Ejemplo

En el ejemplo siguiente se define un temporizador. Cuando presione la tecla Esc, el panel Salida muestra información que incluye el tiempo que ha tardado en presionar la tecla.

```
var keyListener:Object = new Object();
keyListener.onKeyDown = function() {
  if (Key.isDown(Key.ESCAPE)) {
  // Get the current timer, convert the value to seconds and round it to two
  decimal places.
  var timer:Number = Math.round(getTimer()/10)/100;
  trace("you pressed the Esc key: "+getTimer()+" ms ("+timer+" s)");
  }
};
Key.addListener(keyListener);
```

Cuando utilice este ejemplo, asegúrese de seleccionar Control > Deshabilitar métodos abreviados de teclado en el entorno de prueba.

getAscii (método Key.getAscii)

```
public static getAscii() : Number
```

Devuelve el código ASCII de la última tecla presionada o soltada. Los valores ASCII devueltos son los correspondientes a un teclado inglés. Por ejemplo, si presiona Mayús+2, `Key.getAscii()` devolverá `@` en un teclado japonés, que es el mismo resultado que en un teclado inglés.

Una aplicación Flash sólo puede controlar los eventos de teclado que se produzcan en la propia aplicación. Una aplicación Flash no puede detectar eventos de teclado de otra aplicación.

Disponibilidad: ActionScript 1.0; Flash Player 5

Valor devuelto

`Number` - Devuelve el valor ASCII de la última tecla presionada. Este método devuelve 0 si no se ha presionado ni soltado ninguna tecla, o si no se puede acceder al código de tecla por motivos de seguridad.

Ejemplo

En el ejemplo siguiente se llama al método `getAscii()` siempre que se presiona una tecla. En el ejemplo se crea un objeto detector denominado `keyListener` y se define una función que responde al evento `onKeyDown` realizando una llamada a `Key.getAscii()`. Después se registra el objeto `keyListener` en el objeto `Key`, que difunde el mensaje `onKeyDown` siempre que se presione una tecla durante la ejecución del archivo SWF.

```
var keyListener:Object = new Object();
keyListener.onKeyDown = function() {
  trace("The ASCII code for the last key typed is: "+Key.getAscii());
};
Key.addListener(keyListener);
```

Cuando utilice este ejemplo, asegúrese de seleccionar Control > Deshabilitar métodos abreviados de teclado en el entorno de prueba.

En el ejemplo siguiente se añade una llamada a `Key.getAscii()` para señalar la diferencia entre los dos métodos. La principal diferencia es que `Key.getAscii()` distingue entre caracteres en mayúsculas y en minúsculas, mientras que `Key.getCode()` no hace distinciones.

```
var keyListener:Object = new Object();
keyListener.onKeyDown = function() {
  trace("For the last key typed:");
  trace("\tThe Key code is: "+Key.getCode());
  trace("\tThe ASCII value is: "+Key.getAscii());
  trace("");
};
Key.addListener(keyListener);
```

Cuando utilice este ejemplo, asegúrese de seleccionar Control > Deshabilitar métodos abreviados de teclado en el entorno de prueba.

Véase también

`isAccessible (método Key.isAccessible)`

getCode (método Key.getCode)

```
public static getCode() : Number
```

Devuelve el valor de código de la última tecla presionada.

Nota: La implementación de Flash Lite de este método devuelve una cadena o un número, según el código de tecla pasado en la plataforma. Los únicos códigos de teclas válidos son los códigos estándar que acepta esta clase y los códigos especiales que se muestran como propiedades de la clase `ExtendedKey`.

Una aplicación Flash sólo puede controlar los eventos de teclado que se produzcan en la propia aplicación. Una aplicación Flash no puede detectar eventos de teclado de otra aplicación.

Disponibilidad: ActionScript 1.0; Flash Player 5

Valor devuelto

`Number` - El código de tecla de la última tecla presionada. Este método devuelve 0 si no se ha presionado ni soltado ninguna tecla, o si no se puede acceder al código de tecla por motivos de seguridad.

Ejemplo

En el ejemplo siguiente se llama al método `getCode()` siempre que se presiona una tecla. En el ejemplo se crea un objeto detector denominado `keyListener` y se define una función que responde al evento `onKeyDown` realizando una llamada a `Key.getCode()`. Después se registra el objeto `keyListener` en el objeto `Key`, que difunde el mensaje `onKeyDown` siempre que se presione una tecla durante la reproducción del archivo SWF.

```
var keyListener:Object = new Object();
keyListener.onKeyDown = function() {
  // compare return value of getCode() to constant
  if (Key.getCode() == Key.ENTER) {
    trace ("Virtual key code: "+Key.getCode()+" (ENTER key)");
  }
  else {
    trace("Virtual key code: "+Key.getCode());
  }
};
Key.addListener(keyListener);
```

Cuando utilice este ejemplo, asegúrese de seleccionar Control > Deshabilitar métodos abreviados de teclado en el entorno de prueba.

En el ejemplo siguiente se añade una llamada a `Key.getAscii()` para señalar la diferencia entre los dos métodos. La principal diferencia es que `Key.getAscii()` distingue entre caracteres en mayúsculas y en minúsculas, mientras que `Key.getCode()` no hace distinciones.

```
var keyListener:Object = new Object();
keyListener.onKeyDown = function() {
  trace("For the last key typed:");
  trace("\tThe Key code is: "+Key.getCode());
  trace("\tThe ASCII value is: "+Key.getAscii());
  trace("");
};
Key.addListener(keyListener);
```

Cuando utilice este ejemplo, asegúrese de seleccionar Control > Deshabilitar métodos abreviados de teclado en el entorno de prueba.

Véase también

`getAscii (método Key.getAscii)`, `isAccessible (método Key.isAccessible)`

HOME (propiedad Key.HOME)

```
public static HOME : Number
```

El valor de código de tecla asociado a la tecla Inicio (36).

Disponibilidad: ActionScript 1.0; Flash Player 5

Ejemplo

En el ejemplo siguiente se asocia un clip de película que se puede arrastrar denominado `car_mc` en las coordenadas *x* e *y* de 0,0. Cuando se presiona a la tecla Inicio, `car_mc` vuelve a las coordenadas 0,0. Cree un clip de película que tenga el identificador de vinculación `car_id` y agregue el siguiente código ActionScript al fotograma 1 de la línea de tiempo:

```
this.attachMovie("car_id", "car_mc", this.getNextHighestDepth(), {_x:0,
  _y:0});
car_mc.onPress = function() {
  this.startDrag();
};
car_mc.onRelease = function() {
  this.stopDrag();
};
var keyListener:Object = new Object();
keyListener.onKeyDown = function() {
  if (Key.isDown(Key.HOME)) {
  car_mc._x = 0;
  car_mc._y = 0;
  }
};
Key.addListener(keyListener);
```

El método `MovieClip.getNextHighestDepth()` utilizado en este ejemplo necesita Flash Player 7 o una versión posterior. Si el archivo SWF incluye un componente de la versión 2, utilice la clase DepthManager de los componentes de la versión 2 en lugar del método `MovieClip.getNextHighestDepth()`.

INSERT (propiedad Key.INSERT)

```
public static INSERT : Number
```

El valor de código de tecla asociado a la tecla Insert (45).

Disponibilidad: ActionScript 1.0; Flash Player 5

Ejemplo

En el ejemplo siguiente se crea un nuevo objeto detector y se define una función para `onKeyDown`. En la última línea se utiliza `addListener()` para registrar el detector con el objeto Key de modo que se pueda recibir notificación del evento que se produce al presionar la tecla y mostrar información en el panel Salida.

```
var keyListener:Object = new Object();
keyListener.onKeyDown = function() {
  if (Key.isDown(Key.INSERT)) {
  trace("You pressed the Insert key.");
  }
};
Key.addListener(keyListener);
```

isAccessible (método Key.isAccessible)

```
public static isAccessible() : Boolean
```

Devuelve un valor booleano que indica, según las restricciones de seguridad, si otros archivos SWF pueden acceder a la última tecla presionada. De forma predeterminada, el código de un archivo SWF de un dominio no puede acceder a una pulsación de tecla generada en un archivo SWF de otro dominio. Para obtener más información sobre la seguridad entre dominios, consulte "Aspectos básicos de la seguridad" in *Aprendizaje de ActionScript 2.0 en Flash*.

Disponibilidad: ActionScript 1.0; Flash Player 8

Valor devuelto

`Boolean` - El valor `true` si se puede acceder a la última tecla presionada. Si no se permite el acceso, este método devuelve `false`.

isDown (método Key.isDown)

```
public static isDown(code:Number) : Boolean
```

Devuelve `true` si está presionada la tecla especificada en *`keycode`* ; `false` en caso contrario. En Macintosh, los valores de códigos de tecla para Bloq Mayús y Bloq Num son idénticos.

Una aplicación Flash sólo puede controlar los eventos de teclado que se produzcan en la propia aplicación. Una aplicación Flash no puede detectar eventos de teclado de otra aplicación.

Disponibilidad: ActionScript 1.0; Flash Player 5

Parámetros

code:Number - El valor de código de tecla asignado a una tecla específica o una propiedad de clase Key asociada con una determinada tecla.

Valor devuelto

Boolean - El valor true si está presionada la tecla especificada en *keycode* ; false en caso contrario.

Ejemplo

Con el script siguiente es posible controlar la ubicación de un clip de película (car_mc):

```
car_mc.onEnterFrame = function() {
 if (Key.isDown(Key.RIGHT)) {
 this._x += 10;
 } else if (Key.isDown(Key.LEFT)) {
 this._x -= 10;
 }
};
```

isToggled (método Key.isToggled)

```
public static isToggled(code:Number) : Boolean
```

Devuelve true si está activada la tecla Bloq Mayús o Bloq Num (en estado de activación); false en caso contrario. Aunque el término *toggled* (conmutado) normalmente significa que algo puede cambiarse entre dos opciones posibles, el método Key.isToggled() sólo devuelve true si la tecla se está conmutada (toggled) en el estado de activación. En Macintosh, los valores de códigos de tecla para Bloq Mayús y Bloq Num son idénticos.

Una aplicación Flash sólo puede controlar los eventos de teclado que se produzcan en la propia aplicación. Una aplicación Flash no puede detectar eventos de teclado de otra aplicación.

Disponibilidad: ActionScript 1.0; Flash Player 5

Parámetros

code:Number - El código de la tecla Bloq Mayús (20) o Bloq Num (144).

Valor devuelto

`true` - El valor `true` si está activada la tecla Bloq Mayús o Bloq Num (en estado de activación); `false` en caso contrario.

Ejemplo

El ejemplo llama al método `isToggled()` siempre que se presiona una tecla y ejecuta una sentencia trace siempre que la tecla Bloq Mayús se conmuta (toggled) a un estado de activación. En el ejemplo se crea un objeto detector denominado `keyListener` y se define una función que responde al evento `onKeyDown` realizando una llamada a `Key.isToggled()`. Después se registra el objeto `keyListener` en el objeto `Key`, que difunde el mensaje `onKeyDown` siempre que se presione una tecla durante la reproducción del archivo SWF.

```
var keyListener:Object = new Object();
keyListener.onKeyDown = function() {
  if (Key.isDown(Key.CAPSLOCK)) {
  trace("you pressed the Caps Lock key.");
  trace("\tCaps Lock == "+Key.isToggled(Key.CAPSLOCK));
  }
};
Key.addListener(keyListener);
```

Aparece información en el panel Salida al presionar la tecla Bloq Mayús. El panel Salida muestra `true` o `false`, en función de si se activa la tecla Bloq Mayús mediante el método isToggled.

El ejemplo siguiente crea dos campos de texto que se actualizan cuando se conmuta el estado de las teclas Bloq Mayús y Bloq Num . Cada campo de texto muestra true cuando se activa la tecla y false cuando la tecla se desactiva.

```
this.createTextField("capsLock_txt", this.getNextHighestDepth(), 0, 0, 100,
  22);
capsLock_txt.autoSize = true;
capsLock_txt.html = true;
this.createTextField("numLock_txt", this.getNextHighestDepth(), 0, 22, 100,
  22);
numLock_txt.autoSize = true;
numLock_txt.html = true;
//
var keyListener:Object = new Object();
keyListener.onKeyDown = function() {
  capsLock_txt.htmlText = "<b>Caps Lock:</b> "+Key.isToggled(Key.CAPSLOCK);
  numLock_txt.htmlText = "<b>Num Lock:</b> "+Key.isToggled(144);
};
Key.addListener(keyListener);
```

El método `MovieClip.getNextHighestDepth()` utilizado en este ejemplo necesita Flash Player 7 o una versión posterior. Si el archivo SWF incluye un componente de la versión 2, utilice la clase DepthManager de los componentes de la versión 2 en lugar del método `MovieClip.getNextHighestDepth()`.

LEFT (propiedad Key.LEFT)

```
public static LEFT : Number
```

El valor de código de tecla asociado a la tecla de flecha izquierda (37).

Disponibilidad: ActionScript 1.0; Flash Player 5

Ejemplo

El ejemplo siguiente mueve un clip de película denominado `car_mc` hasta una distancia constante (10) al presionar las teclas de flecha. Se reproduce un sonido cuando se presiona la Barra espaciadora. Para este ejemplo, asigne a un sonido de la biblioteca el identificador de vinculación `horn_id`.

```
var DISTANCE:Number = 10;
var horn_sound:Sound = new Sound();
horn_sound.attachSound("horn_id");
var keyListener_obj:Object = new Object();
keyListener_obj.onKeyDown = function() {
  switch (Key.getCode()) {
  case Key.SPACE :
  horn_sound.start();
  break;
  case Key.LEFT :
  car_mc._x -= DISTANCE;
  break;
  case Key.UP :
  car_mc._y -= DISTANCE;
  break;
  case Key.RIGHT :
  car_mc._x += DISTANCE;
  break;
  case Key.DOWN :
  car_mc._y += DISTANCE;
  break;
  }
};
Key.addListener(keyListener_obj);
```

_listeners (propiedad Key._listeners)

```
public static _listeners : Array [read-only]
```

Una lista de las referencias a todos los objetos detectores registrados con el objeto Key. Esta propiedad es para uso interno, aunque puede resultar útil para determinar el número de detectores registrados actualmente en el objeto Key. Los objetos se añaden y quitan de esta matriz mediante llamadas a los métodos `addListener()` y `removelistener()`.

Disponibilidad: ActionScript 1.0; Flash Player 6

Ejemplo

En el ejemplo siguiente se muestra cómo utilizar la propiedad `length` para determinar el número de objetos detectores que están registrados actualmente en el objeto Key.

```
var myListener:Object = new Object();
myListener.onKeyDown = function () {
trace ("You pressed a key.");
}
Key.addListener(myListener);

trace(Key._listeners.length); // Output: 1
```

onKeyDown (detector de eventos Key.onKeyDown)

```
onKeyDown = function() {}
```

Se notifica cuando se presiona una tecla. Para utilizar `onKeyDown`, deberá crear un objeto detector. Posteriormente podrá definir una función para `onKeyDown` y utilizar `addListener()` para registrar el detector en el objeto Key, como se muestra en el siguiente ejemplo:

```
var keyListener:Object = new Object();
keyListener.onKeyDown = function() {
  trace("DOWN -> Code: "+Key.getCode()+"\tACSII: "+Key.getAscii()+"\tKey:
  "+chr(Key.getAscii()));
};
keyListener.onKeyUp = function() {
  trace("UP -> Code: "+Key.getCode()+"\tACSII: "+Key.getAscii()+"\tKey:
  "+chr(Key.getAscii()));
};
Key.addListener(keyListener);
```

Los detectores permiten que diversas partes del código cooperen, ya que varios detectores pueden recibir notificación de un solo evento.

Una aplicación Flash sólo puede controlar los eventos de teclado que se produzcan en la propia aplicación. Una aplicación Flash no puede detectar eventos de teclado de otra aplicación.

Disponibilidad: ActionScript 1.0; Flash Player 6

Véase también

addListener (método Key.addListener)

onKeyDown (detector de eventos Key.onKeyUp)

```
onKeyUp = function() {}
```

Se notifica cuando se suelta una tecla. Para utilizar `onKeyUp`, deberá crear un objeto detector. Posteriormente podrá definir una función para `onKeyUp` y utilizar `addListener()` para registrar el detector en el objeto Key, como se muestra en el siguiente ejemplo:

```
var keyListener:Object = new Object();
keyListener.onKeyDown = function() {
  trace("DOWN -> Code: "+Key.getCode()+"\tACSII: "+Key.getAscii()+"\tKey:
  "+chr(Key.getAscii()));
};
keyListener.onKeyUp = function() {
  trace("UP -> Code: "+Key.getCode()+"\tACSII: "+Key.getAscii()+"\tKey:
  "+chr(Key.getAscii()));
};
Key.addListener(keyListener);
```

Los detectores permiten que diversas partes del código cooperen, ya que varios detectores pueden recibir notificación de un solo evento.

Una aplicación Flash sólo puede controlar los eventos de teclado que se produzcan en la propia aplicación. Una aplicación Flash no puede detectar eventos de teclado de otra aplicación.

Disponibilidad: ActionScript 1.0; Flash Player 6

Véase también

addListener (método Key.addListener)

PGDN (propiedad Key.PGDN)

```
public static PGDN : Number
```

El valor de código de tecla asociado a la tecla AvPág (34).

Disponibilidad: ActionScript 1.0; Flash Player 5

Ejemplo

El ejemplo siguiente gira un clip de película denominado `car_mc` utilizando las teclas AvPág y RePág.

```
var keyListener:Object = new Object();
keyListener.onKeyDown = function() {
```

```
  if (Key.isDown(Key.PGDN)) {
  car_mc._rotation += 5;
  } else if (Key.isDown(Key.PGUP)) {
  car_mc._rotation -= 5;
  }
};
Key.addListener(keyListener);
```

PGUP (propiedad Key.PGUP)

```
public static PGUP : Number
```

El valor de código de tecla asociado a la tecla RePág (33).

Disponibilidad: ActionScript 1.0; Flash Player 5

Ejemplo

El ejemplo siguiente gira un clip de película denominado `car_mc` utilizando las teclas AvPág y RePág.

```
var keyListener:Object = new Object();
keyListener.onKeyDown = function() {
  if (Key.isDown(Key.PGDN)) {
  car_mc._rotation += 5;
  } else if (Key.isDown(Key.PGUP)) {
  car_mc._rotation -= 5;
  }
};
Key.addListener(keyListener);
```

removeListener (método Key.removeListener)

```
public static removeListener(listener:Object) : Boolean
```

Elimina un objeto registrado previamente con `Key.addListener()`.

Disponibilidad: ActionScript 1.0; Flash Player 6

Parámetros

`listener``:Object` - Un objeto.

Valor devuelto

`Boolean` - Si el *`listener`* se ha eliminado correctamente, el método devolverá `true`. Si el *`listener`* no se ha eliminado correctamente (por ejemplo, porque el *`listener`* no se encontraba en la lista de detectores del objeto Key), el método devolverá `false`.

Ejemplo

El ejemplo siguiente mueve un clip de película denominado `car_mc` utilizando las teclas de flecha izquierda y derecha. Cuando se presiona la tecla Esc, se elimina el detector y `car_mc` ya no podrá moverse.

```
var keyListener:Object = new Object();
keyListener.onKeyDown = function() {
  switch (Key.getCode()) {
  case Key.LEFT :
  car_mc._x -= 10;
  break;
  case Key.RIGHT :
  car_mc._x += 10;
  break;
  case Key.ESCAPE :
  Key.removeListener(keyListener);
  }
};
Key.addListener(keyListener);
```

RIGHT (propiedad Key.RIGHT)

```
public static RIGHT : Number
```

El valor de código de tecla asociado a la tecla de flecha derecha (39).

Disponibilidad: ActionScript 1.0; Flash Player 5

Ejemplo

El ejemplo siguiente mueve un clip de película denominado `car_mc` hasta una distancia constante (10) al presionar las teclas de flecha. Se reproduce un sonido cuando se presiona la Barra espaciadora. Para este ejemplo, asigne a un sonido de la biblioteca el identificador de vinculación `horn_id`.

```
var DISTANCE:Number = 10;
var horn_sound:Sound = new Sound();
horn_sound.attachSound("horn_id");
var keyListener_obj:Object = new Object();
keyListener_obj.onKeyDown = function() {
  switch (Key.getCode()) {
  case Key.SPACE :
  horn_sound.start();
  break;
  case Key.LEFT :
  car_mc._x -= DISTANCE;
  break;
  case Key.UP :
  car_mc._y -= DISTANCE;
```

```
  break;
  case Key.RIGHT :
  car_mc._x += DISTANCE;
  break;
  case Key.DOWN :
  car_mc._y += DISTANCE;
  break;
  }
};
Key.addListener(keyListener_obj);
```

SHIFT (propiedad Key.SHIFT)

```
public static SHIFT : Number
```

El valor de código de tecla asociado a la tecla Mayús (16).

Disponibilidad: ActionScript 1.0; Flash Player 5

Ejemplo

El ejemplo siguiente cambia la escala de `car_mc` al presionar Mayús.

```
var keyListener:Object = new Object();
keyListener.onKeyDown = function() {
  if (Key.isDown(Key.SHIFT)) {
  car_mc._xscale = 2;
  car_mc._yscale = 2;
  } else if (Key.isDown(Key.CONTROL)) {
  car_mc._xscale /= 2;
  car_mc._yscale /= 2;
  }
};
Key.addListener(keyListener);
```

SPACE (propiedad Key.SPACE)

```
public static SPACE : Number
```

El valor de código de tecla asociado a la tecla Barra espaciadora (32).

Disponibilidad: ActionScript 1.0; Flash Player 5

Ejemplo

El ejemplo siguiente mueve un clip de película denominado `car_mc` hasta una distancia constante (10) al presionar las teclas de flecha. Se reproduce un sonido cuando se presiona la Barra espaciadora. Para este ejemplo, asigne a un sonido de la biblioteca el identificador de vinculación `horn_id`.

```
var DISTANCE:Number = 10;
```

```
var horn_sound:Sound = new Sound();
horn_sound.attachSound("horn_id");
var keyListener_obj:Object = new Object();
keyListener_obj.onKeyDown = function() {
  switch (Key.getCode()) {
  case Key.SPACE :
  horn_sound.start();
  break;
  case Key.LEFT :
  car_mc._x -= DISTANCE;
  break;
  case Key.UP :
  car_mc._y -= DISTANCE;
  break;
  case Key.RIGHT :
  car_mc._x += DISTANCE;
  break;
  case Key.DOWN :
  car_mc._y += DISTANCE;
  break;
  }
};
Key.addListener(keyListener_obj);
```

TAB (propiedad Key.TAB)

```
public static TAB : Number
```

El valor de código de tecla asociado a la tecla Tabulador (9).

Disponibilidad: ActionScript 1.0; Flash Player 5

Ejemplo

El ejemplo siguiente crea un campo de texto y muestra la fecha en el campo de texto al presionar la tecla Tabulador.

```
this.createTextField("date_txt", this.getNextHighestDepth(), 0, 0, 100,
  22);
date_txt.autoSize = true;
var keyListener:Object = new Object();
keyListener.onKeyDown = function() {
  if (Key.isDown(Key.TAB)) {
  var today_date:Date = new Date();
  date_txt.text = today_date.toString();
  }
};
Key.addListener(keyListener);
```

Cuando utilice este ejemplo, asegúrese de seleccionar Control > Deshabilitar métodos abreviados de teclado en el entorno de prueba.

El método `MovieClip.getNextHighestDepth()` utilizado en este ejemplo necesita Flash Player 7 o una versión posterior. Si el archivo SWF incluye un componente de la versión 2, utilice la clase DepthManager de los componentes de la versión 2 en lugar del método `MovieClip.getNextHighestDepth()`.

UP (propiedad Key.UP)

```
public static UP : Number
```

El valor de código de tecla asociado a la tecla de flecha arriba (38).

Disponibilidad: ActionScript 1.0; Flash Player 5

Ejemplo

El ejemplo siguiente mueve un clip de película denominado `car_mc` hasta una distancia constante (10) al presionar las teclas de flecha. Se reproduce un sonido cuando se presiona la Barra espaciadora. Para este ejemplo, asigne a un sonido de la biblioteca el identificador de vinculación `horn_id`.

```
var DISTANCE:Number = 10;
var horn_sound:Sound = new Sound();
horn_sound.attachSound("horn_id");
var keyListener_obj:Object = new Object();
keyListener_obj.onKeyDown = function() {
  switch (Key.getCode()) {
  case Key.SPACE :
  horn_sound.start();
  break;
  case Key.LEFT :
  car_mc._x -= DISTANCE;
  break;
  case Key.UP :
  car_mc._y -= DISTANCE;
  break;
  case Key.RIGHT :
  car_mc._x += DISTANCE;
  break;
  case Key.DOWN :
  car_mc._y += DISTANCE;
  break;
  }
};
Key.addListener(keyListener_obj);
```

LoadVars

```
Object
  |
  +-LoadVars
```

```
public dynamic class LoadVars
extends Object
```

Puede utilizar la clase LoadVars para obtener confirmación de que determinados datos se han cargado correctamente y para controlar el progreso de las descargas. La clase LoadVars es una alternativa a la función `loadVariables()` para la transferencia de variables entre una aplicación Flash y un servidor.

La clase LoadVars permite enviar todas las variables de un objeto a una URL y cargar todas las variables de una URL en un objeto. También le permite enviar variables concretas en lugar de todas las variables, lo que puede contribuir a que la aplicación sea más eficaz. Puede utilizar el controlador `LoadVars.onLoad` para asegurarse de que la aplicación se ejecute cuando se carguen datos y no antes.

La clase LoadVars funciona de forma similar a la clase XML; utiliza los métodos `load()`, `send()` y `sendAndLoad()` para comunicarse con un servidor. La principal diferencia entre la clase LoadVars y la clase XML estriba en que LoadVars transfiere pares de nombre y valor ActionScript, en lugar de un árbol DOM XML almacenado en el objeto XML. La clase LoadVars presenta las mismas restricciones de seguridad que la clase XML.

Disponibilidad: ActionScript 1.0; Flash Player 6

Véase también

`Función loadVariables`, `onLoad (controlador LoadVars.onLoad)`, `XML`

Resumen de propiedades

Modificadores	Propiedad	Descripción
	`contentType:String`	El tipo MIME que se envía al servidor cuando se llama a `LoadVars.send()` o `LoadVars.sendAndLoad()`.
	`loaded:Boolean`	Valor booleano que indica si una operación `load` o `sendAndLoad` ha finalizado; el valor predeterminado es `undefined` (no definido).

Propiedades heredadas de la clase Object

constructor (propiedad Object.constructor), __proto__ (propiedad Object.__proto__), prototype (propiedad Object.prototype), __resolve (propiedad Object.__resolve)

Resumen de eventos

Evento	Descripción
`onData = function(src:String) {}`	Se invoca cuando se han descargado completamente los datos del servidor o cuando se produce un error mientras se están descargando datos de un servidor.
`onHTTPStatus = function(httpStatus:Number) {}`	Se invoca cuando Flash Player recibe un código de estado HTTP del servidor.
`onLoad = function(success:Boolean) {}`	Se invoca cuando ha finalizado una operación `LoadVars.load()` o `LoadVars.sendAndLoad()`.

Resumen de constructores

Firma	Descripción
`LoadVars()`	Crea un objeto LoadVars.

Resumen de métodos

Modificadores	Firma	Descripción
	`addRequestHeader(header:Object, headerValue:String) : Void`	Añade o cambia los encabezados de peticiones HTTP (como `Content-Type` o `SOAPAction`) enviados con las acciones `POST`.
	`decode(queryString:String) : Void`	Convierte la cadena variable en propiedades del objeto LoadVars especificado.
	`getBytesLoaded() : Number`	Devuelve el número de bytes descargados por `LoadVars.load()` o `LoadVars.sendAndLoad()`.
	`getBytesTotal() : Number`	Devuelve el total de bytes descargados por `LoadVars.load()` o `LoadVars.sendAndLoad()`.
	`load(url:String) : Boolean`	Descarga variables de la URL especificada, analiza los datos de la variable y coloca las variables resultantes en `my_lv`.

Modificadores	Firma	Descripción
	`send(url:String, target:String, [method:String]) : Boolean`	Envía las variables contenidas en el objeto *my_lv* a la URL especificada.
	`sendAndLoad(url:String, target:Object, [method:String]) : Boolean`	Coloca las variables contenidas en el objeto `my_lv` en la URL especificada.
	`toString() : String`	Devuelve una cadena que contiene todas las variables enumerables de *my_lv*, en la codificación de contenido MIME *application/x-www-form-urlencoded*.

Métodos heredados de la clase Object

`addProperty (método Object.addProperty)`, `hasOwnProperty (método Object.hasOwnProperty)`, `isPropertyEnumerable (método Object.isPropertyEnumerable)`, `isPrototypeOf (método Object.isPrototypeOf)`, `registerClass (método Object.registerClass)`, `toString (método Object.toString)`, `unwatch (método Object.unwatch)`, `valueOf (método Object.valueOf)`, `watch (método Object.watch)`

addRequestHeader (método LoadVars.addRequestHeader)

`public addRequestHeader(header:Object, headerValue:String) : Void`

Añade o cambia los encabezados de peticiones HTTP (como `Content-Type` o `SOAPAction`) enviados con las acciones `POST`. En la primera sintaxis, se pasan dos cadenas al método: `header` y `headerValue`. En la segunda sintaxis, se pasa una matriz de cadenas, alternando los nombres de los encabezados y los valores de éstos.

Si se realizan varias llamadas para establecer el mismo nombre de encabezado, cada valor sucesivo reemplazará al valor establecido en la llamada anterior.

Los siguientes encabezados HTTP estándar *no se pueden* añadir ni cambiar con este método: `Accept-Ranges`, `Age`, `Allow`, `Allowed`, `Connection`, `Content-Length`, `Content-Location`, `Content-Range`, `ETag`, `Host`, `Last-Modified`, `Locations`, `Max-Forwards`, `Proxy-Authenticate`, `Proxy-Authorization`, `Public`, `Range`, `Retry-After`, `Server`, `TE`, `Trailer`, `Transfer-Encoding`, `Upgrade`, `URI`, `Vary`, `Via`, `Warning` y `WWW-Authenticate`.

Disponibilidad: ActionScript 1.0; Flash Player 6

Parámetros

header:Object - Una cadena o matriz de cadenas que representan un nombre de encabezado de petición HTTP.

headerValue:String - Una cadena que representa el valor asociado con header.

Ejemplo

El ejemplo siguiente añade un encabezado HTTP personalizado denominado SOAPAction con un valor de Foo al objeto my_lv:

```
my_lv.addRequestHeader("SOAPAction", "'Foo'");
```

El ejemplo siguiente crea una matriz headers que contiene dos encabezados HTTP alternativos y sus valores asociados. La matriz se pasa como un argumento a addRequestHeader().

```
var headers = ["Content-Type", "text/plain", "X-ClientAppVersion", "2.0"];
my_lv.addRequestHeader(headers);
```

El ejemplo siguiente crea un nuevo objeto LoadVars que añade un encabezado de petición denominado FLASH-UUID. El encabezado contiene una variable que el servidor puede comprobar.

```
var my_lv:LoadVars = new LoadVars();
my_lv.addRequestHeader("FLASH-UUID", "41472");
my_lv.name = "Mort";
my_lv.age = 26;
my_lv.send("http://flash-mx.com/mm/cgivars.cfm", "_blank", "POST");
```

Véase también

addRequestHeader (método XML.addRequestHeader)

contentType (propiedad LoadVars.contentType)

```
public contentType : String
```

El tipo MIME que se envía al servidor cuando se llama a LoadVars.send() o LoadVars.sendAndLoad(). El valor predeterminado es *application/x-www-form-urlencoded*.

Disponibilidad: ActionScript 1.0; Flash Player 6

Ejemplo

El ejemplo siguiente crea un objeto LoadVars y muestra el tipo de contenido predeterminado de los datos enviados al servidor.

```
var my_lv:LoadVars = new LoadVars();
trace(my_lv.contentType); // output: application/x-www-form-urlencoded
```

Véase también

send (método LoadVars.send), sensendAndLoad (método LoadVars.sendAndLoad)

decode (método LoadVars.decode)

```
public decode(queryString:String) : Void
```

Convierte la cadena variable en propiedades del objeto LoadVars especificado.

El controlador de eventos `LoadVars.onData` utiliza internamente este método. La mayoría de los usuarios no necesitan llamar a este método directamente. Si sustituye el controlador de eventos `LoadVars.onData`, puede llamar explícitamente a `LoadVars.decode()` para analizar una cadena de variables.

Disponibilidad: ActionScript 1.0; Flash Player 7

Parámetros

queryString:String - Una cadena de consulta con codificación URL con pares nombre/valor.

Ejemplo

El ejemplo siguiente rastrea las tres variables:

```
// Create a new LoadVars object
var my_lv:LoadVars = new LoadVars();
//Convert the variable string to properties
my_lv.decode("name=Mort&score=250000");
trace(my_lv.toString());
// Iterate over properties in my_lv
for (var prop in my_lv) {
  trace(prop+" -> "+my_lv[prop]);
}
```

Véase también

onData (controlador LoadVars.onData), parseXML (método XML.parseXML)

getBytesLoaded (método LoadVars.getBytesLoaded)

```
public getBytesLoaded() : Number
```

Devuelve el número de bytes descargados por `LoadVars.load()` o `LoadVars.sendAndLoad()`. Este método devuelve `undefined` si no hay ninguna operación de carga en curso o si no ha comenzado todavía ninguna operación de carga.

Disponibilidad: ActionScript 1.0; Flash Player 6

Valor devuelto

`Number` - Un entero.

Ejemplo

El ejemplo siguiente utiliza una instancia ProgressBar y un objeto LoadVars para descargar un archivo de texto. Cuando pruebe el archivo de texto, se muestran dos cosas en el panel Salida: si el archivo se carga correctamente y la cantidad de datos que se carga en el archivo SWF. Debe reemplazar el parámetro URL del comando `LoadVars.load()` de modo que el parámetro se refiera a un archivo de texto válido que utilice HTTP. Si intenta utilizar este ejemplo para cargar un archivo local que resida en el disco duro, el ejemplo no funcionará correctamente porque, en el modo Probar película, Flash Player carga los archivos locales íntegramente. Para ver cómo funciona este código, añada una instancia ProgressBar denominada `loadvars_pb` en el escenario. A continuación, añada el código ActionScript siguiente al fotograma 1 de la línea de tiempo:

```
var loadvars_pb:mx.controls.ProgressBar;
var my_lv:LoadVars = new LoadVars();
loadvars_pb.mode = "manual";
this.createEmptyMovieClip("timer_mc", 999);
timer_mc.onEnterFrame = function() {
  var lvBytesLoaded:Number = my_lv.getBytesLoaded();
  var lvBytesTotal:Number = my_lv.getBytesTotal();
  if (lvBytesTotal != undefined) {
  trace("Loaded "+lvBytesLoaded+" of "+lvBytesTotal+" bytes.");
  loadvars_pb.setProgress(lvBytesLoaded, lvBytesTotal);
  }
};
my_lv.onLoad = function(success:Boolean) {
  loadvars_pb.setProgress(my_lv.getBytesLoaded(), my_lv.getBytesTotal());
  delete timer_mc.onEnterFrame;
  if (success) {
  trace("LoadVars loaded successfully.");
  } else {
  trace("An error occurred while loading variables.");
  }
```

```
};
my_lv.load("[place a valid URL pointing to a text file here]");
```

Véase también

load (método LoadVars.load), sensendAndLoad (método LoadVars.sendAndLoad)

getBytesTotal (método LoadVars.getBytesTotal)

```
public getBytesTotal() : Number
```

Devuelve el total de bytes descargados por `LoadVars.load()` o `LoadVars.sendAndLoad()`. Este método devuelve `undefined` si no hay ninguna operación en curso o si no ha comenzado ninguna operación de carga. Este método también devuelve `undefined` si el número total de bytes no se puede determinar (por ejemplo, si la descarga se inició pero el servidor no transmitió un content-length HTTP).

Disponibilidad: ActionScript 1.0; Flash Player 6

Valor devuelto

`Number` - Un entero.

Ejemplo

El ejemplo siguiente utiliza una instancia ProgressBar y un objeto LoadVars para descargar un archivo de texto. Cuando pruebe el archivo de texto, se muestran dos cosas en el panel Salida: si el archivo se carga correctamente y la cantidad de datos que se carga en el archivo SWF. Debe reemplazar el parámetro URL del comando `LoadVars.load()` de modo que el parámetro se refiera a un archivo de texto válido que utilice HTTP. Si intenta utilizar este ejemplo para cargar un archivo local que resida en el disco duro, el ejemplo no funcionará correctamente porque, en el modo Probar película, Flash Player carga los archivos locales íntegramente. Para ver cómo funciona este código, añada una instancia ProgressBar denominada `loadvars_pb` en el escenario. A continuación, añada el código ActionScript siguiente al fotograma 1 de la línea de tiempo:

```
var loadvars_pb:mx.controls.ProgressBar;
var my_lv:LoadVars = new LoadVars();
loadvars_pb.mode = "manual";
this.createEmptyMovieClip("timer_mc", 999);
timer_mc.onEnterFrame = function() {
  var lvBytesLoaded:Number = my_lv.getBytesLoaded();
  var lvBytesTotal:Number = my_lv.getBytesTotal();
  if (lvBytesTotal != undefined) {
  trace("Loaded "+lvBytesLoaded+" of "+lvBytesTotal+" bytes.");
  loadvars_pb.setProgress(lvBytesLoaded, lvBytesTotal);
  }
```

```
};
my_lv.onLoad = function(success:Boolean) {
  loadvars_pb.setProgress(my_lv.getBytesLoaded(), my_lv.getBytesTotal());
  delete timer_mc.onEnterFrame;
  if (success) {
  trace("LoadVars loaded successfully.");
  } else {
  trace("An error occurred while loading variables.");
  }
};
my_lv.load("[place a valid URL pointing to a text file here]");
```

Véase también

load (método LoadVars.load), sensendAndLoad (método LoadVars.sendAndLoad)

load (método LoadVars.load)

```
public load(url:String) : Boolean
```

Descarga variables de la URL especificada, analiza los datos de la variable y coloca las variables resultantes en `my_lv`. Las propiedades existentes en `my_lv` que tengan los mismos nombres que las variables descargadas se sobrescribirán. Las propiedades existentes en `my_lv` que tengan nombres diferentes a los de las variables descargadas no se eliminarán. Esta es una acción asíncrona.

Los datos descargados deben tener el tipo de contenido MIME *application/x-www-form-urlencoded*.

Es el mismo formato que se utiliza en `loadVariables()`.

Asimismo, en archivos publicados para Flash Player 7, la distinción entre mayúsculas y minúsculas se admite para variables externas cargadas con `LoadVars.load()`.

Este método es similar a `XML.load()`.

Nota: Si un archivo que se está cargando contiene caracteres no ASCII (como ocurre en muchos idiomas, no así en inglés), se recomienda que guarde el archivo con codificación UTF-8 o UTF-16, en lugar de utilizar un formato no Unicode, como ASCII.

Cuando utilice este método, puede ser conveniente usar el modelo de seguridad de Flash Player:

Para Flash Player 8:

- No se permite la carga de datos si el archivo SWF que realiza la llamada se encuentra en el entorno limitado local con sistema de archivos y el recurso se encuentra en un entorno limitado de red.
- Tampoco se permite la carga de datos si el archivo SWF que realiza la llamada se encuentra en un entorno limitado de red y el recurso de destino es local.

Para más información, consulte lo siguiente:

- Capítulo 17, "Aspectos básicos de la seguridad," en *Aprendizaje de ActionScript 2.0 en Flash*
- El documento técnico sobre la seguridad de Flash Player 8 en http://www.macromedia.com/go/fp8_security
- El documento técnico sobre interfaces API relativas a seguridad de Flash Player 8 en http://www.macromedia.com/go/fp8_security_apis

Para Flash Player 7 y versiones posteriores, los sitios Web pueden permitir el acceso de distintos dominios a un recurso mediante un archivo de política de varios dominios. En archivos SWF de cualquier versión que se ejecuten en Flash Player 7 y posterior, `url` debe estar exactamente en el mismo dominio. Por ejemplo, un archivo SWF de www.someDomain.com puede cargar datos únicamente de orígenes que estén también en www.someDomain.com.

En archivos SWF que se ejecuten en una versión del reproductor anterior a Flash Player 7, `url` debe estar en el mismo superdominio que el archivo SWF que emite esta llamada. El superdominio puede obtenerse eliminando el componente situado más a la izquierda de la URL de un archivo. Por ejemplo, un archivo SWF situado en www.someDomain.com puede cargar datos de fuentes situadas en store.someDomain.com porque ambos archivos se encuentran en el mismo superdominio, denominado someDomain.com.

Disponibilidad: ActionScript 1.0; Flash Player 6

Parámetros

`url``:String` - Una cadena; la URL desde la que se descargarán las variables. Si el archivo SWF que realiza esta llamada se ejecuta en un navegador Web, el valor `url` debe pertenecer al mismo dominio que el archivo SWF.

Valor devuelto

`Boolean` - `false` si no se pasa ningún parámetro (null); `true`, en caso contrario. Utilice el controlador de eventos `onLoad()` para comprobar el estado de los datos cargados.

Ejemplo

El código siguiente define una función de controlador `onLoad` que emite una señal cuando se devuelven datos a la aplicación Flash desde un archivo de script PHP de servidor y carga los datos en passvars.php.

```
var my_lv:LoadVars = new LoadVars();
my_lv.onLoad = function(success:Boolean) {
  if (success) {
  trace(this.toString());
```

```
  } else {
  trace("Error loading/parsing LoadVars.");
  }
};
my_lv.load("http://www.helpexamples.com/flash/params.txt");
```

Para ver otro ejemplo, consulte el archivo guestbook.fla en la carpeta de ejemplos de ActionScript. Estas son rutas de acceso habituales a esta carpeta:

- Windows: *unidad de inicio*\Archivos de programa\Macromedia\Flash 8\Samples and Tutorials\Samples\ActionScript
- Macintosh: *Disco duro de Macintosh*/Applications/Macromedia Flash 8/Samples and Tutorials/Samples/ActionScript

Véase también

`load (método XML.load)`, `loaded (propiedad LoadVars.loaded)`, `onLoad (controlador LoadVars.onLoad)`, `useCodepage (propiedad System.useCodepage)`

loaded (propiedad LoadVars.loaded)

```
public loaded : Boolean
```

Valor booleano que indica si una operación `load` o `sendAndLoad` ha finalizado; el valor predeterminado es `undefined` (no definido). Cuando se inicia una operación `LoadVars.load()` o `LoadVars.sendAndLoad()`, la propiedad `loaded` es `false`; cuando finaliza la operación, la propiedad `loaded` es `true`. Si la operación no ha finalizado o ha fallado con un error, la propiedad `loaded` continuará establecida en `false`.

Esta propiedad es similar a `XML.loaded`.

Disponibilidad: ActionScript 1.0; Flash Player 6

Ejemplo

El ejemplo siguiente carga un archivo de texto y muestra información en el panel Salida al finalizar la operación.

```
var my_lv:LoadVars = new LoadVars();
my_lv.onLoad = function(success:Boolean) {
  trace("LoadVars loaded successfully: "+this.loaded);
};
my_lv.load("http://www.helpexamples.com/flash/params.txt");
```

Véase también

`load (método LoadVars.load)`, `sensendAndLoad (método LoadVars.sendAndLoad)`, `load (método XML.load)`

LoadVars, constructor

```
public LoadVars()
```

Crea un objeto LoadVars. Posteriormente podrá utilizar los métodos de dicho objeto LoadVars para enviar y cargar los datos.

Disponibilidad: ActionScript 1.0; Flash Player 6

Ejemplo

En el ejemplo siguiente se crea un objeto LoadVars denominado `my_lv`:

```
var my_lv:LoadVars = new LoadVars();
```

onData (controlador LoadVars.onData)

```
onData = function(src:String) {}
```

Se invoca cuando se han descargado completamente los datos del servidor o cuando se produce un error mientras se están descargando datos de un servidor. Este controlador se invoca antes de que se analicen los datos, por lo que puede utilizarse para llamar a una rutina de análisis personalizada en lugar de la incorporada en Flash Player. El valor del parámetro `src` pasado a la función asignada a `LoadVars.onData` puede ser `undefined` (no definido) o una cadena que contenga los pares nombre-valor con codificación URL descargados del servidor. Si el parámetro `src` tiene el valor `undefined`, ello indica que se ha producido un error al descargar los datos del servidor.

La implementación predeterminada de `LoadVars.onData` llama a `LoadVars.onLoad`. Puede sustituir esta implementación predeterminada asignando una función personalizada a `LoadVars.onData`, pero no se llama a `LoadVars.onLoad` a menos que lo haga en su implementación de `LoadVars.onData`.

Disponibilidad: ActionScript 1.0; Flash Player 6

Parámetros

`src``:String` - Una cadena o `undefined`; los datos sin analizar de una llamada a un método `LoadVars.load()` o `LoadVars.sendAndLoad()`.

Ejemplo

El ejemplo siguiente carga un archivo de texto y muestra su contenido en una instancia TextArea denominada `content_ta` al finalizar la operación. Si se produce un error, aparecerá información en el panel Salida.

```
var my_lv:LoadVars = new LoadVars();
my_lv.onData = function(src:String) {
```

```
  if (src == undefined) {
    trace("Error loading content.");
    return;
  }
  content_ta.text = src;
};
my_lv.load("content.txt", my_lv, "GET");
```

Véase también

onLoad (controlador LoadVars.onLoad), onLoad (controlador LoadVars.onLoad), load (método LoadVars.load), sensendAndLoad (método LoadVars.sendAndLoad)

onData (controlador LoadVars.onHTTPStatus)

```
onHTTPStatus = function(httpStatus:Number) {}
```

Se invoca cuando Flash Player recibe un código de estado HTTP del servidor. Este controlador permite capturar y trabajar con códigos de estado HTTP.

El controlador `onHTTPStatus` se llama antes que `onData`, que activa llamadas a `onLoad` con un valor de `undefined` si falla la carga. Tras activar `onHTTPStatus`, se activa *siempre* `onData`, tanto si sustituye `onHTTPStatus` como si no. Para utilizar mejor el controlador `onHTTPStatus`, debe escribir una función para capturar el resultado de la llamada a `onHTTPStatus`; a continuación puede utilizar el resultado en sus controladores `onData` y `onLoad`. Si no se llama a `onHTTPStatus`, el reproductor no ha intentado realizar la petición URL. Esto puede ocurrir debido a que la petición quebranta las reglas de seguridad del entorno limitado para el archivo SWF.

Si Flash Player no puede obtener el código de estado del servidor o si no puede comunicarse con el servidor, se pasa el valor predeterminado 0 al código ActionScript. Es posible generar un valor 0 en cualquier reproductor (por ejemplo, si se solicita una dirección URL mal formada) y el plug-in de Flash Player siempre genera un valor 0 cuando se ejecuta en los navegadores siguientes, ya que no pasan códigos de estado HTTP al reproductor: Netscape, Mozilla, Safari, Opera e Internet Explorer para Macintosh.

Disponibilidad: ActionScript 1.0; Flash Player 8

Parámetros

`httpStatus``:Number` - El código de estado HTTP que devuelve el servidor. Por ejemplo, 404 indica que el servidor no encontró nada que coincida con el URI solicitado. Los códigos de estado HTTP se encuentran en las secciones 10.4 y 10.5 de la especificación HTTP en ftp://ftp.isi.edu/in-notes/rfc2616.txt.

Ejemplo

El siguiente ejemplo muestra cómo utilizar `onHTTPStatus()` para facilitar la depuración. El ejemplo recoge códigos de estado HTTP y asigna su valor y tipo a una instancia del objeto LoadVars. (En este ejemplo se crean los miembros de instancia `this.httpStatus` y `this.httpStatusType` en tiempo de ejecución.) El método `onData` utiliza estos miembros de instancia para controlar información sobre la respuesta HTTP que puede resultar útil en la depuración.

```
var myLoadVars:LoadVars = new LoadVars();

myLoadVars.onHTTPStatus = function(httpStatus:Number) {
  this.httpStatus = httpStatus;
  if(httpStatus < 100) {
    this.httpStatusType = "flashError";
  }
  else if(httpStatus < 200) {
    this.httpStatusType = "informational";
  }
  else if(httpStatus < 300) {
    this.httpStatusType = "successful";
  }
  else if(httpStatus < 400) {
    this.httpStatusType = "redirection";
  }
  else if(httpStatus < 500) {
    this.httpStatusType = "clientError";
  }
  else if(httpStatus < 600) {
    this.httpStatusType = "serverError";
  }
}

myLoadVars.onData = function(src:String) {
  trace(">> " + this.httpStatusType + ": " + this.httpStatus);
  if(src != undefined) {
    this.decode(src);
    this.loaded = true;
    this.onLoad(true);
  }
  else {
    this.onLoad(false);
  }
}

myLoadVars.onLoad = function(success:Boolean) {
}

myLoadVars.load("http://weblogs.macromedia.com/mxna/flashservices/
  getMostRecentPosts.cfm");
```

Véase también

onHTTPStatus (controlador XML.onHTTPStatus), load (método LoadVars.load), sensendAndLoad (método LoadVars.sendAndLoad)

onLoad (controlador LoadVars.onLoad)

```
onLoad = function(success:Boolean) {}
```

Se invoca cuando ha finalizado una operación `LoadVars.load()` o `LoadVars.sendAndLoad()`. Si la operación se ha realizado correctamente, `my_lv` se llena con las variables descargadas por la operación y dichas variables estarán disponibles cuando se invoque este controlador.

De manera predeterminada, este controlador tiene el valor undefined (no definido).

Este controlador de eventos es similar a `XML.onLoad`.

Disponibilidad: ActionScript 1.0; Flash Player 6

Parámetros

success:Boolean - Un valor booleano que indica si la operación de carga ha finalizado correctamente (`true`) o si ha fallado (`false`).

Ejemplo

El ejemplo siguiente añade una instancia de TextInput denominada `name_ti`, una instancia TextArea denominada `result_ta` y una instancia Button denominada `submit_button` en el escenario. Cuando un usuario hace clic en la instancia de botón Login, se crean dos objetos LoadVars: `send_lv` y `result_lv`. El objeto `send_lv` copia el nombre de la instancia `name_ti` y envía los datos a greeting.cfm. El resultado de este script se carga en el objeto `result_lv` y la respuesta del servidor aparece en la instancia TextArea (`result_ta`). Añada el siguiente código ActionScript en el fotograma 1 de la línea de tiempo:

```
var submitListener:Object = new Object();
submitListener.click = function(evt:Object) {
  var result_lv:LoadVars = new LoadVars();
  result_lv.onLoad = function(success:Boolean) {
    if (success) {
      result_ta.text = result_lv.welcomeMessage;
    } else {
      result_ta.text = "Error connecting to server.";
    }
  };
  var send_lv:LoadVars = new LoadVars();
  send_lv.name = name_ti.text;
  send_lv.sendAndLoad("http://www.flash-mx.com/mm/greeting.cfm",
  result_lv, "POST");
```

```
};
submit_button.addEventListener("click", submitListener);
```

Para ver un ejemplo más complejo, consulte el archivo login.fla de la carpeta de ejemplos de ActionScript. Estas son rutas de acceso habituales a esta carpeta:

- Windows: *unidad de inicio*\Archivos de programa\Macromedia\Flash 8\Samples and Tutorials\Samples\ActionScript
- Macintosh: *Disco duro de Macintosh*/Applications/Macromedia Flash 8/Samples and Tutorials/Samples/ActionScript

Véase también

`onLoad (controlador XML.onLoad)`, `loaded (propiedad LoadVars.loaded)`, `load (método LoadVars.load)`, `sensendAndLoad (método LoadVars.sendAndLoad)`

send (método LoadVars.send)

```
public send(url:String, target:String, [method:String]) : Boolean
```

Envía las variables contenidas en el objeto *`my_lv`* a la URL especificada. De manera predeterminada, todas las variables enumerables existentes en *`my_lv`* se concatenan en una cadena en el formato *`application/x-www-form-urlencoded`* y la cadena se coloca en la URL empleando el método HTTP `POST`. Es el mismo formato que utiliza `loadVariables()`. El tipo de contenido MIME enviado en los encabezados de peticiones HTTP es el valor de `my_lv.contentType` o el valor predeterminado *`application/x-www-form-urlencoded`*. Se utiliza el método `POST` a no ser que se especifique `GET`.

Debe especificar el parámetro `target` para asegurarse de que se ejecute el script o la aplicación de la URL especificada. Si omite el parámetro `target`, la función devolverá `true`, pero el script o aplicación no se ejecutará.

El método `send()` resulta útil si se desea la respuesta del servidor para:

- Reemplazar el contenido del SWF (utilice `"_self"` como parámetro de `target`);
- Que aparezca en una nueva ventana (utilice `"_blank"` como parámetro de `target`);
- Que aparezca en el marco superior o del nivel más alto (utilice `"_parent"` o `"_top"` como parámetro de `target`);
- Que aparezca en un marco sin nombre (utilice el nombre del marco como cadena para el parámetro `target`).

Una llamada correcta al método `send()` siempre abre una nueva ventana del navegador o reemplaza el contenido de una ventana o un marco existente. Si prefiere enviar la información a un servidor y continuar reproduciendo el archivo SWF sin abrir una ventana nueva ni reemplazar el contenido de una ventana o un marco existente, deberá utilizar `LoadVars.sendAndLoad()`.

Este método es similar a `XML.send()`.

El entorno de prueba de Flash siempre utiliza el método GET. Si desea probar el método POST, debe asegurarse de que intenta utilizarlo desde un navegador.

Cuando utilice este método, puede ser conveniente usar el modelo de seguridad de Flash Player:

- En el caso de Flash Player 8, el método no se admite si el archivo SWF que realiza la llamada se encuentra en un entorno limitado local que no es de confianza.
- En el caso de Flash Player 7, el método no se admite si el archivo SWF que realiza la llamada es un archivo local.

Para más información, consulte lo siguiente:

- Capítulo 17, "Aspectos básicos de la seguridad," en *Aprendizaje de ActionScript 2.0 en Flash*
- El documento técnico sobre la seguridad de Flash Player 8 en http://www.macromedia.com/go/fp8_security
- El documento técnico sobre interfaces API relativas a seguridad de Flash Player 8 en http://www.macromedia.com/go/fp8_security_apis

Disponibilidad: ActionScript 1.0; Flash Player 6

Parámetros

`url``:String` - Una cadena; la URL a la que se cargarán las variables.

`target``:String` - Una cadena; la ventana o marco del navegador en el que aparecerá la respuesta. Puede introducir el nombre de una ventana específica o seleccionarlo de entre los siguientes nombres de destino reservados:

- `"_self"` especifica el fotograma actual en la ventana actual.
- `"_blank"` especifica una nueva ventana.
- `"_parent"` especifica el elemento principal del fotograma actual.
- `_top` especifica el fotograma de nivel superior de la ventana actual.

`method``:String` [opcional] - Una cadena; el método `GET` o `POST` del protocolo HTTP. El valor predeterminado es `POST`.

Valor devuelto

`Boolean` - Un valor booleano; `false` si no se especifica ningún parámetro; `true` en caso contrario.

Ejemplo

El ejemplo siguiente copia dos valores de campos de texto y envía los datos a un script CFM, utilizado para manejar la información. Por ejemplo, el script puede comprobar si el usuario ha obtenido una alta puntuación e insertar los datos en una tabla de base de datos.

```
var my_lv:LoadVars = new LoadVars();
my_lv.playerName = playerName_txt.text;
my_lv.playerScore = playerScore_txt.text;
my_lv.send("setscore.cfm", "_blank", "POST");
```

Véase también

`sensendAndLoad (método LoadVars.sendAndLoad)`, `send (método XML.send)`

sensendAndLoad (método LoadVars.sendAndLoad)

```
public sendAndLoad(url:String, target:Object, [method:String]) : Boolean
```

Coloca las variables contenidas en el objeto `my_lv` en la URL especificada. La respuesta del servidor se descarga, se analiza como datos variables y las variables resultantes se colocan en el objeto `target`.

Las variables se colocan de la misma forma que en el caso de `LoadVars.send()`. Las variables se descargan en `target` de la misma forma que en el caso de `LoadVars.load()`.

Cuando utilice este método, puede ser conveniente usar el modelo de seguridad de Flash Player:

Para Flash Player 8:

- No se permite la carga de datos si el archivo SWF que realiza la llamada se encuentra en el entorno limitado local con sistema de archivos y el recurso se encuentra en un entorno limitado de red.
- Tampoco se permite la carga de datos si el archivo SWF que realiza la llamada se encuentra en un entorno limitado de red y el recurso de destino es local.

Para más información, consulte lo siguiente:

- Capítulo 17, "Aspectos básicos de la seguridad," en *Aprendizaje de ActionScript 2.0 en Flash*
- El documento técnico sobre la seguridad de Flash Player 8 en http://www.macromedia.com/go/fp8_security
- El documento técnico sobre interfaces API relativas a seguridad de Flash Player 8 en http://www.macromedia.com/go/fp8_security_apis

Flash Player 7 y versiones anteriores:

- Los sitios Web pueden conceder acceso a varios dominios a un recurso mediante un archivo de política de varios dominios.
- En archivos SWF de cualquier versión que se ejecuten en Flash Player 7 y posterior, `url` debe estar exactamente en el mismo dominio. Por ejemplo, un archivo SWF de www.someDomain.com puede cargar datos únicamente de orígenes que estén también en www.someDomain.com.

En archivos SWF que se ejecuten en una versión del reproductor anterior a Flash Player 7, `url` debe estar en el mismo superdominio que el archivo SWF que emite esta llamada. El superdominio puede obtenerse eliminando el componente situado más a la izquierda de la URL de un archivo. Por ejemplo, un archivo SWF situado en www.someDomain.com puede cargar datos de fuentes situadas en store.someDomain.com porque ambos archivos se encuentran en el mismo superdominio, denominado someDomain.com.

Este método es similar a `XML.sendAndLoad()`.

Disponibilidad: ActionScript 1.0; Flash Player 6

Parámetros

`url``:String` - Una cadena; la URL a la que se cargarán las variables. Si el archivo SWF que realiza esta llamada se ejecuta en un navegador Web, el valor `url` debe pertenecer al mismo dominio que el archivo SWF.

`target``:Object` - El objeto LoadVars o XML que recibe las variables descargadas.

`method``:String` [opcional] - Una cadena; el método `GET` o `POST` del protocolo HTTP. El valor predeterminado es `POST`.

Valor devuelto

`Boolean` - Valor booleano.

Ejemplo

Para el ejemplo siguiente, añada una instancia de TextInput denominada `name_ti`, una instancia TextArea denominada `result_ta` y una instancia Button denominada `submit_button` en el escenario. Cuando un usuario hace clic en la instancia de botón Login en el ejemplo siguiente, se crean dos objetos LoadVars: `send_lv` y `result_lv`. El objeto `send_lv` copia el nombre de la instancia `name_ti` y envía los datos a greeting.cfm. El resultado de este script se carga en el objeto `result_lv` y la respuesta del servidor aparece en la instancia TextArea (`result_ta`). Añada el siguiente código ActionScript al fotograma 1 de la línea de tiempo:

```
var submitListener:Object = new Object();
submitListener.click = function(evt:Object) {
  var result_lv:LoadVars = new LoadVars();
  result_lv.onLoad = function(success:Boolean) {
  if (success) {
    result_ta.text = result_lv.welcomeMessage;
  } else {
    result_ta.text = "Error connecting to server.";
  }
  };
  var send_lv:LoadVars = new LoadVars();
  send_lv.name = name_ti.text;
  send_lv.sendAndLoad("http://www.flash-mx.com/mm/greeting.cfm",
  result_lv, "POST");
};
submit_button.addEventListener("click", submitListener);
```

Para ver un ejemplo más complejo, consulte el archivo login.fla de la carpeta de ejemplos de ActionScript. La carpeta de ejemplos de ActionScript suele estar en rutas como las siguientes:

- Windows: *unidad de inicio*\Archivos de programa\Macromedia\Flash 8\Samples and Tutorials\Samples\ActionScript
- Macintosh: *Disco duro de Macintosh*/Applications/Macromedia Flash 8/Samples and Tutorials/Samples/ActionScript

Véase también

`send (método LoadVars.send)`, `load (método LoadVars.load)`, `sendAndLoad (método XML.sendAndLoad)`

toString (método LoadVars.toString)

```
public toString() : String
```

Devuelve una cadena que contiene todas las variables enumerables de *my_lv* , en la codificación de contenido MIME *application/x-www-form-urlencoded*.

Disponibilidad: ActionScript 1.0; Flash Player 6

Valor devuelto

`String` - Una cadena.

Ejemplo

El ejemplo siguiente crea una instancia de un nuevo objeto `LoadVars()`, crea dos propiedades y utiliza `toString()` para devolver una cadena que contiene las dos propiedades en un formato URL codificado:

```
var my_lv:LoadVars = new LoadVars();
my_lv.name = "Gary";
my_lv.age = 26;
trace (my_lv.toString()); //output: age=26&name=Gary
```

LocalConnection

```
Object
  |
  +-LocalConnection
```

```
public dynamic class LocalConnection
extends Object
```

La clase LocalConnection le permite desarrollar archivos SWF que pueden enviarse instrucciones sin utilizar `fscommand()` ni JavaScript. Los objetos LocalConnection sólo pueden comunicarse entre archivos SWF que se ejecuten en el mismo equipo cliente, aunque pueden estar ejecutándose en diferentes aplicaciones: por ejemplo, un archivo SWF que se esté ejecutando en un navegador y un archivo SWF que se esté ejecutando en un proyector. Puede utilizar los objetos LocalConnection para enviar y recibir datos dentro de un mismo archivo SWF, aunque ésta no es una implementación estándar; todos los elementos de esta sección ilustran la comunicación entre archivos SWF diferentes.

Los principales métodos utilizados para enviar y recibir datos son `LocalConnection.send()` y `LocalConnection.connect()`. Como mínimo, el código que cree deberá implementar los siguientes comandos; observe que los comandos `LocalConnection.send()` y `LocalConnection.connect()`especifican el mismo nombre de conexión, `lc_name`:

```
// Code in the receiving SWF file
```

```
this.createTextField("result_txt", 1, 10, 10, 100, 22);
result_txt.border = true;
var receiving_lc:LocalConnection = new LocalConnection();
receiving_lc.methodToExecute = function(param1:Number, param2:Number) {
result_txt.text = param1+param2;
};
receiving_lc.connect("lc_name");

// Code in the sending SWF file
var sending_lc:LocalConnection = new LocalConnection();
sending_lc.send("lc_name", "methodToExecute", 5, 7);
```

La forma más simple de utilizar un objeto LocalConnection consiste en permitir la comunicación solamente entre objetos LocalConnection ubicados en el mismo dominio, ya que con ello evitará problemas de seguridad. No obstante, si necesita permitir la comunicación entre dominios, existen diversas formas de implementar medidas de seguridad. Para obtener más información, consulte la descripción del parámetro `connectionName` en `LocalConnection.send()` y las entradas `LocalConnection.allowDomain` y `LocalConnection.domain()`.

Disponibilidad: ActionScript 1.0; Flash Player 6

Resumen de propiedades

Propiedades heredadas de la clase Object

`constructor (propiedad Object.constructor), __proto__ (propiedad Object.__proto__), prototype (propiedad Object.prototype), __resolve (propiedad Object.__resolve)`

Resumen de eventos

Evento	Descripción
`allowDomain = function([sendingDomain:String]) {}`	Se invoca cuando receiving_lc recibe una solicitud de invocación de un método de un objeto LocalConnection emisor.
`allowInsecureDomain = function([sendingDomain:String]) {}`	Se invoca cuando receiving_lc, que se encuentra en un archivo SWF alojado en un dominio que utiliza un protocolo seguro (HTTPS), recibe una solicitud de invocación de un método de un objeto LocalConnection emisor que se encuentra en un archivo SWF alojado en un protocolo no seguro.
`onStatus = function(infoObject:Object) {}`	Se invoca después de que un objeto LocalConnection emisor intente enviar un comando a un objeto LocalConnection receptor.

Resumen de constructores

Firma	Descripción
`LocalConnection()`	Crea un objeto LocalConnection.

Resumen de métodos

Modificadores	Firma	Descripción
	`close() : Void`	Cierra (desconecta) un objeto LocalConnection.
	`connect(connectionName:String) : Boolean`	Prepara un objeto LocalConnection para recibir comandos de un comando `LocalConnection.send()` (denominado *objeto LocalConnection emisor*).
	`domain() : String`	Devuelve una cadena que representa el dominio de la ubicación del archivo SWF actual.
	`send(connectionName:String, methodName:String, [args:Object]) : Boolean`	Invoca el método denominado `method` en una conexión abierta con el comando `LocalConnection.connect( connectionName )`(el objeto LocalConnection receptor).

Métodos heredados de la clase Object

addProperty (método Object.addProperty), hasOwnProperty (método Object.hasOwnProperty), isPropertyEnumerable (método Object.isPropertyEnumerable), isPrototypeOf (método Object.isPrototypeOf), registerClass (método Object.registerClass), toString (método Object.toString), unwatch (método Object.unwatch), valueOf (método Object.valueOf), watch (método Object.watch)

allowDomain (controlador LocalConnection.allowDomain)

```
allowDomain = function([sendingDomain:String]) {}
```

Se invoca cuando receiving_lc recibe una solicitud de invocación de un método de un objeto LocalConnection emisor. Flash espera que el código que implemente en este controlador devuelva un valor booleano true o false. Si el controlador no devuelve true, la solicitud del objeto emisor no se tiene en cuenta y el método no se invoca.

Cuando este controlador de eventos no está presente, Flash Player aplica una política de seguridad predeterminada que equivale al siguiente código:

```
my_lc.allowDomain = function (sendingDomain)
{
  return (sendingDomain == this.domain());
}
```

Utilice `LocalConnection.allowDomain` para permitir de manera explícita que los objetos LocalConnection de dominios concretos, o de cualquier dominio, ejecuten métodos del objeto LocalConnection receptor. Si no declara el parámetro sendingDomain, probablemente desee que se acepten los comandos de cualquier dominio, por lo que el código del controlador simplemente devolverá true. Si declara sendingDomain, probablemente desee comparar el valor de sendingDomain con los dominios de los que desea aceptar comandos. En los siguientes ejemplos se muestran ambas implementaciones.

En los archivos editados para Flash Player 6, el parámetro `sendingDomain` contiene el superdominio que origina la llamada. En los archivos editados para Flash Player 7, el parámetro `sendingDomain` contiene el dominio exacto que origina la llamada. En este último caso, para permitir el acceso de archivos SWF alojados en www.domain.com o store.domain.com, deberá permitir explícitamente el acceso desde ambos dominios.

```
// For Flash Player 6
receiving_lc.allowDomain = function(sendingDomain) {
  return(sendingDomain=="domain.com");
}
// For Flash Player 7 or later
receiving_lc.allowDomain = function(sendingDomain) {
  return(sendingDomain=="www.domain.com" ||
    sendingDomain=="store.domain.com");
}
```

Asimismo, en el caso de archivos editados para Flash Player 7 o posterior, no podrá utilizar este método para permitir que los archivos SWF alojados mediante un protocolo seguro (HTTPS) permitan el acceso desde archivos SWF alojados en protocolos no seguros; en este caso deberá utilizar el controlador de eventos `LocalConnection.allowInsecureDomain`.

Es posible que, en ocasiones, se encuentre con la siguiente situación. Supongamos que carga un archivo SWF secundario de un dominio diferente. Usted desea implementar este método de manera que el archivo SWF secundario pueda realizar llamadas LocalConnection al archivo SWF principal, pero no conoce el dominio final del que procederá el archivo SWF secundario. Esto puede ocurrir, por ejemplo, cuando se utilizan redirecciones de reparto de carga o servidores de terceros.

En esta situación, puede utilizar la propiedad `MovieClip._url` en su implementación de este método. Por ejemplo, si carga un archivo SWF en my_mc, podrá implementar este método comprobando si el argumento de dominio coincide con el dominio de `my_mc._url`. (Debe analizar el dominio fuera de la URL completa contenida en `my_mc._url`.)

Si lo hace, asegúrese de que espera a que se cargue el archivo SWF de `my_mc`, ya que la propiedad _url no dispondrá de su valor final correcto hasta que el archivo esté completamente cargado. La mejor forma de determinar cuándo termina de cargarse un archivo SWF secundario es a través de `MovieClipLoader.onLoadComplete`.

También puede darse la situación contraria: Puede crear un archivo SWF secundario que quiera aceptar llamadas LocalConnection de su archivo principal pero que no conozca el dominio de su archivo principal. En este caso, implemente este método comprobando si el argumento de dominio coincide con el dominio de _parent._url. Una vez más, deberá analizar el dominio fuera de la URL completa de _parent._url. En este caso, no tendrá que esperar a que se cargue el archivo SWF principal; el archivo principal ya estará cargado en el momento en que se cargue el archivo secundario.

Disponibilidad: ActionScript 1.0; Flash Player 7

Parámetros

`sendingDomain``:String` [opcional] - Una cadena que especifica el dominio del archivo SWF que contiene el objeto LocalConnection de origen.

Ejemplo

El ejemplo siguiente muestra como un objeto LocalConnection en un archivo SWF receptor puede permitir que los archivos SWF de cualquier dominio puedan invocar sus métodos. Compare este código con el ejemplo de `LocalConnection.connect()`, en el que sólo los archivos SWF del mismo dominio pueden invocar el método `trace()` en el archivo SWF receptor. Para ver una descripción del uso del subrayado (_) en el nombre de conexión, consulte `LocalConnection.send()`.

```
this.createTextField("welcome_txt", this.getNextHighestDepth(), 10, 10,
  100, 20);
var my_lc:LocalConnection = new LocalConnection();
my_lc.allowDomain = function(sendingDomain:String) {
  domain_txt.text = sendingDomain;
  return true;
};
my_lc.allowInsecureDomain = function(sendingDomain:String) {
  return (sendingDomain == this.domain());
}
```

```
my_lc.sayHello = function(name:String) {
  welcome_txt.text = "Hello, "+name;
};
my_lc.connect("_mylc");
```

El ejemplo siguiente envía una cadena al archivo SWF anterior y muestra un mensaje de estado que indica si la conexión local ha podido conectar con el archivo. Para mostrar contenido se utiliza un componente TextInput denominado `name_ti`, una instancia TextArea denominada `status_ta` y una instancia Button denominada `send_button`.

```
var sending_lc:LocalConnection;
var sendListener:Object = new Object();
sendListener.click = function(evt:Object) {
  sending_lc = new LocalConnection();
  sending_lc.onStatus = function(infoObject:Object) {
  switch (infoObject.level) {
  case 'status' :
    status_ta.text = "LocalConnection connected successfully.";
    break;
  case 'error' :
    status_ta.text = "LocalConnection encountered an error.";
    break;
  }
  };
  sending_lc.send("_mylc", "sayHello", name_ti.text);
};
send_button.addEventListener("click", sendListener);
```

Si el archivo SWF incluye un componente de la versión 2, utilice la clase DepthManager de los componentes de la versión 2 en lugar del método `MovieClip.getNextHighestDepth()` utilizado en el ejemplo anterior.

En el ejemplo siguiente, el archivo SWF receptor que reside en `thisDomain.com`, sólo acepta comandos de archivos SWF ubicados en `thisDomain.com` o en `thatDomain.com`:

```
var aLocalConn:LocalConnection = new LocalConnection();
aLocalConn.Trace = function(aString) {
  aTextField += aString+newline;
};
aLocalConn.allowDomain = function(sendingDomain) {
  return (sendingDomain == this.domain() || sendingDomain ==
  "www.macromedia.com");
};
aLocalConn.connect("_mylc");
```

Si se publica para Flash Player 7 o posterior, debe coincidir el dominio exacto. Esto significa que el ejemplo fallará si los archivos SWF se encuentran en www.thatDomain.com pero funcionarán si estos archivos se encuentran en thatDomain.com.

Véase también

`connect (método LocalConnection.connect)`, `domain (método LocalConnection.domain)`, `send (método LocalConnection.send)`, `_url (propiedad MovieClip._url)`, `onLoadComplete (detector de eventos MovieClipLoader.onLoadComplete)`, `Propiedad _parent`

allowInsecureDomain (controlador LocalConnection.allowInsecureDomain)

```
allowInsecureDomain = function([sendingDomain:String]) {}
```

Se invoca cuando receiving_lc, que se encuentra en un archivo SWF alojado en un dominio que utiliza un protocolo seguro (HTTPS), recibe una solicitud de invocación de un método de un objeto LocalConnection emisor que se encuentra en un archivo SWF alojado en un protocolo no seguro. Flash espera que el código que implemente en este controlador devuelva un valor booleano `true` o `false`. Si el controlador no devuelve true, la solicitud del objeto emisor no se tiene en cuenta y el método no se invoca.

De manera predeterminada, sólo los archivos SWF alojados mediante un protocolo HTTPS pueden acceder a otros archivos SWF alojados mediante el protocolo HTTPS. Esta implementación mantiene la integridad que ofrece el protocolo HTTPS.

No es recomendable utilizar este método para sustituir el comportamiento predeterminado, ya que pone en riesgo la seguridad de HTTPS. Sin embargo, es posible que tenga que hacerlo, por ejemplo, si necesita permitir el acceso a archivos HTTPS publicados para Flash Player 7 o posterior desde archivos HTTP publicados para Flash Player 6.

Un archivo SWF publicado para Flash Player 6 puede utilizar el controlador de eventos `LocalConnection.allowDomain` para permitir que HTTP acceda a HTTPS. Sin embargo, dado que la seguridad se implementa de manera distinta en Flash Player 7, deberá utilizar el método `LocalConnection.allowInsecureDomain()` para permitir dicho acceso en archivos SWF publicados para Flash Player 7 o posterior.

Disponibilidad: ActionScript 1.0; Flash Player 7

Parámetros

`sendingDomain``:String` [opcional] - Una cadena que especifica el dominio del archivo SWF que contiene el objeto LocalConnection de origen.

Ejemplo

El ejemplo siguiente permite conexiones desde el dominio actual o desde www.macromedia.com, o permite conexiones no seguras sólo desde el dominio actual.

```
this.createTextField("welcome_txt", this.getNextHighestDepth(), 10, 10,
  100, 20);
var my_lc:LocalConnection = new LocalConnection();
my_lc.allowDomain = function(sendingDomain:String) {
  domain_txt.text = sendingDomain;
  return (sendingDomain == this.domain() || sendingDomain ==
  "www.macromedia.com");
};
my_lc.allowInsecureDomain = function(sendingDomain:String) {
  return (sendingDomain == this.domain());
}
my_lc.sayHello = function(name:String) {
  welcome_txt.text = "Hello, "+name;
};
my_lc.connect("lc_name");
```

Si el archivo SWF incluye un componente de la versión 2, utilice la clase DepthManager de los componentes de la versión 2 en lugar del método `MovieClip.getNextHighestDepth()` utilizado en este ejemplo.

Véase también

`allowDomain (controlador LocalConnection.allowDomain)`, `connect (método LocalConnection.connect)`

close (método LocalConnection.close)

```
public close() : Void
```

Cierra (desconecta) un objeto LocalConnection. Envíe este comando cuando ya no desee que el objeto acepte comandos (por ejemplo, cuando desee enviar un comando `LocalConnection.connect()` utilizando el mismo parámetro `connectionName` en otro archivo SWF.

Disponibilidad: ActionScript 1.0; Flash Player 6

Ejemplo

El ejemplo siguiente cierra una conexión denominada `receiving_lc` cuando hace clic en una instancia de componente Button denominada `close_button`:

```
this.createTextField("welcome_txt", this.getNextHighestDepth(), 10, 10,
  100, 22);
this.createTextField("status_txt", this.getNextHighestDepth(), 10, 42,
  100,44);
```

```
var receiving_lc:LocalConnection = new LocalConnection();
receiving_lc.sayHello = function(name:String) {
  welcome_txt.text = "Hello, "+name;
};
receiving_lc.connect("lc_name");
var closeListener:Object = new Object();
closeListener.click = function(evt:Object) {
  receiving_lc.close();
  status_txt.text = "connection closed";
};
close_button.addEventListener("click", closeListener);
```

El método `MovieClip.getNextHighestDepth()` utilizado en este ejemplo necesita Flash Player 7 o una versión posterior. Si el archivo SWF incluye un componente de la versión 2, utilice la clase DepthManager de los componentes de la versión 2 en lugar del método `MovieClip.getNextHighestDepth()`.

Véase también

`connect (método LocalConnection.connect)`

connect (método LocalConnection.connect)

```
public connect(connectionName:String) : Boolean
```

Prepara un objeto LocalConnection para recibir comandos de un comando `LocalConnection.send()` (denominado *objeto LocalConnection emisor*). El objeto utilizado con este comando se conoce como *objeto LocalConnection receptor*. Los objetos receptor y emisor deben estar en ejecución en el mismo equipo cliente.

Asegúrese de que define los métodos asociados a *`receiving_lc`* antes de llamar a este método, como se muestra en todos los ejemplos de esta sección.

De forma predeterminada, Flash Player resuelve `connectionName` como un valor "*`superdomain`*`:connectionName`", donde *`superdomain`* es el superdominio del archivo SWF que contiene el comando `LocalConnection.connect()`. Por ejemplo, si el archivo SWF que contiene el objeto LocalConnection receptor está ubicado en www.someDomain.com, `connectionName` se resuelve como `"someDomain.com:connectionName"`". (Si un archivo SWF está situado en un equipo cliente, el valor asignado a `superdomain` es `"localhost"`.)

También de manera predeterminada, Flash Player permite que el objeto LocalConnection receptor acepte solamente comandos de objetos LocalConnection emisores cuyo nombre de conexión también se resuelva en el valor "*`superdomain`*`:connectionName`". De esta forma, Flash facilita la comunicación entre archivos SWF situados en el mismo dominio.

Sí sólo va a implementar la comunicación entre archivos SWF del mismo dominio, especifique una cadena para `connectionName` que no comience por un carácter de subrayado (_) y que no especifique un nombre de dominio (por ejemplo, `"myDomain:connectionName"`). Utilice la misma cadena en el comando `LocalConnection.connect( connectionName )`.

Si va a implementar comunicación entre archivos SWF de diferentes dominios, la especificación de una cadena para *`connectionName`* que comience por un carácter de subrayado (_) permitirá que el archivo SWF que tiene el objeto LocalConnection receptor presente una mayor portabilidad entre dominios. Estos son los dos casos posibles:

- Si la cadena para *`connectionName`* no comienza por un carácter de subrayado (_), Flash Player añade un prefijo con el superdominio y dos puntos (por ejemplo, `"myDomain:connectionName"`). Aunque esto garantiza que la conexión no entre en conflicto con conexiones de otros dominios que tengan el mismo nombre, los objetos LocalConnection emisores deben especificar este superdominio (por ejemplo, `"myDomain:connectionName"`). Si el archivo SWF que contiene el objeto LocalConnection receptor se traslada a otro dominio, el reproductor cambiará el prefijo para reflejar el nuevo superdominio (por ejemplo, `"anotherDomain:connectionName"`). Sería necesario editar manualmente todos los objetos LocalConnection emisores para que apuntaran al nuevo superdominio.
- Si la cadena de *`connectionName`* comienza por un carácter de subrayado (por ejemplo, `"_connectionName"`), Flash Player no añadirá ningún prefijo a la cadena. Esto significa que los objetos LocalConnection receptores y emisores utilizarán idénticas cadenas para `connectionName`. Si el objeto receptor utiliza `LocalConnection.allowDomain` para especificar que se acepten las conexiones de cualquier dominio, el SWF que contiene el objeto LocalConnection receptor podrá trasladarse a otro dominio sin modificar ningún objeto LocalConnection emisor.

Para obtener más información, consulte la descripción de `connectionName` en `LocalConnection.send()` y las entradas `LocalConnection.allowDomain` y `LocalConnection.domain()`.

Nota: Los dos puntos se utilizan como caracteres especiales que separan el superdominio de la cadena `connectionName`. La cadena de `connectionName` no puede contener dos puntos.

Disponibilidad: ActionScript 1.0; Flash Player 6

Parámetros

`connectionName``:String` - Una cadena que se corresponde con el nombre de conexión especificado en el comando `LocalConnection.send()` que desea comunicarse con *`receiving_lc`* .

Valor devuelto

`Boolean` - Valor booleano: `true` si ningún otro proceso en ejecución en el mismo equipo cliente ha emitido aún este comando empleando el mismo valor para el parámetro *`connectionName`* ; `false` en caso contrario.

Ejemplo

El ejemplo siguiente muestra cómo un archivo SWF de un dominio determinado puede invocar a un método con el nombre de `printOut` en un archivo SWF receptor en el mismo dominio.

Primero, debe crear un archivo SWF con el código siguiente:

```
this.createTextField("tf", this.getNextHighestDepth(), 10, 10, 300, 100);
var aLocalConnection:LocalConnection = new LocalConnection();
aLocalConnection.connect("demoConnection");
aLocalConnection.printOut = function(aString:String):Void{
  tf.text += aString;
}
```

Después, debe crear otro con este código:

```
var sending_lc:LocalConnection = new LocalConnection();
sending_lc.send("demoConnection", "printOut", "This is a message from file
  B. Hello.");
```

Para probar este ejemplo, ejecute el primer archivo SWF y después ejecute el segundo.

A continuación, se muestra otro ejemplo: SWF 1 contiene el código siguiente que crea un nuevo objeto Sound que reproduce un archivo MP3 en tiempo de ejecución. Aparece un objeto ProgressBar denominado `playback_pb` para mostrar la reproducción del archivo MP3. Una instancia de componente Label denominada `song_lbl` muestra el nombre del archivo MP3. Para controlar la reproducción se utilizarán botones de diferentes archivos SWF utilizando un objeto LocalConnection.

```
var playback_pb:mx.controls.ProgressBar;
var my_sound:Sound;
playback_pb.setStyle("themeColor", "haloBlue");
this.createEmptyMovieClip("timer_mc", this.getNextHighestDepth());
var receiving_lc:LocalConnection = new LocalConnection();
receiving_lc.playMP3 = function(mp3Path:String, mp3Name:String) {
  song_lbl.text = mp3Name;
  playback_pb.indeterminate = true;
  my_sound = new Sound();
  my_sound.onLoad = function(success:Boolean) {
  playback_pb.indeterminate = false;
  };
  my_sound.onSoundComplete = function() {
  delete timer_mc.onEnterFrame;
  };
```

```
  timer_mc.onEnterFrame = function() {
  playback_pb.setProgress(my_sound.position, my_sound.duration);
  };
  my_sound.loadSound(mp3Path, true);
};
receiving_lc.connect("lc_name");
```

SWF 2 contiene un botón denominado `play_btn`. Al hacer clic en este botón, se conecta a SWF 1 y pasa dos variables. La primera variable contiene el archivo MP3 que se reproducirá y la segunda variable es el nombre de archivo que se mostrará en la instancia de componente Label en SWF 1.

```
play_btn.onRelease = function() {
  var sending_lc:LocalConnection = new LocalConnection();
  sending_lc.send("lc_name", "playMP3", "song1.mp3", "Album - 01 - Song");
};
```

SWF 3 contiene un botón denominado `play_btn`. Al hacer clic en este botón, se conecta a SWF 1 y pasa dos variables. La primera variable contiene el archivo MP3 que se reproducirá y la segunda variable es el nombre de archivo que se mostrará en la instancia de componente Label en SWF 1.

```
play_btn.onRelease = function() {
  var sending_lc:LocalConnection = new LocalConnection();
  sending_lc.send("lc_name", "playMP3", "song2.mp3", "Album - 02 - Another
  Song");
};
```

El método `MovieClip.getNextHighestDepth()` utilizado en estos ejemplos necesita Flash Player 7 o una versión posterior. Si el archivo SWF incluye un componente de la versión 2, utilice la clase DepthManager de los componentes de la versión 2 en lugar del método `MovieClip.getNextHighestDepth()`.

Véase también

`send (método LocalConnection.send)`, `allowDomain (controlador LocalConnection.allowDomain)`, `domain (método LocalConnection.domain)`

domain (método LocalConnection.domain)

```
public domain() : String
```

Devuelve una cadena que representa el dominio de la ubicación del archivo SWF actual.

En archivos SWF publicados para Flash Player 6, la cadena devuelta es el superdominio del archivo SWF actual. Por ejemplo, si el archivo SWF está situado en www.macromedia.com, este comando devolverá `"macromedia.com"`.

En archivos SWF publicados para Flash Player 7 o posterior, la cadena devuelta es el dominio exacto del archivo SWF actual. Por ejemplo, si el archivo SWF está situado en www.macromedia.com, este comando devolverá `"www.macromedia.com"`.

Si el archivo SWF actual es un archivo local que reside en el equipo cliente, este comando devolverá `"localhost"`.

Este comando se utiliza habitualmente incluyendo el nombre del dominio del objeto LocalConnection emisor como parámetro del método que tiene previsto invocar en el objeto LocalConnection receptor o con `LocalConnection.allowDomain` para aceptar comandos de un dominio determinado. Si desea permitir la comunicación sólo entre objetos LocalConnection que se encuentren en el mismo dominio, probablemente no necesite utilizar este comando.

Disponibilidad: ActionScript 1.0; Flash Player 6

Valor devuelto

`String` - Una cadena que representa al dominio de la ubicación del archivo SWF actual; para más información, consulte la sección Descripción.

Ejemplo

En el ejemplo siguiente, un archivo SWF receptor sólo acepta comandos de archivos SWF ubicados en el mismo dominio o en macromedia.com:

```
// If both the sending and receiving SWF files are Flash Player 6,
// then use the superdomain
var my_lc:LocalConnection = new LocalConnection();
my_lc.allowDomain = function(sendingDomain):String{
  return (sendingDomain==this.domain() || sendingDomain=="macromedia.com");
}

// If either the sending or receiving SWF file is Flash Player 7 or later,
// then use the exact domain. In this case, commands from a SWF file posted
// at www.macromedia.com will be accepted, but those from one posted at
// a different subdomain, e.g. livedocs.macromedia.com, will not.
var my_lc:LocalConnection = new LocalConnection();
my_lc.allowDomain = function(sendingDomain):String{
  return (sendingDomain==this.domain() ||
  sendingDomain=="www.macromedia.com");
}
```

En el ejemplo siguiente, un archivo SWF emisor ubicado en www.yourdomain.com invoca a un método de un archivo SWF receptor ubicado en www.mydomain.com. El archivo SWF emisor incluye su nombre de dominio como un parámetro para el método que invoca, de este modo el archivo SWF receptor puede devolver un valor a un objeto LocalConnection en el dominio correcto. El archivo SWF emisor también especifica que sólo aceptará comandos de archivos SWF de mydomain.com.

Los números de líneas se incluyen para referencia. La secuencia de eventos se describe en la siguiente lista:

- El archivo SWF receptor se prepara para recibir comandos en una conexión denominada `"sum"` (línea 11). Flash Player resuelve el nombre de esta conexión como `"mydomain.com:sum"` (consulte `LocalConnection.connect()`).
- El archivo SWF emisor se prepara para recibir una respuesta en el objeto LocalConnection denominado `"result"` (línea 67). También especifica que sólo aceptará comandos de archivos SWF de mydomain.com (líneas 51 a 53).
- El archivo SWF emisor invoca al método `aSum` de una conexión denominada `"mydomain.com:sum"` (línea 68) y pasa los parámetros siguientes: su superdominio, el nombre de la conexión que recibirá la respuesta (`"result"`) y los valores que utilizará `aSum` (123 y 456).
- El método `aSum` (línea 6) se invoca con los valores siguientes: sender = `"mydomain.com:result"`, `replyMethod` = `"aResult"`, `n1` = 123 y `n2` = 456. A continuación, ejecuta la siguiente línea de código:

```
this.send("mydomain.com:result", "aResult", (123 + 456));
```

- El método `aResult` (línea 54) muestra el valor que devuelve `aSum` (579).

```
// The receiving SWF at http://www.mydomain.com/folder/movie.swf
// contains the following code

1 var aLocalConnection:LocalConnection = new LocalConnection();
2 aLocalConnection.allowDomain = function()
3 {
  // Allow connections from any domain
4 return true;
5 }
6 aLocalConnection.aSum = function(sender, replyMethod, n1, n2)
7 {
8 this.send(sender, replyMethod, (n1 + n2));
9 }
10
11 aLocalConnection.connect("sum");

// The sending SWF at http://www.yourdomain.com/folder/movie.swf
// contains the following code
```

```
50 var lc:LocalConnection = new LocalConnection();
51 lc.allowDomain = function(aDomain) {
     // Allow connections only from mydomain.com
52 return (aDomain == "mydomain.com");
53 }
54 lc.aResult = function(aParam) {
55 trace("The sum is " + aParam);
56 }
   // determine our domain and see if we need to truncate it
57 var channelDomain:String = lc.domain();
58 if (getVersion() >= 7 && this.getSWFVersion() >= 7)
59 {
     // split domain name into elements
60 var domainArray:Array = channelDomain.split(".");

     // if more than two elements are found,
     // chop off first element to create superdomain
61 if (domainArray.length > 2)
62 {
63 domainArray.shift();
64 channelDomain = domainArray.join(".");
65 }
66 }

67 lc.connect("result");
68 lc.send("mydomain.com:sum", "aSum", channelDomain + ':' + "result",
"aResult", 123, 456);
```

Véase también

allowDomain (controlador LocalConnection.allowDomain), connect (método LocalConnection.connect)

Constructor LocalConnection

```
public LocalConnection()
```

Crea un objeto LocalConnection.

Disponibilidad: ActionScript 1.0; Flash Player 6

Ejemplo

El ejemplo siguiente muestra cómo los archivos SWF receptor y emisor crean objetos LocalConnnection. Los dos archivos SWF pueden utilizar el mismo nombre o nombres diferentes para sus correspondientes objetos LocalConnection. En este ejemplo utilizan nombres diferentes.

```
// Code in the receiving SWF file
this.createTextField("result_txt", 1, 10, 10, 100, 22);
result_txt.border = true;
var receiving_lc:LocalConnection = new LocalConnection();
receiving_lc.methodToExecute = function(param1:Number, param2:Number) {
  result_txt.text = param1+param2;
};
receiving_lc.connect("lc_name");
```

El siguiente archivo SWF envía la petición al primer archivo SWF.

```
// Code in the sending SWF file
var sending_lc:LocalConnection = new LocalConnection();
sending_lc.send("lc_name", "methodToExecute", 5, 7);
```

Véase también

connect (método LocalConnection.connect), send (método LocalConnection.send)

onStatus (controlador LocalConnection.onStatus)

```
onStatus = function(infoObject:Object) {}
```

Se invoca después de que un objeto LocalConnection emisor intente enviar un comando a un objeto LocalConnection receptor. Si desea responder a este controlador de eventos, deberá crear una función que procese el objeto de información enviado por el objeto LocalConnection.

Si el objeto de información devuelto por este controlador de eventos contiene un valor de nivel de estado, ello indicará que Flash ha enviado correctamente el comando a un objeto LocalConnection receptor. Esto no significa que Flash haya invocado correctamente el método especificado del objeto LocalConnection receptor; simplemente indica que Flash ha logrado enviar el comando. Por ejemplo, el método no se invoca si el objeto LocalConnection receptor no permite conexiones desde el dominio emisor o si el método no existe. La única forma de saber si el método se ha invocado es haciendo que el objeto receptor envíe una respuesta al objeto emisor.

Si el objeto de información devuelto por este controlador de eventos contiene un valor de nivel de error, ello indicará que Flash no ha podido enviar el comando a un objeto LocalConnection receptor, probablemente porque no haya ningún objeto LocalConnection receptor que coincida con el nombre especificado en el comando `sending_lc.send()` que invocó este controlador.

Además de este controlador `onStatus`, Flash también proporciona una "super" función denominada `System.onStatus`. Si se llama a `onStatus` para un objeto concreto y no hay ninguna función asignada para responder, Flash procesa una función asignada a `System.onStatus`, si existe.

Normalmente sólo implementará este controlador para responder a situaciones de error, como se muestra en el siguiente ejemplo.

Disponibilidad: ActionScript 1.0; Flash Player 6

Parámetros

`infoObject``:Object` - Un parámetro definido de acuerdo con el mensaje de estado. Para obtener detalles sobre este parámetro, consulte la sección Descripción.

Ejemplo

El ejemplo siguiente muestra un mensaje de estado indicando si el archivo SWF se conecta a otro objeto de conexión local denominado `lc_name`. Para mostrar contenido se utiliza un componente TextInput denominado `name_ti`, una instancia TextArea denominada `status_ta` y una instancia Button denominada `send_button`.

```
var sending_lc:LocalConnection;
var sendListener:Object = new Object();
sendListener.click = function(evt:Object) {
  sending_lc = new LocalConnection();
  sending_lc.onStatus = function(infoObject:Object) {
    switch (infoObject.level) {
    case 'status' :
      status_ta.text = "LocalConnection connected successfully.";
        break;
    case 'error' :
      status_ta.text = "LocalConnection encountered an error.";
      break;
    }
  };
  sending_lc.send("lc_name", "sayHello", name_ti.text);
};
send_button.addEventListener("click", sendListener);
```

Véase también

send (método LocalConnection.send), onStatus (controlador System.onStatus)

send (método LocalConnection.send)

```
public send(connectionName:String, methodName:String, [args:Object]) :
  Boolean
```

Invoca el método denominado `method` en una conexión abierta con el comando `LocalConnection.connect( connectionName )`(el objeto LocalConnection receptor). El objeto utilizado con este comando se conoce como objeto LocalConnection emisor. Los archivos SWF que contienen los objetos emisor y receptor deben estar en ejecución en el mismo equipo cliente.

Existe un límite de 40 kilobytes para la cantidad de datos que puede pasar en forma de parámetros a este comando. Si el comando devuelve `false` pero la sintaxis es correcta, intente dividir las peticiones de `LocalConnection.send()` en varios comandos, cada uno con menos de 40 Kb de datos.

Como se describe en la entrada `LocalConnection.connect()`, Flash añade el superdominio actual a `connectionName` de manera predeterminada. Si está implementando la comunicación entre diferentes dominios, deberá definir `connectionName` en los objetos LocalConnection emisor y receptor de manera que Flash no añada el superdominio actual a `connectionName`. Puede hacerlo de una de las dos formas siguientes:

- Utilice un carácter de subrayado (_) al comienzo de `connectionName` en los objetos LocalConnection emisor y receptor. En el archivo SWF que contiene el objeto receptor, utilice `LocalConnection.allowDomain` para especificar que se acepten las conexiones procedentes de cualquier dominio. Esta implementación le permite almacenar los archivos SWF emisor y receptor en cualquier dominio.
- Incluya el superdominio en `connectionName` en el objeto LocalConnection emisor (por ejemplo, `myDomain.com:myConnectionName`). En el objeto receptor, utilice `LocalConnection.allowDomain` para especificar que se acepten las conexiones desde el superdominio especificado (en este caso, myDomain.com) o que se acepten las conexiones desde cualquier dominio.

Nota: No puede especificar un superdominio en `connectionName` en el objeto LocalConnection receptor (sólo puede hacerlo en el objeto LocalConnection emisor).

Cuando utilice este método puede usar el modelo de seguridad de Flash Player. De forma predeterminada, un objeto LocalConnection está asociado con el entorno limitado del archivo SWF que lo creó y no se admiten llamadas de varios dominios a objetos LocalConnection a menos que se haya llamado al método `LocalConnection.allowDomain()`.

Para más información, consulte lo siguiente:

- Capítulo 17, "Aspectos básicos de la seguridad," en *Aprendizaje de ActionScript 2.0 en Flash*
- El documento técnico sobre la seguridad de Flash Player 8 en http://www.macromedia.com/go/fp8_security
- El documento técnico sobre interfaces API relativas a seguridad de Flash Player 8 en http://www.macromedia.com/go/fp8_security_apis

Disponibilidad: ActionScript 1.0; Flash Player 6

Parámetros

connectionName:String - Una cadena que se corresponde con el nombre de conexión especificado en el comando `LocalConnection.send()` que desea comunicarse con *sending_lc* .

methodName:String - Una cadena que especifica el nombre del método que se invoca en el objeto receptor LocalConnection. Los siguientes nombres de método provocan que el comando falle: `send`, `connect`, `close`, `domain`, `onStatus` y `allowDomain`.

args:Object [opcional] - Argumentos que se deben pasar al método especificado.

Valor devuelto

`Boolean` - Valor booleano: `true` si Flash puede realizar la solicitud; `false` en caso contrario.

Nota: Aunque se devuelva el valor `true`, ello no indica necesariamente que Flash haya conectado correctamente con un objeto LocalConnection receptor; sólo significa que el comando es sintácticamente correcto. Para determinar si la conexión ha sido correcta, consulte `LocalConnection.onStatus`.

Ejemplo

Para ver un ejemplo de la comunicación entre objetos LocalConnection ubicados en el mismo dominio, consulte `LocalConnection.connect()`. Para ver un ejemplo de la comunicación entre objetos LocalConnection ubicados en cualquier dominio, consulte `LocalConnection.allowDomain`. Para ver un ejemplo de la comunicación entre objetos LocalConnection ubicados en dominios específicos, consulte `LocalConnection.allowDomain` y `LocalConnection.domain()`.

Véase también

`allowDomain (controlador LocalConnection.allowDomain)`, `connect (método LocalConnection.connect)`, `domain (método LocalConnection.domain)`, `onStatus (controlador LocalConnection.onStatus)`

Locale (mx.lang.Locale)

```
Object
  |
  +-mx.lang.Locale
```

```
public class Locale
extends Object
```

La clase mx.lang.Locale permite controlar cómo aparece el texto en varios idiomas en un archivo SWF. El panel Cadenas de Flash permite utilizar ID de cadenas en lugar de literales de cadenas en campos de texto dinámicos. Esto permite crear un archivo SWF que muestre el texto cargado desde un archivo XML de un idioma específico. El archivo XML debe tener el formato XLIFF (XML Localization Interchange File Format). Existen tres formas de mostrar las cadenas de un idioma específico de los archivos XLIFF:

- `"automatically at runtime"`: Flash Player sustituye los ID con cadenas del archivo XML que coincidan con el código de idioma de sistema predeterminado devuelto por System.capabilities.language.
- `"manually using stage language"`: durante la compilación se sustituyen los ID por cadenas y Flash Player no los puede modificar.
- `"via ActionScript at runtime"`: la sustitución de ID se controla mediante ActionScript durante la ejecución. Esta opción permite controlar la sincronización y el idioma de la sustitución de ID de cadena.

Puede utilizar las propiedades y los métodos de esta clase cuando quiera sustituir los ID de cadena "mediante ActionScript en tiempo de ejecución".

Toda las propiedades y métodos disponibles son estáticos, lo que significa que es posible acceder a los mismos a través de la propia clase mx.lang.Locale en lugar de hacerlo a través de una instancia de la clase.

Nota: La clase Locale es distinta de otras clases de Referencia del lenguaje ActionScript de Flash, ya que no forma parte de Flash Player. Esta clase se instala automáticamente en la ruta de clases del entorno de edición de Flash, por lo que se compila automáticamente en sus archivos SWF. Si se utiliza la clase Locale, aumenta ligeramente el tamaño del archivo del SWF, ya que se compila en él.

Disponibilidad: ActionScript 2.0; Flash Player 7

Resumen de propiedades

Modificadores	Propiedad	Descripción
`static`	`autoReplace:Boolean`	Determina si las cadenas se sustituyen automáticamente después de cargar el archivo XML.
`static`	`languageCodeArray:Array` [read-only]	Una matriz con los códigos de los idiomas que se han especificado o cargado en el archivo FLA.
`static`	`stringIDArray:Array` [read-only]	Una matriz que contiene todos los ID de cadena del archivo FLA.

Propiedades heredadas de la clase Object

`constructor (propiedad Object.constructor)`, `__proto__ (propiedad Object.__proto__)`, `prototype (propiedad Object.prototype)`, `__resolve (propiedad Object.__resolve)`

Resumen de métodos

Modificadores	Firma	Descripción
`static`	`addDelayedInstance(instance:Object, stringID:String) : Void`	Agrega el par {instancia, ID de cadena} en la matriz interna para utilizarlo en otro momento.
`static`	`addXMLPath(langCode:String, path:String) : Void`	Agrega el par {languageCode y languagePath} en la matriz interna para utilizarlo en otro momento.
`static`	`checkXMLStatus() : Boolean`	Devuelve `true` si se ha cargado del archivo XML; `false` en caso contrario.
`static`	`getDefaultLang() : String`	El código de idioma predeterminado definido en el cuadro de diálogo del panel Cadenas o al llamar al método `setDefaultLang()`.
`static`	`initialize() : Void`	Determina automáticamente el idioma que se va a utilizar y carga el archivo de idioma XML.
`static`	`loadLanguageXML(xmlLanguageCode:String, customXmlCompleteCallback:Function) : Void`	Carga el archivo de idioma XML especificado.
`static`	`loadString(id:String) : String`	Devuelve el valor de cadena asociado con el ID de cadena indicado en el idioma actual.

Modificadores	Firma	Descripción
static	loadStringEx(stringID:String, languageCode:String) : String	Devuelve el valor de cadena asociado con el ID de cadena indicado y el código de idioma.
static	setDefaultLang(langCode:String) : Void	Define el código de idioma predeterminado.
static	setLoadCallback(loadCallback:Function) : Void	Define la función callback que se llamará después de cargar el archivo XML.
static	setString(stringID:String, languageCode:String, stringValue:String) : Void	Define el nuevo valor de cadena de un ID de cadena y código de idioma indicado.

Métodos heredados de la clase Object

addProperty (método Object.addProperty), hasOwnProperty (método Object.hasOwnProperty), isPropertyEnumerable (método Object.isPropertyEnumerable), isPrototypeOf (método Object.isPrototypeOf), registerClass (método Object.registerClass), toString (método Object.toString), unwatch (método Object.unwatch), valueOf (método Object.valueOf), watch (método Object.watch)

addDelayedInstance (método Locale.addDelayedInstance)

```
public static addDelayedInstance(instance:Object, stringID:String) : Void
```

Agrega el par {instancia, ID de cadena} en la matriz interna para utilizarlo en otro momento. Flash lo utiliza principalmente cuando el método de sustitución de cadenas es `"automatically at runtime"`.

Disponibilidad: ActionScript 2.0; Flash Player 7

Parámetros

instance:Object - Nombre de instancia del campo de texto que se va a llenar.

stringID:String - ID de cadena de idioma.

Ejemplo

El ejemplo siguiente utiliza la propiedad `autoReplace` y el método `addDelayedInstance()` para llenar un campo de texto en el escenario con la cadena `IDS_GREETING` del archivo de idioma XML inglés.

```
import mx.lang.Locale;
greeting_txt.autoSize = "left";
Locale.autoReplace = true;
Locale.addDelayedInstance(greeting_txt, "IDS_GREETING");
Locale.loadLanguageXML("en");
```

addXMLPath (método Locale.addXMLPath)

```
public static addXMLPath(langCode:String, path:String) : Void
```

Agrega el par {languageCode y languagePath} en la matriz interna para utilizarlo en otro momento. Flash lo utiliza principalmente cuando el método de sustitución de cadenas es `"automatically at runtime"` o `"via ActionScript at runtime"`.

Disponibilidad: ActionScript 2.0; Flash Player 7

Parámetros

`langCode``:String` - El código de idioma.

`path``:String` - La ruta XML que se agregará.

Ejemplo

El ejemplo siguiente utiliza el método `setInterval()` para comprobar si el archivo de idioma XML se ha cargado correctamente.

```
import mx.lang.Locale;
Locale.setLoadCallback(localeCallback);
Locale.loadLanguageXML("en");
// create interval to check if language XML file is loaded
var locale_int:Number = setInterval(checkLocaleStatus, 10);
function checkLocaleStatus():Void {
  if (Locale.checkXMLStatus()) {
    clearInterval(locale_int);
    trace("clearing interval @ " + getTimer() + " ms");
  }
}
// callback function for Locale.setLoadCallback()
function localeCallback(success:Boolean):Void {
  greeting_txt.text = Locale.loadString("IDS_GREETING");
}
```

autoReplace (propiedad Locale.autoReplace)

```
public static autoReplace : Boolean
```

Determina si las cadenas se sustituyen automáticamente después de cargar el archivo XML. Si es `true`, el método de sustitución de texto es equivalente al del panel Cadenas `"automatically at runtime"`. Esto significa que Flash Player establecerá el idioma predeterminado del entorno host y mostrará automáticamente el texto en dicho idioma. Si es `false`, el método de sustitución de texto es equivalente al del panel Cadenas `"via ActionScript at runtime"`. Esto significa que el usuario es responsable de la carga del archivo XML correcto para mostrar el texto.

El valor predeterminado de esta propiedad refleja la configuración seleccionada para Reemplazar cadenas en el panel Cadenas: `true` para `"automatically at runtime"` (predeterminado) y `false` para "mediante ActionScript en tiempo de ejecución".

Disponibilidad: ActionScript 2.0; Flash Player 8

Ejemplo

El ejemplo siguiente utiliza la propiedad `Locale.autoReplace` para llenar el campo de texto `greeting_txt` que se ha creado de forma dinámica en el escenario con el contenido de la cadena `IDS_GREETING` del archivo XML inglés. En el panel Cadenas, haga clic en el botón Configuración para abrir el cuadro de diálogo Configuración. Puede agregar dos idiomas activos en el cuadro de diálogo Configuración: inglés (en) y francés (fr), definir el botón de opción de sustitución de cadenas como `"via ActionScript at runtime"` y hacer clic en Aceptar. Por último, introduzca un ID de cadena *IDS_GREETING* en el panel Cadenas y añada texto para cada idioma activo.

```
import mx.lang.Locale;
this.createTextField("greeting_txt", 10, 40, 40, 200, 20);
greeting_txt.autoSize = "left";
Locale.autoReplace = true;
Locale.addDelayedInstance(greeting_txt, "IDS_GREETING");
Locale.loadLanguageXML("en");
```

checkXMLStatus (método Locale.checkXMLStatus)

```
public static checkXMLStatus() : Boolean
```

Devuelve `true` si se ha cargado del archivo XML; `false` en caso contrario.

Disponibilidad: ActionScript 2.0; Flash Player 7

Valor devuelto

`Boolean` - Devuelve `true` si se ha cargado del archivo XML; `false` en caso contrario.

Ejemplo

El ejemplo siguiente utiliza un intervalo para comprobar cada 10 milisegundos si el archivos de idioma se ha cargado correctamente. Una vez que el archivo XML se ha cargado, el campo de texto `greeting_txt` del escenario se llena con la cadena `IDS_GREETING` del archivo XML de idioma.

```
import mx.lang.Locale;
Locale.setLoadCallback(localeCallback);
Locale.loadLanguageXML("en");
// create interval to check if language XML file is loaded
var locale_int:Number = setInterval(checkLocaleStatus, 10);
function checkLocaleStatus():Void {
  if (Locale.checkXMLStatus()) {
    clearInterval(locale_int);
    trace("clearing interval @ " + getTimer() + " ms");
  }
}
// callback function for Locale.setLoadCallback()
function localeCallback(success:Boolean):Void {
  greeting_txt.text = Locale.loadString("IDS_GREETING");
}
```

getDefaultLang (método Locale.getDefaultLang)

```
public static getDefaultLang() : String
```

El código de idioma predeterminado definido en el cuadro de diálogo del panel Cadenas o al llamar al método `setDefaultLang()`.

Disponibilidad: ActionScript 2.0; Flash Player 8

Valor devuelto

`String` - Devuelve el código de idioma predeterminado.

Ejemplo

El ejemplo siguiente crea una variable denominada `defLang`, utilizada para el idioma predeterminado inicial para el documento de Flash. En el panel Cadenas, haga clic en el botón Configuración para abrir el cuadro de diálogo Configuración. A continuación agregue dos idiomas activos: inglés (en) y francés (fr), defina el botón de opción de sustitución de cadenas como `"via ActionScript at runtime"` y haga clic en Aceptar. En el panel Cadenas, añada un ID de cadena *IDS_GREETING* y, a continuación, añada texto para cada idioma activo.

```
import mx.lang.Locale;
var defLang:String = "fr";
Locale.setDefaultLang(defLang);
Locale.setLoadCallback(localeCallback);
```

```
Locale.loadLanguageXML(Locale.getDefaultLang());
function localeCallback(success:Boolean) {
  if (success) {
    trace(Locale.stringIDArray); // IDS_GREETING
    trace(Locale.loadString("IDS_GREETING"));
  } else {
    trace("unable to load XML");
  }
}
```

Véase también

`setDefaultLang (método Locale.setDefaultLang)`

initialize (método Locale.initialize)

```
public static initialize() : Void
```

Determina automáticamente el idioma que se va a utilizar y carga el archivo de idioma XML. Flash lo utiliza principalmente cuando el método de sustitución de cadenas es `"automatically at runtime"`.

Disponibilidad: ActionScript 2.0; Flash Player 7

Ejemplo

Este ejemplo muestra cómo usar el método `initialize()` para llenar automáticamente el campo de texto `greeting_txt` en el escenario con el idioma actual del SO del usuario. En lugar de utilizar el método `initialize()` directamente, utilice el método de sustitución de cadenas de `"automatically at runtime"`.

```
import mx.lang.Locale;
trace(System.capabilities.language);
Locale.autoReplace = true;
Locale.addDelayedInstance(greeting_txt, "IDS_GREETING");
Locale.initialize();
```

languageCodeArray (propiedad Locale.languageCodeArray)

```
public static languageCodeArray : Array [read-only]
```

Una matriz con los códigos de los idiomas que se han especificado o cargado en el archivo FLA. Los códigos de idioma no se ordenan alfabéticamente.

Disponibilidad: ActionScript 2.0; Flash Player 8

Ejemplo

El ejemplo siguiente carga un archivo XML de idioma basado en el valor actual de un componente ComboBox. Arrastre un componente ComboBox al escenario y asígnele un nombre de instancia `lang_cb`. Con la herramienta Texto, cree un campo de texto dinámico y asígnele el nombre de instancia `greeting_txt`. En el panel Cadenas, añada al menos dos idiomas activos, defina el botón de opción de sustitución de cadenas como `"via ActionScript at runtime"` y haga clic en Aceptar. A continuación, añada un ID de cadena *IDS_GREETING* e introduzca texto para cada idioma activo. Por último, añada el siguiente código ActionScript al fotograma 1 de la línea de tiempo principal:

```
import mx.lang.Locale;
Locale.setLoadCallback(localeListener);
lang_cb.dataProvider = Locale.languageCodeArray.sort();
lang_cb.addEventListener("change", langListener);

function langListener(eventObj:Object):Void {
  Locale.loadLanguageXML(eventObj.target.value);
}
function localeListener(success:Boolean):Void {
  if (success) {
    greeting_txt.text = Locale.loadString("IDS_GREETING");
  } else {
    greeting_txt.text = "unable to load language XML file.";
  }
}
```

loadLanguageXML (método Locale.loadLanguageXML)

```
public static loadLanguageXML(xmlLanguageCode:String,
  customXmlCompleteCallback:Function) : Void
```

Carga el archivo de idioma XML especificado.

Disponibilidad: ActionScript 2.0; Flash Player 8

Parámetros

`xmlLanguageCode``:String` - El código de idioma para el archivo de idioma XML que desea cargar.

`customXmlCompleteCallback``:Function` - Función callback que se va a llamar cuando se cargue un archivo de idioma XML.

Ejemplo

El ejemplo siguiente utiliza el método `loadLanguageXML()` para cargar el archivo de idioma XML inglés (en). Una vez que está cargado el archivo de idioma, se llama al método `localeCallback()` que llena el campo de texto `greeting_txt` del escenario con el contenido de la cadena `IDS_GREETING` del archivo XML.

```
import mx.lang.Locale;
Locale.setLoadCallback(localeCallback);
Locale.loadLanguageXML("en");
// create interval to check if language XML file is loaded
var locale_int:Number = setInterval(checkLocaleStatus, 10);
function checkLocaleStatus():Void {
  if (Locale.checkXMLStatus()) {
    clearInterval(locale_int);
    trace("clearing interval @ " + getTimer() + " ms");
  }
}
// callback function for Locale.setLoadCallback()
function localeCallback(success:Boolean):Void {
  greeting_txt.text = Locale.loadString("IDS_GREETING");
}
```

loadString (método Locale.loadString)

```
public static loadString(id:String) : String
```

Devuelve el valor de cadena asociado con el ID de cadena indicado en el idioma actual.

Disponibilidad: ActionScript 2.0; Flash Player 7

Parámetros

`id``:String` - El número de identificación (ID) de la cadena que se va a cargar.

Valor devuelto

`String` - Devuelve el valor de cadena asociado con el ID de cadena indicado en el idioma actual.

Ejemplo

El ejemplo siguiente utiliza un intervalo para comprobar cada 10 milisegundos si el archivos de idioma se ha cargado correctamente. Una vez que el archivo XML se ha cargado, el campo de texto `greeting_txt` del escenario se llena con la cadena `IDS_GREETING` del archivo XML de idioma.

```
import mx.lang.Locale;
Locale.setLoadCallback(localeCallback);
Locale.loadLanguageXML("en");
// create interval to check if language XML file is loaded
var locale_int:Number = setInterval(checkLocaleStatus, 10);
function checkLocaleStatus():Void {
  if (Locale.checkXMLStatus()) {
    clearInterval(locale_int);
    trace("clearing interval @ " + getTimer() + " ms");
  }
}
// callback function for Locale.setLoadCallback()
function localeCallback(success:Boolean):Void {
  greeting_txt.text = Locale.loadString("IDS_GREETING");
}
```

Véase también

`loadStringEx (método Locale.loadStringEx)`

loadStringEx (método Locale.loadStringEx)

```
public static loadStringEx(stringID:String, languageCode:String) : String
```

Devuelve el valor de cadena asociado con el ID de cadena y el código de idioma indicado. Para evitar la carga del archivo XML no planificada, `loadStringEx()` no carga el archivo de idioma XML si no está ya cargado. Debe decidir el momento correcto para llamar a `loadLanguageXML()` si desea cargar el archivo de idioma XML.

Disponibilidad: ActionScript 2.0; Flash Player 8

Parámetros

`stringID``:String` - El número de identificación (ID) de la cadena que se va a cargar.

`languageCode``:String` - El código de idioma.

Valor devuelto

`String` - El valor de cadena asociado con el ID de cadena indicado en el idioma especificado por el parámetro `languageCode`.

Ejemplo

El ejemplo siguiente utiliza el método `loadStringEx()` para controlar el valor de la cadena `IDS_GREETING` para el archivo XML de idioma francés cargado.

```
import mx.lang.Locale;
Locale.setLoadCallback(localeCallback);
Locale.loadLanguageXML("fr");
function localeCallback(success:Boolean) {
  trace(success);
  trace(Locale.stringIDArray); // IDS_GREETING
  trace(Locale.loadStringEx("IDS_GREETING", "fr")); // bonjour
}
```

Véase también

`loadString (método Locale.loadString)`

setDefaultLang (método Locale.setDefaultLang)

```
public static setDefaultLang(langCode:String) : Void
```

Define el código de idioma predeterminado.

Disponibilidad: ActionScript 2.0; Flash Player 7

Parámetros

`langCode``:String` - Una cadena que representa un código de idioma.

Ejemplo

El ejemplo siguiente crea una variable denominada `defLang`, utilizada para el idioma predeterminado inicial para el documento de Flash. En el panel Cadenas, haga clic en el botón Configuración para abrir el cuadro de diálogo Configuración. A continuación agregue dos idiomas activos: inglés (en) y francés (fr), defina el botón de opción de sustitución de cadenas como `"via ActionScript at runtime"` y haga clic en Aceptar. En el panel Cadenas, añada un ID de cadena *IDS_GREETING* y, a continuación, añada texto para cada idioma activo.

```
import mx.lang.Locale;
var defLang:String = "fr";
Locale.setDefaultLang(defLang);
Locale.setLoadCallback(localeCallback);
Locale.loadLanguageXML(Locale.getDefaultLang());
```

```
function localeCallback(success:Boolean) {
  if (success) {
    trace(Locale.stringIDArray); // IDS_GREETING
    trace(Locale.loadString("IDS_GREETING"));
  } else {
    trace("unable to load XML");
  }
}
```

Véase también

getDefaultLang (método Locale.getDefaultLang)

setLoadCallback (método Locale.setLoadCallback)

```
public static setLoadCallback(loadCallback:Function) : Void
```

Define la función callback que se llamará después de cargar el archivo XML.

Disponibilidad: ActionScript 2.0; Flash Player 7

Parámetros

loadCallback:Function - Función que se va a llamar cuando se cargue un archivo de idioma XML.

Ejemplo

El ejemplo siguiente utiliza un intervalo para comprobar cada 10 milisegundos si el archivos de idioma se ha cargado correctamente. Una vez que el archivo XML se ha cargado, el campo de texto `greeting_txt` del escenario se llena con la cadena `IDS_GREETING` del archivo XML de idioma.

```
import mx.lang.Locale;
Locale.setLoadCallback(localeCallback);
Locale.loadLanguageXML("en");
// create interval to check if language XML file is loaded
var locale_int:Number = setInterval(checkLocaleStatus, 10);
function checkLocaleStatus():Void {
  if (Locale.checkXMLStatus()) {
    clearInterval(locale_int);
    trace("clearing interval @ " + getTimer() + " ms");
  }
}
// callback function for Locale.setLoadCallback()
function localeCallback(success:Boolean):Void {
  greeting_txt.text = Locale.loadString("IDS_GREETING");
}
```

setString (método Locale.setString)

```
public static setString(stringID:String, languageCode:String,
  stringValue:String) : Void
```

Define el nuevo valor de cadena de un ID de cadena y código de idioma indicado.

Disponibilidad: ActionScript 2.0; Flash Player 8

Parámetros

stringID:String - El número de identificación (ID) de la cadena que va a definir.

languageCode:String - El código de idioma.

stringValue:String - Un valor de la cadena.

Ejemplo

El ejemplo siguiente utiliza el método `setString()` para definir la cadena `IDS_WELCOME` para inglés (en) y francés (fr).

```
import mx.lang.Locale;
Locale.setString("IDS_WELCOME", "en", "hello");
Locale.setString("IDS_WELCOME", "fr", "bonjour");
trace(Locale.loadStringEx("IDS_WELCOME", "en")); // hello
```

stringIDArray (propiedad Locale.stringIDArray)

```
public static stringIDArray : Array [read-only]
```

Una matriz que contiene todos los ID de cadena del archivo FLA. Los ID de cadena no se ordenan alfabéticamente.

Disponibilidad: ActionScript 2.0; Flash Player 8

Ejemplo

El ejemplo siguiente controla la propiedad `Locale.stringIDArray` para el archivo de idioma XML cargado. En el panel Cadenas, haga clic en el botón Configuración para abrir el cuadro de diálogo Configuración. A continuación agregue dos idiomas activos: inglés (en) y francés (fr), defina el botón de opción de sustitución de cadenas como `"via ActionScript at runtime"` y haga clic en Aceptar. En el panel Cadenas, añada un ID de cadena *IDS_GREETING* y, a continuación, añada texto para cada idioma activo.

```
import mx.lang.Locale;
Locale.setLoadCallback(localeCallback);
Locale.loadLanguageXML("fr");
```

```
function localeCallback(success:Boolean) {
  trace(success);
  trace(Locale.stringIDArray); // IDS_GREETING
  trace(Locale.loadStringEx("IDS_GREETING", "fr")); // bonjour
}
```

Math

```
Object
  |
  +-Math
```

`public class` ***Math***
`extends Object`

La clase Math es una clase de nivel superior cuyos métodos y propiedades puede utilizar sin emplear un constructor.

Utilice los métodos y propiedades de esta clase para acceder a las constantes y funciones matemáticas y manipularlas. Todos los métodos y propiedades de la clase Math son estáticos y deben llamarse utilizando la sintaxis `Math.method( parameter )` o `Math.constant` En ActionScript, las constantes se definen con la máxima precisión de los números de coma flotante IEEE-754 de doble precisión.

Varios métodos de la clase Math utilizan la medida de un ángulo en radianes como parámetro. Puede utilizar la siguiente ecuación para calcular el valor en radianes antes de llamar al método y luego proporcionar el valor calculado como parámetro, o bien puede proporcionar toda la parte derecha de la ecuación (con la medida del ángulo en radianes en lugar de `degrees`) como el parámetro radian.

Para calcular un valor en radianes, utilice la siguiente fórmula:

```
radians = degrees * Math.PI/180
```

A continuación se ofrece un ejemplo en el que se pasa la ecuación como parámetro para calcular el seno de un ángulo de 45°:

`Math.sin(45 * Math.PI/180)` es lo mismo que `Math.sin(.7854)`

Disponibilidad: ActionScript 1.0; Flash Player 5

Resumen de propiedades

Modificadores	Propiedad	Descripción
`static`	`E:Number`	Constante matemática de la base de los logaritmos neperianos expresada como número *e*.
`static`	`LN10:Number`	Constante matemática del logaritmo neperiano de 10, expresada como loge10, con un valor aproximado de 2,302585092994046.
`static`	`LN2:Number`	Constante matemática del logaritmo neperiano de 2, expresada como loge2, con un valor aproximado de 0,6931471805599453.
`static`	`LOG10E:Number`	Constante matemática del logaritmo en base 10 de la constante e (`Math.E`), expresada como log10e, con un valor aproximado de 0,4342944819032518.
`static`	`LOG2E:Number`	Constante matemática del logaritmo en base 2 de la constante e (`Math.E`), expresada como log2e, con un valor aproximado de 1,442695040888963387.
`static`	`PI:Number`	Constante matemática del radio de la circunferencia de un círculo con respecto a su diámetro, expresada como pi, con un valor aproximado de 3,141592653589793.
`static`	`SQRT1_2:Number`	Constante matemática para la raíz cuadrada de un medio, con un valor aproximado de 0,7071067811865476.
`static`	`SQRT2:Number`	Constante matemática para la raíz cuadrada de 2, con un valor aproximado de 1,4142135623730951.

Propiedades heredadas de la clase Object

```
constructor (propiedad Object.constructor), __proto__ (propiedad
Object.__proto__), prototype (propiedad Object.prototype), __resolve
(propiedad Object.__resolve)
```

Resumen de métodos

Modificadores	Firma	Descripción
`static`	`abs(x:Number) : Number`	Calcula y devuelve un valor absoluto para el número especificado por el parámetro `x`.
`static`	`acos(x:Number) : Number`	Calcula y devuelve el arco coseno del número especificado por el parámetro `x`, en radianes.

Modificadores	Firma	Descripción
static	asin(x:Number) : Number	Calcula y devuelve el arco seno del número especificado por el parámetro x, en radianes.
static	atan(tangent:Number) : Number	Calcula y devuelve el valor en radianes del ángulo cuya tangente se especifica en el parámetro tangent.
static	atan2(y:Number, x:Number) : Number	Calcula y devuelve el ángulo del punto y/ x en radianes cuando se mide en sentido contrario al de las agujas del reloj desde el eje *x* de un círculo (siendo 0,0 el centro del círculo).
static	ceil(x:Number) : Number	Devuelve el valor redondeado al alza del número o expresión especificada.
static	cos(x:Number) : Number	Calcula y devuelve el coseno del ángulo especificado en radianes.
static	exp(x:Number) : Number	Devuelve el valor de la base del logaritmo neperiano (*e*) elevado a la potencia del exponente especificado en el parámetro x.
static	floor(x:Number) : Number	Devuelve el valor redondeado a la baja del número o expresión especificada en el parámetro x.
static	log(x:Number) : Number	Devuelve el logaritmo del parámetro x.
static	max(x:Number, y:Number) : Number	Calcula x e y y devuelve el valor mayor.
static	min(x:Number, y:Number) : Number	Calcula x e y y devuelve el valor más pequeño.
static	pow(x:Number, y:Number) : Number	Calcula y devuelve x elevado a la potencia de y.
static	random() : Number	Devuelve un número n seudo-aleatorio, siendo 0 <= n < 1.
static	round(x:Number) : Number	Redondea el valor del parámetro x hacia arriba o hacia abajo con el entero más próximo y devuelve el valor resultante.
static	sin(x:Number) : Number	Calcula y devuelve el seno del ángulo especificado en radianes.
static	sqrt(x:Number) : Number	Calcula y devuelve la raíz cuadrada del número especificado.
static	tan(x:Number) : Number	Calcula y devuelve la tangente del ángulo especificado.

Métodos heredados de la clase Object

addProperty (método Object.addProperty), hasOwnProperty (método Object.hasOwnProperty), isPropertyEnumerable (método Object.isPropertyEnumerable), isPrototypeOf (método Object.isPrototypeOf), registerClass (método Object.registerClass), toString (método Object.toString), unwatch (método Object.unwatch), valueOf (método Object.valueOf), watch (método Object.watch)

abs (método Math.abs)

```
public static abs(x:Number) : Number
```

Calcula y devuelve un valor absoluto para el número especificado por el parámetro `x`.

Disponibilidad: ActionScript 1.0; Flash Player 5

Parámetros

x:`Number` - Un número.

Valor devuelto

`Number` - Un número.

Ejemplo

El ejemplo siguiente muestra cómo `Math.abs()` devuelve el valor absoluto de un número y no afecta al valor del parámetro `x` (denominado `num` en este ejemplo):

```
var num:Number = -12;
var numAbsolute:Number = Math.abs(num);
trace(num); // output: -12
trace(numAbsolute); // output: 12
```

acos (método Math.acos)

```
public static acos(x:Number) : Number
```

Calcula y devuelve el arco coseno del número especificado por el parámetro `x`, en radianes.

Disponibilidad: ActionScript 1.0; Flash Player 5

Parámetros

x:`Number` - Un número entre -1.0 y 1.0.

Valor devuelto

`Number` - Un número; el arco coseno del parámetro `x`.

Ejemplo

El ejemplo siguiente muestra el arco coseno de varios valores.

```
trace(Math.acos(-1)); // output: 3.14159265358979
trace(Math.acos(0)); // output: 1.5707963267949
trace(Math.acos(1)); // output: 0
```

Véase también

asin (método Math.asin), atan (método Math.atan), atan2 (método Math.atan2), cos (método Math.cos), sin (método Math.sin), tan (método Math.tan)

asin (método Math.asin)

```
public static asin(x:Number) : Number
```

Calcula y devuelve el arco seno del número especificado por el parámetro `x`, en radianes.

Disponibilidad: ActionScript 1.0; Flash Player 5

Parámetros

x:`Number` - Un número entre -1.0 y 1.0.

Valor devuelto

`Number` - Un número comprendido entre pi negativo dividido por 2 y pi positivo dividido por 2.

Ejemplo

El ejemplo siguiente muestra el arco seno de varios valores.

```
trace(Math.asin(-1)); // output: -1.5707963267949
trace(Math.asin(0)); // output: 0
trace(Math.asin(1)); // output: 1.5707963267949
```

Véase también

acos (método Math.acos), atan (método Math.atan), atan2 (método Math.atan2), cos (método Math.cos), sin (método Math.sin), tan (método Math.tan)

atan (método Math.atan)

```
public static atan(tangent:Number) : Number
```

Calcula y devuelve el valor en radianes del ángulo cuya tangente se especifica en el parámetro `tangent`. El valor devuelto está comprendido entre pi negativo dividido por 2 y pi positivo dividido por 2.

Disponibilidad: ActionScript 1.0; Flash Player 5

Parámetros

`tangent``:Number` - Un número que representa la tangente de un ángulo.

Valor devuelto

`Number` - Un número comprendido entre pi negativo dividido por 2 y pi positivo dividido por 2.

Ejemplo

El ejemplo siguiente muestra el valor de ángulo para varias tangentes.

```
trace(Math.atan(-1)); // output: -0.785398163397448
trace(Math.atan(0)); // output: 0
trace(Math.atan(1)); // output: 0.785398163397448
```

Véase también

`acos (método Math.acos)`, `asin (método Math.asin)`, `atan2 (método Math.atan2)`, `cos (método Math.cos)`, `sin (método Math.sin)`, `tan (método Math.tan)`

atan2 (método Math.atan2)

```
public static atan2(y:Number, x:Number) : Number
```

Calcula y devuelve el ángulo del punto `y`/ `x` en radianes cuando se mide en sentido contrario al de las agujas del reloj desde el eje *x* de un círculo (siendo 0,0 el centro del círculo). El valor devuelto está comprendido entre pi positivo y pi negativo.

Disponibilidad: ActionScript 1.0; Flash Player 5

Parámetros

`y``:Number` - Un número que especifica la coordenada *y* del punto.

`x``:Number` - Un número que especifica la coordenada *x* del punto.

Valor devuelto

`Number` - Un número.

Ejemplo

El ejemplo siguiente devuelve el ángulo, en radianes, del punto especificado por las coordenadas (0, 10), de forma que x = 0 e y = 10. Observe que el primer parámetro de atan2 es siempre la coordenada y.

```
trace(Math.atan2(10, 0)); // output: 1.5707963267949
```

Véase también

`acos (método Math.acos)`, `asin (método Math.asin)`, `atan (método Math.atan)`, `cos (método Math.cos)`, `sin (método Math.sin)`, `tan (método Math.tan)`

ceil (método Math.ceil)

```
public static ceil(x:Number) : Number
```

Devuelve el valor redondeado al alza del número o expresión especificada. El valor redondeado al alza de un número es el número entero más cercano mayor o igual al número.

Disponibilidad: ActionScript 1.0; Flash Player 5

Parámetros

x`:Number` - Un número o expresión.

Valor devuelto

`Number` - Un entero que es, al mismo tiempo, el más próximo y mayor o igual que el parámetro `x`.

Ejemplo

El código siguiente devuelve un valor de 13:

```
Math.ceil(12.5);
```

Véase también

`floor (método Math.floor)`, `round (método Math.round)`

cos (método Math.cos)

```
public static cos(x:Number) : Number
```

Calcula y devuelve el coseno del ángulo especificado en radianes. Para calcular un radián, consulte la descripción incluida en la entrada de la clase Math.

Disponibilidad: ActionScript 1.0; Flash Player 5

Parámetros

x:Number - Un número que representa un ángulo medido en radianes.

Valor devuelto

Number - Un número entre -1.0 y 1.0.

Ejemplo

El ejemplo siguiente muestra el coseno de varios ángulos diferentes.

```
trace (Math.cos(0)); // 0 degree angle. Output: 1
trace (Math.cos(Math.PI/2)); // 90 degree angle. Output: 6.12303176911189e-
  17
trace (Math.cos(Math.PI)); // 180 degree angle. Output: -1
trace (Math.cos(Math.PI*2)); // 360 degree angle. Output: 1
```

Nota: El coseno de un ángulo de 90 grados es cero, pero debido a la inherente falta de precisión de los cálculos decimales utilizando números binarios, Flash Player generará un número muy cercano pero que no es exactamente igual a cero.

Véase también

acos (método Math.acos), asin (método Math.asin), atan (método Math.atan), atan2 (método Math.atan2), sin (método Math.sin), tan (método Math.tan)

E (propiedad Math.E)

```
public static E : Number
```

Constante matemática de la base de los logaritmos neperianos expresada como número *e*. El valor aproximado de *e* es 2.71828182845905.

Disponibilidad: ActionScript 1.0; Flash Player 5

Ejemplo

El ejemplo siguiente muestra cómo se puede utilizar `Math.E` para calcular continuamente intereses compuestos en un sencillo caso de un interés del 100 por cien en un periodo de un año.

```
var principal:Number = 100;
var simpleInterest:Number = 100;
var continuouslyCompoundedInterest:Number = (100 * Math.E) - principal;

trace ("Beginning principal: $" + principal);
trace ("Simple interest after one year: $" + simpleInterest);
trace ("Continuously compounded interest after one year: $" +
  continuouslyCompoundedInterest);

//
Output:
Beginning principal: $100
Simple interest after one year: $100
Continuously compounded interest after one year: $171.828182845905
```

exp (método Math.exp)

```
public static exp(x:Number) : Number
```

Devuelve el valor de la base del logaritmo neperiano (*e*) elevado a la potencia del exponente especificado en el parámetro `x`. La constante `Math.E` puede proporcionar el valor de *e*.

Disponibilidad: ActionScript 1.0; Flash Player 5

Parámetros

`x``:Number` - El exponente; un número o expresión.

Valor devuelto

`Number` - Un número.

Ejemplo

El ejemplo siguiente muestra el logaritmo de dos números.

```
trace(Math.exp(1)); // output: 2.71828182845905
trace(Math.exp(2)); // output: 7.38905609893065
```

Véase también

`E (propiedad Math.E)`

floor (método Math.floor)

```
public static floor(x:Number) : Number
```

Devuelve el valor redondeado a la baja del número o expresión especificada en el parámetro `x`. El valor redondeado a la baja es el entero más cercano inferior o igual al número o expresión especificada.

Disponibilidad: ActionScript 1.0; Flash Player 5

Parámetros

x`:Number` - Un número o expresión.

Valor devuelto

`Number` - El entero que es, al mismo tiempo, el más próximo y menor o igual que el parámetro `x`.

Ejemplo

El código siguiente devuelve un valor de 12:

```
Math.floor(12.5);
```

El código siguiente devuelve un valor de 7:

```
Math.floor(-6.5);
```

LN10 (propiedad Math.LN10)

```
public static LN10 : Number
```

Constante matemática del logaritmo neperiano de 10, expresada como loge10, con un valor aproximado de 2,302585092994046.

Disponibilidad: ActionScript 1.0; Flash Player 5

Ejemplo

Este ejemplo rastrea el valor de `Math.LN10`.

```
trace(Math.LN10);
// output: 2.30258509299405
```

LN2 (propiedad Math.LN2)

```
public static LN2 : Number
```

Constante matemática del logaritmo neperiano de 2, expresada como loge2, con un valor aproximado de 0,6931471805599453.

Disponibilidad: ActionScript 1.0; Flash Player 5

log (método Math.log)

```
public static log(x:Number) : Number
```

Devuelve el logaritmo del parámetro `x`.

Disponibilidad: ActionScript 1.0; Flash Player 5

Parámetros

`x``:Number` - Un número o expresión con un valor mayor que 0.

Valor devuelto

`Number` - Devuelve el logaritmo del parámetro `x`.

Ejemplo

El ejemplo siguiente muestra el logaritmo de tres números.

```
trace(Math.log(0)); // output: -Infinity
trace(Math.log(1)); // output: 0
trace(Math.log(2)); // output: 0.693147180559945
trace(Math.log(Math.E)); // output: 1
```

LOG10E (propiedad Math.LOG10E)

```
public static LOG10E : Number
```

Constante matemática del logaritmo en base 10 de la constante e (`Math.E`), expresada como log10e, con un valor aproximado de 0,4342944819032518.

El método `Math.log()` calcula el logaritmo neperiano de un número. Multiplique el resultado de `Math.log()` por `Math.LOG10E` para obtener el logaritmo de base 10.

Disponibilidad: ActionScript 1.0; Flash Player 5

Ejemplo

Este ejemplo muestra cómo obtener el logaritmo de base 10 de un número:

```
trace(Math.log(1000) * Math.LOG10E);
// Output: 3
```

LOG2E (Math.LOG2E property)

```
public static LOG2E : Number
```

Constante matemática del logaritmo en base 2 de la constante e (`Math.E`), expresada como log2e, con un valor aproximado de 1,442695040888963387.

El método `Math.log` calcula el logaritmo neperiano de un número. Multiplique el resultado de `Math.log()` por `Math.LOG2E` para obtener el logaritmo de base 2.

Disponibilidad: ActionScript 1.0; Flash Player 5

Ejemplo

Este ejemplo muestra cómo obtener el logaritmo de base 2 de un número:

```
trace(Math.log(16) * Math.LOG2E);
// Output: 4
```

max (método Math.max)

```
public static max(x:Number, y:Number) : Number
```

Calcula `x` e `y` y devuelve el valor mayor.

Disponibilidad: ActionScript 1.0; Flash Player 5

Parámetros

x`:Number` - Un número o expresión.

y`:Number` - Un número o expresión.

Valor devuelto

`Number` - Un número.

Ejemplo

El ejemplo siguiente muestra `Thu Dec 30 00:00:00 GMT-0700 2004`, que es la mayor de las expresiones evaluadas.

```
var date1:Date = new Date(2004, 11, 25);
var date2:Date = new Date(2004, 11, 30);
var maxDate:Number = Math.max(date1.getTime(), date2.getTime());
trace(new Date(maxDate).toString());
```

Véase también

`min (método Math.min)`, `Date`

min (método Math.min)

```
public static min(x:Number, y:Number) : Number
```

Calcula `x` e `y` y devuelve el valor más pequeño.

Disponibilidad: ActionScript 1.0; Flash Player 5

Parámetros

x:`Number` - Un número o expresión.

y:`Number` - Un número o expresión.

Valor devuelto

`Number` - Un número.

Ejemplo

El ejemplo siguiente muestra `Sat Dec 25 00:00:00 GMT-0700 2004`, que es la menor de las expresiones evaluadas.

```
var date1:Date = new Date(2004, 11, 25);
var date2:Date = new Date(2004, 11, 30);
var minDate:Number = Math.min(date1.getTime(), date2.getTime());
trace(new Date(minDate).toString());
```

Véase también

`max (método Math.max)`

PI (propiedad Math.PI)

```
public static PI : Number
```

Constante matemática del radio de la circunferencia de un círculo con respecto a su diámetro, expresada como pi, con un valor aproximado de 3,141592653589793.

Disponibilidad: ActionScript 1.0; Flash Player 5

Ejemplo

El ejemplo siguiente dibuja un círculo utilizando la constante matemática pi y la API de dibujo.

```
drawCircle(this, 100, 100, 50);
//
function drawCircle(mc:MovieClip, x:Number, y:Number, r:Number):Void {
  mc.lineStyle(2, 0xFF0000, 100);
  mc.moveTo(x+r, y);
```

```
  mc.curveTo(r+x, Math.tan(Math.PI/8)*r+y, Math.sin(Math.PI/4)*r+x,
  Math.sin(Math.PI/4)*r+y);
  mc.curveTo(Math.tan(Math.PI/8)*r+x, r+y, x, r+y);
  mc.curveTo(-Math.tan(Math.PI/8)*r+x, r+y, -Math.sin(Math.PI/4)*r+x,
  Math.sin(Math.PI/4)*r+y);
  mc.curveTo(-r+x, Math.tan(Math.PI/8)*r+y, -r+x, y);
  mc.curveTo(-r+x, -Math.tan(Math.PI/8)*r+y, -Math.sin(Math.PI/4)*r+x, -
  Math.sin(Math.PI/4)*r+y);
  mc.curveTo(-Math.tan(Math.PI/8)*r+x, -r+y, x, -r+y);
  mc.curveTo(Math.tan(Math.PI/8)*r+x, -r+y, Math.sin(Math.PI/4)*r+x, -
  Math.sin(Math.PI/4)*r+y);
  mc.curveTo(r+x, -Math.tan(Math.PI/8)*r+y, r+x, y);
}
```

pow (método Math.pow)

```
public static pow(x:Number, y:Number) : Number
```

Calcula y devuelve `x` elevado a la potencia de `y`.

Disponibilidad: ActionScript 1.0; Flash Player 5

Parámetros

`x``:Number` - Un número elevado a una potencia.

`y``:Number` - Un número que especifica la potencia a la que se eleva el parámetro `x`.

Valor devuelto

`Number` - Un número.

Ejemplo

El ejemplo siguiente utiliza `Math.pow` y `Math.sqrt` para calcular la longitud de una línea.

```
this.createEmptyMovieClip("canvas_mc", this.getNextHighestDepth());
var mouseListener:Object = new Object();
mouseListener.onMouseDown = function() {
  this.origX = _xmouse;
  this.origY = _ymouse;
};
mouseListener.onMouseUp = function() {
  this.newX = _xmouse;
  this.newY = _ymouse;
  var minY = Math.min(this.origY, this.newY);
  var nextDepth:Number = canvas_mc.getNextHighestDepth();
  var line_mc:MovieClip =
  canvas_mc.createEmptyMovieClip("line"+nextDepth+"_mc", nextDepth);
  line_mc.moveTo(this.origX, this.origY);
  line_mc.lineStyle(2, 0x000000, 100);
```

```
  line_mc.lineTo(this.newX, this.newY);
  var hypLen:Number = Math.sqrt(Math.pow(line_mc._width,
  2)+Math.pow(line_mc._height, 2));
  line_mc.createTextField("length"+nextDepth+"_txt",
  canvas_mc.getNextHighestDepth(), this.origX, this.origY-22, 100, 22);
  line_mc['length'+nextDepth+'_txt'].text = Math.round(hypLen) +" pixels";
};
Mouse.addListener(mouseListener);
```

El método `MovieClip.getNextHighestDepth()` utilizado en este ejemplo necesita Flash Player 7 o una versión posterior. Si el archivo SWF incluye un componente de la versión 2, utilice la clase DepthManager de los componentes de la versión 2 en lugar del método `MovieClip.getNextHighestDepth()`.

random (método Math.random)

```
public static random() : Number
```

Devuelve un número n seudo-aleatorio, siendo 0 <= n < 1. El número devuelto es seudo-aleatorio porque no se genera mediante un verdadero fenómeno aleatorio natural, como una caída radioactiva.

Disponibilidad: ActionScript 1.0; Flash Player 5

Valor devuelto

`Number` - Un número.

Ejemplo

El siguiente ejemplo obtiene 100 enteros aleatorios entre 4 y 11 (inclusive):

```
function randRange(min:Number, max:Number):Number {
  var randomNum:Number = Math.floor(Math.random() * (max - min + 1)) + min;
  return randomNum;
}
for (var i = 0; i < 100; i++) {
  var n:Number = randRange(4, 11)
  trace(n);
}
```

round (método Math.round)

```
public static round(x:Number) : Number
```

Redondea el valor del parámetro `x` hacia arriba o hacia abajo con el entero más próximo y devuelve el valor resultante. Si el parámetro `x` es equidistante de los dos enteros más próximos a él (es decir, que el número termina en ,5), el valor se redondea al alza con el siguiente entero mayor.

Disponibilidad: ActionScript 1.0; Flash Player 5

Parámetros

x:`Number` - Un número.

Valor devuelto

`Number` - Un número; un entero.

Ejemplo

El ejemplo siguiente devuelve un número aleatorio comprendido entre dos enteros especificados.

```
function randRange(min:Number, max:Number):Number {
  var randomNum:Number = Math.round(Math.random() * (max-min+1) + (min-
  .5));
  return randomNum;
}
for (var i = 0; i<25; i++) {
  trace(randRange(4, 11));
}
```

Véase también

`ceil (método Math.ceil)`, `floor (método Math.floor)`

sin (método Math.sin)

```
public static sin(x:Number) : Number
```

Calcula y devuelve el seno del ángulo especificado en radianes. Para calcular un radián, consulte la descripción incluida en la entrada de la clase Math.

Disponibilidad: ActionScript 1.0; Flash Player 5

Parámetros

x:`Number` - Un número que representa un ángulo medido en radianes.

Valor devuelto

`Number` - Un número; el seno del ángulo especificado (entre -1.0 y 1.0).

Ejemplo

El ejemplo siguiente dibuja un círculo utilizando la constante matemática pi, el seno de un ángulo y la API de dibujo

```
drawCircle(this, 100, 100, 50);
//
function drawCircle(mc:MovieClip, x:Number, y:Number, r:Number):Void {
  mc.lineStyle(2, 0xFF0000, 100);
  mc.moveTo(x+r, y);
  mc.curveTo(r+x, Math.tan(Math.PI/8)*r+y, Math.sin(Math.PI/4)*r+x,
  Math.sin(Math.PI/4)*r+y);
  mc.curveTo(Math.tan(Math.PI/8)*r+x, r+y, x, r+y);
  mc.curveTo(-Math.tan(Math.PI/8)*r+x, r+y, -Math.sin(Math.PI/4)*r+x,
  Math.sin(Math.PI/4)*r+y);
  mc.curveTo(-r+x, Math.tan(Math.PI/8)*r+y, -r+x, y);
  mc.curveTo(-r+x, -Math.tan(Math.PI/8)*r+y, -Math.sin(Math.PI/4)*r+x, -
  Math.sin(Math.PI/4)*r+y);
  mc.curveTo(-Math.tan(Math.PI/8)*r+x, -r+y, x, -r+y);
  mc.curveTo(Math.tan(Math.PI/8)*r+x, -r+y, Math.sin(Math.PI/4)*r+x, -
  Math.sin(Math.PI/4)*r+y);
  mc.curveTo(r+x, -Math.tan(Math.PI/8)*r+y, r+x, y);
}
```

Véase también

`acos (método Math.acos)`, `asin (método Math.asin)`, `atan (método Math.atan)`, `atan2 (método Math.atan2)`, `cos (método Math.cos)`, `tan (método Math.tan)`

sqrt (método Math.sqrt)

```
public static sqrt(x:Number) : Number
```

Calcula y devuelve la raíz cuadrada del número especificado.

Disponibilidad: ActionScript 1.0; Flash Player 5

Parámetros

x:`Number` - Un número o expresión mayor o igual que 0.

Valor devuelto

`Number` - Un número si el parámetro *x* es mayor o igual a cero; en caso contrario, NaN (no es un número).

Ejemplo

El ejemplo siguiente utiliza `Math.pow` y `Math.sqrt` para calcular la longitud de una línea.

```
this.createEmptyMovieClip("canvas_mc", this.getNextHighestDepth());
var mouseListener:Object = new Object();
mouseListener.onMouseDown = function() {
  this.origX = _xmouse;
  this.origY = _ymouse;
};
mouseListener.onMouseUp = function() {
  this.newX = _xmouse;
  this.newY = _ymouse;
  var minY = Math.min(this.origY, this.newY);
  var nextDepth:Number = canvas_mc.getNextHighestDepth();
  var line_mc:MovieClip =
  canvas_mc.createEmptyMovieClip("line"+nextDepth+"_mc", nextDepth);
  line_mc.moveTo(this.origX, this.origY);
  line_mc.lineStyle(2, 0x000000, 100);
  line_mc.lineTo(this.newX, this.newY);
  var hypLen:Number = Math.sqrt(Math.pow(line_mc._width,
  2)+Math.pow(line_mc._height, 2));
  line_mc.createTextField("length"+nextDepth+"_txt",
  canvas_mc.getNextHighestDepth(), this.origX, this.origY-22, 100, 22);
  line_mc['length'+nextDepth+'_txt'].text = Math.round(hypLen) +" pixels";
};
Mouse.addListener(mouseListener);
```

SQRT1_2 (propiedad Math.SQRT1_2)

```
public static SQRT1_2 : Number
```

Constante matemática para la raíz cuadrada de un medio, con un valor aproximado de 0,7071067811865476.

Disponibilidad: ActionScript 1.0; Flash Player 5

Ejemplo

Este ejemplo rastrea el valor de `Math.SQRT1_2`..

```
trace(Math.SQRT1_2);
// Output: 0.707106781186548
```

SQRT2 (propiedad Math.SQRT2)

```
public static SQRT2 : Number
```

Constante matemática para la raíz cuadrada de 2, con un valor aproximado de 1,4142135623730951.

Disponibilidad: ActionScript 1.0; Flash Player 5

Ejemplo

Este ejemplo rastrea el valor de `Math.SQRT2`..

```
trace(Math.SQRT2);
// Output: 1.4142135623731
```

tan (método Math.tan)

```
public static tan(x:Number) : Number
```

Calcula y devuelve la tangente del ángulo especificado. Para calcular un radián, utilice la información descrita en la introducción a la clase Math.

Disponibilidad: ActionScript 1.0; Flash Player 5

Parámetros

x`:Number` - Un número que representa un ángulo medido en radianes.

Valor devuelto

`Number` - Un número; tangente del parámetro `x`.

Ejemplo

El ejemplo siguiente dibuja un círculo utilizando la constante matemática pi, la tangente de un ángulo y la API de dibujo.

```
drawCircle(this, 100, 100, 50);
//
function drawCircle(mc:MovieClip, x:Number, y:Number, r:Number):Void {
  mc.lineStyle(2, 0xFF0000, 100);
  mc.moveTo(x+r, y);
  mc.curveTo(r+x, Math.tan(Math.PI/8)*r+y, Math.sin(Math.PI/4)*r+x,
  Math.sin(Math.PI/4)*r+y);
  mc.curveTo(Math.tan(Math.PI/8)*r+x, r+y, x, r+y);
  mc.curveTo(-Math.tan(Math.PI/8)*r+x, r+y, -Math.sin(Math.PI/4)*r+x,
  Math.sin(Math.PI/4)*r+y);
  mc.curveTo(-r+x, Math.tan(Math.PI/8)*r+y, -r+x, y);
  mc.curveTo(-r+x, -Math.tan(Math.PI/8)*r+y, -Math.sin(Math.PI/4)*r+x, -
  Math.sin(Math.PI/4)*r+y);
```

```
  mc.curveTo(-Math.tan(Math.PI/8)*r+x, -r+y, x, -r+y);
  mc.curveTo(Math.tan(Math.PI/8)*r+x, -r+y, Math.sin(Math.PI/4)*r+x, -
  Math.sin(Math.PI/4)*r+y);
  mc.curveTo(r+x, -Math.tan(Math.PI/8)*r+y, r+x, y);
}
```

Véase también

acos (método Math.acos), asin (método Math.asin), atan (método Math.atan), atan2 (método Math.atan2), cos (método Math.cos), sin (método Math.sin)

Matrix (flash.geom.Matrix)

```
Object
  |
  +-flash.geom.Matrix
```

public class *Matrix*
extends Object

La clase flash.geom.Matrix representa una matriz de transformación que determina cómo asignar puntos de un espacio de coordenadas a otro. Al definir las propiedades del objeto Matrix y aplicarlas a un objeto MovieClip o BitmapData es posible realizar varias transformaciones gráficas en el objeto. Estas funciones de transformación son la traslación (cambio de posición *x* e *y*), giro, cambio de escala y sesgo.

En su conjunto estas transformaciones se conocen como *transformaciones afines*. Las transformaciones afines preservan la rectilineidad de las líneas durante la transformación, de este modo las paralelas siguen siendo paralelas.

Para aplicar una matriz de transformación a un clip de película, se debe crear un objeto flash.geom.Transform y definir su propiedad Matrix con la matriz de transformación. Los objetos Matrix también se utilizan como parámetros de algunos métodos, como el método `draw()` de la clase flash.display.BitmapData.

Un objeto de matriz de transformación se considera una matriz de 3 x 3 con el contenido siguiente:

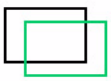
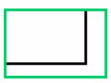
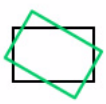
$$\begin{bmatrix} a & b & t_x \\ c & d & t_y \\ u & v & w \end{bmatrix}$$

En las matrices de transformación tradicionales, las propiedades `u`, `v` y `w` ofrecen capacidades extra. La clase Matrix sólo puede funcionar en un espacio bidimensional de modo que siempre presupone que los valores de propiedad `u` y `v` son 0.0, y que el valor de propiedad `w` es 1.0. En pocas palabras, los valores efectivos de la matriz son los siguientes:

$$\begin{bmatrix} a & b & t_x \\ c & d & t_y \\ 0 & 0 & 1 \end{bmatrix}$$

Es posible obtener y establecer los valores de las otras seis propiedades de un objeto Matrix: `a`, `b`, `c`, `d`, `tx` y `ty`.

La clase Matrix admite las cuatro funciones principales de transformación: traslación, escala, rotación y sesgo. Existen métodos especializados para tres de estas funciones, tal como se describe en la tabla siguiente.

Transformación	Método	Valores Matrix	Resultado mostrado	Descripción
Traslación (desplazamiento)	`translate(tx, ty)`	$\begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix}$		Desplaza la imagen `tx` píxeles hacia la derecha y `ty` píxeles hacia abajo.
Escala	`scale(sx, sy)`	$\begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix}$		Cambia el tamaño de la imagen multiplicando la ubicación de cada píxel por `sx` en el eje *x* y por `sy` en el eje *y*.
Rotación	`rotate(q)`	$\begin{bmatrix} \cos(q) & \sin(q) & 0 \\ -\sin(q) & \cos(q) & 0 \\ 0 & 0 & 1 \end{bmatrix}$		Gira la imagen según el ángulo `q`, medido en radianes

Transformación	Método	Valores Matrix	Resultado mostrado	Descripción
Sesgo o inclinación	Ninguno; deben definirse las propiedades `b` y `c`.	$\begin{bmatrix} 0 & sk_y & 0 \\ sk_x & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$		Desplaza de forma progresiva la imagen en una dirección paralela al eje *x* o *y*. El valor `skx` actúa como un multiplicador que controla la distancia del desplazamiento a lo largo del eje *x*; `sky` controla la distancia del desplazamiento a lo largo del eje *y*.

Cada función de transformación altera las propiedades de la matriz actual de forma que puede combinar distintas transformaciones. Para ello, llama a más de una función de transformación antes de aplicar la matriz a su clip de película o destino de mapa de bits.

Disponibilidad: ActionScript 1.0; Flash Player 8

Véase también

`transform (propiedad MovieClip.transform)`, `Transform (flash.geom.Transform)`, `draw (método BitmapData.draw)`, `a (propiedad Matrix.a)`, `b (propiedad Matrix.b)`, `c (propiedad Matrix.c)`, `d (propiedad Matrix.d)`, `tx (propiedad Matrix.tx)`, `ty (propiedad Matrix.ty)`, `translate (método Matrix.translate)`, `scale (método Matrix.scale)`, `rotate (método Matrix.rotate)`

Resumen de propiedades

Modificadores	Propiedad	Descripción
	`a:Number`	El valor de la primera fila y la primera columna del objeto Matrix, que afecta a la posición de los píxeles a lo largo del eje *x* cuando se cambia la escala o gira una imagen.
	`b:Number`	El valor de la primera fila y la segunda columna del objeto Matrix, que afecta a la posición de los píxeles a lo largo del eje *y* cuando se gira o sesga una imagen.

Modificadores	Propiedad	Descripción
	`c:Number`	El valor de la segunda fila y la primera columna del objeto Matrix, que afecta a la posición de los píxeles a lo largo del eje *x* cuando se gira o sesga una imagen.
	`d:Number`	El valor de la segunda fila y la segunda columna del objeto Matrix, que afecta a la posición de los píxeles a lo largo del eje *y* cuando se cambia la escala o gira una imagen.
	`tx:Number`	La distancia que se trasladará cada punto a lo largo del eje *x*.
	`ty:Number`	La distancia que se trasladará cada punto a lo largo del eje *y*.

Propiedades heredadas de la clase Object

```
constructor (propiedad Object.constructor), __proto__ (propiedad
Object.__proto__), prototype (propiedad Object.prototype), __resolve
(propiedad Object.__resolve)
```

Resumen de constructores

Firma	Descripción
`Matrix([a:Number], [b:Number], [c:Number], [d:Number], [tx:Number], [ty:Number])`	Crea un nuevo objeto Matrix con los parámetros especificados.

Resumen de métodos

Modificadores	Firma	Descripción
	`clone() : Matrix`	Devuelve un nuevo objeto Matrix que es una copia de esta matriz, con una copia exacta del objeto contenido.
	`concat(m:Matrix) : Void`	Concatena una matriz con la matriz actual y combina eficazmente los efectos geométricos de ambas matrices.

Modificadores	Firma	Descripción
	`createBox(scaleX:Number, scaleY:Number, [rotation:Number], [tx:Number], [ty:Number]) : Void`	Incluye parámetros para escala, rotación y traslación.
	`createGradientBox(width:Number, height:Number, [rotation:Number], [tx:Number], [ty:Number]) : Void`	Crea el estilo específico de matriz que el método `MovieClip.beginGradientFill()` espera.
	`deltaTransformPoint(pt:Point) : Point`	Tras especificar un punto en el espacio de coordenadas previas a la transformación, devuelve las coordenadas de dicho punto después de la transformación.
	`identity() : Void`	Establece cada propiedad de matriz en un valor que hace que un clip de película o construcción geométrica transformada sea idéntica a la original.
	`invert() : Void`	Realiza la transformación inversa de la matriz original.
	`rotate(angle:Number) : Void`	Establece los valores en la matriz actual, de forma que pueda utilizarse para aplicar una transformación de rotación.
	`scale(sx:Number, sy:Number) : Void`	Modifica una matriz de forma que su aplicación provoque un cambio de tamaño de una imagen.
	`toString() : String`	Devuelve un valor de texto con las propiedades del objeto Matrix.
	`transformPoint(pt:Point) : Point`	Aplica en el punto especificado la transformación geométrica representada por el objeto Matrix.
	`translate(tx:Number, ty:Number) : Void`	Modifica un objeto Matrix de forma que su transformación provoque el desplazamiento de un objeto por los ejes *x* e *y*.

addProperty (método Object.addProperty), hasOwnProperty (método Object.hasOwnProperty), isPropertyEnumerable (método Object.isPropertyEnumerable), isPrototypeOf (método Object.isPrototypeOf), registerClass (método Object.registerClass), toString (método Object.toString), unwatch (método Object.unwatch), valueOf (método Object.valueOf), watch (método Object.watch)

a (propiedad Matrix.a)

```
public a : Number
```

El valor de la primera fila y la primera columna del objeto Matrix, que afecta a la posición de los píxeles a lo largo del eje *x* cuando se cambia la escala o gira una imagen.

Disponibilidad: ActionScript 1.0; Flash Player 8

Ejemplo

El ejemplo siguiente crea el objeto Matrix `myMatrix` y define su valor `a`.

```
import flash.geom.Matrix;

var myMatrix:Matrix = new Matrix();
trace(myMatrix.a); // 1

myMatrix.a = 2;
trace(myMatrix.a); // 2
```

b (propiedad Matrix.b)

```
public b : Number
```

El valor de la primera fila y la segunda columna del objeto Matrix, que afecta a la posición de los píxeles a lo largo del eje *y* cuando se gira o sesga una imagen.

Disponibilidad: ActionScript 1.0; Flash Player 8

Ejemplo

El ejemplo siguiente crea el objeto Matrix `myMatrix` y define su valor `b`.

```
import flash.geom.Matrix;

var myMatrix:Matrix = new Matrix();
trace(myMatrix.b); // 0

var degrees:Number = 45;
```

```
var radians:Number = (degrees/180) Math.PI;
myMatrix.b = radians;
trace(myMatrix.b); // 0.785398163397448
```

c (propiedad Matrix.c)

```
public c : Number
```

El valor de la segunda fila y la primera columna del objeto Matrix, que afecta a la posición de los píxeles a lo largo del eje *x* cuando se gira o sesga una imagen.

Disponibilidad: ActionScript 1.0; Flash Player 8

Ejemplo

El ejemplo siguiente crea el objeto Matrix `myMatrix` y define su valor `c`.

```
import flash.geom.Matrix;

var myMatrix:Matrix = new Matrix();
trace(myMatrix.c); // 0

var degrees:Number = 45;
var radians:Number = (degrees/180) Math.PI;
myMatrix.c = radians;
trace(myMatrix.c); // 0.785398163397448
```

clone (método Matrix.clone)

```
public clone() : Matrix
```

Devuelve un nuevo objeto Matrix que es una copia de esta matriz, con una copia exacta del objeto contenido.

Disponibilidad: ActionScript 1.0; Flash Player 8

Valor devuelto

`flash.geom.Matrix` - Un objeto Matrix.

Ejemplo

El ejemplo siguiente crea la variable `clonedMatrix` a partir de la variable `myMatrix`. La clase Matrix no tiene un método `equals`, por lo tanto el ejemplo siguiente utiliza una función personalizada para comprobar la igualdad de las dos matrices.

```
import flash.geom.Matrix;

var myMatrix:Matrix = new Matrix(2, 0, 0, 2, 0, 0);
var clonedMatrix:Matrix = new Matrix();
```

```
trace(myMatrix); // (a=2, b=0, c=0, d=2, tx=0, ty=0)
trace(clonedMatrix); // (a=1, b=0, c=0, d=1, tx=0, ty=0)
trace(equals(myMatrix, clonedMatrix)); // false

clonedMatrix = myMatrix.clone();

trace(myMatrix); // (a=2, b=0, c=0, d=2, tx=0, ty=0)
trace(clonedMatrix); // (a=2, b=0, c=0, d=2, tx=0, ty=0)
trace(equals(myMatrix, clonedMatrix)); // true

function equals(m1:Matrix, m2:Matrix):Boolean {
  return m1.toString() == m2.toString();
}
```

concat (método Matrix.concat)

```
public concat(m:Matrix) : Void
```

Concatena una matriz con la matriz actual y combina eficazmente los efectos geométricos de ambas matrices. En términos matemáticos, la concatenación de matrices equivale a combinarlas utilizando la multiplicación de matrices.

Por ejemplo, si la matriz `m1` aplica a un objeto una escala de factor cuatro y la matriz `m2` gira un objeto 1,5707963267949 radianes (`Math.PI/2`), `m1.concat(m2)` transforma `m1` en una matriz que aplica a un objeto una escala cuatro y gira el objeto `Math.PI/2` radianes.

Este método sustituye la matriz original con la concatenada. Si desea concatenar dos matrices sin alterar ninguna de las originales, puede copiar primero la matriz de origen con el método clone() tal como se muestra en la sección de ejemplos.

Disponibilidad: ActionScript 1.0; Flash Player 8

Parámetros

m:`flash.geom.Matrix` - La matriz que se concatenará con la matriz original.

Ejemplo

El ejemplo siguiente crea tres matrices que definen transformaciones para tres clips de película rectangulares. Las primeras dos matrices `rotate45Matrix` y `doubleScaleMatrix` se aplican a dos rectángulos: `rectangleMc_1` y `rectangleMc_2`. La tercera matriz se crea utilizando el método `concat()` en `rotate45Matrix` y `doubleScaleMatrix` para crear `scaleAndRotateMatrix`. Finalmente se aplica esta matriz sobre `rectangleMc_3` para cambiar la escala y girarlo.

```
import flash.geom.Matrix;
import flash.geom.Transform;
```

```
var rectangleMc_0:MovieClip = createRectangle(20, 80, 0x000000);
var rectangleMc_1:MovieClip = createRectangle(20, 80, 0xFF0000);
var rectangleMc_2:MovieClip = createRectangle(20, 80, 0x00FF00);
var rectangleMc_3:MovieClip = createRectangle(20, 80, 0x0000FF);

var rectangleTrans_1:Transform = new Transform(rectangleMc_1);
var rectangleTrans_2:Transform = new Transform(rectangleMc_2);
var rectangleTrans_3:Transform = new Transform(rectangleMc_3);

var rotate45Matrix:Matrix = new Matrix();
rotate45Matrix.rotate(Math.PI/4);
rectangleTrans_1.matrix = rotate45Matrix;
rectangleMc_1._x = 100;
trace(rotate45Matrix.toString()); // (a=0.707106781186548,
  b=0.707106781186547, c=-0.707106781186547, d=0.707106781186548, tx=0,
  ty=0)

var doubleScaleMatrix:Matrix = new Matrix();
doubleScaleMatrix.scale(2, 2);
rectangleTrans_2.matrix = doubleScaleMatrix;
rectangleMc_2._x = 200;
trace(doubleScaleMatrix.toString()); // (a=2, b=0, c=0, d=2, tx=0, ty=0)

var scaleAndRotateMatrix:Matrix = doubleScaleMatrix.clone();
scaleAndRotateMatrix.concat(rotate45Matrix);
rectangleTrans_3.matrix = scaleAndRotateMatrix;
rectangleMc_3._x = 300;
trace(scaleAndRotateMatrix.toString()); // (a=1.4142135623731,
  b=1.41421356237309, c=-1.41421356237309, d=1.4142135623731, tx=0, ty=0)

function createRectangle(width:Number, height:Number,
  color:Number):MovieClip {
  var depth:Number = this.getNextHighestDepth();
  var mc:MovieClip = this.createEmptyMovieClip("mc_" + depth, depth);
  mc.beginFill(color);
  mc.lineTo(0, height);
  mc.lineTo(width, height);
  mc.lineTo(width, 0);
  mc.lineTo(0, 0);
  return mc;
}
```

createBox (método Matrix.createBox)

```
public createBox(scaleX:Number, scaleY:Number, [rotation:Number],
  [tx:Number], [ty:Number]) : Void
```

Incluye parámetros para escala, rotación y traslación. Cuando se aplica a una matriz define sus valores en función de dichos parámetros.

El método `createBox()` permite obtener la misma matriz que si aplicara los métodos `identity()`, `rotate()`, `scale()` y `translate()` de forma consecutiva. Por ejemplo, `mat1.createBox(2,2,Math.PI/5, 100, 100)` tiene el mismo efecto que lo siguiente:

```
import flash.geom.Matrix;

var mat1:Matrix = new Matrix();
mat1.identity();
mat1.rotate(Math.PI/4);
mat1.scale(2,2);
mat1.translate(10,20);
```

Disponibilidad: ActionScript 1.0; Flash Player 8

Parámetros

`scaleX``:Number` - El factor de escala horizontal.

`scaleY``:Number` - El factor de escala vertical.

`rotation``:Number` [opcional] - La cantidad de giro, en radianes. El valor predeterminado es 0.

`tx``:Number` [opcional] - El número de píxeles que se trasladarán (moverán) hacia la derecha a lo largo del eje *x*. El valor predeterminado es 0.

`ty``:Number` [opcional] - El número de píxeles que se trasladarán (moverán) hacia abajo a lo largo del eje *y*. El valor predeterminado es 0.

Ejemplo

El ejemplo siguiente define la escala `scaleX`, `scaleY`, rotación, ubicación *x* y ubicación *y* de `myMatrix` llamando a su método `createBox()`.

```
import flash.geom.Matrix;
import flash.geom.Transform;

var myMatrix:Matrix = new Matrix();
trace(myMatrix.toString()); // (a=1, b=0, c=0, d=1, tx=0, ty=0)

myMatrix.createBox(1, 2, Math.PI/4, 100, 200);
trace(myMatrix.toString()); // (a=0.707106781186548, b=1.41421356237309,
  c=-0.707106781186547, d=1.4142135623731, tx=100, ty=200)
```

```
var rectangleMc:MovieClip = createRectangle(20, 80, 0xFF0000);
var rectangleTrans:Transform = new Transform(rectangleMc);
rectangleTrans.matrix = myMatrix;
```

Véase también

createGradientBox (método Matrix.createGradientBox)

```
public createGradientBox(width:Number, height:Number, [rotation:Number],
  [tx:Number], [ty:Number]) : Void
```

Crea el estilo específico de matriz que el método `MovieClip.beginGradientFill()` espera. La anchura y altura se cambian a un valor de par `scaleX`/`scaleY` y los valores `tx`/`ty` se desplazan con la mitad de la anchura y altura.

Disponibilidad: ActionScript 1.0; Flash Player 8

Parámetros

`width``:Number` - La anchura del cuadro de degradado

`height``:Number` - La altura del cuadro de degradado

`rotation``:Number` [opcional] - La cantidad de giro, en radianes. El valor predeterminado es 0.

`tx``:Number` [opcional] - La distancia en píxeles que se trasladará (moverá) hacia la derecha a lo largo del eje *x*. Este valor se desplazará la mitad del parámetro de anchura. El valor predeterminado es 0.

`ty``:Number` [opcional] - La distancia en píxeles que se trasladará (moverá) hacia abajo a lo largo del eje *y*. Este valor se desplazará la mitad del parámetro de altura. El valor predeterminado es 0.

Ejemplo

El ejemplo siguiente utiliza `myMatrix` como un parámetro para el método `beginGradientFill()` del objeto MovieClip.

```
import flash.geom.Matrix;

var myMatrix:Matrix = new Matrix();
trace(myMatrix.toString()); // (a=1, b=0, c=0, d=1, tx=0, ty=0)

myMatrix.createGradientBox(200, 200, 0, 50, 50);
trace(myMatrix.toString()); // (a=0.1220703125, b=0, c=0, d=0.1220703125,
  tx=150, ty=150)
```

```
var depth:Number = this.getNextHighestDepth();
var mc:MovieClip = this.createEmptyMovieClip("mc_" + depth, depth);
var colors:Array = [0xFF0000, 0x0000FF];
var alphas:Array = [100, 100];
var ratios:Array = [0, 0xFF];
mc.beginGradientFill("linear", colors, alphas, ratios, myMatrix);
mc.lineTo(0, 300);
mc.lineTo(300, 300);
mc.lineTo(300, 0);
mc.lineTo(0, 0);
```

Véase también

beginGradientFill (método MovieClip.beginGradientFill)

d (propiedad Matrix.d)

```
public d : Number
```

El valor de la segunda fila y la segunda columna del objeto Matrix, que afecta a la posición de los píxeles a lo largo del eje *y* cuando se cambia la escala o se gira una imagen.

Disponibilidad: ActionScript 1.0; Flash Player 8

Ejemplo

El ejemplo siguiente crea el objeto Matrix `myMatrix` y define su valor `d`.

```
import flash.geom.Matrix;

var myMatrix:Matrix = new Matrix();
trace(myMatrix.d); // 1

myMatrix.d = 2;
trace(myMatrix.d); // 2
```

deltaTransformPoint (método Matrix.deltaTransformPoint)

```
public deltaTransformPoint(pt:Point) : Point
```

Tras especificar un punto en el espacio de coordenadas previas a la transformación, devuelve las coordenadas de dicho punto después de la transformación. A diferencia de la transformación estándar aplicada con el método transformPoint(), la transformación del método deltaTransformPoint() no tiene en cuenta los parámetros de traslación `tx` y `ty`.

Disponibilidad: ActionScript 1.0; Flash Player 8

Parámetros

pt:flash.geom.Point - Un objeto Point.

Valor devuelto

flash.geom.Point - El nuevo objeto Point.

Ejemplo

El ejemplo siguiente utiliza el método deltaTransformPoint() para crear deltaTransformedPoint a partir de myPoint. En el ejemplo, el método translate() no altera la posición del punto deltaTransformedPoint. Sin embargo, el método scale() no afecta a la posición de ese punto. Aumenta el valor x del punto un factor de tres desde 50 a 150.

```
import flash.geom.Matrix;
import flash.geom.Point;

var myMatrix:Matrix = new Matrix();
trace(myMatrix); // (a=1, b=0, c=0, d=1, tx=0, ty=0)

myMatrix.translate(100, 0);
trace(myMatrix); // (a=1, b=0, c=0, d=1, tx=100, ty=0)

myMatrix.scale(3, 3);
trace(myMatrix); // (a=3, b=0, c=0, d=3, tx=300, ty=0)

var myPoint:Point = new Point(50,0);
trace(myPoint); // (50, 0)

var deltaTransformedPoint:Point = myMatrix.deltaTransformPoint(myPoint);
trace(deltaTransformedPoint); // (150, 0)

var pointMc_0:MovieClip = createRectangle(10, 10, 0xFF0000);
pointMc_0._x = myPoint.x;

var pointMc_1:MovieClip = createRectangle(10, 10, 0x00FF00);
pointMc_1._x = deltaTransformedPoint.x;

function createRectangle(width:Number, height:Number,
  color:Number):MovieClip {
  var depth:Number = this.getNextHighestDepth();
  var mc:MovieClip = this.createEmptyMovieClip("mc_" + depth, depth);
  mc.beginFill(color);
  mc.lineTo(0, height);
  mc.lineTo(width, height);
  mc.lineTo(width, 0);
  mc.lineTo(0, 0);
  return mc;
}
```

identity (método Matrix.identity)

```
public identity() : Void
```

Establece cada propiedad de matriz en un valor que hace que un clip de película o construcción geométrica transformada sea idéntica a la original.

Después de llamar al método `identity()`, la matriz resultante tiene las propiedades siguientes: `a=1, b=0, c=0, d=1, tx=0, ty=0`.

En la notación de matrices, la matriz de identidad tendrá este aspecto:

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Disponibilidad: ActionScript 1.0; Flash Player 8

Ejemplo

El ejemplo siguiente muestra cómo la llamada al método `identity()` convierte el objeto Matrix que realiza la llamada en un objeto Matrix de identidad. El número y los tipos de transformaciones aplicadas previamente al objeto Matrix original no son relevantes. Si se llama a `identity()`, los valores de la matriz se convierten a (a=1, b=0, c=0, d=1, tx=0, ty=0).

```
import flash.geom.Matrix;

var myMatrix:Matrix = new Matrix(2, 0, 0, 2, 0 ,0);
trace(myMatrix.toString()); // (a=2, b=0, c=0, d=2, tx=0, ty=0)

myMatrix.rotate(Math.atan(3/4));
trace(myMatrix.toString()); // (a=1.6, b=1.2, c=-1.2, d=1.6, tx=0, ty=0)

myMatrix.translate(100,200);
trace(myMatrix.toString()); // (a=1.6, b=1.2, c=-1.2, d=1.6, tx=100,
  ty=200)

myMatrix.scale(2, 2);
trace(myMatrix.toString()); // (a=3.2, b=2.4, c=-2.4, d=3.2, tx=200,
  ty=400)

myMatrix.identity();
trace(myMatrix.toString()); // (a=1, b=0, c=0, d=1, tx=0, ty=0)
```

invert (método Matrix.invert)

```
public invert() : Void
```

Realiza la transformación inversa de la matriz original. Puede aplicar la matriz inversa a un objeto para deshacer la transformación realizada al aplicar la matriz original.

Disponibilidad: ActionScript 1.0; Flash Player 8

Ejemplo

El ejemplo siguiente crea `halfScaleMatrix` llamando al método `invert()` de `doubleScaleMatrix` y muestra cómo son matrices inversas, es decir, matrices que deshacen las transformaciones que realiza la otra. El ejemplo muestra esta inversión creando `originalAndInverseMatrix`, que es igual a `noScaleMatrix`.

```
import flash.geom.Matrix;
import flash.geom.Transform;

var rectangleMc_0:MovieClip = createRectangle(20, 80, 0xFF0000);
var rectangleMc_1:MovieClip = createRectangle(20, 80, 0x00FF00);
var rectangleMc_2:MovieClip = createRectangle(20, 80, 0x0000FF);
var rectangleMc_3:MovieClip = createRectangle(20, 80, 0x000000);

var rectangleTrans_0:Transform = new Transform(rectangleMc_0);
var rectangleTrans_1:Transform = new Transform(rectangleMc_1);
var rectangleTrans_2:Transform = new Transform(rectangleMc_2);
var rectangleTrans_3:Transform = new Transform(rectangleMc_3);

var doubleScaleMatrix:Matrix = new Matrix(2, 0, 0, 2, 0, 0);
rectangleTrans_0.matrix = doubleScaleMatrix;
trace(doubleScaleMatrix.toString()); // (a=2, b=0, c=0, d=2, tx=0, ty=0)

var noScaleMatrix:Matrix = new Matrix(1, 0, 0, 1, 0, 0);
rectangleTrans_1.matrix = noScaleMatrix;
rectangleMc_1._x = 100;
trace(noScaleMatrix.toString()); // (a=1, b=0, c=0, d=1, tx=0, ty=0)

var halfScaleMatrix:Matrix = doubleScaleMatrix.clone();
halfScaleMatrix.invert();
rectangleTrans_2.matrix = halfScaleMatrix;
rectangleMc_2._x = 200;
trace(halfScaleMatrix.toString()); // (a=0.5, b=0, c=0, d=0.5, tx=0, ty=0)

var originalAndInverseMatrix:Matrix = doubleScaleMatrix.clone();
originalAndInverseMatrix.concat(halfScaleMatrix);
rectangleTrans_3.matrix = originalAndInverseMatrix;
rectangleMc_3._x = 300;
trace(originalAndInverseMatrix.toString()); // (a=1, b=0, c=0, d=1, tx=0,
  ty=0)
```

```
function createRectangle(width:Number, height:Number,
  color:Number):MovieClip {
  var depth:Number = this.getNextHighestDepth();
  var mc:MovieClip = this.createEmptyMovieClip("mc_" + depth, depth);
  mc.beginFill(color);
  mc.lineTo(0, height);
  mc.lineTo(width, height);
  mc.lineTo(width, 0);
  mc.lineTo(0, 0);
  return mc;
}
```

Matrix, constructor

```
public Matrix([a:Number], [b:Number], [c:Number], [d:Number], [tx:Number],
  [ty:Number])
```

Crea un nuevo objeto Matrix con los parámetros especificados. En la notación de matrices, las propiedades se organizarán de este modo:

$$\begin{bmatrix} a & b & t_x \\ c & d & t_y \\ 0 & 0 & 1 \end{bmatrix}$$

Si no proporciona ningún parámetro para el nuevo constructor `Matrix()`, se crea una "matriz de identidad" con los valores siguientes:

`a = 1`	`b = 0`
`c = 0`	`d = 1`
`tx = 0`	`ty = 0`

En la notación de matrices, la matriz de identidad tendrá este aspecto:

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Disponibilidad: ActionScript 1.0; Flash Player 8

Parámetros

a:Number [opcional] - Valor de la primera fila y la primera columna del nuevo objeto Matrix.

b:Number [opcional] - Valor de la primera fila y la segunda columna del nuevo objeto Matrix.

c:Number [opcional] - Valor de la segunda fila y la primera columna del nuevo objeto Matrix.

d:Number [opcional] - Valor de la segunda fila y la segunda columna del nuevo objeto Matrix.

tx:Number [opcional] - Valor de la tercera fila y la primera columna del nuevo objeto Matrix.

ty:Number [opcional] - Valor de la tercera fila y la segunda columna del nuevo objeto Matrix.

Ejemplo

El ejemplo siguiente crea `matrix_1` no enviando ningún parámetro al constructor `Matrix` y `matrix_2` enviando parámetros a dicho constructor. El objeto Matrix `matrix_1`, que se crea sin parámetros, es una matriz de identidad con los valores (a=1, b=0, c=0, d=1, tx=0, ty=0).

```
import flash.geom.Matrix;

var matrix_1:Matrix = new Matrix();
trace(matrix_1); // (a=1, b=0, c=0, d=1, tx=0, ty=0)

var matrix_2:Matrix = new Matrix(1, 2, 3, 4, 5, 6);
trace(matrix_2); // (a=1, b=2, c=3, d=4, tx=5, ty=6)
```

rotate (método Matrix.rotate)

```
public rotate(angle:Number) : Void
```

Establece los valores en la matriz actual, de forma que pueda utilizarse para aplicar una transformación de rotación.

El método `rotate()` modifica las propiedades `a` y `d` del objeto Matrix. En la notación de matrices aparece del modo siguiente:

$$\begin{bmatrix} \cos(q) & \sin(q) & 0 \\ -\sin(q) & \cos(q) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Disponibilidad: ActionScript 1.0; Flash Player 8

Parámetros

angle:Number - El ángulo de rotación expresado en radianes.

Ejemplo

El ejemplo siguiente muestra cómo el método `rotate()` rota `rectangleMc` 30° en sentido de las agujas del reloj. Al aplicar `myMatrix` a `rectangleMc` se restablece el valor de *_x*, por lo que debe restablecerlo a 100 de forma manual.

```
import flash.geom.Matrix;
import flash.geom.Transform;

var myMatrix:Matrix = new Matrix();
trace(myMatrix.toString()); // (a=1, b=0, c=0, d=1, tx=0, ty=0)

var degrees:Number = 30;
var radians:Number = (degrees/180) Math.PI;
myMatrix.rotate(radians);
trace(myMatrix.toString()); // (a=0.866025403784439, b=0.5, c=-0.5,
  d=0.866025403784439, tx=0, ty=0)

var rectangleMc:MovieClip = createRectangle(20, 80, 0xFF0000);
trace(rectangleMc._x); // 0
rectangleMc._x = 100;
trace(rectangleMc._x); // 100

var rectangleTrans:Transform = new Transform(rectangleMc);
rectangleTrans.matrix = myMatrix;
trace(rectangleMc._x); // 0
rectangleMc._x = 100;
trace(rectangleMc._x); // 100

function createRectangle(width:Number, height:Number,
  color:Number):MovieClip {
  var depth:Number = this.getNextHighestDepth();
  var mc:MovieClip = this.createEmptyMovieClip("mc_" + depth, depth);
  mc.beginFill(color);
  mc.lineTo(0, height);
  mc.lineTo(width, height);
  mc.lineTo(width, 0);
  mc.lineTo(0, 0);
  return mc;
}
```

El ejemplo anterior utiliza la propiedad _x del objeto MovieClip para situar rectangleMc. Normalmente, cuando tenga que posicionar matrices, no se recomienda mezclar técnicas de posicionamiento. El ejemplo anterior escrito con la sintaxis correcta concatenaría una matriz de traslación con myMatrix para cambiar la ubicación horizontal de rectangleMc. En el ejemplo siguiente se incluye una demostración.

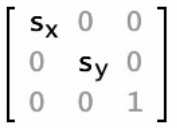
```
import flash.geom.Matrix;
import flash.geom.Transform;

var myMatrix:Matrix = new Matrix();
trace(myMatrix.toString()); // (a=1, b=0, c=0, d=1, tx=0, ty=0)

var degrees:Number = 30;
var radians:Number = (degrees/180) Math.PI;
myMatrix.rotate(radians);
trace(myMatrix.toString()); // (a=0.866025403784439, b=0.5, c=-0.5,
  d=0.866025403784439, tx=0, ty=0)

var translateMatrix:Matrix = new Matrix();
translateMatrix.translate(100, 0);
myMatrix.concat(translateMatrix);
trace(myMatrix.toString()); // (a=0.866025403784439, b=0.5, c=-0.5,
  d=0.866025403784439, tx=100, ty=0)

var rectangleMc:MovieClip = createRectangle(20, 80, 0xFF0000);
trace(rectangleMc._x); // 0
rectangleMc._x = 100;
trace(rectangleMc._x); // 100

var rectangleTrans:Transform = new Transform(rectangleMc);
rectangleTrans.matrix = myMatrix;
trace(rectangleMc._x); // 100

function createRectangle(width:Number, height:Number,
  color:Number):MovieClip {
  var depth:Number = this.getNextHighestDepth();
  var mc:MovieClip = this.createEmptyMovieClip("mc_" + depth, depth);
  mc.beginFill(color);
  mc.lineTo(0, height);
  mc.lineTo(width, height);
  mc.lineTo(width, 0);
  mc.lineTo(0, 0);
  return mc;
}
```

scale (método Matrix.scale)

```
public scale(sx:Number, sy:Number) : Void
```

Modifica una matriz de forma que su aplicación provoque un cambio de tamaño de una imagen. En la imagen redimensionada, se multiplica la ubicación de cada píxel por `sx` en el eje *x* y por `sy` en el eje *y*.

El método `scale()` modifica las propiedades `a` y `d` del objeto Matrix. En la notación de matrices aparece del modo siguiente:



Disponibilidad: ActionScript 1.0; Flash Player 8

Parámetros

sx:`Number` - Un multiplicador utilizado para aplicar una escala al objeto a lo largo del eje *x*.

sy:`Number` - Un multiplicador utilizado para aplicar una escala al objeto a lo largo del eje *y*.

Ejemplo

El ejemplo siguiente utiliza el método `scale()` para cambiar la escala de `myMatrix` según un factor tres horizontalmente y un factor 4 verticalmente.

```
import flash.geom.Matrix;

var myMatrix:Matrix = new Matrix(2, 0, 0, 2, 100, 100);
trace(myMatrix.toString()); // (a=2, b=0, c=0, d=2, tx=100, ty=100)

myMatrix.scale(3, 4);
trace(myMatrix.toString()); // (a=6, b=0, c=0, d=8, tx=300, ty=400)
```

toString (método Matrix.toString)

```
public toString() : String
```

Devuelve un valor de texto con las propiedades del objeto Matrix.

Disponibilidad: ActionScript 1.0; Flash Player 8

Valor devuelto

`String` - Una cadena con los valores de las propiedades del objeto Matrix: `a`, `b`, `c`, `d`, `tx` y `ty`.

Ejemplo

El ejemplo siguiente crea `myMatrix` y convierte sus valores en una cadena con el formato (a=A, b=B, c=C, d=D, tx=TX, ty=TY).

```
import flash.geom.Matrix;

var myMatrix:Matrix = new Matrix();
trace("myMatrix: " + myMatrix.toString()); // (a=1, b=0, c=0, d=1, tx=0,
  ty=0)
```

transformPoint (método Matrix.transformPoint)

```
public transformPoint(pt:Point) : Point
```

Aplica en el punto especificado la transformación geométrica representada por el objeto Matrix.

Disponibilidad: ActionScript 1.0; Flash Player 8

Parámetros

pt`:flash.geom.Point` - El punto (*x*,*y*) que se transformará.

Valor devuelto

`flash.geom.Point` - El nuevo objeto Point.

Ejemplo

El ejemplo siguiente utiliza el método `transformPoint()` para crear `transformedPoint` a partir de `myPoint`. El método `translate()` no tiene ningún efecto sobre la posición de `transformedPoint`. En el ejemplo, `scale()` triplica el valor `x` original de 50 a 150 y `translate()` aumenta `x` en 300 hasta un valor total de 450.

```
import flash.geom.Matrix;
import flash.geom.Point;

var myMatrix:Matrix = new Matrix();
trace(myMatrix); // (a=1, b=0, c=0, d=1, tx=0, ty=0)

myMatrix.translate(100, 0);
trace(myMatrix); // (a=1, b=0, c=0, d=1, tx=100, ty=0)

myMatrix.scale(3, 3);
trace(myMatrix); // (a=3, b=0, c=0, d=3, tx=300, ty=0)

var myPoint:Point = new Point(50,0);
trace(myPoint); // (50, 0)
```

```
var transformedPoint:Point = myMatrix.transformPoint(myPoint);
trace(transformedPoint); // (450, 0)

var pointMc_0:MovieClip = createRectangle(10, 10, 0xFF0000);
pointMc_0._x = myPoint.x;

var pointMc_1:MovieClip = createRectangle(10, 10, 0x00FF00);
pointMc_1._x = transformedPoint.x;

function createRectangle(width:Number, height:Number,
  color:Number):MovieClip {
  var depth:Number = this.getNextHighestDepth();
  var mc:MovieClip = this.createEmptyMovieClip("mc_" + depth, depth);
  mc.beginFill(color);
  mc.lineTo(0, height);
  mc.lineTo(width, height);
  mc.lineTo(width, 0);
  mc.lineTo(0, 0);
  return mc;
}
```

translate (método Matrix.translate)

```
public translate(tx:Number, ty:Number) : Void
```

Modifica un objeto Matrix de forma que su transformación provoque el desplazamiento de un objeto por los ejes *x* e *y*.

Disponibilidad: ActionScript 1.0; Flash Player 8

Parámetros

tx:Number - El movimiento total hacia la derecha a lo largo del eje *x*, en píxeles.

ty:Number - El movimiento total hacia abajo a lo largo del eje *y*, en píxeles.

Ejemplo

El ejemplo siguiente utiliza el método `translate()` para colocar `rectangleMc` x:100 e y:50. El método `translate()` afecta a las propiedades de traslación `tx` y `ty`, pero no afecta a las propiedades `a`, `b`, `c` o `d`.

```
import flash.geom.Matrix;

var myMatrix:Matrix = new Matrix(2, 0, 0, 2, 100, 100);
trace(myMatrix.toString()); // (a=2, b=0, c=0, d=2, tx=100, ty=100)

myMatrix.translate(100, 50);
trace(myMatrix.toString()); // (a=2, b=0, c=0, d=2, tx=200, ty=150)
```

tx (propiedad Matrix.tx)

```
public tx : Number
```

La distancia que se trasladará cada punto a lo largo del eje *x*. Representa el valor de la tercera fila y la primera columna del nuevo objeto Matrix.

Disponibilidad: ActionScript 1.0; Flash Player 8

Ejemplo

El ejemplo siguiente crea el objeto Matrix `myMatrix` y define su valor `tx`.

```
import flash.geom.Matrix;

var myMatrix:Matrix = new Matrix();
trace(myMatrix.tx); // 0

myMatrix.tx = 50; // 50
trace(myMatrix.tx);
```

ty (propiedad Matrix.ty)

```
public ty : Number
```

La distancia que se trasladará cada punto a lo largo del eje *y*. Representa el valor de la tercera fila y la segunda columna del nuevo objeto Matrix.

Disponibilidad: ActionScript 1.0; Flash Player 8

Ejemplo

El ejemplo siguiente crea el objeto Matrix `myMatrix` y define su valor `ty`.

```
import flash.geom.Matrix;

var myMatrix:Matrix = new Matrix();
trace(myMatrix.ty); // 0

myMatrix.ty = 50;
trace(myMatrix.ty); // 50
```

Microphone

```
Object
  |
  +-Microphone
```

```
public class Microphone
extends Object
```

La clase Microphone le permite capturar sonido de un micrófono conectado al equipo que está ejecutando Flash Player.

La clase Microphone está pensada principalmente para su uso con Flash Communication Server, aunque puede utilizarse de forma limitada sin el servidor (por ejemplo, para transmitir sonido del micrófono a través de los altavoces del sistema local).

Precaución: Flash Player muestra un cuadro de diálogo de Privacidad que permite al usuario decidir si autoriza o deniega el acceso al micrófono. Asegúrese de que el tamaño del escenario sea de al menos 215 x 138 píxeles; este es el tamaño mínimo que exige Flash para mostrar el cuadro de diálogo.

Los usuarios con permisos de Usuario y Administrador también pueden desactivar el acceso de micrófono para cada sitio o de forma global.

Para crear o hacer referencia a un objeto Microphone, utilice el método `Microphone.get()`.

Disponibilidad: ActionScript 1.0; Flash Player 6

Resumen de propiedades

Modificadores	Propiedad	Descripción
	`activityLevel:Number` [read-only]	Valor numérico que especifica la cantidad de sonido que está detectando el micrófono.
	`gain:Number` [read-only]	Valor con el que el micrófono debe reforzar la señal.
	`index:Number` [read-only]	Entero basado en cero que especifica el índice del micrófono, como se refleja en la matriz devuelta por `Microphone.names`.
	`muted:Boolean` [read-only]	Valor booleano que especifica si el usuario ha denegado el acceso al micrófono (`true`) o lo ha autorizado (`false`).
	`name:String` [read-only]	Cadena que especifica el nombre del dispositivo de captura de sonido, tal y como lo ha devuelto el hardware de captura de sonido.

Modificadores	Propiedad	Descripción
`static`	`names:Array` [read-only]	Recupera una matriz de cadenas que reflejan los nombres de todos los dispositivos de captura de sonido disponibles sin mostrar el panel Parámetros de privacidad de Flash Player.
	`rate:Number` [read-only]	Frecuencia a la que está capturando el sonido el micrófono, expresada en kHz.
	`silenceLevel:Number` [read-only]	Un entero que especifica el nivel de sonido necesario para activar el micrófono e invocar `Microphone.onActivity(true)`.
	`silenceTimeOut:Number` [read-only]	Valor numérico que representa los milisegundos que transcurren entre el momento en que el micrófono deja de detectar sonido y la invocación de `Microphone.onActivity(false)`.
	`useEchoSuppression:Boolean`	Propiedad (sólo lectura); valor booleano `true` si está activada la supresión de eco; `false` en caso contrario.

Propiedades heredadas de la clase Object

`constructor (propiedad Object.constructor), __proto__ (propiedad Object.__proto__), prototype (propiedad Object.prototype), __resolve (propiedad Object.__resolve)`

Resumen de eventos

Evento	Descripción
`onActivity = function(active:Boolean) {}`	Se invoca cuando el micrófono comienza o deja de detectar sonido.
`onStatus = function(infoObject:Object) {}`	Se invoca cuando el usuario autoriza o deniega el acceso al micrófono.

Resumen de métodos

Modificadores	Firma	Descripción
`static`	`get([index:Number]) : Microphone`	Devuelve una referencia a un objeto Microphone para capturar sonido.
	`setGain(gain:Number) : Void`	Establece la ganancia del micrófono (es decir, el valor por el que el micrófono debe multiplicar la señal antes de transmitirla).

Modificadores	Firma	Descripción
	`setRate(rate:Number) : Void`	Establece la frecuencia, expresada en kHz, con la que el micrófono debe capturar el sonido.
	`setSilenceLevel(silenceLevel:Number, [timeOut:Number]) : Void`	Establece el nivel mínimo de la entrada que debe considerarse como sonido y (opcionalmente) el tiempo de silencio que indica que efectivamente ha comenzado el silencio.
	`setUseEchoSuppression(useEchoSuppression:Boolean) : Void`	Especifica si debe utilizarse la función de supresión de eco del códec de audio.

Métodos heredados de la clase Object

`addProperty (método Object.addProperty), hasOwnProperty (método Object.hasOwnProperty), isPropertyEnumerable (método Object.isPropertyEnumerable), isPrototypeOf (método Object.isPrototypeOf), registerClass (método Object.registerClass), toString (método Object.toString), unwatch (método Object.unwatch), valueOf (método Object.valueOf), watch (método Object.watch)`

activityLevel (propiedad Microphone.activityLevel)

`public activityLevel : Number [read-only]`

Valor numérico que especifica la cantidad de sonido que está detectando el micrófono. El rango de valores válidos es de 0 (no se detecta sonido) a 100 (se detecta un sonido alto). El valor de esta propiedad permite determinar un valor adecuado para el método `Microphone.setSilenceLevel()`.

Si el micrófono está disponible pero no se está utilizando todavía porque no se ha llamado a `Microphone.get()`, esta propiedad se establece con el valor -1.

Disponibilidad: ActionScript 1.0; Flash Player 6

Ejemplo

El ejemplo siguiente muestra el nivel de actividad del micrófono actual en una instancia ProgressBar denominada `activityLevel_pb`.

```
var activityLevel_pb:mx.controls.ProgressBar;
activityLevel_pb.mode = "manual";
activityLevel_pb.label = "Activity Level: %3%%";
activityLevel_pb.setStyle("themeColor", "0xFF0000");
this.createEmptyMovieClip("sound_mc", this.getNextHighestDepth());
var active_mic:Microphone = Microphone.get();
sound_mc.attachAudio(active_mic);
```

```
this.onEnterFrame = function() {
  activityLevel_pb.setProgress(active_mic.activityLevel, 100);
};
active_mic.onActivity = function(active:Boolean) {
  if (active) {
  var haloTheme_str:String = "haloGreen";
  } else {
  var haloTheme_str:String = "0xFF0000";
  }
  activityLevel_pb.setStyle("themeColor", haloTheme_str);
};
```

El método `MovieClip.getNextHighestDepth()` utilizado en este ejemplo necesita Flash Player 7 o una versión posterior. Si el archivo SWF incluye un componente de la versión 2, utilice la clase DepthManager de los componentes de la versión 2 en lugar del método `MovieClip.getNextHighestDepth()`.

Véase también

`get (propiedad Microphone.get)`, `setSilenceLevel (método Microphone.setSilenceLevel)`, `setGain (método Microphone.setGain)`

gain (propiedad Microphone.gain)

```
public gain : Number [read-only]
```

Valor con el que el micrófono debe reforzar la señal. Los valores válidos son de 0 a 100. El valor predeterminado es 50.

Disponibilidad: ActionScript 1.0; Flash Player 6

Ejemplo

El ejemplo siguiente utiliza una instancia de ProgressBar denominada `gain_pb` y una instancia NumericStepper denominada `gain_nstep` para definir el valor de ganancia del micrófono.

```
this.createEmptyMovieClip("sound_mc", this.getNextHighestDepth());
var active_mic:Microphone = Microphone.get();
sound_mc.attachAudio(active_mic);

gain_pb.label = "Gain: %3";
gain_pb.mode = "manual";
gain_pb.setProgress(active_mic.gain, 100);
gain_nstep.value = active_mic.gain;

function changeGain() {
  active_mic.setGain(gain_nstep.value);
  gain_pb.setProgress(active_mic.gain, 100);
```

```
}
gain_nstep.addEventListener("change", changeGain);
```

El método `MovieClip.getNextHighestDepth()` utilizado en este ejemplo necesita Flash Player 7 o una versión posterior. Si el archivo SWF incluye un componente de la versión 2, utilice la clase DepthManager de los componentes de la versión 2 en lugar del método `MovieClip.getNextHighestDepth()`.

Véase también

`setGain (método Microphone.setGain)`

get (propiedad Microphone.get)

```
public static get([index:Number]) : Microphone
```

Devuelve una referencia a un objeto Microphone para capturar sonido. Para comenzar a capturar audio, deberá asociar el objeto Microphone a un objeto MovieClip (véase `MovieClip.attachAudio()`.

A diferencia de los objetos que cree utilizando el constructor `new`, varias llamadas a `Microphone.get()` hacen referencia al mismo micrófono. Por consiguiente, si el script contiene las líneas `mic1 = Microphone.get()` y `mic2 = Microphone.get()`, tanto `mic1` como `mic2` harán referencia al mismo micrófono (predeterminado).

En general, no deberá pasar un valor para *`index`* ; simplemente utilice `Microphone.get()` para devolver una referencia al micrófono predeterminado. A través del panel de configuración del micrófono (descrito posteriormente en esta sección), el usuario puede especificar el micrófono que debe utilizar Flash. Si pasa un valor para *`index`* , es posible que esté intentando hacer referencia a un micrófono diferente al que desea el usuario. Podría utilizar *`index`* en casos poco frecuentes (por ejemplo, si la aplicación está capturando audio de dos micrófonos a la vez.

Cuando un archivo SWF intenta acceder al micrófono devuelto por el método `Microphone.get()` (por ejemplo, al enviar `MovieClip.attachAudio()`), Flash Player muestra un cuadro de diálogo Privacidad que permite al usuario autorizar o denegar el acceso al micrófono. (Asegúrese de que el tamaño del escenario sea de al menos 215 x 138 píxeles; este es el tamaño mínimo que exige Flash para mostrar el cuadro de diálogo.)

Cuando el usuario responde a este cuadro de diálogo, el controlador de eventos `Microphone.onStatus` devuelve un objeto de información que indica la respuesta del usuario. Para determinar si el usuario ha denegado o autorizado el acceso al micrófono sin procesar este controlador de eventos, utilice `Microphone.muted`.

El usuario también puede especificar una configuración de privacidad permanente para un dominio concreto haciendo clic con el botón derecho del ratón (Windows) o haciendo clic con la tecla Control presionada (Macintosh) mientras se reproduce un archivo SWF, seleccionando Configuración, abriendo el panel Privacidad y seleccionando Recordar.

No puede utilizar ActionScript para establecer el valor Allow (autorizar) o Deny (denegar) para un usuario, pero puede mostrar el panel Privacidad para el usuario utilizando `System.showSettings(0)`. Si el usuario selecciona Recordar, Flash Player ya no mostrará el cuadro de diálogo Privacidad para archivos SWF de este dominio.

Si `Microphone.get()` devuelve el valor `null` (nulo), ello indicará que otra aplicación utiliza el micrófono o que no hay micrófonos instalados en el sistema. Para comprobar si hay micrófonos instalados, utilice `Microphones.names.length`. Para ver el panel de configuración del micrófono de Flash Player, en el que el usuario puede elegir el micrófono al que hará referencia `Microphone.get()`, utilice `System.showSettings(2)`.

Disponibilidad: ActionScript 1.0; Flash Player 6

Parámetros

`index``:Number` [opcional] - Entero basado en cero que especifica el micrófono que debe utilizarse, como se determina a partir de la la matriz contenida en `Microphone.names`. Para utilizar el micrófono predeterminado (recomendado para la mayoría de las aplicaciones), omita este parámetro.

Valor devuelto

`Microphone` -

- Si no se especifica *`index`* , este método devuelve una referencia al micrófono predeterminado o, si no está disponible, al primer micrófono que lo esté. Si no hay micrófonos disponibles o instalados, el método devuelve el valor `null` (nulo).
- Si se especifica *`index`* , este método devuelve una referencia al micrófono solicitado o `null` si no está disponible.

Ejemplo

El ejemplo siguiente permite especificar el micrófono predeterminado, después captura audio y lo reproduce localmente. Para evitar problemas de acoplamiento, se recomienda realizar la prueba de este código con auriculares.

```
this.createEmptyMovieClip("sound_mc", this.getNextHighestDepth());
System.showSettings(2);
var active_mic:Microphone = Microphone.get();
sound_mc.attachAudio(active_mic);
```

El método `MovieClip.getNextHighestDepth()` utilizado en este ejemplo necesita Flash Player 7 o una versión posterior. Si el archivo SWF incluye un componente de la versión 2, utilice la clase DepthManager de los componentes de la versión 2 en lugar del método `MovieClip.getNextHighestDepth()`.

Véase también

`get (propiedad Microphone.get)`, `index (propiedad Microphone.index)`, `muted (propiedad Microphone.muted)`, `names (propiedad Microphone.names)`, `onStatus (controlador Microphone.onStatus)`, `attachAudio (método MovieClip.attachAudio)`, `showSettings (método System.showSettings)`

index (propiedad Microphone.index)

```
public index : Number [read-only]
```

Entero basado en cero que especifica el índice del micrófono, como se refleja en la matriz devuelta por `Microphone.names`.

Disponibilidad: ActionScript 1.0; Flash Player 6

Ejemplo

El ejemplo siguiente muestra los nombres de los dispositivos de captura de sonido disponibles en el equipo del usuario en una instancia ComboBox denominada `mic_cb`. Una instancia del componente Label denominada `mic_lbl` muestra el micrófono index. Puede utilizar la instancia ComboBox para cambiar entre dispositivos.

```
var mic_lbl:mx.controls.Label;
var mic_cb:mx.controls.ComboBox;

this.createEmptyMovieClip("sound_mc", this.getNextHighestDepth());
var active_mic:Microphone = Microphone.get();
sound_mc.attachAudio(active_mic);
mic_lbl.text = "["+active_mic.index+"] "+active_mic.name;
mic_cb.dataProvider = Microphone.names;
mic_cb.selectedIndex = active_mic.index;

var cbListener:Object = new Object();
cbListener.change = function(evt:Object) {
  active_mic = Microphone.get(evt.target.selectedIndex);
  sound_mc.attachAudio(active_mic);
  mic_lbl.text = "["+active_mic.index+"] "+active_mic.name;
};
mic_cb.addEventListener("change", cbListener);
```

El método `MovieClip.getNextHighestDepth()` utilizado en este ejemplo necesita Flash Player 7 o una versión posterior. Si el archivo SWF incluye un componente de la versión 2, utilice la clase DepthManager de los componentes de la versión 2 en lugar del método `MovieClip.getNextHighestDepth()`.

Véase también

`get (propiedad Microphone.get)`, `names (propiedad Microphone.names)`

muted (propiedad Microphone.muted)

```
public muted : Boolean [read-only]
```

Valor booleano que especifica si el usuario ha denegado el acceso al micrófono (`true`) o lo ha autorizado (`false`). Cuando este valor cambia, se invoca `Microphone.onStatus`. Para más información, consulte `Microphone.get()`.

Disponibilidad: ActionScript 1.0; Flash Player 6

Ejemplo

Este ejemplo obtiene el micrófono predeterminado y comprueba si está sin sonido.

```
var active_mic:Microphone = Microphone.get();
trace(active_mic.muted);
```

Véase también

`get (propiedad Microphone.get)`, `onStatus (controlador Microphone.onStatus)`

name (propiedad Microphone.name)

```
public name : String [read-only]
```

Cadena que especifica el nombre del dispositivo de captura de sonido, tal y como lo ha devuelto el hardware de captura de sonido.

Disponibilidad: ActionScript 1.0; Flash Player 6

Ejemplo

El ejemplo siguiente muestra información sobre el dispositivo o dispositivos de captura de sonido del sistema del usuario, incluido una matriz de nombres y el dispositivo predeterminado.

```
var status_ta:mx.controls.TextArea;
status_ta.html = false;
status_ta.setStyle("fontSize", 9);
var microphone_array:Array = Microphone.names;
```

```
var active_mic:Microphone = Microphone.get();
status_ta.text = "The default device is: "+active_mic.name+newline+newline;
status_ta.text += "You have "+microphone_array.length+" device(s)
  installed."+newline+newline;
for (var i = 0; i<microphone_array.length; i++) {
  status_ta.text += "["+i+"] "+microphone_array[i]+newline;
}
```

Véase también

`get (propiedad Microphone.get)`, `names (propiedad Microphone.names)`

names (propiedad Microphone.names)

```
public static names : Array [read-only]
```

Recupera una matriz de cadenas que reflejan los nombres de todos los dispositivos de captura de sonido disponibles sin mostrar el panel Parámetros de privacidad de Flash Player. Esta matriz se comporta de la misma forma que cualquier otra matriz ActionScript y proporciona implícitamente el índice basado en cero de cada dispositivo de captura de sonido y el número de dispositivos de captura de sonido existentes en el sistema (mediante `Microphone.names.length`. Para más información, consulte la entrada de la clase de matriz Microphone.names.

La llamada a `Microphone.names` exige un examen amplio del hardware y puede tardar varios segundos en crear la matriz. En la mayoría de los casos, podrá utilizar simplemente el micrófono predeterminado.

Disponibilidad: ActionScript 1.0; Flash Player 6

Ejemplo

El ejemplo siguiente muestra información sobre el dispositivo o dispositivos de captura de sonido del sistema del usuario, incluido una matriz de nombres y el dispositivo predeterminado.

```
var status_ta:mx.controls.TextArea;
status_ta.html = false;
status_ta.setStyle("fontSize", 9);
var microphone_array:Array = Microphone.names;
var active_mic:Microphone = Microphone.get();
status_ta.text = "The default device is: "+active_mic.name+newline+newline;
status_ta.text += "You have "+microphone_array.length+" device(s)
  installed."+newline+newline;
for (var i = 0; i<microphone_array.length; i++) {
  status_ta.text += "["+i+"] "+microphone_array[i]+newline;
}
```

Por ejemplo, se puede mostrar la siguiente información:

```
The default device is: Logitech USB Headset
You have 2 device(s) installed.
[0] Logitech USB Headset
[1] YAMAHA AC-XG WDM Audio
```

Véase también

`name (propiedad Microphone.name)`, `get (propiedad Microphone.get)`

onActivity (controlador Microphone.onActivity)

```
onActivity = function(active:Boolean) {}
```

Se invoca cuando el micrófono comienza o deja de detectar sonido. Si desea responder a este controlador de eventos, deberá crear una función que procese su valor de actividad.

Para especificar la cantidad de sonido necesario para invocar `Microphone.onActivity(true)` y el tiempo que debe transcurrir sin que exista sonido para que se invoque `Microphone.onActivity(false)`, utilice `Microphone.setSilenceLevel()`.

Disponibilidad: ActionScript 1.0; Flash Player 6

Parámetros

`active``:Boolean` - Un valor booleano definido en true cuando el micrófono comienza detectar sonido y en false cuando deja de hacerlo.

Ejemplo

El ejemplo siguiente muestra el nivel de actividad en una instancia ProgressBar denominada `activityLevel_pb`. Cuando el micrófono detecte sonido, invocará la función `onActivity` que modifica la instancia ProgressBar.

```
var activityLevel_pb:mx.controls.ProgressBar;
activityLevel_pb.mode = "manual";
activityLevel_pb.label = "Activity Level: %3%%";
this.createEmptyMovieClip("sound_mc", this.getNextHighestDepth());
var active_mic:Microphone = Microphone.get();
sound_mc.attachAudio(active_mic);
active_mic.onActivity = function(active:Boolean) {
  if (active) {
    activityLevel_pb.indeterminate = false;
    activityLevel_pb.label = "Activity Level: %3%%";
  } else {
    activityLevel_pb.indeterminate = true;
    activityLevel_pb.label = "Activity Level: (inactive)";
  }
```

```
};
this.onEnterFrame = function() {
  activityLevel_pb.setProgress(active_mic.activityLevel, 100);
};
```

El método `MovieClip.getNextHighestDepth()` utilizado en este ejemplo necesita Flash Player 7 o una versión posterior. Si el archivo SWF incluye un componente de la versión 2, utilice la clase DepthManager de los componentes de la versión 2 en lugar del método `MovieClip.getNextHighestDepth()`.

Véase también

`setSilenceLevel (método Microphone.setSilenceLevel)`

onStatus (controlador Microphone.onStatus)

```
onStatus = function(infoObject:Object) {}
```

Se invoca cuando el usuario autoriza o deniega el acceso al micrófono. Si desea responder a este controlador de eventos, deberá crear una función que procese el objeto de información generado por el micrófono.

Cuando un archivo SWF intenta acceder al micrófono, Flash Player muestra un cuadro de diálogo Privacidad que permite al usuario autorizar o denegar el acceso.

- Si el usuario autoriza el acceso, la propiedad Microphone.muted se establece como false y se invoca su controlador de eventos con un objeto de información cuya propiedad de código es "Microphone.Unmuted" y cuya propiedad de nivel es "Status".
- Si el usuario deniega el acceso, la propiedad Microphone.muted se establece como true y se invoca su controlador de eventos con un objeto de información cuya propiedad de código es "Microphone.Muted" y cuya propiedad de nivel es "Status".

Para determinar si el usuario ha denegado o autorizado el acceso al micrófono sin procesar este controlador de eventos, utilice `Microphone.muted`.

Nota: Si el usuario opta por permitir o denegar permanentemente el acceso a todos los archivos SWF de un determinado dominio, no se invocará este método para archivos SWF de dicho dominio a no ser que el usuario posteriormente cambie la configuración de privacidad.

Para más información, consulte `Microphone.get()`.

Disponibilidad: ActionScript 1.0; Flash Player 6

Parámetros

`infoObject``:Object` - Un parámetro definido de acuerdo con el mensaje de estado.

Ejemplo

El ejemplo siguiente muestra el cuadro de diálogo de Privacidad que permite al usuario decidir si autoriza o deniega el acceso al micrófono cuando se haga clic en un hipervínculo . Si el usuario decide denegar el acceso, *muted* aparecerá con grandes letras rojas. Si se permite el acceso al micrófono, el usuario no verá este texto.

```
this.createTextField("muted_txt", this.getNextHighestDepth(), 10, 10, 100,
  22);
muted_txt.autoSize = true;
muted_txt.html = true;
muted_txt.selectable = false;
muted_txt.htmlText = "<a href=\"asfunction:System.showSettings\"><u>Click
  Here</u></a> to Allow/Deny access.";
this.createEmptyMovieClip("sound_mc", this.getNextHighestDepth());
var active_mic:Microphone = Microphone.get();
sound_mc.attachAudio(active_mic);
active_mic.onStatus = function(infoObj:Object) {
  status_txt._visible = active_mic.muted;
  muted_txt.htmlText = "Status: <a
  href=\"asfunction:System.showSettings\"><u>"+infoObj.code+"</u></a>";
};
this.createTextField("status_txt", this.getNextHighestDepth(), 0, 0, 100,
  22);
status_txt.html = true;
status_txt.autoSize = true;
status_txt.htmlText = "<font size='72' color='#FF0000'>muted</font>";
status_txt._x = (Stage.width-status_txt._width)/2;
status_txt._y = (Stage.height-status_txt._height)/2;
status_txt._visible = active_mic.muted;
```

El método `MovieClip.getNextHighestDepth()` utilizado en este ejemplo necesita Flash Player 7 o una versión posterior. Si el archivo SWF incluye un componente de la versión 2, utilice la clase DepthManager de los componentes de la versión 2 en lugar del método `MovieClip.getNextHighestDepth()`.

Véase también

`get (propiedad Microphone.get)`, `muted (propiedad Microphone.muted)`, `showSettings (método System.showSettings)`, `onStatus (controlador System.onStatus)`

rate (propiedad Microphone.rate)

`public rate : Number [read-only]`

Frecuencia a la que está capturando el sonido el micrófono, expresada en kHz. El valor predeterminado es 8 kHz si su dispositivo de captura de sonido admite este valor. En caso contrario, el valor predeterminado será el siguiente nivel de captura disponible por encima de 8 kHz que admita su dispositivo de captura de sonido, generalmente 11 kHz.

Para definir este valor, utilice `Microphone.setRate()`.

Disponibilidad: ActionScript 1.0; Flash Player 6

Ejemplo

En el código siguiente se utiliza una instancia ComboBox, denominada `rate_cb` para cambiar la frecuencia con la que el micrófono captura del sonido. La frecuencia actual se muestra en una instancia Label denominada `rate_lbl`.

```
this.createEmptyMovieClip("sound_mc", this.getNextHighestDepth());
var active_mic:Microphone = Microphone.get();
sound_mc.attachAudio(active_mic);
var rate_array:Array = new Array(5, 8, 11, 22, 44);
rate_cb.dataProvider = rate_array;
rate_cb.labelFunction = function(item:Object) {
  return (item+" kHz");
};
for (var i = 0; i<rate_array.length; i++) {
  if (rate_cb.getItemAt(i) == active_mic.rate) {
  rate_cb.selectedIndex = i;
  break;
  }
}
function changeRate() {
  active_mic.setRate(rate_cb.selectedItem);
  rate_lbl.text = "Current rate: "+active_mic.rate+" kHz";
}
rate_cb.addEventListener("change", changeRate);
rate_lbl.text = "Current rate: "+active_mic.rate+" kHz";
```

El método `MovieClip.getNextHighestDepth()` utilizado en este ejemplo necesita Flash Player 7 o una versión posterior. Si el archivo SWF incluye un componente de la versión 2, utilice la clase DepthManager de los componentes de la versión 2 en lugar del método `MovieClip.getNextHighestDepth()`.

Véase también

`setRate (método Microphone.setRate)`

setGain (método Microphone.setGain)

```
public setGain(gain:Number) : Void
```

Establece la ganancia del micrófono (es decir, el valor por el que el micrófono debe multiplicar la señal antes de transmitirla). El valor 0 indica a Flash que multiplique por 0, lo que hace que el micrófono no transmita ningún sonido.

Esta configuración puede concebirse como el control de volumen de un equipo de música: 0 equivale a ningún volumen y 50 a un volumen normal; los números situados por debajo de 50 especifican un volumen inferior al normal, mientras que los números situados por encima de 50 especifican un volumen superior al normal.

Disponibilidad: ActionScript 1.0; Flash Player 6

Parámetros

gain:Number - Un entero que indica un valor con el que el micrófono debe reforzar la señal. Los valores válidos están comprendidos entre 0 y 100. El valor predeterminado es 50; sin embargo, es posible cambiar este valor en el panel Parámetros del micrófono de Flash Player.

Ejemplo

El ejemplo siguiente utiliza una instancia de ProgressBar denominada `gain_pb` y una instancia NumericStepper denominada `gain_nstep` para definir el valor de ganancia del micrófono.

```
this.createEmptyMovieClip("sound_mc", this.getNextHighestDepth());
var active_mic:Microphone = Microphone.get();
sound_mc.attachAudio(active_mic);

gain_pb.label = "Gain: %3";
gain_pb.mode = "manual";
gain_pb.setProgress(active_mic.gain, 100);
gain_nstep.value = active_mic.gain;

function changeGain() {
  active_mic.setGain(gain_nstep.value);
  gain_pb.setProgress(active_mic.gain, 100);
}
gain_nstep.addEventListener("change", changeGain);
```

El método `MovieClip.getNextHighestDepth()` utilizado en este ejemplo necesita Flash Player 7 o una versión posterior. Si el archivo SWF incluye un componente de la versión 2, utilice la clase DepthManager de los componentes de la versión 2 en lugar del método `MovieClip.getNextHighestDepth()`.

Véase también

gain (propiedad Microphone.gain), setUseEchoSuppression (método Microphone.setUseEchoSuppression)

setRate (método Microphone.setRate)

```
public setRate(rate:Number) : Void
```

Establece la frecuencia, expresada en kHz, con la que el micrófono debe capturar el sonido.

Disponibilidad: ActionScript 1.0; Flash Player 6

Parámetros

rate:Number - Frecuencia a la que el micrófono debe capturar el sonido, expresada en kHz. Los valores aceptables son 5, 8, 11, 22 y 44. El valor predeterminado es 8 kHz si su dispositivo de captura de sonido admite este valor. En caso contrario, el valor predeterminado será el siguiente nivel de captura disponible por encima de 8 kHz que admita su dispositivo de captura de sonido, generalmente 11 kHz.

Ejemplo

El ejemplo siguiente define la frecuencia del micrófono con la preferencia del usuario (que se ha asignado a la variable `userRate`), si es uno de los valores siguientes: 5, 8, 11, 22 o 44. Si no es uno de estos valores, el valor se redondeará al valor aceptable más cercano que admita el dispositivo de captura de sonido.

```
active_mic.setRate(userRate);
```

En el ejemplo siguiente se utiliza una instancia ComboBox, denominada `rate_cb` para cambiar la frecuencia con la que el micrófono captura el sonido. La frecuencia actual se muestra en una instancia Label denominada `rate_lbl`.

```
this.createEmptyMovieClip("sound_mc", this.getNextHighestDepth());
var active_mic:Microphone = Microphone.get();
sound_mc.attachAudio(active_mic);
var rate_array:Array = new Array(5, 8, 11, 22, 44);
rate_cb.dataProvider = rate_array;
rate_cb.labelFunction = function(item:Object) {
  return (item+" kHz");
};
for (var i = 0; i<rate_array.length; i++) {
  if (rate_cb.getItemAt(i) == active_mic.rate) {
  rate_cb.selectedIndex = i;
  break;
  }
}
function changeRate() {
```

```
  active_mic.setRate(rate_cb.selectedItem);
  rate_lbl.text = "Current rate: "+active_mic.rate+" kHz";
}
rate_cb.addEventListener("change", changeRate);
rate_lbl.text = "Current rate: "+active_mic.rate+" kHz";
```

El método `MovieClip.getNextHighestDepth()` utilizado en este ejemplo necesita Flash Player 7 o una versión posterior. Si el archivo SWF incluye un componente de la versión 2, utilice la clase DepthManager de los componentes de la versión 2 en lugar del método `MovieClip.getNextHighestDepth()`.

Véase también

`rate (propiedad Microphone.rate)`

setSilenceLevel (método Microphone.setSilenceLevel)

```
public setSilenceLevel(silenceLevel:Number, [timeOut:Number]) : Void
```

Establece el nivel mínimo de la entrada que debe considerarse como sonido y (opcionalmente) el tiempo de silencio que indica que efectivamente ha comenzado el silencio.

- Para impedir que el micrófono detecte sonido, pase el valor 100 para *`level`*; `Microphone.onActivity` nunca se invocará.
- Para determinar la cantidad de sonido que está detectando actualmente el micrófono, utilice `Microphone.activityLevel`.

La detección de actividad es la capacidad de detectar cuándo los niveles de audio sugieren que una persona está hablando. Cuando no hay nadie hablando, puede ahorrarse ancho de banda porque no es necesario enviar el flujo de audio asociado. Esta información también se utiliza como información para que los usuarios sepan que ellos (u otras personas) están en silencio.

Los valores de silencio se corresponden directamente con los valores de actividad. El silencio total tiene un valor de actividad 0. El ruido alto y constante (el máximo que puede registrarse con la configuración de ganancia actual) tiene un valor de actividad 100. Una vez ajustada adecuadamente la ganancia, el valor de actividad es inferior al valor de silencio cuando no está hablando; cuando está hablando, el valor de actividad supera el valor de silencio.

Este método tiene una finalidad similar a la de `Camera.setMotionLevel()`; ambos métodos se utilizan para especificar cuándo debe invocarse el controlador de eventos `onActivity`. No obstante, estos métodos tienen efectos muy distintos en la publicación de flujos:

- `Camera.setMotionLevel()` está diseñado para detectar movimiento y no afecta al uso del ancho de banda. Aunque un flujo de vídeo no detecte movimiento, el vídeo continúa enviándose.

- `Microphone.setSilenceLevel()` está diseñado para optimizar el ancho de banda. Cuando se considera que un flujo de audio está en silencio, no se envían datos de audio. Por el contrario, se envía un único mensaje que indica que el silencio ha comenzado.

Disponibilidad: ActionScript 1.0; Flash Player 6

Parámetros

`silenceLevel``:Number` - Entero que especifica el nivel de sonido necesario para activar el micrófono e invocar `Microphone.onActivity(true)`. El rango de valores admitidos va de 0 a 100. El valor predeterminado es 10.

`timeOut``:Number` [opcional] - Un entero que especifica el número de milisegundos que deben transcurrir sin que exista actividad para que Flash considere que el sonido se ha detenido e invoque `Microphone.onActivity(false)`. El valor predeterminado es 2000 (2 segundos).

Ejemplo

El ejemplo siguiente cambia el nivel de silencio en función de la entrada del usuario en una instancia NumericStepper denominada `silenceLevel_nstep`. La instancia ProgressBar denominada `silenceLevel_pb` modifica su aspecto en función de si el flujo de audio se considera silencio. En caso contrario, muestra el nivel de actividad del flujo de audio.

```
var silenceLevel_pb:mx.controls.ProgressBar;
var silenceLevel_nstep:mx.controls.NumericStepper;

this.createEmptyMovieClip("sound_mc", this.getNextHighestDepth());
var active_mic:Microphone = Microphone.get();
sound_mc.attachAudio(active_mic);

silenceLevel_pb.label = "Activity level: %3";
silenceLevel_pb.mode = "manual";
silenceLevel_nstep.minimum = 0;
silenceLevel_nstep.maximum = 100;
silenceLevel_nstep.value = active_mic.silenceLevel;

var nstepListener:Object = new Object();
nstepListener.change = function(evt:Object) {
  active_mic.setSilenceLevel(evt.target.value, active_mic.silenceTimeOut);
};
silenceLevel_nstep.addEventListener("change", nstepListener);

this.onEnterFrame = function() {
  silenceLevel_pb.setProgress(active_mic.activityLevel, 100);
};
active_mic.onActivity = function(active:Boolean) {
  if (active) {
```

```
  silenceLevel_pb.indeterminate = false;
  silenceLevel_pb.setStyle("themeColor", "haloGreen");
  silenceLevel_pb.label = "Activity level: %3";
  } else {
  silenceLevel_pb.indeterminate = true;
  silenceLevel_pb.setStyle("themeColor", "0xFF0000");
  silenceLevel_pb.label = "Activity level: (inactive)";
  }
};
```

El método `MovieClip.getNextHighestDepth()` utilizado en este ejemplo necesita Flash Player 7 o una versión posterior. Si el archivo SWF incluye un componente de la versión 2, utilice la clase DepthManager de los componentes de la versión 2 en lugar del método `MovieClip.getNextHighestDepth()`.

Véase también

`setMotionLevel (método Camera.setMotionLevel)`, `activityLevel (propiedad Microphone.activityLevel)`, `onActivity (controlador Microphone.onActivity)`, `setGain (método Microphone.setGain)`, `silenceLevel (propiedad Microphone.silenceLevel)`, `silenceTimeOut (propiedad Microphone.silenceTimeOut)`

setUseEchoSuppression (método Microphone.setUseEchoSuppression)

```
public setUseEchoSuppression(useEchoSuppression:Boolean) : Void
```

Especifica si debe utilizarse la función de supresión de eco del códec de audio. El valor predeterminado es `false` a no ser que el usuario haya seleccionado la reducción de eco en el panel Configuración del micrófono de Flash Player.

La supresión del eco es un intento de reducción de los efectos del acoplamiento acústico, que tiene lugar cuando el sonido que sale del altavoz es detectado por el micrófono del mismo equipo. (Esto es diferente a la cancelación del eco, que elimina completamente el acoplamiento.)

Por lo general, la supresión del eco es aconsejable cuando el sonido que se está capturando se reproduce a través de los altavoces (en lugar de a través de auriculares) del mismo equipo. Si el archivo SWF permite a los usuarios especificar el dispositivo de salida del sonido, puede que resulte conveniente llamar a `Microphone.setUseEchoSuppression(true)` si éstos indican que están utilizando altavoces y que van a utilizar el micrófono también.

Los usuarios también pueden ajustar estos parámetros en el panel de configuración del micrófono de Flash Player.

Disponibilidad: ActionScript 1.0; Flash Player 6

Parámetros

useEchoSuppression:Boolean - Valor booleano que indica si debe utilizarse la supresión de eco (true) o no (false).

Ejemplo

El ejemplo siguiente activa la supresión de eco si el usuario selecciona una instancia CheckBox denominada useEchoSuppression_ch. La instancia ProgressBar denominada activityLevel_pb muestra el nivel de actividad actual del flujo de audio.

```
var useEchoSuppression_ch:mx.controls.CheckBox;
var activityLevel_pb:mx.controls.ProgressBar;

this.createEmptyMovieClip("sound_mc", this.getNextHighestDepth());
var active_mic:Microphone = Microphone.get();
sound_mc.attachAudio(active_mic);

activityLevel_pb.mode = "manual";
activityLevel_pb.label = "Activity Level: %3";
useEchoSuppression_ch.selected = active_mic.useEchoSuppression;
this.onEnterFrame = function() {
  activityLevel_pb.setProgress(active_mic.activityLevel, 100);
};
var chListener:Object = new Object();
chListener.click = function(evt:Object) {
  active_mic.setUseEchoSuppression(evt.target.selected);
};
useEchoSuppression_ch.addEventListener("click", chListener);
```

El método MovieClip.getNextHighestDepth() utilizado en este ejemplo necesita Flash Player 7 o una versión posterior. Si el archivo SWF incluye un componente de la versión 2, utilice la clase DepthManager de los componentes de la versión 2 en lugar del método MovieClip.getNextHighestDepth().

Véase también

setUseEchoSuppression (método Microphone.setUseEchoSuppression),
useEchoSuppression (propiedad Microphone.useEchoSuppression)

silenceLevel (propiedad Microphone.silenceLevel)

```
public silenceLevel : Number [read-only]
```

Un entero que especifica el nivel de sonido necesario para activar el micrófono e invocar Microphone.onActivity(true). El valor predeterminado es 10.

Disponibilidad: ActionScript 1.0; Flash Player 6

Ejemplo

El ejemplo siguiente cambia el nivel de silencio en función de la entrada del usuario en una instancia NumericStepper denominada `silenceLevel_nstep`. La instancia ProgressBar denominada `silenceLevel_pb` modifica su aspecto en función de si el flujo de audio se considera silencio. En caso contrario, muestra el nivel de actividad del flujo de audio.

```
var silenceLevel_pb:mx.controls.ProgressBar;
var silenceLevel_nstep:mx.controls.NumericStepper;

this.createEmptyMovieClip("sound_mc", this.getNextHighestDepth());
var active_mic:Microphone = Microphone.get();
sound_mc.attachAudio(active_mic);

silenceLevel_pb.label = "Activity level: %3";
silenceLevel_pb.mode = "manual";
silenceLevel_nstep.minimum = 0;
silenceLevel_nstep.maximum = 100;
silenceLevel_nstep.value = active_mic.silenceLevel;

var nstepListener:Object = new Object();
nstepListener.change = function(evt:Object) {
  active_mic.setSilenceLevel(evt.target.value, active_mic.silenceTimeOut);
};
silenceLevel_nstep.addEventListener("change", nstepListener);

this.onEnterFrame = function() {
  silenceLevel_pb.setProgress(active_mic.activityLevel, 100);
};
active_mic.onActivity = function(active:Boolean) {
  if (active) {
  silenceLevel_pb.indeterminate = false;
  silenceLevel_pb.setStyle("themeColor", "haloGreen");
  silenceLevel_pb.label = "Activity level: %3";
  } else {
  silenceLevel_pb.indeterminate = true;
  silenceLevel_pb.setStyle("themeColor", "0xFF0000");
  silenceLevel_pb.label = "Activity level: (inactive)";
  }
};
```

El método `MovieClip.getNextHighestDepth()` utilizado en este ejemplo necesita Flash Player 7 o una versión posterior. Si el archivo SWF incluye un componente de la versión 2, utilice la clase DepthManager de los componentes de la versión 2 en lugar del método `MovieClip.getNextHighestDepth()`.

Véase también

gain (propiedad Microphone.gain), setSilenceLevel (método Microphone.setSilenceLevel)

silenceTimeOut (propiedad Microphone.silenceTimeOut)

```
public silenceTimeOut : Number [read-only]
```

Valor numérico que representa los milisegundos que transcurren entre el momento en que el micrófono deja de detectar sonido y la invocación de `Microphone.onActivity(false)`. El valor predeterminado es 2000 (2 segundos).

Para definir este valor, utilice `Microphone.setSilenceLevel()`.

Disponibilidad: ActionScript 1.0; Flash Player 6

Ejemplo

El ejemplo siguiente permite que el usuario pueda controlar el tiempo que transcurrirá desde el momento en que el micrófono deje de detectar sonido y la invocación de `Microphone.onActivity(false)`. El usuario puede controlar este valor utilizando una instancia NumericStepper denominada silenceTimeOut_nstep. La instancia ProgressBar denominada `silenceLevel_pb` modifica su aspecto en función de si el flujo de audio se considera silencio. En caso contrario, muestra el nivel de actividad del flujo de audio.

```
var silenceLevel_pb:mx.controls.ProgressBar;
var silenceTimeOut_nstep:mx.controls.NumericStepper;

this.createEmptyMovieClip("sound_mc", this.getNextHighestDepth());
var active_mic:Microphone = Microphone.get();
sound_mc.attachAudio(active_mic);

silenceLevel_pb.label = "Activity level: %3";
silenceLevel_pb.mode = "manual";
silenceTimeOut_nstep.minimum = 0;
silenceTimeOut_nstep.maximum = 10;
silenceTimeOut_nstep.value = active_mic.silenceTimeOut/1000;

var nstepListener:Object = new Object();
nstepListener.change = function(evt:Object) {
  active_mic.setSilenceLevel(active_mic.silenceLevel, evt.target.value
  1000);
};
silenceTimeOut_nstep.addEventListener("change", nstepListener);

this.onEnterFrame = function() {
```

```
  silenceLevel_pb.setProgress(active_mic.activityLevel, 100);
};
active_mic.onActivity = function(active:Boolean) {
  if (active) {
  silenceLevel_pb.indeterminate = false;
  silenceLevel_pb.setStyle("themeColor", "haloGreen");
  silenceLevel_pb.label = "Activity level: %3";
  } else {
  silenceLevel_pb.indeterminate = true;
  silenceLevel_pb.setStyle("themeColor", "0xFF0000");
  silenceLevel_pb.label = "Activity level: (inactive)";
  }
};
```

El método `MovieClip.getNextHighestDepth()` utilizado en este ejemplo necesita Flash Player 7 o una versión posterior. Si el archivo SWF incluye un componente de la versión 2, utilice la clase DepthManager de los componentes de la versión 2 en lugar del método `MovieClip.getNextHighestDepth()`.

Véase también

`setSilenceLevel (método Microphone.setSilenceLevel)`

useEchoSuppression (propiedad Microphone.useEchoSuppression)

```
public useEchoSuppression : Boolean
```

Propiedad (sólo lectura); valor booleano `true` si está activada la supresión de eco; `false` en caso contrario. El valor predeterminado es `false` a no ser que el usuario haya seleccionado la reducción de eco en el panel Configuración del micrófono de Flash Player.

Disponibilidad: ActionScript 1.0; Flash Player 6

Ejemplo

El ejemplo siguiente activa la supresión de eco si el usuario selecciona una instancia CheckBox denominada `useEchoSuppression_ch`. La instancia ProgressBar denominada `activityLevel_pb` muestra el nivel de actividad actual del flujo de audio.

```
var useEchoSuppression_ch:mx.controls.CheckBox;
var activityLevel_pb:mx.controls.ProgressBar;

this.createEmptyMovieClip("sound_mc", this.getNextHighestDepth());
var active_mic:Microphone = Microphone.get();
sound_mc.attachAudio(active_mic);

activityLevel_pb.mode = "manual";
```

```
activityLevel_pb.label = "Activity Level: %3";
useEchoSuppression_ch.selected = active_mic.useEchoSuppression;
this.onEnterFrame = function() {
  activityLevel_pb.setProgress(active_mic.activityLevel, 100);
};
var chListener:Object = new Object();
chListener.click = function(evt:Object) {
  active_mic.setUseEchoSuppression(evt.target.selected);
};
useEchoSuppression_ch.addEventListener("click", chListener);
```

El método `MovieClip.getNextHighestDepth()` utilizado en este ejemplo necesita Flash Player 7 o una versión posterior. Si el archivo SWF incluye un componente de la versión 2, utilice la clase DepthManager de los componentes de la versión 2 en lugar del método `MovieClip.getNextHighestDepth()`.

Véase también

`setUseEchoSuppression (método Microphone.setUseEchoSuppression)`

Mouse

```
Object
  |
  +-Mouse
```

`public class` ***Mouse***
`extends Object`

La clase Mouse es una clase de nivel superior a cuyos métodos y propiedades puede acceder sin emplear un constructor. Puede utilizar los métodos de la clase Mouse para ocultar y mostrar el puntero del ratón (cursor) en el archivo SWF. El puntero del ratón es visible de forma predeterminada, pero es posible ocultarlo y emplear un puntero personalizado creado con un clip de película.

Una aplicación Flash sólo puede controlar los eventos de ratón que se produzcan en la propia aplicación. Una aplicación Flash no puede detectar eventos de ratón de otra aplicación.

Disponibilidad: ActionScript 1.0; Flash Player 5

Resumen de propiedades

Propiedades heredadas de la clase Object

`constructor (propiedad Object.constructor), __proto__ (propiedad Object.__proto__), prototype (propiedad Object.prototype), __resolve (propiedad Object.__resolve)`

Resumen de eventos

Evento	Descripción
`onMouseDown = function() {}`	Se notifica cuando se presiona el ratón.
`onMouseMove = function() {}`	Se notifica cuando se mueve el ratón.
`onMouseUp = function() {}`	Se notifica cuando se suelta el botón del ratón.
`onMouseWheel = function([delta:Number], [scrollTarget:String]) {}`	Se notifica cuando el usuario hace girar la rueda del ratón.

Resumen de métodos

Modificadores	Firma	Descripción
`static`	`addListener(listener:Object) : Void`	Registra un objeto para recibir notificaciones de los detectores `onMouseDown`, `onMouseMove`, `onMouseUp` y `onMouseWheel`.
`static`	`hide() : Number`	Oculta el puntero en un archivo SWF.
`static`	`removeListener(listener:Object) : Boolean`	Elimina un objeto registrado previamente con `addListener()`.
`static`	`show() : Number`	Muestra el puntero del ratón en un archivo SWF.

Métodos heredados de la clase Object

`addProperty (método Object.addProperty), hasOwnProperty (método Object.hasOwnProperty), isPropertyEnumerable (método Object.isPropertyEnumerable), isPrototypeOf (método Object.isPrototypeOf), registerClass (método Object.registerClass), toString (método Object.toString), unwatch (método Object.unwatch), valueOf (método Object.valueOf), watch (método Object.watch)`

addListener (método Mouse.addListener)

```
public static addListener(listener:Object) : Void
```

Registra un objeto para recibir notificaciones de los detectores `onMouseDown`, `onMouseMove`, `onMouseUp` y `onMouseWheel`. (El detector `onMouseWheel` sólo es compatible con Windows.)

El parámetro `listener` debe contener un objeto que tenga un método definido al menos para uno de los detectores.

Cuando el ratón se presiona, mueve, suelta o utiliza para desplazamiento, con independencia de cuál sea la selección de entrada, se invocará el método `onMouseDown`, `onMouseMove`, `onMouseUp` u `onMouseWheel` de todos los objetos a la escucha que estén registrados con este método. Puede haber varios objetos a la escucha de notificaciones de ratón. Si el `listener` ya se ha registrado, no se producirá ningún cambio.

Disponibilidad: ActionScript 1.0; Flash Player 6

Parámetros

`listener``:Object` - Un objeto.

Ejemplo

Este ejemplo está extraído del archivo *animation.fla* de la carpeta de ejemplos de ActionScript.

```
// Create a mouse listener object
var mouseListener:Object = new Object();

// Every time the mouse cursor moves within the SWF file,
  update the position of the crosshair movie clip
  instance on the Stage.
mouseListener.onMouseMove = function() {
  crosshair_mc._x = _xmouse;
  crosshair_mc._y = _ymouse;
};

// When you click the mouse, check to see if the cursor is within the
  boundaries of the Stage. If so, increment the number of shots.
mouseListener.onMouseDown = function() {
  if (bg_mc.hitTest(_xmouse, _ymouse, false)) {
  _global.shots++;
  }
};
Mouse.addListener(mouseListener);
```

Para ver el script completo, consulte el archivo *animation.fla* de la carpeta de ejemplo de ActionScript. La lista siguiente muestra rutas habituales a la carpeta ActionScript:

- Windows: *unidad de inicio*\Archivos de programa\Macromedia\Flash 8\Samples and Tutorials\Samples\ActionScript
- Macintosh: *Disco duro de Macintosh*/Applications/Macromedia Flash 8/Samples and Tutorials/Samples/ActionScript

Véase también

`onMouseDown (detector de eventos Mouse.onMouseDown)`, `onMouseMove (detector de eventos Mouse.onMouseMove)`, `onMouseUp (detector de eventos Mouse.onMouseUp)`, `onMouseWheel (detector de eventos Mouse.onMouseWheel)`

hide (método Mouse.hide)

```
public static hide() : Number
```

Oculta el puntero en un archivo SWF. El puntero está visible de manera predeterminada.

Disponibilidad: ActionScript 1.0; Flash Player 5

Valor devuelto

`Number` - un entero; `0` o `1`. Si el puntero del ratón se ocultó antes de la llamada a `Mouse.hide()`,, el valor devuelto es `0`. Si el puntero del ratón era visible antes de la llamada a `Mouse.hide()`, el valor devuelto es `1`.

Ejemplo

El código siguiente oculta el puntero estándar del ratón y establece las posiciones *x* e *y* de la instancia de clip de película `pointer_mc` en las posiciones *x* e *y* del puntero. Cree un clip de película y defina su identificador de vinculación en `pointer_id`. Añada el siguiente código ActionScript al fotograma 1 de la línea de tiempo:

```
// to use this script you need a symbol
// in your library with a Linkage Identifier of "pointer_id".
this.attachMovie("pointer_id", "pointer_mc", this.getNextHighestDepth());
Mouse.hide();
var mouseListener:Object = new Object();
mouseListener.onMouseMove = function() {
  pointer_mc._x = _xmouse;
  pointer_mc._y = _ymouse;
  updateAfterEvent();
};
Mouse.addListener(mouseListener);
```

El método `MovieClip.getNextHighestDepth()` utilizado en este ejemplo necesita Flash Player 7 o una versión posterior. Si el archivo SWF incluye un componente de la versión 2, utilice la clase DepthManager de los componentes de la versión 2 en lugar del método `MovieClip.getNextHighestDepth()`.

Véase también

`show (método Mouse.show)`, `_xmouse (propiedad MovieClip._xmouse)`, `_ymouse (propiedad MovieClip._ymouse)`

onMouseDown (detector de eventos Mouse.onMouseDown)

```
onMouseDown = function() {}
```

Se notifica cuando se presiona el ratón. Para utilizar el detector `onMouseDown`, debe crear un objeto detector. Posteriormente podrá definir una función para `onMouseDown` y utilizar `addListener()` para registrar el detector en el objeto Mouse, como se muestra en el siguiente código:

```
var someListener:Object = new Object();
someListener.onMouseDown = function () { ... };
Mouse.addListener(someListener);
```

Los detectores permiten que diversas partes del código cooperen, ya que varios detectores pueden recibir notificación de un solo evento.

Una aplicación Flash sólo puede controlar los eventos de ratón que se produzcan en la propia aplicación. Una aplicación Flash no puede detectar eventos de ratón de otra aplicación.

Disponibilidad: ActionScript 1.0; Flash Player 6

Ejemplo

El ejemplo siguiente utiliza la interfaz API de dibujo para dibujar un rectángulo siempre que el usuario hace clic, arrastra y suelta el ratón en tiempo de ejecución.

```
this.createEmptyMovieClip("canvas_mc", this.getNextHighestDepth());
var mouseListener:Object = new Object();
mouseListener.onMouseDown = function() {
  this.isDrawing = true;
  this.orig_x = _xmouse;
  this.orig_y = _ymouse;
  this.target_mc = canvas_mc.createEmptyMovieClip("",
  canvas_mc.getNextHighestDepth());
};
```

```
mouseListener.onMouseMove = function() {
  if (this.isDrawing) {
    this.target_mc.clear();
    this.target_mc.lineStyle(1, 0xFF0000, 100);
    this.target_mc.moveTo(this.orig_x, this.orig_y);
    this.target_mc.lineTo(_xmouse, this.orig_y);
    this.target_mc.lineTo(_xmouse, _ymouse);
    this.target_mc.lineTo(this.orig_x, _ymouse);
    this.target_mc.lineTo(this.orig_x, this.orig_y);
  }
  updateAfterEvent();
};
mouseListener.onMouseUp = function() {
  this.isDrawing = false;
};
Mouse.addListener(mouseListener);
```

El método `MovieClip.getNextHighestDepth()` utilizado en este ejemplo necesita Flash Player 7 o una versión posterior. Si el archivo SWF incluye un componente de la versión 2, utilice la clase DepthManager de los componentes de la versión 2 en lugar del método `MovieClip.getNextHighestDepth()`.

Véase también

`addListener (método Mouse.addListener)`

onMouseMove (detector de eventos Mouse.onMouseMove)

```
onMouseMove = function() {}
```

Se notifica cuando se mueve el ratón. Para utilizar el detector `onMouseMove`, debe crear un objeto detector. Posteriormente podrá definir una función para `onMouseMove` y utilizar `addListener()` para registrar el detector en el objeto Mouse, como se muestra en el siguiente código:

```
var someListener:Object = new Object();
someListener.onMouseMove = function () { ... };
Mouse.addListener(someListener);
```

Los detectores permiten que diversas partes del código cooperen, ya que varios detectores pueden recibir notificación de un solo evento.

Una aplicación Flash sólo puede controlar los eventos de ratón que se produzcan en la propia aplicación. Una aplicación Flash no puede detectar eventos de ratón de otra aplicación.

Disponibilidad: ActionScript 1.0; Flash Player 6

Ejemplo

El ejemplo siguiente utiliza el puntero del ratón como herramienta para dibujar líneas con `onMouseMove` y la interfaz API de dibujo. El usuario dibuja una línea al arrastrar el puntero del ratón.

```
this.createEmptyMovieClip("canvas_mc", this.getNextHighestDepth());
var mouseListener:Object = new Object();
mouseListener.onMouseDown = function() {
  this.isDrawing = true;
  canvas_mc.lineStyle(2, 0xFF0000, 100);
  canvas_mc.moveTo(_xmouse, _ymouse);
};
mouseListener.onMouseMove = function() {
  if (this.isDrawing) {
    canvas_mc.lineTo(_xmouse, _ymouse);
  }
  updateAfterEvent();
};
mouseListener.onMouseUp = function() {
  this.isDrawing = false;
};
Mouse.addListener(mouseListener);
```

El método `MovieClip.getNextHighestDepth()` utilizado en este ejemplo necesita Flash Player 7 o una versión posterior. Si el archivo SWF incluye un componente de la versión 2, utilice la clase DepthManager de los componentes de la versión 2 en lugar del método `MovieClip.getNextHighestDepth()`.

El ejemplo siguiente oculta el puntero estándar del ratón y establece las posiciones *x* e *y* de la instancia de clip de película `pointer_mc` en las posiciones *x* e *y* del puntero. Cree un clip de película y defina su identificador de vinculación en `pointer_id`. Añada el siguiente código ActionScript al fotograma 1 de la línea de tiempo:

```
// to use this script you need a symbol
// in your library with a Linkage Identifier of "pointer_id".
this.attachMovie("pointer_id", "pointer_mc", this.getNextHighestDepth());
Mouse.hide();
var mouseListener:Object = new Object();
mouseListener.onMouseMove = function() {
  pointer_mc._x = _xmouse;
  pointer_mc._y = _ymouse;
  updateAfterEvent();
};
Mouse.addListener(mouseListener);
```

El método `MovieClip.getNextHighestDepth()` utilizado en este ejemplo necesita Flash Player 7 o una versión posterior. Si el archivo SWF incluye un componente de la versión 2, utilice la clase DepthManager de los componentes de la versión 2 en lugar del método `MovieClip.getNextHighestDepth()`.

Véase también

addListener (método Mouse.addListener)

onMouseUp (detector de eventos Mouse.onMouseUp)

```
onMouseUp = function() {}
```

Se notifica cuando se suelta el botón del ratón. Para utilizar el detector `onMouseUp`, debe crear un objeto detector. Posteriormente podrá definir una función para `onMouseUp` y utilizar `addListener()` para registrar el detector en el objeto Mouse, como se muestra en el siguiente código:

```
var someListener:Object = new Object();
someListener.onMouseUp = function () { ... };
Mouse.addListener(someListener);
```

Los detectores permiten que diversas partes del código cooperen, ya que varios detectores pueden recibir notificación de un solo evento.

Una aplicación Flash sólo puede controlar los eventos de ratón que se produzcan en la propia aplicación. Una aplicación Flash no puede detectar eventos de ratón de otra aplicación.

Disponibilidad: ActionScript 1.0; Flash Player 6

Ejemplo

El ejemplo siguiente utiliza el puntero del ratón como herramienta para dibujar líneas con `onMouseMove` y la interfaz API de dibujo. El usuario dibuja una línea al arrastrar el puntero del ratón. El usuario deja de dibujar la línea al soltar el botón del ratón.

```
this.createEmptyMovieClip("canvas_mc", this.getNextHighestDepth());
var mouseListener:Object = new Object();
mouseListener.onMouseDown = function() {
  this.isDrawing = true;
  canvas_mc.lineStyle(2, 0xFF0000, 100);
  canvas_mc.moveTo(_xmouse, _ymouse);
};
mouseListener.onMouseMove = function() {
  if (this.isDrawing) {
    canvas_mc.lineTo(_xmouse, _ymouse);
  }
  updateAfterEvent();
};
mouseListener.onMouseUp = function() {
  this.isDrawing = false;
};
Mouse.addListener(mouseListener);
```

El método `MovieClip.getNextHighestDepth()` utilizado en este ejemplo necesita Flash Player 7 o una versión posterior. Si el archivo SWF incluye un componente de la versión 2, utilice la clase DepthManager de los componentes de la versión 2 en lugar del método `MovieClip.getNextHighestDepth()`.

Véase también

`addListener (método Mouse.addListener)`

onMouseWheel (detector de eventos Mouse.onMouseWheel)

```
onMouseWheel = function([delta:Number], [scrollTarget:String]) {}
```

Se notifica cuando el usuario hace girar la rueda del ratón. Para utilizar el detector `onMouseWheel`, debe crear un objeto detector. Posteriormente podrá definir una función para `onMouseWheel` y utilizar `addListener()` para registrar el detector en el objeto Mouse.

Nota: Los detectores de eventos de rueda del ratón sólo están disponibles para las versiones Windows de Flash Player.

Una aplicación Flash sólo puede controlar los eventos de ratón que se produzcan en la propia aplicación. Una aplicación Flash no puede detectar eventos de ratón de otra aplicación.

Disponibilidad: ActionScript 1.0; Flash Player 6

Parámetros

`delta``:Number` [opcional] - Número que indica el número de líneas que debe desplazarse la visualización por cada muesca de la rueda del ratón que haga rodar el usuario. Un valor de `delta` positivo indica un desplazamiento hacia arriba; un valor negativo indica un desplazamiento hacia abajo. Los valores típicos son de 1 a 3; el desplazamiento a mayor velocidad puede producir valores mayores.

`scrollTarget``:String` [opcional] - Parámetro que indica la instancia de clip de película de nivel superior situada bajo el puntero del ratón cuando se hace rodar la rueda del ratón. Si desea especificar un valor para `scrollTarget` pero no desea especificar un valor para `delta`, pase el valor `null` para `delta`.

Ejemplo

El ejemplo siguiente muestra cómo crear un objeto detector que responda a los eventos de rueda del ratón. En este ejemplo, la coordenada *x* de un objeto de clip de película denominado `clip_mc` cambia cada vez que el usuario hace rodar la rueda del ratón:

```
var mouseListener:Object = new Object();
```

```
mouseListener.onMouseWheel = function(delta) {
  clip_mc._x += delta;
}
Mouse.addListener(mouseListener);
```

El ejemplo siguiente dibuja una línea que gira al hacer rodar la rueda del ratón. Haga clic en el archivo SWF en tiempo de ejecución y haga rodar la rueda del ratón para ver el clip de película en acción.

```
this.createEmptyMovieClip("line_mc", this.getNextHighestDepth());
line_mc.lineStyle(2, 0xFF0000, 100);
line_mc.moveTo(0, 100);
line_mc.lineTo(0, 0);
line_mc._x = 200;
line_mc._y = 200;

var mouseListener:Object = new Object();
mouseListener.onMouseWheel = function(delta:Number) {
  line_mc._rotation += delta;
};
mouseListener.onMouseDown = function() {
  trace("Down");
};
Mouse.addListener(mouseListener);
```

El método `MovieClip.getNextHighestDepth()` utilizado en este ejemplo necesita Flash Player 7 o una versión posterior. Si el archivo SWF incluye un componente de la versión 2, utilice la clase DepthManager de los componentes de la versión 2 en lugar del método `MovieClip.getNextHighestDepth()`.

Véase también

`addListener (método Mouse.addListener)`, `mouseWheelEnabled (propiedad TextField.mouseWheelEnabled)`

removeListener (método Mouse.removeListener)

```
public static removeListener(listener:Object) : Boolean
```

Elimina un objeto registrado previamente con `addListener()`.

Disponibilidad: ActionScript 1.0; Flash Player 6

Parámetros

`listener``:Object` - Un objeto.

Valor devuelto

`Boolean` - Si el objeto `listener` se elimina correctamente, el método devolverá `true`; si el objeto `listener` no se elimina correctamente (por ejemplo, si el objeto `listener` no se encontraba en la lista de detectores del objeto Mouse), el método devolverá `false`.

Ejemplo

El ejemplo siguiente asocia tres botones al escenario y permite que el usuario dibuje líneas en el archivo SWF con el puntero del ratón en tiempo de ejecución. Un botón borra todas las líneas del archivo SWF. El segundo botón elimina el detector del ratón, por lo que el usuario no puede dibujar líneas. El tercer botón añade el detector del ratón cuando se ha eliminado, de modo que el usuario puede volver a dibujar líneas. Añada el siguiente código ActionScript al fotograma 1 de la línea de tiempo:

```
this.createClassObject(mx.controls.Button, "clear_button",
  this.getNextHighestDepth(), {_x:10, _y:10, label:'clear'});
this.createClassObject(mx.controls.Button, "stopDrawing_button",
  this.getNextHighestDepth(), {_x:120, _y:10, label:'stop drawing'});
this.createClassObject(mx.controls.Button, "startDrawing_button",
  this.getNextHighestDepth(), {_x:230, _y:10, label:'start drawing'});
startDrawing_button.enabled = false;
//
this.createEmptyMovieClip("canvas_mc", this.getNextHighestDepth());
var mouseListener:Object = new Object();
mouseListener.onMouseDown = function() {
  this.isDrawing = true;
  canvas_mc.lineStyle(2, 0xFF0000, 100);
  canvas_mc.moveTo(_xmouse, _ymouse);
};
mouseListener.onMouseMove = function() {
  if (this.isDrawing) {
  canvas_mc.lineTo(_xmouse, _ymouse);
  }
  updateAfterEvent();
};
mouseListener.onMouseUp = function() {
  this.isDrawing = false;
};
Mouse.addListener(mouseListener);
var clearListener:Object = new Object();
clearListener.click = function() {
  canvas_mc.clear();
};
clear_button.addEventListener("click", clearListener);
//
var stopDrawingListener:Object = new Object();
stopDrawingListener.click = function(evt:Object) {
  Mouse.removeListener(mouseListener);
```

```
  evt.target.enabled = false;
  startDrawing_button.enabled = true;
};
stopDrawing_button.addEventListener("click", stopDrawingListener);
var startDrawingListener:Object = new Object();
startDrawingListener.click = function(evt:Object) {
  Mouse.addListener(mouseListener);
  evt.target.enabled = false;
  stopDrawing_button.enabled = true;
};
startDrawing_button.addEventListener("click", startDrawingListener);
```

El método `MovieClip.getNextHighestDepth()` utilizado en este ejemplo necesita Flash Player 7 o una versión posterior. Si el archivo SWF incluye un componente de la versión 2, utilice la clase DepthManager de los componentes de la versión 2 en lugar del método `MovieClip.getNextHighestDepth()`.

show (método Mouse.show)

```
public static show() : Number
```

Muestra el puntero del ratón en un archivo SWF. El puntero está visible de manera predeterminada.

Disponibilidad: ActionScript 1.0; Flash Player 5

Valor devuelto

`Number` - un entero; `0` o `1`. Si el puntero del ratón se ocultó antes de la llamada a `Mouse.show()`,, el valor devuelto es `0`. Si el puntero del ratón era visible antes de la llamada a `Mouse.show()`, el valor devuelto es `1`.

Ejemplo

El ejemplo siguiente asocia un cursor personalizado de la biblioteca al desplazarse sobre un clip de película denominado `my_mc`. Asigne a un clip de película de la biblioteca el identificador de vinculación `cursor_help_id` y añada el siguiente código ActionScript al fotograma 1 de la línea de tiempo:

```
my_mc.onRollOver = function() {
  Mouse.hide();
  this.attachMovie("cursor_help_id", "cursor_mc",
  this.getNextHighestDepth(), {_x:this._xmouse, _y:this._ymouse});
};
my_mc.onMouseMove = function() {
  this.cursor_mc._x = this._xmouse;
  this.cursor_mc._y = this._ymouse;
};
my_mc.onRollOut = function() {
```

```
  Mouse.show();
  this.cursor_mc.removeMovieClip();
};
```

El método `MovieClip.getNextHighestDepth()` utilizado en este ejemplo necesita Flash Player 7 o una versión posterior. Si el archivo SWF incluye un componente de la versión 2, utilice la clase DepthManager de los componentes de la versión 2 en lugar del método `MovieClip.getNextHighestDepth()`.

Véase también

`hide (método Mouse.hide)`, `_xmouse (propiedad MovieClip._xmouse)`, `_ymouse (propiedad MovieClip._ymouse)`

MovieClip

```
Object
  |
  +-MovieClip
```

`public dynamic class` ***MovieClip***
`extends Object`

Los métodos de la clase MovieClip proporcionan la misma funcionalidad que las acciones que van dirigidas a clips de película. También existen métodos adicionales que carecen de acciones equivalentes en el cuadro de herramientas Acciones del panel Acciones.

Para crear un clip de película no tendrá que utilizar un método constructor. Es posible elegir entre tres métodos para crear instancias de clip de película:

- El método `attachMovie()` permite crear una instancia de clip de película basada en un símbolo de clip de película que existe en la biblioteca.
- El método `createEmptyMovieClip()` permite crear una instancia vacía de clip de película como elemento secundario de otro clip de película.
- El método `duplicateMovieClip()` permite crear una instancia de clip de película basada en otro clip de película.

Para llamar a los métodos de la clase MovieClip, debe hacer referencia a las instancias de clip de película por su nombre y utilizar la siguiente sintaxis, donde *`my_mc`* es una instancia de clip de película:

```
my_mc.play();
my_mc.gotoAndPlay(3);
```

Puede ampliar los métodos y los controladores de eventos de la clase MovieClip creando una subclase.

Disponibilidad: ActionScript 1.0; Flash Player 3

Resumen de propiedades

Modificadores	Propiedad	Descripción
	`_alpha:Number`	Valor de transparencia alfa del clip de película.
	`blendMode:Object`	Modo de mezcla de este clip de película.
	`cacheAsBitmap:Boolean`	Si se define como `true`, Flash Player deja en caché una representación interna de mapa de bits del clip de película.
	`_currentframe:Number` [read-only]	Devuelve el número del fotograma en el que está situada la cabeza lectora en la línea de tiempo del clip de película.
	`_droptarget:String` [read-only]	Devuelve la ruta absoluta en notación de sintaxis con barras de la instancia de clip de película en la que se ha colocado dicho clip de película.
	`enabled:Boolean`	Valor booleano que indica si un clip de película está activado.
	`filters:Array`	Matriz indexada que contiene todos los objetos de filtro actualmente asociados al clip de película.
	`focusEnabled:Boolean`	Si el valor es `undefined` o `false`, un clip de película no puede quedar resaltado a menos que sea un botón.
	`_focusrect:Boolean`	Valor booleano que especifica si un clip de película debe mostrar un rectángulo amarillo a su alrededor cuando se selecciona con el teclado.
	`_framesloaded:Number` [read-only]	Número de fotogramas que se han cargado de un flujo de archivo SWF.
	`_height:Number`	Altura del clip de película, expresada en píxeles.
	`_highquality:Number`	*Desfasada* desde Flash Player 7. Esta propiedad está desfasada y en su lugar debe utilizarse `MovieClip._quality`. Especifica el nivel de visualización suavizada que se aplica al archivo SWF actual.
	`hitArea:Object`	Designa otro clip de película para que actúe como área activa de un clip de película.
	`_lockroot:Boolean`	Un valor booleano que especifica a qué hace referencia `_root` cuando se carga un archivo SWF en un clip de película.
	`menu:ContextMenu`	Asocia el objeto ContextMenu especificado al clip de película.

Modificadores	Propiedad	Descripción
	`_name:String`	Nombre de instancia del clip de película.
	`opaqueBackground:Number`	Color del fondo opaco (no transparente) del clip de película del color especificado por el número (valor RGB hexadecimal).
	`_parent:MovieClip`	Referencia al clip de película u objeto que contiene el clip de película u objeto actual.
	`_quality:String`	Establece o recupera la calidad de representación que se utiliza para un archivo SWF.
	`_rotation:Number`	Especifica el giro del clip de película, expresado en grados, con respecto a su orientación original.
	`scale9Grid:Rectangle`	Región rectangular que define las nueve regiones de escala del clip de película.
	`scrollRect:Object`	La propiedad `scrollRect` permite desplazar con rapidez el contenido del clip de película y abrir una ventana que muestre mayor cantidad de contenido.
	`_soundbuftime:Number`	Especifica el número de segundos que un sonido debe acumularse previamente en el búfer antes de que comience a reproducirse sin interrupción.
	`tabChildren:Boolean`	Determina si los elementos secundarios de un clip de película se incluyen en el orden de tabulación automático.
	`tabEnabled:Boolean`	Especifica si el clip de película se incluye en el orden de tabulación automático.
	`tabIndex:Number`	Permite personalizar el orden de tabulación de los objetos de una película.
	`_target:String` [read-only]	Devuelve la ruta de destino de la instancia de clip de película en notación con barras.
	`_totalframes:Number` [read-only]	Devuelve el número total de fotogramas de la instancia de clip de película especificada en el parámetro `MovieClip`.
	`trackAsMenu:Boolean`	Valor booleano que indica si otros botones o clips de película pueden recibir eventos de liberación del botón del ratón.
	`transform:Transform`	Objeto con propiedades pertenecientes a una matriz, transformación de color y límites de píxel de un clip de película.

Modificadores	Propiedad	Descripción
	`_url:String` [read-only]	Recupera la URL del archivo SWF, JPEG, GIF o PNG del que se descargó el clip de película.
	`useHandCursor:Boolean`	Valor booleano que indica si debe aparecer el puntero de mano (cursor de mano) cuando el ratón pasa por encima de un clip de película.
	`_visible:Boolean`	Valor booleano que indica si un clip de película está visible.
	`_width:Number`	Anchura del clip de película, expresada en píxeles.
	`_x:Number`	Entero que establece la coordenada *x* de un clip de película con respecto a las coordenadas locales del clip de película principal.
	`_xmouse:Number` [read-only]	Devuelve la coordenada *x* de la posición del ratón.
	`_xscale:Number`	Determina la escala horizontal (*`percentage`*) del clip de película aplicada desde el punto de registro del clip de película.
	`_y:Number`	Establece la coordenada *y* de un clip de película con respecto a las coordenadas locales del clip de película principal.
	`_ymouse:Number` [read-only]	Indica la coordenada *y* de la posición del ratón.
	`_yscale:Number`	Establece la escala vertical (*`percentage`*) del clip de película aplicada desde el punto de registro del clip de película.

Propiedades heredadas de la clase Object

```
constructor (propiedad Object.constructor), __proto__ (propiedad
Object.__proto__), prototype (propiedad Object.prototype), __resolve
(propiedad Object.__resolve)
```

Resumen de eventos

Evento	Descripción
`onData = function() {}`	Se invoca cuando un clip de película recibe datos de una llamada `MovieClip.loadVariables()` o `MovieClip.loadMovie()`.
`onDragOut = function() {}`	Se invoca cuando se presiona el botón del ratón y el puntero se desplaza fuera del objeto.

Evento	Descripción
`onDragOver = function() {}`	Se invoca cuando el puntero se arrastra fuera del clip de película y luego se pasa por encima de éste.
`onEnterFrame = function() {}`	Se invoca de manera repetida con la velocidad de fotogramas del archivo SWF.
`onKeyDown = function() {}`	Se invoca cuando un clip de película está resaltado y se presiona una tecla.
`onKeyUp = function() {}`	Se invoca cuando se suelta una tecla.
`onKillFocus = function(newFocus:Object) {}`	Se invoca cuando un clip de película deja de estar seleccionado con el teclado.
`onLoad = function() {}`	Se invoca cuando se crea una instancia del clip de película y ésta aparece en la línea de tiempo.
`onMouseDown = function() {}`	Se invoca cuando se presiona el botón del ratón.
`onMouseMove = function() {}`	Se invoca cuando se mueve el ratón.
`onMouseUp = function() {}`	Se invoca cuando se suelta el botón del ratón.
`onPress = function() {}`	Se invoca cuando el usuario hace clic en el ratón mientras el puntero se encuentra sobre el clip de película.
`onRelease = function() {}`	Se invoca cuando se suelta el botón del ratón sobre un clip de película.
`onReleaseOutside = function() {}`	Se invoca después de que se presione el botón del ratón dentro del área de un clip de película y luego se suelte fuera del área del clip de película.
`onRollOut = function() {}`	Se invoca cuando el puntero se desplaza fuera del área de un clip de película.
`onRollOver = function() {}`	Se invoca cuando el puntero se desplaza sobre un área de un clip de película.
`onSetFocus = function(oldFocus:Object) {}`	Se invoca cuando un clip de película se selecciona con el teclado.
`onUnload = function() {}`	Se invoca en el primer fotograma después de que el clip de película se elimine de la línea de tiempo.

Resumen de métodos

Modificadores	Firma	Descripción
	`attachAudio(id:Object) : Void`	Especifica la fuente de audio que debe reproducirse.
	`attachBitmap(bmp:BitmapData, depth:Number, [pixelSnapping:String], [smoothing:Boolean]) : Void`	Asocia una imagen de mapa de bits a un clip de película.
	`attachMovie(id:String, name:String, depth:Number, [initObject:Object]) : MovieClip`	Localiza un símbolo de la biblioteca y lo asocia al clip de película.
	`beginBitmapFill(bmp:BitmapData, [matrix:Matrix], [repeat:Boolean], [smoothing:Boolean]) : Void`	Rellena un área de dibujo con una imagen de mapa de bits.
	`beginFill(rgb:Number, [alpha:Number]) : Void`	Indica el comienzo de un nuevo trazo de dibujo.
	`beginGradientFill(fillType:String, colors:Array, alphas:Array, ratios:Array, matrix:Object, [spreadMethod:String], [interpolationMethod:String], [focalPointRatio:Number]) : Void`	Indica el comienzo de un nuevo trazo de dibujo.
	`clear() : Void`	Elimina todos los gráficos creados en tiempo de ejecución empleando los métodos de dibujo del clip de película, incluidos los estilos de línea especificados con `MovieClip.lineStyle()`.

Modificadores	Firma	Descripción
	`createEmptyMovieClip(name:String, depth:Number) : MovieClip`	Crea un clip de película vacío como elemento secundario de un clip de película existente.
	`createTextField(instanceName:String, depth:Number, x:Number, y:Number, width:Number, height:Number) : TextField`	Crea un nuevo campo de texto vacío como elemento secundario del clip de película que llama a este método.
	`curveTo(controlX:Number, controlY:Number, anchorX:Number, anchorY:Number) : Void`	Dibuja una curva con el estilo de línea actual desde la posición de dibujo actual hasta (`anchorX`, `anchorY`) utilizando el punto de control especificado con (`controlX`, `controlY`).
	`duplicateMovieClip(name:String, depth:Number, [initObject:Object]) : MovieClip`	Crea una instancia del clip de película especificado mientras se reproduce el archivo SWF.
	`endFill() : Void`	Aplica un relleno a las líneas y curvas añadidas desde la última llamada a `beginFill()` o `beginGradientFill()`.
	`getBounds(bounds:Object) : Object`	Devuelve las propiedades que son los valores de coordenada mínimo y máximo *x* e *y* del clip de película para el parámetro `bounds`.
	`getBytesLoaded() : Number`	Devuelve el número de bytes que se han cargado (reproducido sin interrupción) para el clip de película.
	`getBytesTotal() : Number`	Devuelve el tamaño en bytes del clip de película.
	`getDepth() : Number`	Devuelve la profundidad de una instancia de clip de película.
	`getInstanceAtDepth(depth:Number) : MovieClip`	Determina si una profundidad específica ya está ocupada por un clip de película.

Modificadores	Firma	Descripción
	`getNextHighestDepth() : Number`	Le permite determinar un valor de profundidad que puede pasar a `MovieClip.attachMovie()`, `MovieClip.duplicateMovieClip()` o `MovieClip.createEmptyMovieClip()` para asegurarse de que Flash representa el clip de película delante de todos los demás objetos situados en el mismo nivel y capa del clip de película actual.
	`getRect(bounds:Object) : Object`	Devuelve las propiedades que son los valores de coordenada mínimo y máximo *x* e *y* del clip de película para el parámetro `bounds`, excepto cualquier trazo en formas..
	`getSWFVersion() : Number`	Devuelve un entero que indica la versión de Flash Player para la que se ha publicado el clip de película.
	`getTextSnapshot() : TextSnapshot`	Devuelve un objeto TextSnapshot que contiene el texto de todos los campos de texto estáticos existentes en el clip de película especificado; el texto contenido en los clips de película secundarios no se incluye.
	`getURL(url:String, [window:String], [method:String]) : Void`	Carga un documento de la URL especificada en la ventana indicada.
	`globalToLocal(pt:Object) : Void`	Convierte el objeto `pt` de las coordenadas del escenario (globales) en coordenadas del clip de película (locales).
	`gotoAndPlay(frame:Object) : Void`	Inicia la reproducción del archivo SWF en el fotograma especificado.
	`gotoAndStop(frame:Object) : Void`	Traslada la cabeza lectora al fotograma especificado del clip de película y la detiene en dicho lugar.
	`hitTest() : Boolean`	Evalúa el clip de película para comprobar si se superpone o corta con el área activa identificada mediante los parámetros de coordenadas `target` o `x` e `y`.

Modificadores	Firma	Descripción
	`lineGradientStyle(fillType:String, colors:Array, alphas:Array, ratios:Array, matrix:Object, [spreadMethod:String], [interpolationMethod:String], [focalPointRatio:Number]) : Void`	Especifica un estilo de línea que Flash utilizará para las posteriores llamadas a `lineTo()` y `curveTo()` hasta que llame a `lineStyle()` o `lineGradientStyle()` con otros parámetros.
	`lineStyle(thickness:Number, rgb:Number, alpha:Number, pixelHinting:Boolean, noScale:String, capsStyle:String, jointStyle:String, miterLimit:Number) : Void`	Especifica un estilo de línea que Flash utilizará para las posteriores llamadas a `lineTo()` y `curveTo()` hasta que llame a `lineStyle()` con otros parámetros.
	`lineTo(x:Number, y:Number) : Void`	Dibuja una línea utilizando el estilo de línea actual desde la posición de dibujo actual hasta (x, y); la posición de dibujo actual se establece posteriormente como (x, y).
	`loadMovie(url:String, [method:String]) : Void`	Carga un archivo SWF, JPEG, GIF o PNG en un clip de película en Flash Player mientras se reproduce el archivo SWF original.
	`loadVariables(url:String, [method:String]) : Void`	Lee datos de un archivo externo y establece los valores de variables en el clip de película.
	`localToGlobal(pt:Object) : Void`	Convierte el objeto `pt` de coordenadas del clip de película (locales) en coordenadas del escenario (globales).
	`moveTo(x:Number, y:Number) : Void`	Mueve la posición de dibujo actual a (x, y).
	`nextFrame() : Void`	Traslada la cabeza lectora al siguiente fotograma y la detiene en dicho punto.

Modificadores	Firma	Descripción
	`play() : Void`	Mueve la cabeza lectora por la línea de tiempo del clip de película.
	`prevFrame() : Void`	Traslada la cabeza lectora al fotograma anterior y la detiene en dicho punto.
	`removeMovieClip() : Void`	Elimina una instancia de clip de película creada por `duplicateMovieClip()`, `MovieClip.duplicateMovieClip()`, `MovieClip.createEmptyMovieClip()` o `MovieClip.attachMovie()`.
	`setMask(mc:Object) : Void`	Convierte el clip de película del parámetro `mc` en una máscara que revela el clip de película especificado por el clip de película.
	`startDrag([lockCenter:Boolean], [left:Number], [top:Number], [right:Number], [bottom:Number]) : Void`	Permite al usuario arrastrar el clip de película especificado.
	`stop() : Void`	Detiene el clip de película que se está reproduciendo actualmente.
	`stopDrag() : Void`	Finaliza un método `MovieClip.startDrag()`.
	`swapDepths(target:Object) : Void`	Intercambia el apilamiento o nivel de profundidad (z-order), de este clip de película con el clip de película especificado por el parámetro `target` o con el clip de película que ocupa actualmente el nivel de profundidad especificado en el parámetro `target`.
	`unloadMovie() : Void`	Elimina el contenido de una instancia de clip de película.

Métodos heredados de la clase Object

addProperty (método Object.addProperty), hasOwnProperty (método Object.hasOwnProperty), isPropertyEnumerable (método Object.isPropertyEnumerable), isPrototypeOf (método Object.isPrototypeOf), registerClass (método Object.registerClass), toString (método Object.toString), unwatch (método Object.unwatch), valueOf (método Object.valueOf), watch (método Object.watch)

_alpha (propiedad MovieClip._alpha)

```
public _alpha : Number
```

Valor de transparencia alfa del clip de película. Los valores válidos son los comprendidos entre 0 (totalmente transparente) y 100 (totalmente opaco). El valor predeterminado es 100. Los objetos existentes en un clip de película que tenga configurado `_alpha` con el valor 0 continuarán activos aunque no sean visibles. Por ejemplo, puede hacer clic en un botón de un clip de película aunque su propiedad `_alpha` esté configurada con el valor 0. Para desactivar el botón completamente, puede establecer la propiedad `_visible` del clip de película con el valor false.

Puede ampliar los métodos y los controladores de eventos de la clase MovieClip creando una subclase.

Disponibilidad: ActionScript 1.0; Flash Player 4

Ejemplo

El código siguiente define en 50% la propiedad `_alpha` de un clip de película creado dinámicamente y denominado `triangle` cuando el ratón se desplaza sobre el clip de película. Añada el siguiente código ActionScript al archivo FLA o AS:

```
this.createEmptyMovieClip("triangle", this.getNextHighestDepth());

triangle.beginFill(0x0000FF, 100);
triangle.moveTo(10, 10);
triangle.lineTo(10, 100);
triangle.lineTo(100, 10);
triangle.lineTo(10, 10);

triangle.onRollOver = function() {
  this._alpha = 50;
};
triangle.onRollOut = function() {
  this._alpha = 100;
};
```

El método `MovieClip.getNextHighestDepth()` utilizado en este ejemplo necesita Flash Player 7 o una versión posterior. Si el archivo SWF incluye un componente de la versión 2, utilice la clase DepthManager de los componentes de la versión 2 en lugar del método `MovieClip.getNextHighestDepth()`.

Véase también

`_alpha (propiedad Button._alpha)`, `_alpha (propiedad TextField._alpha)`, `_visible (propiedad MovieClip._visible)`

attachAudio (método MovieClip.attachAudio)

```
public attachAudio(id:Object) : Void
```

Especifica la fuente de audio que debe reproducirse. Para detener la reproducción de la fuente de audio, pase `false` como valor de `id`.

Puede ampliar los métodos y los controladores de eventos de la clase MovieClip creando una subclase.

Disponibilidad: ActionScript 1.0; Flash Player 6

Parámetros

`id``:Object` - Objeto que contiene el audio que ha de reproducirse. Los valores válidos son un objeto Microphone, un objeto NetStream que reproduce un archivo FLV y `false` (detiene la reproducción de audio).

Ejemplo

El ejemplo siguiente crea una nueva conexión NetStream. Añada un nuevo símbolo de vídeo; para ello, abra el panel Biblioteca y seleccione Nuevo vídeo en el menú Opciones de Biblioteca. Asigne a la instancia del símbolo el nombre `my_video`. Cargue dinámicamente el vídeo FLV en tiempo de ejecución. Utilice el método `attachAudio()` para asociar el audio del archivo FLV a un clip de película del escenario. A continuación puede controlar el audio del clip de película mediante la clase Sound y dos botones denominados `volUp_btn` y `volDown_btn`:

```
var my_nc:NetConnection = new NetConnection();
my_nc.connect(null);
var my_ns:NetStream = new NetStream(my_nc);
my_video.attachVideo(my_ns);
my_ns.play("yourVideo.flv");
this.createEmptyMovieClip("flv_mc", this.getNextHighestDepth());
flv_mc.attachAudio(my_ns);
var audio_sound:Sound = new Sound(flv_mc);

// Add volume buttons.
volUp_btn.onRelease = function() {
  if (audio_sound.getVolume()<100) {
  audio_sound.setVolume(audio_sound.getVolume()+10);
  updateVolume();
  }
};
volDown_btn.onRelease = function() {
  if (audio_sound.getVolume()>0) {
  audio_sound.setVolume(audio_sound.getVolume()-10);
  updateVolume();
  }
```

```
};

// Updates the volume.
this.createTextField("volume_txt", this.getNextHighestDepth(), 0, 0, 100,
  22);
updateVolume();
function updateVolume() {
  volume_txt.text = "Volume: "+audio_sound.getVolume();
}
```

El ejemplo siguiente especifica un micrófono como fuente de audio de una instancia de clip de película creada dinámicamente y denominada `audio_mc`:

```
var active_mic:Microphone = Microphone.get();
this.createEmptyMovieClip("audio_mc", this.getNextHighestDepth());
audio_mc.attachAudio(active_mic);
```

El método `MovieClip.getNextHighestDepth()` utilizado en este ejemplo necesita Flash Player 7 o una versión posterior. Si el archivo SWF incluye un componente de la versión 2, utilice la clase DepthManager de los componentes de la versión 2 en lugar del método `MovieClip.getNextHighestDepth()`.

Véase también

`Microphone`, `play (método NetStream.play)`, `Sound`, `attachVideo (método Video.attachVideo)`

attachBitmap (método MovieClip.attachBitmap)

```
public attachBitmap(bmp:BitmapData, depth:Number, [pixelSnapping:String],
  [smoothing:Boolean]) : Void
```

Asocia una imagen de mapa de bits a un clip de película.

Una vez asociado el mapa de bits al clip de película, en este último se hace una referencia al objeto de mapa de bits. Al asociar un mapa de bits puede especificar los parámetros `pixelSnapping` y `smoothing` para modificar el aspecto del mapa de bits.

Tras añadir un objeto al clip de película, deja de ser un objeto accesible. Los parámetros `depth`, `pixelSnapping` y `smoothing` sólo se pueden definir en una llamada al método `attachBitmap()` y no se pueden modificar después.

Utilice primero `createEmptyMovieClip()` para crear un clip de película vacío y después emplee el método `attachBitmap()`. De este modo, podrá aplicar transformaciones al clip de película para transformar el mapa de bits, por ejemplo, mediante la propiedad `matrix` del clip de película.

El ajuste en píxeles fuerza la posición del mapa de bits al valor de píxel completo más cercano en lugar de situarlo en un píxel parcial. Hay tres modos de ajuste en píxeles:

- El modo automático (Auto) realiza el ajuste en píxeles siempre que el mapa de bits no esté expandido o girado.
- El modo Always realiza el ajuste en píxeles independientemente de la expansión o rotación del mapa de bits.
- El modo Never desactiva el ajuste de píxeles para el clip de película.

El modo de suavizado afecta al aspecto de la imagen cuando se modifica la escala.

Disponibilidad: ActionScript 1.0; Flash Player 8

Parámetros

bmp:`flash.display.BitmapData` - Una imagen de mapa de bits transparente u opaca.

depth:`Number` - Un entero que especifica el nivel de profundidad del clip de película en el que debe colocarse la imagen de mapa de bits.

pixelSnapping:`String` [opcional] - Los modos de ajuste en píxeles son `auto`, `always` y `never`. El modo predeterminado es `auto`.

smoothing:`Boolean` [opcional] - El modo de suavizado es `true` para activado y `false` para desactivado. El modo predeterminado es desactivado.

Ejemplo

El ejemplo siguiente asocia un mapa de bits muy básico a un clip de película:

```
import flash.display.*;

this.createEmptyMovieClip("bmp1", this.getNextHighestDepth());
var bmpData1:BitmapData = new BitmapData(200, 200, false, 0xaa3344);
bmp1.attachBitmap(bmpData1, 2, "auto", true);
```

Si el archivo SWF incluye un componente de la versión 2, utilice la clase DepthManager de los componentes de la versión 2 en lugar del método `MovieClip.getNextHighestDepth()` utilizado en este ejemplo.

attachMovie (método MovieClip.attachMovie)

```
public attachMovie(id:String, name:String, depth:Number,
  [initObject:Object]) : MovieClip
```

Localiza un símbolo de la biblioteca y lo asocia al clip de película. Utilice `MovieClip.removeMovieClip()` o `MovieClip.unloadMovie()` para eliminar un archivo SWF asociado mediante `attachMovie()`.

Puede ampliar los métodos y los controladores de eventos de la clase MovieClip creando una subclase.

Disponibilidad: ActionScript 1.0; Flash Player 5

Parámetros

id:String - Nombre de vinculación del símbolo de clip de película de la biblioteca que va a asociarse a un clip de película del escenario. Es el nombre introducido en el campo Identificador del cuadro de diálogo Propiedades de vinculación.

name:String - Nombre de instancia exclusivo del clip de película que va a asociarse al clip de película.

depth:Number - Entero que especifica el nivel de profundidad en el que está situado el archivo SWF.

initObject:Object [opcional] - (Admitido en Flash Player 6 y versiones posteriores) Objeto que contiene propiedades con las que rellenar el clip de película recién asociado. Este parámetro permite a los clips de película creados dinámicamente recibir parámetros de clip. Si initObject no es un objeto, se omite. Todas las propiedades de initObject se copian en la nueva instancia. Las propiedades especificadas con initObject están disponibles para la función constructora.

Valor devuelto

MovieClip - Una referencia a la nueva instancia creada.

Ejemplo

El ejemplo siguiente asocia el símbolo con el identificador de vinculación circle a la instancia de clip de película, situada en el escenario en el archivo SWF:

```
this.attachMovie("circle", "circle1_mc", this.getNextHighestDepth());
this.attachMovie("circle", "circle2_mc", this.getNextHighestDepth(),
  {_x:100, _y:100});
```

El método MovieClip.getNextHighestDepth() utilizado en este ejemplo necesita Flash Player 7 o una versión posterior. Si el archivo SWF incluye un componente de la versión 2, utilice la clase DepthManager de los componentes de la versión 2 en lugar del método MovieClip.getNextHighestDepth().

Véase también

removeMovieClip (método MovieClip.removeMovieClip), unloadMovie (método MovieClip.unloadMovie), Función removeMovieClip

beginBitmapFill (método MovieClip.beginBitmapFill)

```
public beginBitmapFill(bmp:BitmapData, [matrix:Matrix], [repeat:Boolean],
  [smoothing:Boolean]) : Void
```

Rellena un área de dibujo con una imagen de mapa de bits. El mapa de bits rellena el área a base de repeticiones o mosaicos.

Disponibilidad: ActionScript 1.0; Flash Player 8

Parámetros

bmp:flash.display.BitmapData - Una imagen de mapa de bits transparente u opaca.

matrix:flash.geom.Matrix [opcional] - Objeto matrix (de la clase flash.geom.Matrix), que se puede utilizar para definir transformaciones en el mapa de bits. Por ejemplo, es posible utilizar la siguiente matriz para rotar un mapa de bits 45 grados (pi/4 radianes):

```
var matrix = new flash.geom.Matrix();
matrix.rotate(Math.PI/4);
```

repeat:Boolean [opcional] - Si el valor es `true`, la imagen de mapa de bits se repite en un patrón. Si el valor es `false`, la imagen no se repite y los bordes del mapa de bits se utilizan para cualquier área de relleno que se extienda más allá del mapa de bits.

Por ejemplo, considere el mapa de bits siguiente (un patrón de tablero de ajedrez de 20 x 20 píxeles):

Cuando `repeat` se define en `true` (como en el ejemplo siguiente), el relleno repite el mapa de bits:

Cuando `repeat` se define en `false`, el relleno de mapa de bits utiliza los píxeles del borde para el área de relleno fuera del mapa de bits:

smoothing:Boolean [opcional] - Si el valor es `false` , las imágenes de mapa de bits que aumentan de escala se representan con un algoritmo de vecino más cercano y se obtiene un aspecto pixelado. Si el valor es `true`, las imágenes de mapa de bits que aumentan de escala se representan con un algoritmo de interpolación bilineal. La representación con el algoritmo de vecino más cercano suele ser mucho más rápida. El valor predeterminado de este parámetro es `false`.

Ejemplo

El código siguiente define un mapa de bits sencillo y después utiliza `beginBitmapFill()` para rellenar un clip de película con un mosaico de ese mapa de bits:

```
import flash.display.*;
import flash.geom.*;

var bmpd:BitmapData = new BitmapData(20,20);
var rect1:Rectangle = new Rectangle(0,0,10,10);
var rect2:Rectangle = new Rectangle(0, 10, 10, 20);
var rect3:Rectangle = new Rectangle(10, 0, 20, 10);
var rect4:Rectangle = new Rectangle(10, 10, 20, 20);
bmpd.fillRect(rect1, 0xAA0000FF);
bmpd.fillRect(rect2, 0xAA00FF00);
bmpd.fillRect(rect3, 0xAAFF0000);
bmpd.fillRect(rect4, 0xAA999999);

this.createEmptyMovieClip("bmp_fill_mc", this.getNextHighestDepth());
with (bmp_fill_mc) {
  matrix = new Matrix();
  matrix.rotate(Math.PI/8);
  repeat = true;
  smoothing = true;
  beginBitmapFill(bmpd, matrix, repeat, smoothing);
  moveTo(0, 0);
  lineTo(0, 60);
  lineTo(60, 60);
  lineTo(60, 0);
  lineTo(0, 0);
  endFill();
}

bmp_fill_mc._xscale = 200;
bmp_fill_mc._yscale = 200;
```

Si el archivo SWF incluye un componente de la versión 2, utilice la clase DepthManager de los componentes de la versión 2 en lugar del método `MovieClip.getNextHighestDepth()` utilizado en este ejemplo.

beginFill (método MovieClip.beginFill)

```
public beginFill(rgb:Number, [alpha:Number]) : Void
```

Indica el comienzo de un nuevo trazo de dibujo. Si existe un trazado abierto (es decir, si la posición de dibujo actual no es la misma que la posición anterior especificada en un método `MovieClip.moveTo()` y tiene un relleno asociado, dicho trazado se cerrará con una línea y luego se rellenará. Se trata de un resultado similar al que se obtiene cuando se llama a `MovieClip.endFill()`.

Puede ampliar los métodos y los controladores de eventos de la clase MovieClip creando una subclase.

Disponibilidad: ActionScript 1.0; Flash Player 6

Parámetros

rgb:Number - Valor de color hexadecimal (por ejemplo, rojo es 0xFF0000, azul es 0x0000FF, etc.). Si no se proporciona este valor o es undefined, no se crea un relleno.

alpha:Number [opcional] - Entero entre 0 y 100 que especifica el valor alfa del relleno. Si no se proporciona este valor, se utiliza 100 (continuo). Si el valor es inferior a 0, Flash utiliza 0. Si el valor es mayor que 100, Flash utiliza 100.

Ejemplo

El ejemplo siguiente crea un cuadrado con relleno rojo en el escenario:

```
this.createEmptyMovieClip("square_mc", this.getNextHighestDepth());
square_mc.beginFill(0xFF0000);
square_mc.moveTo(10, 10);
square_mc.lineTo(100, 10);
square_mc.lineTo(100, 100);
square_mc.lineTo(10, 100);
square_mc.lineTo(10, 10);
square_mc.endFill();
```

El método `MovieClip.getNextHighestDepth()` utilizado en este ejemplo necesita Flash Player 7 o una versión posterior. Si el archivo SWF incluye un componente de la versión 2, utilice la clase DepthManager de los componentes de la versión 2 en lugar del método `MovieClip.getNextHighestDepth()`.

También hay un ejemplo en el archivo drawingapi.fla de la carpeta Samples\ActionScript\DrawingAPI. La lista siguiente muestra rutas habituales a esta carpeta:

- Windows: \Archivos de programa\Macromedia\Flash 8\Samples and Tutorials\Samples\
- Macintosh: HD/Applications/Macromedia Flash 8/Samples and Tutorials/Samples/

Véase también

`moveTo (método MovieClip.moveTo)`, `endFill (método MovieClip.endFill)`, `beginGradientFill (método MovieClip.beginGradientFill)`

beginGradientFill (método MovieClip.beginGradientFill)

```
public beginGradientFill(fillType:String, colors:Array, alphas:Array,
  ratios:Array, matrix:Object, [spreadMethod:String],
  [interpolationMethod:String], [focalPointRatio:Number]) : Void
```

Indica el comienzo de un nuevo trazo de dibujo. Si el primer parámetro es `undefined`, o si no se han pasado parámetros, el trazado no tendrá relleno. Si existe un trazado abierto (es decir, si la posición de dibujo actual no es la misma que la posición anterior especificada en un método `MovieClip.moveTo()`) y tiene un relleno asociado, dicho trazado se cerrará con una línea y luego se rellenará. Se trata de un resultado similar al que se obtiene cuando se llama a `MovieClip.endFill()`.

Este método falla si se da una de las situaciones siguientes:

- El número de elementos de los parámetros `colors`, `alphas` y `ratios` no coincide.
- El parámetro `fillType` no es `"linear"` ni `"radial"`.
- Alguno de los campos del objeto correspondiente al parámetro `matrix` no están presentes o no son válidos.

Puede ampliar los métodos y los controladores de eventos de la clase MovieClip creando una subclase.

Disponibilidad: ActionScript 1.0; Flash Player 6

Parámetros

`fillType``:String` - Los valores válidos son la cadena `"linear"` y al cadena `"radial"`.

`colors``:Array` - Matriz de valores de color RGB hexadecimales que se utiliza en el degradado (por ejemplo, rojo es 0xFF0000, azul es 0x0000FF, etc.). Puede utilizar hasta 15 colores. Para cada color, asegúrese de especificar un valor correspondiente en los parámetros `alphas` y `ratios`.

`alphas``:Array` - Matriz de valores alfa para los colores correspondientes de la matriz `colors`; los valores válidos son del 0 al 100. Si el valor es inferior a 0, Flash utiliza 0. Si el valor es mayor que 100, Flash utiliza 100.

`ratios``:Array` - Una matriz de proporciones de distribución de colores; los valores válidos son del 0 al 255. Este valor define el porcentaje de la anchura donde el color se muestrea al 100%. Introduzca un valor con el parámetro `colors`.

Por ejemplo, en el caso de un degradado lineal con dos colores, azul y verde, la siguiente figura ilustra la colocación de los colores en el degradado en función de los diferentes valores en la matriz `ratios`:

`ratios`	**Degradado**
`[0, 127]`	
`[0, 255]`	
`[127, 255]`	

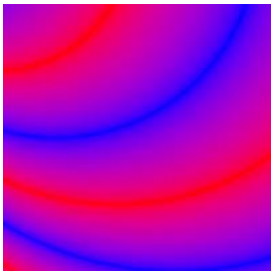
Los valores de la matriz deben aumentar secuencialmente; por ejemplo, `[0, 63, 127, 190, 255]`.

`matrix``:Object` - Una matriz de transformación que puede encontrarse en una de tres formas:

- Objeto matrix (sólo se admite en Flash Player 8 y versiones posteriores) definido en la clase flash.geom.Matrix. La clase flash.geom.Matrix incluye un método `createGradientBox()`, que permite configurar cómodamente la matriz para utilizarla con el método `beginGradientFill()` de la clase MovieClip. Macromedia recomienda utilizar esta forma de matriz con Flash Player 8 y versiones posteriores.
- El ejemplo siguiente utiliza el método `beginGradientFill()` con un parámetro `matrix`de este tipo:

```
import flash.geom.*

this.createEmptyMovieClip("gradient_mc", this.getNextHighestDepth());
with (gradient_mc)
{
  colors = [0xFF0000, 0x0000FF];
  fillType = "radial"
  alphas = [100, 100];
  ratios = [0, 0xFF];
  spreadMethod = "reflect";
  interpolationMethod = "linearRGB";
  focalPointRatio = 0.9;
  matrix = new Matrix();
  matrix.createGradientBox(100, 100, Math.PI, 0, 0);
  beginGradientFill(fillType, colors, alphas, ratios, matrix,
    spreadMethod, interpolationMethod, focalPointRatio);
  moveTo(100, 100);
  lineTo(100, 300);
  lineTo(300, 300);
  lineTo(300, 100);
  lineTo(100, 100);
  endFill();
}
```

- Este código dibuja la imagen siguiente en la pantalla:



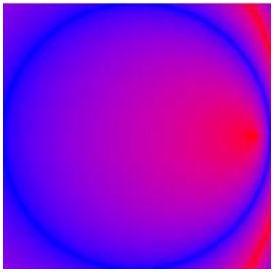
- Puede utilizar las propiedades `a`, `b`, `c`, `d`, `e`, `f`, `g`, `h` y `i` para describir una matriz de 3 x 3 de la forma siguiente:

```
a b c
d e f
g h i
```

- *Nota:* Con Flash Player 8 y versiones posteriores, Macromedia recomienda definir el parámetro matrix con la forma de objeto `flash.geom.Matrix` (descrito en el primer punto de viñeta de esta lista).
- El ejemplo siguiente utiliza el método `beginGradientFill()` con un parámetro `matrix`de este tipo:

```
this.createEmptyMovieClip("gradient_mc", this.getNextHighestDepth());
with (gradient_mc)
{
  colors = [0xFF0000, 0x0000FF];
  fillType = "radial"
  alphas = [100, 100];
  ratios = [0, 0xFF];
  spreadMethod = "reflect";
  interpolationMethod = "linearRGB";
  focalPointRatio = 0.9;
  matrix = {a:200, b:0, c:0, d:0, e:200, f:0, g:200, h:200, i:1};
  beginGradientFill(fillType, colors, alphas, ratios, matrix, spreadMethod,
interpolationMethod, focalPointRatio);
  moveTo(100, 100);
  lineTo(100, 300);
  lineTo(300, 300);
  lineTo(300, 100);
  lineTo(100, 100);
  endFill();
}
```

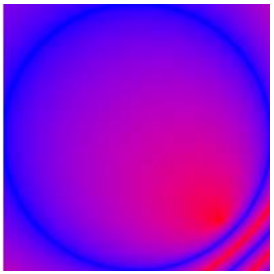
- Este código dibuja la imagen siguiente en la pantalla:



- Un objeto con las propiedades siguientes: `matrixType`, `x`, `y`, `w`, `h`, `r`.
- Las propiedades indican lo siguiente: `matrixType` es la cadena `"box"`, `x` es la posición horizontal de la esquina superior izquierda del degradado con respecto al punto de registro del clip principal, `y` es la posición vertical de la esquina superior izquierda del degradado con respecto al punto de registro del clip principal, `w` es la anchura del degradado, `h` es la altura del degradado y `r` es la rotación en radianes del degradado.
- *Nota:* Con Flash Player 8 y versiones posteriores, Macromedia recomienda definir el parámetro `matrix` con la forma de objeto `flash.geom.Matrix` (descrito en el primer punto de viñeta de esta lista).
- El ejemplo siguiente utiliza el método `beginGradientFill()` con un parámetro `matrix`de este tipo:

```
this.createEmptyMovieClip("gradient_mc", this.getNextHighestDepth());
with (gradient_mc)
{
  colors = [0xFF0000, 0x0000FF];
  fillType = "radial"
  alphas = [100, 100];
  ratios = [0, 0xFF];
  spreadMethod = "reflect";
  interpolationMethod = "linearRGB";
  focalPointRatio = 0.9;
  matrix = {matrixType:"box", x:100, y:100, w:200, h:200, r:(45/
  180)*Math.PI};
  beginGradientFill(fillType, colors, alphas, ratios, matrix,
    spreadMethod, interpolationMethod, focalPointRatio);
  moveTo(100, 100);
  lineTo(100, 300);
  lineTo(300, 300);
  lineTo(300, 100);
  lineTo(100, 100);
  endFill();
}
```

- Este código dibuja la imagen siguiente en la pantalla:

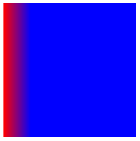


spreadMethod:String [opcional] - Añadido en Flash Player 8. "pad", "reflect" o "repeat", que controla el modo del relleno con degradado. El valor predeterminado es "pad".

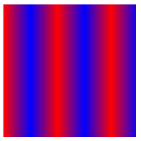
Por ejemplo, imagine un degradado lineal sencillo entre dos colores:

```
import flash.geom.*;
var fillType:String = "linear"
var colors:Array = [0xFF0000, 0x0000FF];
var alphas:Array = [100, 100];
var ratios:Array = [0x00, 0xFF];
var matrix:Matrix = new Matrix();
matrix.createGradientBox(20, 20, 0, 0, 0);
var spreadMethod:String = "pad";
this.beginGradientFill(fillType, colors, alphas, ratios, matrix,
  spreadMethod);
this.moveTo(0, 0);
this.lineTo(0, 100);
this.lineTo(100, 100);
this.lineTo(100, 0);
this.lineTo(0, 0);
this.endFill();
```

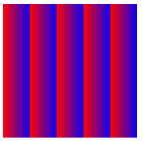
Dado que en este caso se utiliza `"pad"` como método de extensión, el relleno con degradado tendrá este aspecto:



Si tuviera que utilizar `"reflect"` como método de extensión, el relleno con degradado tendría este aspecto:

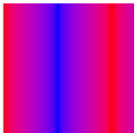


Si tuviera que utilizar `"repeat"` como método de extensión, el relleno con degradado tendría este aspecto:

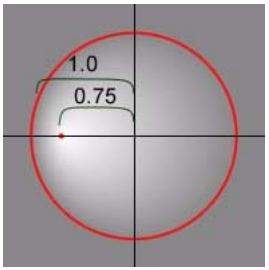


`interpolationMethod``:String` [opcional] - Añadido a Flash Player 8. "RGB" o "linearRGB". Con "linearRGB" los colores se distribuyen de forma lineal en el degradado. El valor predeterminado es "RGB".

Por ejemplo, imagine un degradado lineal sencillo entre dos colores (con el parámetro `spreadMethod` definido en "reflect"). Los diferentes métodos de interpolación afectan al aspecto de este modo:

	
`"linearRGB"`	`"RGB"`

focalPointRatio:Number [opcional] - Añadido en Flash Player 8. Número que controla la ubicación del punto focal del degradado. El valor 0 significa que el punto focal está en el centro. El valor 1 significa que el punto focal se encuentra en un borde del círculo del degradado. El valor -1 significa que el punto focal se encuentra en el otro borde del círculo del degradado. Un valor inferior a -1 o mayor que 1 se redondea a -1 o 1. El siguiente ejemplo muestra `focalPointRatio` definido en 0,75:



Ejemplo

El código siguiente crea un efecto de sombra esférica:

```
import flash.geom.*
this.createEmptyMovieClip("gradient_mc", this.getNextHighestDepth());
with (gradient_mc)
  {
    fillType = "radial"
  colors = [0x000000, 0xFFFFFF];
    alphas = [50, 90];
    ratios = [0, 0xFF];
    spreadMethod = "pad";
    interpolationMethod = "RGB";
  focalPointRatio = 0.3;
  matrix = new Matrix();
  matrix.createGradientBox(100, 100, 0, 0, 0);
  beginGradientFill(fillType, colors, alphas, ratios, matrix,
    spreadMethod, interpolationMethod, focalPointRatio);
  moveTo(0, 0);
  lineTo(0, 100);
  lineTo(100, 100);
  lineTo(100, 0);
  lineTo(0, 0);
  endFill();
}
```

El código dibuja lo siguiente (la escala de la ilustración se ha adaptado al 50%):

Si el archivo SWF incluye un componente de la versión 2, utilice la clase DepthManager de los componentes de la versión 2 en lugar del método `MovieClip.getNextHighestDepth()` utilizado en este ejemplo.

Véase también

`createGradientBox (método Matrix.createGradientBox)`, `beginFill (método MovieClip.beginFill)`, `endFill (método MovieClip.endFill)`, `lineStyle (método MovieClip.lineStyle)`, `lineTo (método MovieClip.lineTo)`, `moveTo (método MovieClip.moveTo)`

blendMode (propiedad MovieClip.blendMode)

```
public blendMode : Object
```

Modo de mezcla de este clip de película. El modo de mezcla afecta al aspecto del clip de película cuando se encuentra en una capa encima de otro objeto de la pantalla.

Flash Player aplica la propiedad `blendMode` a cada píxel del clip de película. Cada píxel consta de los tres colores primarios (o rojo, verde y azul). Cada color primario tiene un valor comprendido entre 0x00 y 0xFF. Flash Player compara cada color primario de un píxel del clip de película con el color correspondiente del píxel en el fondo. Por ejemplo, si `blendMode` se define en `"lighten"`, Flash Player comparar el valor del rojo del clip de película con el valor del rojo del fondo, y utilizará el color más claro de los dos como valor del componente rojo para mostrar el color.

La tabla siguiente describe los valores de `blendMode`: Para definir la propiedad `blendMode`, puede emplear un entero del 1 al 14 o una cadena. Las ilustraciones de la tabla muestran los valores `blendMode` aplicados a un clip de película circular (2) superpuesto a otro objeto (1) de la pantalla.

Valor entero	Valor de la cadena	Ilustración	Descripción
1	`"normal"`		El clip de película aparece delante del fondo. Los valores de píxel del clip de película anulan a los del fondo. Cuando el clip de película sea transparente, el fondo será visible.
2	`"layer"`		Fuerza la creación de un búfer temporal para la composición previa del clip de película. Esto se hace automáticamente si hay más de un objeto secundario en un clip de película y se ha seleccionado un modo `blendMode` diferente de `"normal"` para el objeto secundario.
3	`"multiply"`		Multiplica los valores de los colores primarios del clip de película por los del color del fondo, a continuación, normaliza el resultado dividiéndolo por 0xFF, con esto se obtienen unos colores más oscuros. Este valor se suele utilizar para efectos de sombras y profundidad. Por ejemplo, si un color primario (como el rojo) de un píxel del clip de película y el color correspondiente del píxel del fondo tienen ambos el valor 0x88, el resultado de la multiplicación será 0x4840. Al dividir por 0xFF el resultado es 0x48 para dicho color primario, consiguiéndose un tono más oscuro que el del clip de película o el del fondo.

Valor entero	Valor de la cadena	Ilustración	Descripción
4	`"screen"`		Multiplica el complemento (inverso) del color del clip de película por el complemento del color de fondo, con lo que se obtiene un efecto de decoloración. Este valor se suele utilizar para los resaltados o para eliminar áreas de color negro del clip de película.
5	`"lighten"`		Selecciona el color primario más claro del clip de película y los del fondo (los que tengan valores mayores). Este valor suele emplearse para el tipo de superposición. Por ejemplo, si el clip de película tiene un píxel con un valor RGB de 0xFFCC33, y el píxel del fondo tiene un valor RGB de 0xDDF800, el valor RGB resultante para el píxel mostrado es de 0xFFF833 (debido a que 0xFF > 0xDD, 0xCC < 0xF8 y 0x33 > 0x00 = 33).
6	`"darken"`		Selecciona el color primario más oscuro del clip de película y los del fondo (los que tengan valores menores). Este valor suele emplearse para el tipo de superposición. Por ejemplo, si el clip de película tiene un píxel con un valor RGB de 0xFFCC33, y el píxel del fondo tiene un valor RGB de 0xDDF800, el valor RGB resultante para el píxel mostrado es de 0xDDCC00 (debido a que 0xFF > 0xDD, 0xCC < 0xF8 y 0x33 > 0x00 = 33).

Valor entero	Valor de la cadena	Ilustración	Descripción
7	`"difference"`		Compara los colores primarios del clip de película con los del fondo y resta el valor más oscuro del valor más claro de los colores primarios. Este valor suele emplearse en los colores más vivos. Por ejemplo, si el clip de película tiene un píxel con un valor RGB de 0xFFCC33, y el píxel del fondo tiene un valor RGB de 0xDDF800, el valor RGB resultante para el píxel mostrado es de 0x222C33 (debido a que 0xFF - 0xDD = 0x22, 0xF8 - 0xCC = 0x2C y 0x33 - 0x00 = 0x33).
8	`"add"`		Agrega los valores de los colores primarios del clip de película a los del fondo con un límite de 0xFF. Este valor suele emplearse para animar una disolución de aclarado entre dos objetos. Por ejemplo, si el clip de película tiene un píxel con un valor RGB de 0xAAA633, y el píxel del fondo tiene un valor RGB de 0xDD2200, el valor RGB resultante para el píxel mostrado es de 0xFFC833 (debido a que 0xAA + 0xDD › 0xFF, 0xA6 + 0x22 = 0xC8 y 0x33 + 0x00 = 0x33).

Valor entero	Valor de la cadena	Ilustración	Descripción
9	`"subtract"`		Resta los valores de los colores primarios del clip de película a los del fondo, aplicando un límite inferior de 0. Suele utilizarse para animar una disolución de oscurecimiento entre dos imágenes. Por ejemplo, si el clip de película tiene un píxel con un valor RGB de 0xAA2233, y el píxel del fondo tiene un valor RGB de 0xDDA600, el valor RGB resultante para el píxel mostrado es de 0x338400 (debido a que 0xDD - 0xAA = 0x33, 0xA6 - 0x22 = 0x84 y 0x00 - 0x33 < 0x00).
10	`"invert"`		Invierte el fondo.
11	`"alpha"`		Aplica el valor alfa de cada píxel del clip de película al fondo. Esto requiere que se aplique el valor `blendMode` de "capa" al clip de película principal. Por ejemplo, en la ilustración, el clip de película principal, un fondo blanco, tiene `blendMode = "layer"`.
12	`"erase"`		Borra el fondo en función del valor alfa del clip de película. Esto requiere que se aplique `blendMode` de "capa" al clip de película principal. Por ejemplo, en la ilustración, el clip de película principal, un fondo blanco, tiene `blendMode = "layer"`.

Valor entero	Valor de la cadena	Ilustración	Descripción
13	`"overlay"`		Ajusta el color de cada mapa de bits en función de la oscuridad del fondo. Si el fondo es más claro que un 50% de gris, los colores del clip de película y del fondo se tamizan, consiguiéndose un color más claro. Si el fondo es más oscuro que un 50% de gris, los colores del clip de película y del fondo se multiplican, consiguiéndose un color más oscuro. Este valor suele emplearse para conseguir efectos de sombreado.
14	`"hardlight"`		Ajusta el color de cada mapa de bits en función de la oscuridad del clip de película. Si el clip de película es más claro que un 50% de gris, los colores del clip de película y del fondo se tamizan, consiguiéndose un color más claro. Si el clip de película es más oscuro que un 50% de gris, los colores del clip de película y del fondo se multiplican, consiguiéndose un color más oscuro. Este valor suele emplearse para conseguir efectos de sombreado.

Si intenta establecer la propiedad `blendMode` en cualquier otro valor, Flash Player la establece en `"normal"`.

Sin embargo, tenga en cuenta que si define la propiedad en un entero, Flash Player convierte el valor en la versión de cadena correspondiente:

```
this.createEmptyMovieClip("mclip", this.getNextHighestDepth());
mclip.blendMode = 8;
trace (mclip.blendMode) // add
```

Disponibilidad: ActionScript 1.0; Flash Player 8

Ejemplo

En el ejemplo siguiente se definen dos clips de película con rellenos de degradado y cada segundo cambia el modo de mezcla del clip que se encuentra en primer plano. Para que el modo de mezcla `"alpha"` muestre un efecto, el degradado para el clip de película `mc2` incluye un rango de proporciones alfa y el modo de mezcla `"layer"` se aplica al clip de película principal (`this.blendMode="layer"`).

```
  this.createEmptyMovieClip("mc1", this.getNextHighestDepth());
  this.createEmptyMovieClip("mc2", this.getNextHighestDepth());
  this.blendMode="layer";
  this.createTextField("blendLabel", this.getNextHighestDepth(), 50, 150,
  100, 100)

  fillClip(mc1, 0x00AA00, 0x22FFFF, 100, 100)
  fillClip(mc2, 0xFF0000, 0x2211FF, 100, 50)
  mc2._x = 33;
  mc2._y = 33;

  var blendModeIndex = 0;

  setInterval(changeBlendMode, 1000);
  function changeBlendMode()
  {
    mc2.blendMode = blendModeIndex % 14 + 1 ;
      // values 1 - 14
  blendLabel.text = (blendModeIndex% 14 + 1) + ": " + mc2.blendMode;
    blendModeIndex++;
  }

  function fillClip(mc:MovieClip, color1:Number, color2:Number,
        alpha1:Number, alpha2: Number)
{
    matrix = {a:100, b:0, c:0, d:0, e:100, f:0, g:50, h:20, i:1};
    mc.beginGradientFill("linear", [color1, color2], [alpha1, alpha2], [0,
  0xFF], matrix);
    mc.lineStyle(8,0x888888,100)
    mc.moveTo(0, 0);
    mc.lineTo(0, 100);
    mc.lineTo(100, 100);
    mc.lineTo(100, 0);
    mc.lineTo(0, 0);
    mc.endFill();
  }
```

Si el archivo SWF incluye un componente de la versión 2, utilice la clase DepthManager de los componentes de la versión 2 en lugar del método `MovieClip.getNextHighestDepth()` utilizado en el ejemplo anterior.

cacheAsBitmap (propiedad MovieClip.cacheAsBitmap)

```
public cacheAsBitmap : Boolean
```

Si se define como `true`, Flash Player deja en caché una representación interna de mapa de bits del clip de película. Esto puede aumentar el rendimiento de los clips de película que tienen contenido vectorial complejo.

Todos los datos vectoriales de un clip de película con un mapa de bits en caché se dibujan en el mapa de bits, no en el escenario principal. A continuación, el mapa de bits se copia en el escenario principal como píxeles no expandidos ni rotados, pero ajustados a los límites de píxeles más cercanos. Los píxeles se asignan de uno a uno con el objeto principal. Si los límites del mapa de bits cambian, el mapa de bits se vuelve a crear en lugar de expandirse.

No se crea ningún mapa de bits interno a menos que la propiedad `cacheAsBitmap` esté definida en `true`.

Después de definir la propiedad `cacheAsBitmap` de un clip de película en `true`, la representación no varía, pero el clip de película realiza automáticamente el ajuste en píxeles. La velocidad de animación puede ser notablemente mayor según la complejidad del contenido vectorial.

La propiedad `cacheAsBitmap` se define automáticamente en `true` siempre que se aplica un filtro a un clip de película (cuando su matriz `filter` no está vacía). Si un clip de película tiene un filtro aplicado, `cacheAsBitmap` aparece como `true` para dicho clip de película, incluso si se define la propiedad en `false`. Si se eliminan todos los filtros de un clip de película, el ajuste `cacheAsBitmap` adopta el último valor *definido*.

En los casos siguientes y aunque la propiedad `cacheAsBitmap` se haya definido en `true`, un clip de película no utiliza un mapa de bits y se representa a partir de datos vectoriales:

- El mapa de bits es demasiado grande: mayor que 2.880 píxeles en cualquier dirección.
- No se puede asignar memoria para el mapa de bits (se produce un error de memoria insuficiente).

La propiedad `cacheAsBitmap` es idónea con clips de película que tienen sobre todo contenido estático y no cambian de escala ni giran con frecuencia. Con estos clips, `cacheAsBitmap` puede aumentar el rendimiento cuando se convierte el clip de película (cuando se modifica la posición de *x* e *y*).

Disponibilidad: ActionScript 1.0; Flash Player 8

Ejemplo

El ejemplo siguiente añade una sombra a una instancia de clip de película. Después, averigua el valor de la propiedad `cacheAsBitmap` que se define como `true` cuando se aplica un filtro.

```
import flash.filters.DropShadowFilter;

var container:MovieClip = setUpShape();
trace(container.cacheAsBitmap); // false
var dropShadow:DropShadowFilter = new DropShadowFilter(6, 45, 0x000000, 50,
  5, 5, 1, 2, false, false, false);
container.filters = new Array(dropShadow);
trace(container.cacheAsBitmap); // true

function setUpShape():MovieClip {
  var mc:MovieClip = this.createEmptyMovieClip("container",
  this.getNextHighestDepth());
  mc._x = 10;
  mc._y = 10;
  var w:Number = 50;
  var h:Number = 50;
  mc.beginFill(0xFFCC00);
  mc.lineTo(w, 0);
  mc.lineTo(w, h);
  mc.lineTo(0, h);
  mc.lineTo(0, 0);
  mc.endFill();
  return mc;
}
```

Véase también

`opaqueBackground (propiedad MovieClip.opaqueBackground)`, `cacheAsBitmap (propiedad MovieClip.cacheAsBitmap)`

clear (método MovieClip.clear)

```
public clear() : Void
```

Elimina todos los gráficos creados en tiempo de ejecución empleando los métodos de dibujo del clip de película, incluidos los estilos de línea especificados con `MovieClip.lineStyle()`. Las formas y líneas que se dibujan manualmente durante la etapa de edición (con las herramientas de dibujo de Flash) no se ven afectadas.

Disponibilidad: ActionScript 1.0; Flash Player 6

Ejemplo

El ejemplo siguiente dibuja un recuadro en el escenario: Cuando el usuario hace clic en el gráfico del recuadro, elimina el gráfico del escenario.

```
this.createEmptyMovieClip("box_mc", this.getNextHighestDepth());
box_mc.onRelease = function() {
  this.clear();
};
drawBox(box_mc, 10, 10, 320, 240);
function drawBox(mc:MovieClip, x:Number, y:Number, w:Number, h:Number):Void
  {
  mc.lineStyle(0);
  mc.beginFill(0xEEEEEE);
  mc.moveTo(x, y);
  mc.lineTo(x+w, y);
  mc.lineTo(x+w, y+h);
  mc.lineTo(x, y+h);
  mc.lineTo(x, y);
  mc.endFill();
}
```

El método `MovieClip.getNextHighestDepth()` utilizado en este ejemplo necesita Flash Player 7 o una versión posterior. Si el archivo SWF incluye un componente de la versión 2, utilice la clase DepthManager de los componentes de la versión 2 en lugar del método `MovieClip.getNextHighestDepth()`.

También hay un ejemplo en el archivo drawingapi.fla de la carpeta Samples\ActionScript\DrawingAPI. La lista siguiente muestra rutas habituales a esta carpeta:

- Windows: \Archivos de programa\Macromedia\Flash 8\Samples and Tutorials\Samples\ActionScript
- Macintosh: HD/Applications/Macromedia Flash 8/Samples and Tutorials/Samples/ActionScript

Véase también

`lineStyle (método MovieClip.lineStyle)`

createEmptyMovieClip (método MovieClip.createEmptyMovieClip)

```
public createEmptyMovieClip(name:String, depth:Number) : MovieClip
```

Crea un clip de película vacío como elemento secundario de un clip de película existente. Este método se comporta de forma similar al método `attachMovie()`, aunque no es necesario proporcionar un identificador de vinculación externo para el nuevo clip de película. El punto de registro de un clip de película vacío recién creado es la esquina superior izquierda. Este método falla si falta cualquiera de los parámetros.

Puede ampliar los métodos y los controladores de eventos de la clase MovieClip creando una subclase.

Disponibilidad: ActionScript 1.0; Flash Player 6

Parámetros

`name``:String` - Cadena que identifica el nombre de instancia del nuevo clip de película.

`depth``:Number` - Entero que especifica la profundidad del nuevo clip de película.

Valor devuelto

`MovieClip` - Una referencia al nuevo clip de película creado.

Ejemplo

El ejemplo siguiente crea un objeto MovieClip vacío denominado `container`, crea un nuevo objeto TextField en su interior y define la nueva propiedad `TextField.text`.

```
var container:MovieClip = this.createEmptyMovieClip("container",
  this.getNextHighestDepth());
var label:TextField = container.createTextField("label", 1, 0, 0, 150, 20);
label.text = "Hello World";
```

El método `MovieClip.getNextHighestDepth()` utilizado en este ejemplo necesita Flash Player 7 o una versión posterior. Si el archivo SWF incluye un componente de la versión 2, utilice la clase DepthManager de los componentes de la versión 2 en lugar del método `MovieClip.getNextHighestDepth()`.

Véase también

`attachMovie (método MovieClip.attachMovie)`

createTextField (método MovieClip.createTextField)

```
public createTextField(instanceName:String, depth:Number, x:Number,
  y:Number, width:Number, height:Number) : TextField
```

Crea un nuevo campo de texto vacío como elemento secundario del clip de película que llama a este método. Puede utilizar el método `createTextField()` para crear campos de texto mientras se reproduce un archivo SWF. El parámetro `depth` determina el nivel de profundidad del nuevo campo de texto (posición de orden z) del clip de película. Cada nivel de profundidad sólo puede contener un objeto. Si crea un nuevo campo de texto en una profundidad en la que ya hay un campo de texto, el nuevo campo de texto reemplaza al existente. Para evitar sobrescribir campos de texto existentes, utilice `MovieClip.getInstanceAtDepth()` con el fin de determinar si una profundidad ya está ocupada o el método `MovieClip.getNextHighestDepth()` para determinar la profundidad más alta no ocupada. El campo de texto se sitúa en (*x*, *y*) con las dimensiones `width` (anchura) por *`height`* (altura). Los parámetros `x` e `y` son relativos al clip de película del contenedor; estos parámetros corresponden a las propiedades _`x` e _`y` del campo de texto. Los parámetros `width` y `height` corresponden a las propiedades _`width` y _`height` del campo de texto.

Las propiedades predeterminadas de un campo de texto son las siguientes:

```
type = "dynamic"
border = false
background = false
password = false
multiline = false
html = false
embedFonts = false
selectable = true
wordWrap = false
mouseWheelEnabled = true
condenseWhite = false
restrict = null
variable = null
maxChars = null
styleSheet = undefined
tabInded = undefined
```

Un campo de texto creado con `createTextField()` recibe los ajustes del siguiente objeto TextFormat predeterminado:

```
font = "Times New Roman" // "Times" on Mac OS
size = 12
color = 0x000000
bold = false
italic = false
underline = false
url = ""
target = ""
```

```
align = "left"
leftMargin = 0
rightMargin = 0
indent = 0
leading = 0
blockIndent = 0
bullet = false
display = block
tabStops = [] // (empty array)
```

Puede ampliar los métodos y los controladores de eventos de la clase MovieClip creando una subclase.

Disponibilidad: ActionScript 1.0; Flash Player 6

Parámetros

instanceName:String - Cadena que identifica el nombre de instancia del nuevo campo de texto.

depth:Number - Entero positivo que especifica la profundidad del nuevo campo de texto.

x:Number - Entero que especifica la coordenada *x* del nuevo campo de texto.

y:Number - Entero que especifica la coordenada *y* del nuevo campo de texto.

width:Number - Entero positivo que especifica la anchura del nuevo campo de texto.

height:Number - Entero positivo que especifica la altura del nuevo campo de texto.

Valor devuelto

TextField - Flash Player 8 devuelve una referencia al objeto TextField creado. Las versiones de Flash Player anteriores a la 8 devuelven void.

Ejemplo

El ejemplo siguiente crea un campo de texto con una anchura de 300, una altura de 100, una coordenada *x* de 100, una coordenada *y* de 100, sin bordes, rojo y texto subrayado:

```
this.createTextField("my_txt", 1, 100, 100, 300, 100);
my_txt.multiline = true;
my_txt.wordWrap = true;
var my_fmt:TextFormat = new TextFormat();
my_fmt.color = 0xFF0000;
my_fmt.underline = true;
my_txt.text = "This is my first test field object text.";
my_txt.setTextFormat(my_fmt);
```

También hay un ejemplo en el archivo animation.fla de la carpeta Samples\ActionScriptAnimation. La lista siguiente muestra rutas habituales a esta carpeta:

- Windows: \Archivos de programa\Macromedia\Flash 8\Samples and Tutorials\Samples\
- Macintosh: HD/Applications/Macromedia Flash 8/Samples and Tutorials/Samples/

Véase también

`getInstanceAtDepth (método MovieClip.getInstanceAtDepth)`, `getNextHighestDepth (método MovieClip.getNextHighestDepth)`, `TextFormat`

_currentframe (propiedad MovieClip._currentframe)

```
public _currentframe : Number [read-only]
```

Devuelve el número del fotograma en el que está situada la cabeza lectora en la línea de tiempo del clip de película.

Disponibilidad: ActionScript 1.0; Flash Player 4

Ejemplo

El ejemplo siguiente utiliza la propiedad `_currentframe` para ordenar a la cabeza lectora del clip de película `actionClip_mc` que avance cinco fotogramas con respecto a su posición actual:

```
actionClip_mc.gotoAndStop(actionClip_mc._currentframe + 5);
```

curveTo (método MovieClip.curveTo)

```
public curveTo(controlX:Number, controlY:Number, anchorX:Number,
  anchorY:Number) : Void
```

Dibuja una curva con el estilo de línea actual desde la posición de dibujo actual hasta (`anchorX`, `anchorY`) utilizando el punto de control especificado con (`controlX`, `controlY`). La posición de dibujo actual se establece entonces como `anchorX`, `anchorY`). Si el clip de película en el que está dibujando incluye contenido creado con las herramientas de dibujo de Flash, las llamadas a `curveTo()` se dibujarán debajo de este contenido. Si llama a `curveTo()` antes de realizar ninguna llamada al método `moveTo()`, se adoptará la posición de dibujo predeterminada (0,0). Si falta alguno de los parámetros, este método falla y la posición de dibujo actual no cambia.

Puede ampliar los métodos y los controladores de eventos de la clase MovieClip creando una subclase.

Disponibilidad: ActionScript 1.0; Flash Player 6

Parámetros

controlX:Number - Entero que especifica la posición horizontal del punto de control con respecto al punto de registro del clip de película principal.

controlY:Number - Entero que especifica la posición vertical del punto de control con respecto al punto de registro del clip de película principal.

anchorX:Number - Entero que especifica la posición horizontal del siguiente punto de anclaje con respecto al punto de registro del clip de película principal.

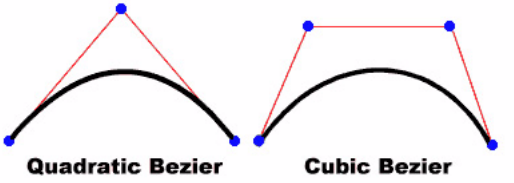
anchorY:Number - Entero que especifica la posición vertical del siguiente punto de anclaje con respecto al punto de registro del clip de película principal.

Ejemplo

El ejemplo siguiente dibuja una curva casi circular con un trazo fino continuo azul y un relleno continuo rojo:

```
this.createEmptyMovieClip("circle_mc", 1);
with (circle_mc) {
  lineStyle(0, 0x0000FF, 100);
  beginFill(0xFF0000);
  moveTo(0, 100);
  curveTo(0,200,100,200);
  curveTo(200,200,200,100);
  curveTo(200,0,100,0);
  curveTo(0,0,0,100);
  endFill();
}
```

La curva dibujada en este ejemplo es una curva cuadrática de Bézier. Las curvas cuadráticas de Bézier constan de dos puntos de anclaje y un punto de control. La curva interpola los dos puntos de anclaje y se dirige al punto de control.



En el script siguiente se utiliza el método `curveTo()` y la clase Math para crear un círculo:

```
this.createEmptyMovieClip("circle2_mc", 2);
circle2_mc.lineStyle(0, 0x000000);
drawCircle(circle2_mc, 100, 100, 100);
function drawCircle(mc:MovieClip, x:Number, y:Number, r:Number):Void {
  mc.moveTo(x+r, y);
```

```
  mc.curveTo(r+x, Math.tan(Math.PI/8)*r+y, Math.sin(Math.PI/4)*r+x,
Math.sin(Math.PI/4)*r+y);
  mc.curveTo(Math.tan(Math.PI/8)*r+x, r+y, x, r+y);
  mc.curveTo(-Math.tan(Math.PI/8)*r+x, r+y, -Math.sin(Math.PI/4)*r+x,
Math.sin(Math.PI/4)*r+y);
  mc.curveTo(-r+x, Math.tan(Math.PI/8)*r+y, -r+x, y);
  mc.curveTo(-r+x, -Math.tan(Math.PI/8)*r+y, -Math.sin(Math.PI/4)*r+x,
-Math.sin(Math.PI/4)*r+y);
  mc.curveTo(-Math.tan(Math.PI/8)*r+x, -r+y, x, -r+y);
  mc.curveTo(Math.tan(Math.PI/8)*r+x, -r+y, Math.sin(Math.PI/4)*r+x,
-Math.sin(Math.PI/4)*r+y);
  mc.curveTo(r+x, -Math.tan(Math.PI/8)*r+y, r+x, y);
}
```

También hay un ejemplo en el archivo drawingapi.fla de la carpeta Samples\ActionScript\DrawingAPI. La lista siguiente muestra rutas habituales a esta carpeta:

- Windows: \Archivos de programa\Macromedia\Flash 8\Samples and Tutorials\Samples\
- Macintosh: HD/Applications/Macromedia Flash 8/Samples and Tutorials/Samples/

Véase también

`beginFill (método MovieClip.beginFill)`, `createEmptyMovieClip (método MovieClip.createEmptyMovieClip)`, `endFill (método MovieClip.endFill)`, `lineStyle (método MovieClip.lineStyle)`, `lineTo (método MovieClip.lineTo)`, `moveTo (método MovieClip.moveTo)`, `Math`

_droptarget (propiedad MovieClip._droptarget)

```
public _droptarget : String [read-only]
```

Devuelve la ruta absoluta en notación de sintaxis con barras de la instancia de clip de película en la que se ha colocado dicho clip de película. La propiedad `_droptarget` siempre devuelve una ruta que comienza con una barra (`/`). Para comparar la propiedad `_droptarget` de una instancia con una referencia, utilice la función `eval()` para convertir el valor devuelto de la sintaxis con barras a una referencia de sintaxis con punto.

Nota: Debe realizar esta conversión si está utilizando ActionScript 2.0, ya que no admite la sintaxis con barras.

Disponibilidad: ActionScript 1.0; Flash Player 4

Ejemplo

El ejemplo siguiente evalúa la propiedad `_droptarget` de la instancia de clip de película `garbage_mc` y utiliza `eval()` para convertirla de sintaxis con barras a una referencia de sintaxis con punto. A continuación, la referencia a `garbage_mc` se compara con la referencia a la instancia de clip de película `trashcan_mc`. Si las dos referencias son equivalentes, la visibilidad de `garbage_mc` se establece en el valor `false`. Si no son equivalentes, la instancia `garbage` recupera su posición original.

```
origX = garbage_mc._x;
origY = garbage_mc._y;
garbage_mc.onPress = function() {
  this.startDrag();
};
garbage_mc.onRelease = function() {
  this.stopDrag();
  if (eval(this._droptarget) == trashcan_mc) {
  this._visible = false;
  } else {
  this._x = origX;
  this._y = origY;
  }
};
```

Véase también

startDrag (método MovieClip.startDrag), stopDrag (método MovieClip.stopDrag), Función eval

duplicateMovieClip (método MovieClip.duplicateMovieClip)

```
public duplicateMovieClip(name:String, depth:Number, [initObject:Object]) :
  MovieClip
```

Crea una instancia del clip de película especificado mientras se reproduce el archivo SWF. Los clips de película duplicados siempre comienzan a reproducirse por el fotograma 1, con independencia del fotograma en el que se encontrara el clip de película original cuando se llamó al método `duplicateMovieClip()`. Las variables del clip de película principal no se copian en el clip de película duplicado. Los clips de película que se han creado con `duplicateMovieClip()` no se duplican si efectúa una llamada a `duplicateMovieClip()` en el clip principal. Si se elimina el clip de película principal, el clip de película duplicado también se elimina. Si ha utilizado `MovieClip.loadMovie()` o la clase MovieClipLoader para cargar un clip de película, el contenido del archivo SWF no se duplica. Esto significa que no puede ahorrar ancho de banda cargando un archivo JPEG, GIF, PNG o SWF y duplicando posteriormente el clip de película.

Compare este método con la versión de función global de `duplicateMovieClip()`. La versión global de este método requiere un parámetro que especifique el clip de película de destino que duplicar. Este parámetro es innecesario con la versión de clase de MovieClip, ya que el objetivo de este método es la instancia de clip de película en la que se invoca este método. Además, la versión global de `duplicateMovieClip()` no admite el parámetro `initobject` ni el valor devuelto de una referencia a la instancia de MovieClip recién creada.

Disponibilidad: ActionScript 1.0; Flash Player 5

Parámetros

`name``:String` - Identificador exclusivo del clip de película duplicado.

`depth``:Number` - Entero exclusivo que especifica la profundidad a la que debe colocarse el nuevo clip de película. Utilice la profundidad -16384 para colocar la nueva instancia de clip de película debajo del contenido creado en el entorno de edición de Flash. Los valores comprendidos entre -16384 y -1, ambos inclusive, están reservados para utilizarse en el entorno de edición y no deben emplearse con este método. Los demás valores de profundidad válidos oscilan entre 0 y 1048575, ambos inclusive.

`initObject``:Object` [opcional] - (Admitido en Flash Player 6 y posteriores.) Objeto que contiene propiedades con las que rellenar el clip de película duplicado. Este parámetro permite a los clips de película creados dinámicamente recibir parámetros de clip. Si `initObject` no es un objeto, se omite. Todas las propiedades de `initObject` se copian en la nueva instancia. Las propiedades especificadas con `initObject` están disponibles para la función constructora.

Valor devuelto

`MovieClip` - Una referencia al clip de película duplicado (admitido en Flash Player 6 y posteriores).

Ejemplo

El ejemplo siguiente duplica varias veces un objeto MovieClip creado recientemente y detecta el destino de cada duplicado.

```
var container:MovieClip = setUpContainer();
var ln:Number = 10;
var spacer:Number = 1;
var duplicate:MovieClip;
for(var i:Number = 1; i < ln; i++) {
  var newY:Number = i * (container._height + spacer);
  duplicate = container.duplicateMovieClip("clip-" + i, i, {_y:newY});
  trace(duplicate); // _level0.clip-[number]
}

function setUpContainer():MovieClip {
  var mc:MovieClip = this.createEmptyMovieClip("container",
  this.getNextHighestDepth());
  var w:Number = 100;
  var h:Number = 20;
  mc.beginFill(0x333333);
  mc.lineTo(w, 0);
  mc.lineTo(w, h);
  mc.lineTo(0, h);
  mc.lineTo(0, 0);
  mc.endFill();
  return mc;
}
```

El método `MovieClip.getNextHighestDepth()` utilizado en este ejemplo necesita Flash Player 7 o una versión posterior. Si el archivo SWF incluye un componente de la versión 2, utilice la clase DepthManager de los componentes de la versión 2 en lugar del método `MovieClip.getNextHighestDepth()`.

Véase también

`loadMovie (método MovieClip.loadMovie)`, `removeMovieClip (método MovieClip.removeMovieClip)`, `Función duplicateMovieClip`

enabled (propiedad MovieClip.enabled)

```
public enabled : Boolean
```

Valor booleano que indica si un clip de película está activado. El valor predeterminado de `enabled` es `true`. Si `enabled` se configura con el valor `false`, ya no se invocarán los métodos callback y controladores de eventos `on` *`action`* del clip de película, al tiempo que se desactivarán los fotogramas Over, Down y Up. La propiedad `enabled` no afecta a la línea de tiempo del clip de película; si se está reproduciendo un clip de película, éste continuará reproduciéndose. El clip de película continuará recibiendo los eventos de clip de película (por ejemplo, `mouseDown`, `mouseUp`, `keyDown` y `keyUp`).

La propiedad `enabled` sólo controla las propiedades de botón de un clip de película. Puede cambiar la propiedad `enabled` en cualquier momento; el clip de película modificado se activa o desactiva de inmediato. La propiedad `enabled` puede leerse de un objeto prototipo. Si `enabled` se establece con el valor `false`, el objeto no se incluirá en el orden de tabulación automático.

Disponibilidad: ActionScript 1.0; Flash Player 6

Ejemplo

El ejemplo siguiente desactiva el clip de película `circle_mc` cuando el usuario hace clic en él:

```
circle_mc.onRelease = function() {
  trace("disabling the "+this._name+" movie clip.");
  this.enabled = false;
};
```

endFill (método MovieClip.endFill)

```
public endFill() : Void
```

Aplica un relleno a las líneas y curvas añadidas desde la última llamada a `beginFill()` o `beginGradientFill()`. Flash utiliza el relleno especificado en la anterior llamada a `beginFill()` o `beginGradientFill()`. Si la posición de dibujo actual no es la misma que la posición anterior especificada en un método `moveTo()` y se ha definido un relleno, el trazado se cerrará con una línea y luego se rellenará.

Disponibilidad: ActionScript 1.0; Flash Player 6

Ejemplo

El ejemplo siguiente crea un cuadrado con relleno rojo en el escenario:

```
this.createEmptyMovieClip("square_mc", this.getNextHighestDepth());
square_mc.beginFill(0xFF0000);
square_mc.moveTo(10, 10);
square_mc.lineTo(100, 10);
square_mc.lineTo(100, 100);
square_mc.lineTo(10, 100);
square_mc.lineTo(10, 10);
square_mc.endFill();
```

El método `MovieClip.getNextHighestDepth()` utilizado en este ejemplo necesita Flash Player 7 o una versión posterior. Si el archivo SWF incluye un componente de la versión 2, utilice la clase DepthManager de los componentes de la versión 2 en lugar del método `MovieClip.getNextHighestDepth()`.

También hay un ejemplo en el archivo drawingapi.fla de la carpeta Samples\ActionScript\DrawingAPI. La lista siguiente muestra rutas habituales a esta carpeta:

- Windows: \Archivos de programa\Macromedia\Flash 8\Samples and Tutorials\Samples\
- Macintosh: HD/Applications/Macromedia Flash 8/Samples and Tutorials/Samples/

Véase también

`beginFill (método MovieClip.beginFill)`, `beginGradientFill (método MovieClip.beginGradientFill)`, `moveTo (método MovieClip.moveTo)`

filters (propiedad MovieClip.filters)

```
public filters : Array
```

Matriz indexada que contiene todos los objetos de filtro actualmente asociados al clip de película. El paquete flash.filters contiene varias clases que definen filtros específicos que puede utilizar.

Los filtros pueden aplicarse en tiempo de diseño en la herramienta del entorno de edición de Flash o en tiempo de ejecución con código ActionScript. Para aplicar un filtro con ActionScript, debe realizar una copia temporal de toda la matriz `MovieClip.filters`, modificar la matriz temporal y volver a asignar el valor de la matriz temporal a la matriz `MovieClip.filters`. A la matriz `MovieClip.filters` no pueden añadírsele directamente objetos filter nuevos. El código siguiente no afecta al clip de película de destino, denominado `myMC`:

```
myMC.filters[0].push(myDropShadow);
```

Para añadir un filtro con ActionScript, debe seguir estos pasos (suponiendo que el clip de película de destino se denomina myMC):

- Cree un nuevo objeto filter con la función constructora de la clase de filtro que haya elegido.
- Asigne el valor de la matriz `myMC.filters` a una matriz temporal, denominada por ejemplo `myFilters`.
- Añada el nuevo objeto filter a la matriz temporal, `myFilters`.
- Asigne el valor de la matriz temporal a la matriz `myMC.filters`.

Si la matriz `filters` está vacía, no tendrá que utilizar una matriz temporal. En lugar de ello, podrá asignar directamente un literal de matriz que contenga uno o varios objetos filter que ha creado.

Para modificar un objeto filter existente, creado tanto en tiempo de diseño o como en tiempo de ejecución, debe utilizar la técnica de modificar una copia de la matriz `filters`:

- Asigne el valor de la matriz `myMC.filters` a una matriz temporal, denominada por ejemplo `myFilters`.
- Modifique la propiedad utilizando la matriz temporal, `myFilters`. Por ejemplo, si desea establecer la propiedad quality del primer filtro de la matriz, podría utilizar el código siguiente: `myList[0].quality = 1;`
- Asigne el valor de la matriz temporal a la matriz `myMC.filters`.

Para borrar los filtros de un clip de película, defina `filters` con una matriz vacía (`[]`).

Durante la carga, si un clip de película tiene un filtro asociado, se marca para guardarlo en caché como mapa de bits transparente. Desde ese momento, siempre y cuando el clip de película tenga una lista de filtros válida, el reproductor guardará el clip de película en caché como mapa de bits. Este mapa de bits de origen se utiliza como imagen de origen para los efectos de filtro. Normalmente, un clip de película tiene dos mapas de bits: uno con el clip de película de origen sin filtrar (original) y otro con la imagen final una vez filtrada. La imagen final es la que se utiliza en representaciones. Mientras el clip de película no cambie, la imagen final no necesita actualización.

Si está trabajando con una matriz `filters` que contiene varios filtros y necesita averiguar el tipo de filtro asignado a cada índice de matriz, puede conservar su propia matriz `filters` y utilizar una estructura de datos aparte para averiguar el tipo de filtro asociado a cada índice de la matriz. No hay ninguna forma sencilla de determinar el tipo de filtro asociado a cada índice de la matriz `filters`.

Disponibilidad: ActionScript 1.0; Flash Player 8

Ejemplo

El ejemplo siguiente añade un filtro de sombra a un clip de película denominado `myMC`:

```
var myDropFilter = new flash.filters.DropShadowFilter();
var myFilters:Array = myMC.filters;
myFilters.push(myDropFilter);
myMC.filters = myFilters;
```

El ejemplo siguiente cambia la configuración de `quality` del primer filtro de la matriz a 15 (este ejemplo sólo funciona si el clip de película `myMC` tiene asociado como mínimo un objeto filter):

```
var myList:Array = myMC.filters;
myList[0].quality = 15;
myMC.filters = myList;
```

Véase también

focusEnabled (propiedad MovieClip.focusEnabled)

```
public focusEnabled : Boolean
```

Si el valor es `undefined` o `false`, un clip de película no puede quedar resaltado a menos que sea un botón. Si el valor de la propiedad `focusEnabled` es `true`, el clip de película podrá seleccionarse con el teclado aunque no sea un botón.

Disponibilidad: ActionScript 1.0; Flash Player 6

Ejemplo

El ejemplo siguiente configura la propiedad `focusEnabledd` del clip de película `my_mc` en `false`:

```
my_mc.focusEnabled = false;
```

_focusrect (propiedad MovieClip._focusrect)

```
public _focusrect : Boolean
```

Valor booleano que especifica si un clip de película debe mostrar un rectángulo amarillo a su alrededor cuando se selecciona con el teclado. Esta propiedad puede sustituir a la propiedad global `_focusrect`. El valor predeterminado de la propiedad `_focusrect` de una instancia de clip de película es `null`, lo que significa que la instancia de clip de película no sustituye a la propiedad global `_focusrect`. Si la propiedad `_focusrect` de una instancia de clip de película se establece como `true` o `false`, ésta sustituirá a la configuración de la propiedad global `_focusrect` para la instancia de clip de película en cuestión.

En archivos SWF de Flash Player 4 o Flash Player 5, la propiedad `_focusrect` controla la propiedad global `_focusrect`. Se trata de un valor booleano. Este comportamiento cambió en Flash Player 6 y versiones posteriores para permitir la personalización de `_focusrect` en un clip de película concreto.

Si la propiedad `_focusrect` se define con el valor `false`, ese clip de película sólo podrá desplazarse con el teclado con la tecla Tabulador. Todas las demás teclas, incluida Intro y las teclas de flecha, quedan anuladas. Para restablecer el desplazamiento completo con el teclado, es preciso configurar `_focusrect` con el valor `true`.

Disponibilidad: ActionScript 1.0; Flash Player 6

Ejemplo

Este ejemplo muestra cómo ocultar el rectángulo amarillo que rodea una determinada instancia de clip de película de un archivo SWF cuando está seleccionada en una ventana del navegador. Cree tres clips de película denominados `mc1_mc`, `mc2_mc` y `mc3_mc`, y añada el siguiente código ActionScript al fotograma 1 de la línea de tiempo:

```
mc1_mc._focusrect = true;
mc2_mc._focusrect = false;
mc3_mc._focusrect = true;

mc1_mc.onRelease = traceOnRelease;
mc3_mc.onRelease = traceOnRelease;

function traceOnRelease() {
  trace(this._name);
}
```

Pruebe el archivo SWF en una ventana del navegador seleccionando Archivo > Vista previa de publicación > HTML. Seleccione el archivo SWF; para ello, haga clic sobre él en la ventana del navegador y pulse la tecla Tabulador para seleccionar cada instancia. Si `_focusrect` está desactivado, no podrá ejecutar código en el navegador para este clip de película con la tecla Intro ni la barra espaciadora.

También puede probar el archivo SWF en el entorno de prueba. Seleccione Control > Deshabilitar métodos abreviados de teclado en el menú principal del entorno de prueba. Esto le permitirá ver el rectángulo de selección alrededor de las instancias del archivo SWF.

Véase también

`Propiedad _focusrect`, `_focusrect (propiedad Button._focusrect)`

_framesloaded (propiedad MovieClip._framesloaded)

```
public _framesloaded : Number [read-only]
```

Número de fotogramas que se han cargado de un flujo de archivo SWF. Esta propiedad es útil para determinar si se ha cargado el contenido de un fotograma específico y de todos los que le preceden y si está disponible localmente en el navegador. También resulta útil para controlar la descarga de archivos SWF grandes. Por ejemplo, puede que desee mostrar un mensaje a los usuarios para indicar que el archivo SWF se está cargando hasta que un fotograma concreto del archivo SWF haya terminado de cargarse.

Disponibilidad: ActionScript 1.0; Flash Player 4

Ejemplo

El ejemplo siguiente utiliza la propiedad `_framesloaded` para iniciar un archivo SWF cuando se han cargado todos los fotogramas. Si no están cargados todos los fotogramas, la propiedad `_xscale` de la instancia de clip de película `bar_mc` aumenta proporcionalmente para crear una barra de progreso.

Introduzca el siguiente código ActionScript en el fotograma 1 de la línea de tiempo:

```
var pctLoaded:Number = Math.round(this.getBytesLoaded()/
  this.getBytesTotal()*100);
bar_mc._xscale = pctLoaded;
```

Añada el siguiente código al fotograma 2:

```
if (this._framesloaded < this._totalframes) {
  this.gotoAndPlay(1);
} else {
  this.gotoAndStop(3);
}
```

Coloque el contenido sobre o detrás del fotograma 3. A continuación añada el código siguiente al fotograma 3:

```
stop();
```

Véase también

`MovieClipLoader`

getBounds (método MovieClip.getBounds)

```
public getBounds(bounds:Object) : Object
```

Devuelve las propiedades que son los valores de coordenada mínimo y máximo *x* e *y* del clip de película para el parámetro `bounds`.

Nota: Utilice `MovieClip.localToGlobal()` y `MovieClip.globalToLocal()` para convertir las coordenadas locales del clip de película en coordenadas de escenario o las coordenadas de escenario en coordenadas locales, respectivamente.

Puede ampliar los métodos y los controladores de eventos de la clase MovieClip creando una subclase.

Disponibilidad: ActionScript 1.0; Flash Player 5

Parámetros

`bounds``:Object` - Ruta de destino de la línea de tiempo cuyo sistema de coordenadas desee utilizar como punto de referencia.

Valor devuelto

`Object` - Un objeto con las propiedades `xMin`, `xMax`, `yMin` y `yMax`.

Ejemplo

El ejemplo siguiente crea un clip de película denominado `square_mc`. El código dibuja un cuadrado para este clip de película y utiliza `MovieClip.getBounds()` para mostrar los valores de coordenada de la instancia en el panel Salida.

```
this.createEmptyMovieClip("square_mc", 1);
square_mc._x = 10;
square_mc._y = 10;
square_mc.beginFill(0xFF0000);
square_mc.moveTo(0, 0);
square_mc.lineTo(100, 0);
square_mc.lineTo(100, 100);
square_mc.lineTo(0, 100);
square_mc.lineTo(0, 0);
square_mc.endFill();

var bounds_obj:Object = square_mc.getBounds(this);
for (var i in bounds_obj) {
  trace(i+" --> "+bounds_obj[i]);
}
```

Aparecerá la información siguiente en el panel Salida:

```
yMax --> 110
yMin --> 10
xMax --> 110
xMin --> 10
```

Véase también

getRect (método MovieClip.getRect), globalToLocal (método MovieClip.globalToLocal), localToGlobal (método MovieClip.localToGlobal)

getBytesLoaded (método MovieClip.getBytesLoaded)

```
public getBytesLoaded() : Number
```

Devuelve el número de bytes que se han cargado (reproducido sin interrupción) para el clip de película. Puede comparar este valor con el devuelto por `MovieClip.getBytesTotal()` para determinar el porcentaje de un clip de película que se ha cargado.

Puede ampliar los métodos y los controladores de eventos de la clase MovieClip creando una subclase.

Disponibilidad: ActionScript 1.0; Flash Player 5

Valor devuelto

`Number` - Un entero que indica el número de bytes cargados.

Ejemplo

El ejemplo siguiente utiliza la propiedad `_framesloaded` para iniciar un archivo SWF cuando se han cargado todos los fotogramas. Si no están cargados todos los fotogramas, la propiedad `_xscale` de la instancia de clip de película `loader` aumenta proporcionalmente para crear una barra de progreso.

Introduzca el código ActionScript siguiente al fotograma 1 de la línea de tiempo:

```
var pctLoaded:Number = Math.round(this.getBytesLoaded()/
  this.getBytesTotal() * 100);
bar_mc._xscale = pctLoaded;
```

Añada el siguiente código al fotograma 2:

```
if (this._framesloaded<this._totalframes) {
  this.gotoAndPlay(1);
} else {
  this.gotoAndStop(3);
}
```

Coloque el contenido sobre o detrás del fotograma 3 y añada el código siguiente al fotograma 3:

```
stop();
```

Véase también

getBytesTotal (método MovieClip.getBytesTotal)

getBytesTotal (método MovieClip.getBytesTotal)

```
public getBytesTotal() : Number
```

Devuelve el tamaño en bytes del clip de película. En el caso de clips de película externos (el archivo SWF raíz o un clip de película que se está cargando en un destino o un nivel), el valor devuelto es el tamaño sin comprimir del archivo SWF.

Puede ampliar los métodos y los controladores de eventos de la clase MovieClip creando una subclase.

Disponibilidad: ActionScript 1.0; Flash Player 5

Valor devuelto

`Number` - Un entero que indica el tamaño total en bytes del clip de película.

Ejemplo

El ejemplo siguiente utiliza la propiedad `_framesloaded` para iniciar un archivo SWF cuando se han cargado todos los fotogramas. Si no están cargados todos los fotogramas, la propiedad `_xscale` de la instancia `loader` de clip de película aumenta proporcionalmente para crear una barra de progreso.

Introduzca el código ActionScript siguiente al fotograma 1 de la línea de tiempo:

```
var pctLoaded:Number = Math.round(this.getBytesLoaded()/
  this.getBytesTotal()*100);
bar_mc._xscale = pctLoaded;
```

Añada el siguiente código al fotograma 2:

```
if (this._framesloaded<this._totalframes) {
  this.gotoAndPlay(1);
} else {
  this.gotoAndStop(3);
}
```

Coloque el contenido sobre o detrás del fotograma 3. A continuación añada el código siguiente al fotograma 3:

```
stop();
```

Véase también

`getBytesLoaded (método MovieClip.getBytesLoaded)`

getDepth (método MovieClip.getDepth)

```
public getDepth() : Number
```

Devuelve la profundidad de una instancia de clip de película.

Cada clip de película, botón y campo de texto tiene asociada una profundidad exclusiva que determina cómo aparece el objeto delante o detrás de otros objetos. Los objetos con valores mayores de profundidad aparecen delante. El contenido creado en tiempo de diseño (en la herramienta de edición) comienza en la profundidad -16383.

Puede ampliar los métodos y los controladores de eventos de la clase MovieClip creando una subclase.

Disponibilidad: ActionScript 1.0; Flash Player 6

Valor devuelto

`Number` - La profundidad del clip de película.

Ejemplo

El código siguiente averigua la profundidad de todas las instancias de clip de película del escenario:

```
for (var i in this) {
  if (typeof (this[i]) == "movieclip") {
  trace("movie clip '"+this[i]._name+"' is at depth "+this[i].getDepth());
  }
}
```

Véase también

`getInstanceAtDepth (método MovieClip.getInstanceAtDepth)`, `getNextHighestDepth (método MovieClip.getNextHighestDepth)`, `swapDepths (método MovieClip.swapDepths)`, `getDepth (método TextField.getDepth)`, `getDepth (método Button.getDepth)`

getInstanceAtDepth (método MovieClip.getInstanceAtDepth)

```
public getInstanceAtDepth(depth:Number) : MovieClip
```

Determina si una profundidad específica ya está ocupada por un clip de película. Puede utilizar ese método antes de utilizar `MovieClip.attachMovie()`, `MovieClip.duplicateMovieClip()` o `MovieClip.createEmptyMovieClip()` para determinar si el parámetro de profundidad que desea pasar a alguno de estos métodos ya contiene un clip de película.

Puede ampliar los métodos y los controladores de eventos de la clase MovieClip creando una subclase.

Disponibilidad: ActionScript 1.0; Flash Player 7

Parámetros

depth:Number - Entero que especifica el nivel de profundidad de la consulta.

Valor devuelto

MovieClip - Una referencia a la instancia MovieClip situada en la profundidad especificada o undefined si no hay ningún clip de película en dicha profundidad.

Ejemplo

El ejemplo siguiente muestra la profundidad ocupada por la instancia de clip de película triangle en el panel Salida:

```
this.createEmptyMovieClip("triangle", 1);

triangle.beginFill(0x0000FF, 100);
triangle.moveTo(100, 100);
triangle.lineTo(100, 150);
triangle.lineTo(150, 100);
triangle.lineTo(100, 100);

trace(this.getInstanceAtDepth(1)); // output: _level0.triangle
```

Véase también

attachMovie (método MovieClip.attachMovie), duplicateMovieClip (método MovieClip.duplicateMovieClip), createEmptyMovieClip (método MovieClip.createEmptyMovieClip), getDepth (método MovieClip.getDepth), getNextHighestDepth (método MovieClip.getNextHighestDepth), swapDepths (método MovieClip.swapDepths)

getNextHighestDepth (método MovieClip.getNextHighestDepth)

`public getNextHighestDepth() : Number`

Le permite determinar un valor de profundidad que puede pasar a `MovieClip.attachMovie()`, `MovieClip.duplicateMovieClip()` o `MovieClip.createEmptyMovieClip()` para asegurarse de que Flash representa el clip de película delante de todos los demás objetos situados en el mismo nivel y capa del clip de película actual. El valor devuelto es 0 o un valor mayor (es decir, no se devuelven números negativos).

Puede ampliar los métodos y los controladores de eventos de la clase MovieClip creando una subclase.

Nota: Si utiliza componentes de la versión 2, no use este método. Si coloca un componente de la versión 2 en el escenario o la biblioteca, el método `getNextHighestDepth()` puede devolver una profundidad de 1048676, valor que queda fuera del rango válido. Si utiliza componentes de la versión 2, debe usar siempre la clase DepthManager de los componentes de la versión 2.

Disponibilidad: ActionScript 1.0; Flash Player 7

Valor devuelto

`Number` - Un entero que refleja el siguiente índice de profundidad disponible que se representaría por encima de todos los demás objetos del mismo nivel y capa dentro del clip de pelula.

Ejemplo

El ejemplo siguiente dibuja tres instancias de clip de película utilizando el método `getNextHighestDepth()` como parámetro `depth` del método `createEmptyMovieClip()`, además etiqueta cada clip de película con su profundidad:

```
for (i = 0; i < 3; i++) {
  drawClip(i);
}

function drawClip(n:Number):Void {
  this.createEmptyMovieClip("triangle" + n, this.getNextHighestDepth());
  var mc:MovieClip = eval("triangle" + n);
  mc.beginFill(0x00aaFF, 100);
  mc.lineStyle(4, 0xFF0000, 100);
  mc.moveTo(0, 0);
  mc.lineTo(100, 100);
  mc.lineTo(0, 100);
```

```
  mc.lineTo(0, 0);
  mc._x = n * 30;
  mc._y = n * 50
  mc.createTextField("label", this.getNextHighestDepth(), 20, 50, 200, 200)
  mc.label.text = mc.getDepth();
}
```

Véase también

getDepth (método MovieClip.getDepth), getInstanceAtDepth (método MovieClip.getInstanceAtDepth), swapDepths (método MovieClip.swapDepths), attachMovie (método MovieClip.attachMovie), duplicateMovieClip (método MovieClip.duplicateMovieClip), createEmptyMovieClip (método MovieClip.createEmptyMovieClip)

getRect (método MovieClip.getRect)

```
public getRect(bounds:Object) : Object
```

Devuelve las propiedades que son los valores de coordenada mínimo y máximo *x* e *y* del clip de película para el parámetro `bounds`, excepto cualquier trazo en formas.. Los valores devueltos por `getRect()` son los mismos o menores que los devueltos por `MovieClip.getBounds()`.

Nota: Utilice `MovieClip.localToGlobal()` y `MovieClip.globalToLocal()` para convertir las coordenadas locales del clip de película en coordenadas de escenario o las coordenadas de escenario en coordenadas locales, respectivamente.

Puede ampliar los métodos y los controladores de eventos de la clase MovieClip creando una subclase.

Disponibilidad: ActionScript 1.0; Flash Player 8

Parámetros

`bounds``:Object` - Ruta de destino de la línea de tiempo cuyo sistema de coordenadas desee utilizar como punto de referencia.

Valor devuelto

`Object` - Un objeto con las propiedades `xMin`, `xMax`, `yMin` y `yMax`.

Ejemplo

El ejemplo siguiente crea un clip de película y dibuja en su interior un cuadrado con un grosor de trazo de 4 píxeles. Después, se realizan llamadas a los métodos `MovieClip.getBounds()` y `MovieClip.getRect()` para mostrar las diferencias existentes entre los dos. El método `getBounds()` devuelve los valores de coordenada máximo y mínimo de todo el clip de película, incluido el grosor de trazo del cuadrado. El método `getRect()` devuelve los valores de coordenada máximo y mínimo excluido el grosor de trazo del cuadrado.

```
this.createEmptyMovieClip("square_mc", 1);
square_mc._x = 10;
square_mc._y = 10;
square_mc.beginFill(0xFF0000);
square_mc.lineStyle(4, 0xFF00FF, 100, true, "none", "round", "miter", 1);
square_mc.moveTo(0, 0);
square_mc.lineTo(100, 0);
square_mc.lineTo(100, 100);
square_mc.lineTo(0, 100);
square_mc.lineTo(0, 0);
square_mc.endFill();

var bounds_obj:Object = square_mc.getBounds(this);
trace("getBounds() output:");
for (var i in bounds_obj) {
  trace(i+" --> "+bounds_obj[i]);
}

var rect_obj:Object = square_mc.getRect(this);
trace("getRect() output:");
for (var i in rect_obj) {
  trace(i+" --> "+rect_obj[i]);
}
```

El resultado de la sentencia `trace()` se muestra en la salida siguiente.

```
getBounds() output:
yMax --> 112
yMin --> 8
xMax --> 112
xMin --> 8
getRect() output:
yMax --> 110
yMin --> 10
xMax --> 110
xMin --> 10
```

Véase también

`getBounds (método MovieClip.getBounds)`, `globalToLocal (método MovieClip.globalToLocal)`, `localToGlobal (método MovieClip.localToGlobal)`

getSWFVersion (método MovieClip.getSWFVersion)

```
public getSWFVersion() : Number
```

Devuelve un entero que indica la versión de Flash Player para la que se ha publicado el clip de película. Si el clip de película es un archivo JPEG, GIF o PNG, o si se produce un error y Flash Player no logra determinar la versión de SWF del clip de película, se devuelve -1.

Puede ampliar los métodos y los controladores de eventos de la clase MovieClip creando una subclase.

Disponibilidad: ActionScript 1.0; Flash Player 7

Valor devuelto

`Number` - Un entero que especifica la versión de Flash Player de destino cuando se publicó el archivo SWF cargado en el clip de película.

Ejemplo

El ejemplo siguiente crea un nuevo contenedor y ofrece como salida el valor de `getSWFVersion()`. Después utiliza MovieClipLoader para cargar un archivo SWF externo que se publicó con Flash Player 7 y muestra el valor de `getSWFVersion()` después de activarse el controlador `onLoadInit`.

```
var container:MovieClip = this.createEmptyMovieClip("container",
  this.getUpperEmptyDepth());
var listener:Object = new Object();
listener.onLoadInit = function(target:MovieClip):Void {
  trace("target: " + target.getSWFVersion()); // target: 7
}
var mcLoader:MovieClipLoader = new MovieClipLoader();
mcLoader.addListener(listener);
trace("container: " + container.getSWFVersion()); // container: 8
mcLoader.loadClip("FlashPlayer7.swf", container);
```

getTextSnapshot (método MovieClip.getTextSnapshot)

```
public getTextSnapshot() : TextSnapshot
```

Devuelve un objeto TextSnapshot que contiene el texto de todos los campos de texto estáticos existentes en el clip de película especificado; el texto contenido en los clips de película secundarios no se incluye. Este método siempre devuelve un objeto TextSnapshot.

Flash concatena texto y lo coloca en el objeto TextSnapshot en un orden que refleja el índice de tabulación de los campos de texto estáticos del clip de película. Los campos de texto que carecen de valores de índice de tabulación se sitúan por orden aleatorio en el objeto y preceden a cualquier texto de campos que sí tienen valores de índice de tabulación. El final de un campo y el comienzo del siguiente no se marca con ningún salto de línea ni ningún formato.

Nota: En Flash, no es posible especificar un valor de índice de tabulación para texto estático. No obstante, es posible que otros productos lo permitan (por ejemplo, Macromedia FlashPaper).

El contenido del objeto TextSnapshot no es dinámico; es decir, si el clip de película se traslada a otro fotograma o se modifica de algún modo (por ejemplo, si se añaden o eliminan objetos del clip de película), es posible que el objeto TextSnapshot no represente al texto actual del clip de película. Para asegurarse de que el contenido del objeto esté actualizado, vuelva a enviar este comando cuando sea necesario.

Puede ampliar los métodos y los controladores de eventos de la clase MovieClip creando una subclase.

Disponibilidad: ActionScript 1.0; Flash Player 7

Valor devuelto

`TextSnapshot` - Un objeto TextSnapshot que contiene el texto estático del clip de película.

Ejemplo

El ejemplo siguiente muestra varios modos de utilizar este método. Para utilizar este código, coloque en el escenario un campo de texto estático que contenga el texto "TextSnapshot Example".

```
var textSnap:TextSnapshot = this.getTextSnapshot();
trace(textSnap.getText(0, textSnap.getCount(), false));
```

Véase también

`TextSnapshot`

getURL (método MovieClip.getURL)

```
public getURL(url:String, [window:String], [method:String]) : Void
```

Carga un documento de la URL especificada en la ventana indicada. El método `getURL()` también puede utilizarse para pasar variables a otra aplicación definida en la URL mediante un método `GET` o `POST`.

Las páginas Web que albergan películas Flash deben definir explícitamente el atributo `allowScriptAccess` para permitir o denegar la creación de scripts en Flash Player a partir del código HTML (etiqueta `PARAM` en Internet Explorer o la etiqueta `EMBED` en Netscape Navigator).

- Cuando `allowScriptAccess` es `"never"`, la creación de scripts salientes falla siempre.
- Cuando `allowScriptAccess` es `"always"`, la creación de scripts salientes siempre es correcta.
- Cuando `allowScriptAccess` es `"sameDomain"` (admitido en los archivos SWF a partir de la versión 8), se permite la creación de scripts salientes si el archivo SWF pertenece al mismo dominio de la página Web que lo aloja.
- Si una página HTML no especifica `allowScriptAccess`, el valor predeterminado es `"sameDomain"` para los archivos SWF de la versión 8; en el caso de archivos SWF de versiones anteriores, el valor predeterminado es `"always"`.

Cuando utilice este método puede usar el modelo de seguridad de Flash Player. En el caso de Flash Player 8, no se admite el método si el archivo SWF que realiza la llamada, se encuentra en el entorno limitado local con sistema de archivos y el recurso no es local.

Para más información, consulte las referencias siguientes:

- Capitulo 17, "Aspectos básicos de la seguridad", de *Aprendizaje de ActionScript 2.0 en Flash*
- El documento técnico sobre la seguridad de Flash Player 8 en http://www.macromedia.com/go/fp8_security
- El documento técnico sobre interfaces API relativas a seguridad de Flash Player 8 en http://www.macromedia.com/go/fp8_security_apis

Puede ampliar los métodos y los controladores de eventos de la clase MovieClip creando una subclase.

Disponibilidad: ActionScript 1.0; Flash Player 5

Parámetros

url:String - URL del cual se obtiene el documento.

window:String [opcional] - Un parámetro que especifica el nombre, fotograma o expresión que especifica la ventana o fotograma HTML en el que se carga el documento. También es posible utilizar uno de los siguientes nombres de destino reservados: `_self` especifica el marco actual de la ventana actual, `_blank` especifica una ventana nueva, `_parent` especifica el nivel superior del marco actual y `_top` especifica el marco del nivel más alto de la ventana actual.

method:String [opcional] - Una cadena (`"GET"` o `"POST"`) que especifica un método para enviar las variables asociadas al archivo SWF que va a cargarse. Si no hay variables, omita este parámetro; de lo contrario, especifique si las variables han de cargarse con el método `GET` o `POST`. El método `GET` añade las variables al final de la URL y se utiliza para un número reducido de variables. El método `POST` envía las variables en un encabezado HTTP independiente y se utiliza para cadenas de variables largas.

Ejemplo

El siguiente código ActionScript crea una instancia de clip de película y abre el sitio Web de Macromedia en una nueva ventana del navegador:

```
this.createEmptyMovieClip("loader_mc", this.getNextHighestDepth());
loader_mc.getURL("http://www.macromedia.com", "_blank");
```

El método `getURL()` también permite enviar variables a un script de servidor remoto, como muestra el código siguiente:

```
this.createEmptyMovieClip("loader_mc", this.getNextHighestDepth());
loader_mc.username = "some user input";
loader_mc.password = "random string";
loader_mc.getURL("http://www.flash-mx.com/mm/viewscope.cfm", "_blank",
  "GET");
```

El método `MovieClip.getNextHighestDepth()` utilizado en estos ejemplos necesita Flash Player 7 o una versión posterior. Si el archivo SWF incluye un componente de la versión 2, utilice la clase DepthManager de los componentes de la versión 2 en lugar del método `MovieClip.getNextHighestDepth()`.

Cuando utilice este método puede usar el modelo de seguridad de Flash Player.

- En el caso de Flash Player 8, no se admite `MovieClip.getURL()` si el archivo SWF que realiza la llamada se encuentra en el entorno limitado local con sistema de archivos y el recurso no es local.

Para más información, consulte las referencias siguientes:

- Capitulo 17, "Aspectos básicos de la seguridad", de *Aprendizaje de ActionScript 2.0 en Flash*
- El documento técnico sobre la seguridad de Flash Player 8 en http://www.macromedia.com/go/fp8_security
- El documento técnico sobre interfaces API relativas a seguridad de Flash Player 8 en http://www.macromedia.com/go/fp8_security_apis

Véase también

`Función getURL`, `sensendAndLoad (método LoadVars.sendAndLoad)`, `send (método LoadVars.send)`

globalToLocal (método MovieClip.globalToLocal)

```
public globalToLocal(pt:Object) : Void
```

Convierte el objeto `pt` de las coordenadas del escenario (globales) en coordenadas del clip de película (locales).

El método `MovieClip.globalToLocal()` permite convertir los valores de las coordenadas *x* e *y* dadas, relativos a la esquina superior izquierda del escenario, en valores relativos a la esquina superior izquierda de un clip de película especificado.

En primer lugar deber crear un objeto genérico con dos propiedades, x e y. Estos valores `x` e `y` (deben llamarse `x` e `y`) se denominan coordenadas globales porque son relativas a la esquina superior izquierda del escenario. La propiedad x representa el desplazamiento horizontal con respecto a la esquina superior izquierda. En otras palabras, representa la distancia del punto hacia la derecha. Por ejemplo, si x = 50, el punto queda 50 píxeles a la derecha de la esquina superior izquierda. La propiedad y representa el desplazamiento vertical con respecto a la esquina superior izquierda. En otras palabras, representa la distancia del punto hacia abajo. Por ejemplo, si y = 20, el punto queda 20 píxeles por debajo de la esquina superior izquierda. El código siguiente crea un objeto genérico con estas coordenadas:

```
var myPoint:Object = new Object();
myPoint.x = 50;
myPoint.y = 20;
```

También es posible crear el objeto y asignarle simultáneamente los valores con un valor literal de objeto:

```
var myPoint:Object = {x:50, y:20};
```

Una vez creado un objeto punto con coordenadas globales, puede convertir las coordenadas en coordenadas locales. El método `globalToLocal()` no devuelve un valor porque cambia los valores de `x` e `y` del objeto genérico enviado como parámetro. Sustituye los valores relativos al escenario (coordenadas globales) por valores relativos a un clip de película específico (coordenadas locales).

Por ejemplo, si crea un clip de película situado en el punto `(_x:100, _y:100)` y pasa el punto global que representa la esquina superior izquierda del escenario `(x:0, y:0)` al método `globalToLocal()`, el método debería convertir los valores de `x` e `y` en coordenadas locales, en este caso `(x:-100, y:-100)`. Esta conversión ocurre porque ahora las coordenadas *x* e *y* se expresan en relación con la esquina superior izquierda del clip de película, no del escenario. Los valores son negativos porque para pasar de la esquina superior izquierda del clip de película a la esquina superior izquierda del escenario, debe desplazarse 100 píxeles a la izquierda (*x* negativo) y 100 píxeles hacia arriba (*y* negativo).

Las coordenadas del clip de película se han expresado con `_x` e `_y` porque son las propiedades de MovieClip utilizadas para definir los valores *x* e *y* para MovieClips. Sin embargo, el objeto genérico utiliza `x` e `y` sin subrayado. El siguiente código convierte los valores de *x* e *y* en coordenadas locales:

```
var myPoint:Object = {x:0, y:0}; // Create your generic point object.
this.createEmptyMovieClip("myMovieClip", this.getNextHighestDepth());
myMovieClip._x = 100; // _x for movieclip x position
myMovieClip._y = 100; // _y for movieclip y position

myMovieClip.globalToLocal(myPoint);
trace ("x: " + myPoint.x); // output: -100
trace ("y: " + myPoint.y); // output: -100
```

Puede ampliar los métodos y los controladores de eventos de la clase MovieClip creando una subclase.

Disponibilidad: ActionScript 1.0; Flash Player 5

Parámetros

pt`:Object` - Nombre o identificador de un objeto creado con la clase Object genérica. El objeto especifica las coordenadas *x* e *y* como propiedades.

Ejemplo

Añada el siguiente código ActionScript a un archivo FLA o AS del mismo directorio como una imagen denominada photo1.jpg:

```
this.createTextField("coords_txt", this.getNextHighestDepth(), 10, 10, 100,
  22);
coords_txt.html = true;
coords_txt.multiline = true;
```

```
coords_txt.autoSize = true;
this.createEmptyMovieClip("target_mc", this.getNextHighestDepth());
target_mc._x = 100;
target_mc._y = 100;
target_mc.loadMovie("photo1.jpg");

var mouseListener:Object = new Object();
mouseListener.onMouseMove = function() {
  var point:Object = {x:_xmouse, y:_ymouse};
  target_mc.globalToLocal(point);
  var rowHeaders = "<b>   \t</b><b>_x\t</b><b>_y</b>";
  var row_1 = "_root\t"+_xmouse+"\t"+_ymouse;
  var row_2 = "target_mc\t"+point.x+"\t"+point.y;
  coords_txt.htmlText = "<textformat tabstops='[100, 150]'>";
  coords_txt.htmlText += rowHeaders;
  coords_txt.htmlText += row_1;
  coords_txt.htmlText += row_2;
  coords_txt.htmlText += "</textformat>";
};
Mouse.addListener(mouseListener);
```

El método `MovieClip.getNextHighestDepth()` utilizado en este ejemplo necesita Flash Player 7 o una versión posterior. Si el archivo SWF incluye un componente de la versión 2, utilice la clase DepthManager de los componentes de la versión 2 en lugar del método `MovieClip.getNextHighestDepth()`.

Véase también

`getBounds (método MovieClip.getBounds)`, `localToGlobal (método MovieClip.localToGlobal)`, `Object`

gotoAndPlay (método MovieClip.gotoAndPlay)

```
public gotoAndPlay(frame:Object) : Void
```

Inicia la reproducción del archivo SWF en el fotograma especificado. Para especificar una escena además de un fotograma, utilice `gotoAndPlay()`.

Puede ampliar los métodos y los controladores de eventos de la clase MovieClip creando una subclase.

Disponibilidad: ActionScript 1.0; Flash Player 5

Parámetros

`frame``:Object` - Número que representa el número de fotograma o la cadena que representa la etiqueta del fotograma al que se envía la cabeza lectora.

Ejemplo

El ejemplo siguiente utiliza la propiedad `_framesloaded` para iniciar un archivo SWF cuando se han cargado todos los fotogramas. Si no están cargados todos los fotogramas, la propiedad `_xscale` de la instancia de clip de película `loader` aumenta proporcionalmente para crear una barra de progreso.

Introduzca el código ActionScript siguiente al fotograma 1 de la línea de tiempo:

```
var pctLoaded:Number = Math.round(this.getBytesLoaded()/
  this.getBytesTotal()*100);
bar_mc._xscale = pctLoaded;
```

Añada el siguiente código al fotograma 2:

```
if (this._framesloaded<this._totalframes) {
  this.gotoAndPlay(1);
} else {
  this.gotoAndStop(3);
}
```

Coloque el contenido sobre o detrás del fotograma 3. A continuación añada el código siguiente al fotograma 3:

```
stop();
```

Véase también

`Función gotoAndPlay`, `Función play`

gotoAndStop (método MovieClip.gotoAndStop)

```
public gotoAndStop(frame:Object) : Void
```

Traslada la cabeza lectora al fotograma especificado del clip de película y la detiene en dicho lugar. Para especificar una escena además de un fotograma, utilice `gotoAndStop()`.

Puede ampliar los métodos y los controladores de eventos de la clase MovieClip creando una subclase.

Disponibilidad: ActionScript 1.0; Flash Player 5

Parámetros

`frame``:Object` - Número del fotograma al que se envía la cabeza lectora.

Ejemplo

El ejemplo siguiente utiliza la propiedad `_framesloaded` para iniciar un archivo SWF cuando se han cargado todos los fotogramas. Si no están cargados todos los fotogramas, la propiedad `_xscale` de la instancia de clip de película `loader` aumenta proporcionalmente para crear una barra de progreso.

Introduzca el código ActionScript siguiente al fotograma 1 de la línea de tiempo:

```
var pctLoaded:Number = Math.round(this.getBytesLoaded()/
  this.getBytesTotal()*100);
bar_mc._xscale = pctLoaded;
```

Añada el siguiente código al fotograma 2:

```
if (this._framesloaded<this._totalframes) {
  this.gotoAndPlay(1);
} else {
  this.gotoAndStop(3);
}
```

Coloque el contenido sobre o detrás del fotograma 3. A continuación añada el código siguiente al fotograma 3:

```
stop();
```

Véase también

`Función gotoAndStop`, `Función stop`

_height (propiedad MovieClip._height)

```
public _height : Number
```

Altura del clip de película, expresada en píxeles.

Disponibilidad: ActionScript 1.0; Flash Player 4

Ejemplo

El siguiente ejemplo de código muestra la altura y anchura de un clip de película en el panel Salida:

```
this.createEmptyMovieClip("image_mc", this.getNextHighestDepth());
var image_mcl:MovieClipLoader = new MovieClipLoader();
var mclListener:Object = new Object();
mclListener.onLoadInit = function(target_mc:MovieClip) {
  trace(target_mc._name+" = "+target_mc._width+" X "+target_mc._height+"
  pixels");
};
image_mcl.addListener(mclListener);

image_mcl.loadClip("example.jpg", image_mc);
```

El método `MovieClip.getNextHighestDepth()` utilizado en este ejemplo necesita Flash Player 7 o una versión posterior. Si el archivo SWF incluye un componente de la versión 2, utilice la clase DepthManager de los componentes de la versión 2 en lugar del método `MovieClip.getNextHighestDepth()`.

La clase MovieClipLoader utilizada en este ejemplo necesita Flash Player 7 o una versión posterior.

Véase también

`_width (propiedad MovieClip._width)`

_highquality (propiedad MovieClip._highquality)

```
public _highquality : Number
```

Desfasada desde Flash Player 7. Esta propiedad está desfasada y en su lugar debe utilizarse `MovieClip._quality`.

Especifica el nivel de visualización suavizada que se aplica al archivo SWF actual. Especifique 2 (mejor calidad) para aplicar alta calidad con el suavizado de mapa de bits siempre activado. Especifique 1 (alta calidad) para aplicar la visualización suavizada, que suaviza los mapas de bits si el archivo SWF no contiene animación. Especifique 0 (baja calidad) para evitar el suavizado. Esta propiedad puede sobrescribir a la propiedad global `_highquality`.

Disponibilidad: ActionScript 1.0; Flash Player 6

Ejemplo

El siguiente código ActionScript especifica la aplicación de visualización suavizada de la mejor calidad al archivo SWF.

```
my_mc._highquality = 2;
```

Véase también

`_quality (propiedad MovieClip._quality)`, `Propiedad _quality`

hitArea (propiedad MovieClip.hitArea)

```
public hitArea : Object
```

Designa otro clip de película para que actúe como área activa de un clip de película. Si la propiedad `hitArea` no existe o tiene el valor `null` o `undefined`, se utilizará el propio clip de película como área activa. El valor de la propiedad `hitArea` puede ser una referencia a un objeto de clip de película.

Puede cambiar la propiedad `hitArea` en cualquier momento; el clip de película modificado adoptará de inmediato el nuevo comportamiento de área activa. No es preciso que el clip de película designado como área activa sea visible; se detectará la condición de "activa" de su forma gráfica aunque no sea visible.

Disponibilidad: ActionScript 1.0; Flash Player 6

Ejemplo

El ejemplo siguiente configura el clip de película `circle_mc` como área activa del clip de película `square_mc`. Coloque los dos clips de película en el escenario y pruebe el documento. Al hacer clic en `circle_mc`, el clip de película `square_mc` detecta que se ha hecho clic.

```
square_mc.hitArea = circle_mc;
square_mc.onRelease = function() {
  trace("hit! "+this._name);
};
```

También es posible configurar como `false` la propiedad `visible` del clip de película `circle_mc` para ocultar el área activa de `square_mc`.

```
circle_mc._visible = false;
```

Véase también

`hitTest (método MovieClip.hitTest)`

hitTest (método MovieClip.hitTest)

```
public hitTest() : Boolean
```

Evalúa el clip de película para comprobar si se superpone o corta con el área activa identificada mediante los parámetros de coordenadas `target` o `x` e `y`.

Sintaxis 1: Compara las coordenadas *x* e *y* con la forma o el recuadro de delimitación de la instancia especificada, según la configuración de `shapeFlag`. Si `shapeFlag` se configura como `true`, sólo se evaluará el área del escenario que ocupa la instancia en sí y, si *x* e *y* se solapan en algún punto, se devolverá el valor `true`. Esta evaluación resulta útil para determinar si el clip de película se encuentra dentro de un área activa especificada.

Sintaxis 2: Evalúa los recuadros de delimitación del destino (`target`) y la instancia especificada y devuelve `true` si se solapan o cortan en algún punto.

Parámetros `x: Number` La coordenada *x* del área activa en el escenario. `y: Number` La coordenada *y* del área activa en el escenario. Las coordenadas *x* e *y* se definen en el espacio de coordenadas globales. `shapeFlag: Boolean` - Valor booleano que especifica si debe evaluarse la forma completa de la instancia especificada (`true`) o sólo el recuadro de delimitación (`false`). Este parámetro sólo puede especificarse si el área activa se identifica empleando los parámetros de coordenadas `x` e `y` `target: Object` El trazado de destino del área activa que puede solaparse con la instancia especificada por el clip de película. El parámetro `target` normalmente representa un botón o campo de introducción de texto.

Disponibilidad: ActionScript 1.0; Flash Player 5

Valor devuelto

`Boolean` - El valor booleano `true` si el clip de película se solapa con el área activa especificada, `false` en caso contrario.

Ejemplo

El ejemplo siguiente utiliza `hitTest()` para determinar si el clip de película `circle_mc` se solapa con el clip de película `square_mc` cuando el usuario suelta el botón del ratón:

```
square_mc.onPress = function() {
  this.startDrag();
};
square_mc.onRelease = function() {
  this.stopDrag();
  if (this.hitTest(circle_mc)) {
  trace("you hit the circle");
  }
};
```

Véase también

`getBounds (método MovieClip.getBounds)`, `globalToLocal (método MovieClip.globalToLocal)`, `localToGlobal (método MovieClip.localToGlobal)`

lineGradientStyle (método MovieClip.lineGradientStyle)

```
public lineGradientStyle(fillType:String, colors:Array, alphas:Array,
  ratios:Array, matrix:Object, [spreadMethod:String],
  [interpolationMethod:String], [focalPointRatio:Number]) : Void
```

Especifica un estilo de línea que Flash utilizará para las posteriores llamadas a `lineTo()` y `curveTo()` hasta que llame a `lineStyle()` o `lineGradientStyle()` con otros parámetros. Puede efectuar una llamada al método `lineGradientStyle()` en mitad de un trazado para especificar diferentes estilos para los distintos segmentos de línea de un trazado.

Nota: Realice una llamada a `lineStyle()` antes de hacerlo a `lineGradientStyle()` para activar un trazo, pues en caso contrario el estilo de línea seguirá siendo `undefined`.

Nota: Las llamadas a `clear()` volverán a establecer el estilo de línea con el valor `undefined`.

Puede ampliar los métodos y los controladores de eventos de la clase MovieClip creando una subclase.

Disponibilidad: ActionScript 1.0; Flash Player 8

Parámetros

`fillType``:String` - Los valores válidos son `"linear"` o `"radial"`.

`colors``:Array` - Matriz de valores de color RGB hexadecimales que se utiliza en el degradado (por ejemplo, rojo es 0xFF0000, azul es 0x0000FF, etc.). Puede utilizar hasta 15 colores. Para cada color, asegúrese de especificar un valor correspondiente en los parámetros `alphas` y `ratios`.

`alphas``:Array` - Matriz de valores alfa para los colores correspondientes de la matriz *`colors`* ; los valores válidos son del 0 al 100. Si el valor es inferior a 0, Flash utiliza 0. Si el valor es mayor que 100, Flash utiliza 100

`ratios``:Array` - Una matriz de proporciones de distribución de colores; los valores válidos son del 0 al 255. Este valor define el porcentaje de la anchura donde el color se muestrea al 100%. Introduzca un valor con el parámetro `colors`.

Por ejemplo, en el caso de un degradado lineal con dos colores, azul y verde, la siguiente figura ilustra la colocación de los colores en el degradado en función de los diferentes valores en la matriz `ratios`:

`ratios`	**Degradado**
`[0, 127]`	
`[0, 255]`	
`[127, 255]`	

Los valores de la matriz deben aumentar secuencialmente; por ejemplo, `[0, 63, 127, 190, 255]`.

`matrix``:Object` - Matriz de transformación que es un objeto con uno de los dos conjuntos de propiedades siguientes:

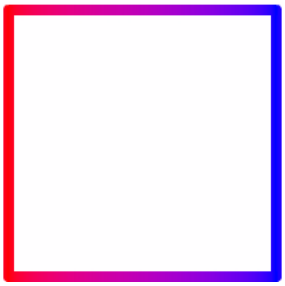
- Puede utilizar las propiedades `a`, `b`, `c`, `d`, `e`, `f`, `g`, `h` y `i` para describir una matriz de 3 x 3 de la forma siguiente:

```
a b c
d e f
g h i
```

- El ejemplo siguiente utiliza el método `lineGradientFill()` con un parámetro `matrix` que es un objeto con estas propiedades:

```
this.createEmptyMovieClip("gradient_mc", 1);
with (gradient_mc) {
  colors = [0xFF0000, 0x0000FF];
    alphas = [100, 100];
    ratios = [0, 0xFF];
    matrix = {a:200, b:0, c:0, d:0, e:200, f:0, g:200, h:200, i:1};
    spreadMethod = "reflect";
    interpolationMethod = "linearRGB";
    focalPointRatio = 0.9;
  lineStyle(8);
    lineGradientStyle("linear", colors, alphas, ratios, matrix,
    spreadMethod, interpolationMethod, focalPointRatio);
    moveTo(100, 100);
    lineTo(100, 300);
    lineTo(300, 300);
    lineTo(300, 100);
    lineTo(100, 100);
    endFill();
}
```

- Este código dibuja la imagen siguiente en la pantalla:



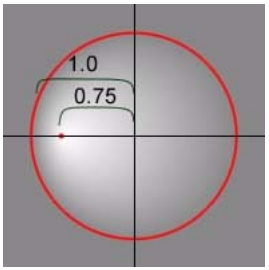
- `matrixType`, `x`, `y`, `w`, `h`, `r`.
- Las propiedades indican lo siguiente: `matrixType` es la cadena "`box`", `x` es la posición horizontal de la esquina superior izquierda del degradado con respecto al punto de registro del clip principal, `y` es la posición vertical de la esquina superior izquierda del degradado con respecto al punto de registro del clip principal, `w` es la anchura del degradado, `h` es la altura del degradado y `r` es la rotación en radianes del degradado.
- El ejemplo siguiente utiliza el método `lineGradientFill()` con un parámetro `matrix` que es un objeto con estas propiedades:

```
this.createEmptyMovieClip("gradient_mc", 1);
with (gradient_mc) {
  colors = [0xFF0000, 0x0000FF];
  alphas = [100, 100];
  ratios = [0, 0xFF];
  matrix = {matrixType:"box", x:100, y:100, w:200, h:200, r:(45/
  180)*Math.PI};
  spreadMethod = "reflect";
  interpolationMethod = "linearRGB";
  lineStyle(8);
  lineGradientStyle("linear", colors, alphas, ratios, matrix,
    spreadMethod, interpolationMethod);
  moveTo(100, 100);
  lineTo(100, 300);
  lineTo(300, 300);
  lineTo(300, 100);
  lineTo(100, 100);
  endFill();
}
```

`spreadMethod``:String` [opcional] - Los valores válidos son "pad", "reflect" o "repeat", que controla el modo del relleno del degradado.

`interpolationMethod``:String` [opcional] - Los valores válidos son "RGB" o "linearRGB".

focalPointRatio:Number [opcional] - Número que controla la ubicación del punto focal del degradado. El valor 0 significa que el punto focal está en el centro. El valor 1 significa que el punto focal se encuentra en un borde del círculo del degradado. El valor -1 significa que el punto focal se encuentra en el otro borde del círculo del degradado. Los valores inferiores a -1 o mayores que 1 se redondean a -1 o 1. El siguiente ejemplo muestra un degradado con `focalPointRatio` definido en -0,75:

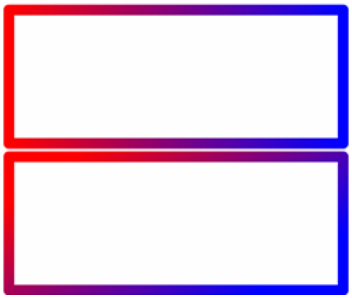


Ejemplo

El código siguiente utiliza ambos métodos para dibujar dos rectángulos apilados y un relleno con degradado de líneas rojas-azules:

```
this.createEmptyMovieClip("gradient_mc", 1);
with (gradient_mc) {
  colors = [0xFF0000, 0x0000FF];
  alphas = [100, 100];
  ratios = [0, 0xFF];
  matrix = {a:500, b:0, c:0, d:0, e:200, f:0, g:350, h:200, i:1};
  lineStyle(16);
  lineGradientStyle("linear", colors, alphas, ratios, matrix);
  moveTo(100, 100);
  lineTo(100, 300);
  lineTo(600, 300);
  lineTo(600, 100);
  lineTo(100, 100);
  endFill();
  matrix2 = {matrixType:"box", x:100, y:310, w:500, h:200, r:(30/
  180)*Math.PI};
  lineGradientStyle("linear", colors, alphas, ratios, matrix2);
  moveTo(100, 320);
  lineTo(100, 520);
  lineTo(600, 520);
  lineTo(600, 320);
  lineTo(100, 320);
  endFill();
}
```

El código dibuja la imagen siguiente (la escala de la ilustración se ha adaptado al 50%):



Véase también

`beginGradientFill (método MovieClip.beginGradientFill)`, `lineStyle (método MovieClip.lineStyle)`, `lineTo (método MovieClip.lineTo)`, `moveTo (método MovieClip.moveTo)`

lineStyle (método MovieClip.lineStyle)

```
public lineStyle(thickness:Number, rgb:Number, alpha:Number,
  pixelHinting:Boolean, noScale:String, capsStyle:String,
  jointStyle:String, miterLimit:Number) : Void
```

Especifica un estilo de línea que Flash utilizará para las posteriores llamadas a `lineTo()` y `curveTo()` hasta que llame a `lineStyle()` con otros parámetros. Puede efectuar una llamada a `lineStyle()` en mitad de un trazado para especificar diferentes estilos para los distintos segmentos de línea de un trazado.

Nota: Las llamadas al método `clear()` volverán a establecer el estilo de línea con el valor `undefined`.

Puede ampliar los métodos y los controladores de eventos de la clase MovieClip creando una subclase.

Disponibilidad: ActionScript 1.0; Flash Player 6

Parámetros

`thickness``:Number` - Entero que indica el grosor de la línea en puntos; los valores válidos son del 0 al 255. Si no se especifica ningún número o si el parámetro es `undefined`, no se traza ninguna línea. Si se pasa un valor inferior a 0, Flash Player utiliza 0. El valor 0 indica grosor muy fino; el grosor máximo es 255. Si se pasa un valor superior a 255, el intérprete de Flash utiliza 255.

rgb:Number - Valor de color hexadecimal de la línea (por ejemplo, rojo es 0xFF0000, azul es 0x0000FF, etc.). Si no se indica ningún valor, Flash utiliza 0x000000 (negro).

alpha:Number - Entero que indica el valor alfa del color de la línea; los valores válidos son del 0 al 100. Si no se indica ningún valor, Flash utiliza 100 (continuo). Si el valor es inferior a 0, Flash utiliza 0; si el valor es mayor que 100, Flash utiliza 100.

pixelHinting:Boolean - Añadido en Flash Player 8. Valor booleano que especifica si sugerir trazos a píxeles completos. Este valor afecta a la posición de los anclajes de una curva y al tamaño propiamente dicho del trazo de la línea. Si no se indica ningún valor, Flash Player no utiliza consejos de píxeles.

noScale:String - Añadido en Flash Player 8. Cadena que especifica cómo cambiar la escala de un trazo. Los posibles valores son los siguientes:

- "normal" - Siempre cambia la escala del grosor (valor predeterminado).
- "none" Nunca cambia la escala del grosor.
- "vertical" No cambia la escala del grosor si el objeto sólo cambia de escala vertical.
- "horizontal" No cambia la escala del grosor si el objeto sólo cambia de escala horizontal.

capsStyle:String - Añadido en Flash Player 8. Cadena que especifica el tipo de final de línea. Los valores válidos son: "round", "square" y "none". Si no se indica ningún valor, Flash utiliza extremos redondos.

Por ejemplo, las ilustraciones siguientes muestran los diferentes ajustes de capsStyle. Para cada ajuste, la ilustración muestra una línea azul con un grosor de 30 (a la que se aplica capsStyle) y una línea superpuesta de color negro con un grosor de 1 (a la que no se aplica capsStyle):

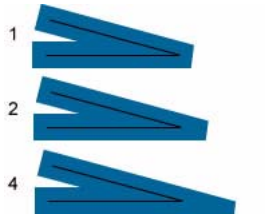
jointStyle:String - Añadido en Flash Player 8. Cadena que especifica el tipo de junta utilizado en los ángulos. Los valores válidos son: "round", "miter" y "bevel". Si no se indica ningún valor, Flash utiliza juntas redondas.

Por ejemplo, las ilustraciones siguientes muestran los diferentes ajustes de capsStyle. Para cada ajuste, la ilustración muestra una línea azul en ángulo con un grosor de 30 (a la que se aplica jointStyle) y una línea superpuesta de color negro en ángulo con un grosor de 1 (a la que no se aplica jointStyle):

Tenga en cuenta que en el caso de `jointStyle` definido como `"miter"`, es posible limitar la longitud del punto de esquina utilizando el parámetro `miterLimit`.

`miterLimit``:Number` - Añadido en Flash Player 8. Número que indica el límite en que se corta una esquina. Los valores posibles están comprendidos entre 1 y 255 (los valores que no estén comprendidos en el rango se redondean a 1 o 255). Este valor sólo se utiliza si `jointStyle` se define como `"miter"`. Si no se indica ningún valor, Flash utiliza 3. El valor de `miterLimit` representa la longitud con la que una esquina se puede extender más allá del punto en que las líneas se cortan para formar una unión. El valor expresa un factor del valor de `thickness` (grosor) de la línea. Por ejemplo, con un factor `miterLimit` de 2,5 y un valor `thickness` (grosor) de 10 píxeles, la esquina se corta a los 25 píxeles.

Por ejemplo, si tenemos las siguientes líneas en ángulo, cada una dibujada con un valor `thickness` (grosor) de 20, pero con `miterLimit` definido en 1, 2 y 4. Se han superpuesto unas líneas de color negro de referencia que muestran los puntos de encuentro de las uniones:



Tenga en cuenta que para un determinado valor `miterLimit`, existe un ángulo máximo específico en el que se corta la esquina. En la siguiente tabla se muestran algunos ejemplos:

Valor de `Value of miterLimit`**:**	**Los ángulos más pequeños se cortan:**
1.414	90 grados
2	60 grados
4	30 grados
8	15 grados

Ejemplo

El código siguiente dibuja un triángulo con una línea continua de 5 píxeles de color magenta sin relleno, con consejos de píxeles, sin cambio de escala de trazo, sin extremos y con juntas de esquina que tienen `miterLimit` definido en 1:

```
this.createEmptyMovieClip("triangle_mc", this.getNextHighestDepth());
triangle_mc.lineStyle(5, 0xff00ff, 100, true, "none", "round", "miter", 1);
triangle_mc.moveTo(200, 200);
```

```
triangle_mc.lineTo(300, 300);
triangle_mc.lineTo(100, 300);
triangle_mc.lineTo(200, 200);
```

Si el archivo SWF incluye un componente de la versión 2, utilice la clase DepthManager de los componentes de la versión 2 en lugar del método `MovieClip.getNextHighestDepth()` utilizado en este ejemplo.

Véase también

`beginFill (método MovieClip.beginFill)`, `beginGradientFill (método MovieClip.beginGradientFill)`, `clear (método MovieClip.clear)`, `curveTo (método MovieClip.curveTo)`, `lineTo (método MovieClip.lineTo)`, `moveTo (método MovieClip.moveTo)`

lineTo (método MovieClip.lineTo)

```
public lineTo(x:Number, y:Number) : Void
```

Dibuja una línea utilizando el estilo de línea actual desde la posición de dibujo actual hasta (x, y); la posición de dibujo actual se establece posteriormente como (x, y). Si el clip de película en el que está dibujando incluye contenido creado con las herramientas de dibujo de Flash, las llamadas a `lineTo()` se dibujarán debajo de este contenido. Si llama a `lineTo()` antes de realizar ninguna llamada al método `moveTo()`, se adoptará la posición de dibujo predeterminada (0,0). Si falta alguno de los parámetros, este método falla y la posición de dibujo actual no cambia.

Puede ampliar los métodos y los controladores de eventos de la clase MovieClip creando una subclase.

Disponibilidad: ActionScript 1.0; Flash Player 6

Parámetros

x`:Number` - Entero que indica la posición horizontal con respecto al punto de registro del clip de película principal.

y`:Number` - Entero que indica la posición vertical con respecto al punto de registro del clip de película principal.

Ejemplo

El ejemplo siguiente dibuja un triángulo con una línea continua magenta de 5 píxeles y un relleno azul parcialmente transparente:

```
this.createEmptyMovieClip("triangle_mc", 1);
triangle_mc.beginFill(0x0000FF, 30);
triangle_mc.lineStyle(5, 0xFF00FF, 100);
```

```
triangle_mc.moveTo(200, 200);
triangle_mc.lineTo(300, 300);
triangle_mc.lineTo(100, 300);
triangle_mc.lineTo(200, 200);
triangle_mc.endFill();
```

Véase también

beginFill (método MovieClip.beginFill), createEmptyMovieClip (método MovieClip.createEmptyMovieClip), endFill (método MovieClip.endFill), lineStyle (método MovieClip.lineStyle), moveTo (método MovieClip.moveTo)

loadMovie (método MovieClip.loadMovie)

```
public loadMovie(url:String, [method:String]) : Void
```

Carga un archivo SWF, JPEG, GIF o PNG en un clip de película en Flash Player mientras se reproduce el archivo SWF original. En Flash Player 8 se ha añadido compatibilidad con archivos GIF no animados, archivos PNG y archivos JPEG progresivos. Si carga un archivo GIF animado, sólo se mostrará el primer fotograma.

Sugerencia: Para controlar el progreso de la descarga, utilice `MovieClipLoader.loadClip()` en lugar del método `loadMovie()`.

Sin el método `loadMovie()`, Flash Player muestra un único archivo SWF y luego se cierra. El método `loadMovie()` permite mostrar varios archivos SWF a la vez y cambiar entre archivos SWF sin cargar otro documento HTML.

Un archivo SWF o imagen cargada en un clip de película hereda las propiedades de posición, giro y escala del clip de película. Puede utilizar la ruta de destino del clip de película para emplear como destino el archivo SWF cargado.

Cuando se realice una llamada al método `loadMovie()`, defina la propiedad `MovieClip._lockroot` en `true` en la película de cargador, tal como se muestra en el ejemplo siguiente. Si no establece `_lockroot` en `true` en la película de cargador, todas las referencias a `_root` que contenga la película cargada señalan a `_root` del cargador en lugar de a `_root` de la película cargada:

```
myMovieClip._lockroot = true;
```

Utilice el método `MovieClip.unloadMovie()` para eliminar los archivos o imágenes SWF cargados con el método `loadMovie()`.

Utilice el método `MovieClip.loadVariables()`, el objeto `XML`, Flash Remoting u objetos compartidos en tiempo de ejecución para mantener el archivo SWF activo y cargar nuevos datos en él.

La utilización de controladores de evento con `MovieClip.loadMovie()` puede producir resultados impredecibles. Si asocia un controlador de eventos a un botón mediante `on()`, o si crea un controlador dinámico empleando un método de controlador de eventos como `MovieClip.onPress()`, y posteriormente efectúa una llamada a `loadMovie()`, el controlador de eventos no permanecerá después de cargarse el nuevo contenido. No obstante, si asocia un controlador de eventos a un clip de película utilizando `onClipEvent()` u `on()` y luego efectúa una llamada a `loadMovie()` en dicho clip de película, el controlador de eventos permanecerá después de cargarse el nuevo contenido.

Cuando utilice este método puede usar el modelo de seguridad de Flash Player.

En Flash Player 8:

- No se admite la carga si el clip de película que realiza la llamada se encuentra en el entorno limitado local con sistema de archivos y el clip de película se encuentra en un entorno limitado de red.
- No se admite la carga si el archivo SWF que realiza la llamada se encuentra en el entorno limitado de red y el clip de película que debe cargarse está en un sistema local.
- Para acceder al entorno limitado de red desde el entorno limitado local de confianza o local con acceso a la red se necesitan permisos del sitio Web a través de un archivo de política de varios dominios.
- Es posible que los clips de película del entorno limitado local con sistema de archivos no puedan crear scripts de clips de película en el entorno limitado local con acceso a la red (y viceversa).

En Flash Player 7 y versiones anteriores:

- Los sitios Web pueden permitir el acceso de varios dominios a un recurso mediante un archivo de política entre varios dominios.
- La creación de scripts entre archivos SWF queda restringida en función del dominio de origen de los archivos SWF. Utilice el método `System.security.allowDomain()` para ajustar estas restricciones.

Para más información, consulte las referencias siguientes:

- Capitulo 17, "Aspectos básicos de la seguridad", de *Aprendizaje de ActionScript 2.0 en Flash*
- El documento técnico sobre la seguridad de Flash Player 8 en http://www.macromedia.com/go/fp8_security
- El documento técnico sobre interfaces API relativas a seguridad de Flash Player 8 en http://www.macromedia.com/go/fp8_security_apis

Puede ampliar los métodos y los controladores de eventos de la clase MovieClip creando una subclase.

Disponibilidad: ActionScript 1.0; Flash Player 5

Parámetros

url:String - URL absoluta o relativa del archivo SWF, JPEG, GIF y PNG que va a cargarse. Una ruta relativa debe ser relativa al archivo SWF en el nivel 0. Las URL absolutas deben incluir la referencia al protocolo, como `http://` o `file:///`.

method:String [opcional] - Especifica un método HTTP para enviar o cargar variables. El parámetro debe ser la cadena `GET` o `POST`. Si no necesita enviar variables, omita este parámetro. El método `GET` añade las variables al final de la URL y se utiliza para un número reducido de variables. El método `POST` envía las variables en un encabezado HTTP independiente y se utiliza para enviar cadenas de variables largas.

Ejemplo

El ejemplo siguiente crea dinámicamente un nuevo de clip de película, después, crea un elemento secundario en dicho clip de película y carga una imagen PNG en el elemento secundario. De este modo el elemento principal conserva los valores de instancia que se asignaron antes de realizar la llamada a `loadMovie`.

```
var mc:MovieClip = this.createEmptyMovieClip("mc",
  this.getNextHighestDepth());
mc.onRelease = function():Void {
  trace(this.image._url); // http://www.w3.org/Icons/w3c_main.png
}
var image:MovieClip = mc.createEmptyMovieClip("image",
  mc.getNextHighestDepth());
image.loadMovie("http://www.w3.org/Icons/w3c_main.png");
```

El método `MovieClip.getNextHighestDepth()` utilizado en este ejemplo necesita Flash Player 7 o una versión posterior. Si el archivo SWF incluye un componente de la versión 2, utilice la clase DepthManager de los componentes de la versión 2 en lugar del método `MovieClip.getNextHighestDepth()`.

Véase también

`_lockroot (propiedad MovieClip._lockroot)`, `unloadMovie (método MovieClip.unloadMovie)`, `loadVariables (método MovieClip.loadVariables)`, `loadMovie (método MovieClip.loadMovie)`, `onPress (controlador MovieClip.onPress)`, `MovieClipLoader`, `Controlador onClipEvent`, `on handler`, `Función loadMovieNum`, `Función unloadMovie`, `Función unloadMovieNum`

loadVariables (método MovieClip.loadVariables)

```
public loadVariables(url:String, [method:String]) : Void
```

Lee datos de un archivo externo y establece los valores de variables en el clip de película. El archivo externo puede ser un archivo de texto generado por Macromedia ColdFusion, un script CGI, Active Server Page (ASP) o un script PHP o cualquier archivo de texto que tenga los formatos correctos. El archivo puede contener cualquier número de variables.

El método `loadVariables()` también puede utilizarse para actualizar variables del clip de película activo con nuevos valores.

El método `loadVariables()` exige que el texto de la URL esté en formato MIME estándar: *application/x-www-form-urlencoded* (formato de script CGI).

En archivos SWF que se ejecuten en una versión anterior a Flash Player 7, `url` debe estar en el mismo superdominio que el archivo SWF que emite esta llamada. El superdominio puede obtenerse eliminando el componente situado más a la izquierda de la URL de un archivo. Por ejemplo, un archivo SWF situado en www.someDomain.com puede cargar datos de una fuente situada en store.someDomain.com porque ambos archivos se encuentran en el mismo superdominio, denominado someDomain.com.

En archivos SWF de cualquier versión que se ejecuten en Flash Player 7 o versiones posteriores, `url` debe estar exactamente en el mismo dominio que el archivo SWF que emite esta llamada. Por ejemplo, un archivo SWF en www.someDomain.com puede cargar datos únicamente desde orígenes que también se encuentren en www.someDomain.com. Para cargar datos de un dominio distinto, puede colocar un archivo de política para distintos dominios en el servidor que aloja el origen de los datos a los que se obtiene acceso.

Para cargar variables en un nivel específico, utilice `loadVariablesNum()` en lugar de `loadVariables()`.

Puede ampliar los métodos y los controladores de eventos de la clase MovieClip creando una subclase.

Disponibilidad: ActionScript 1.0; Flash Player 5

Parámetros

`url``:String` - URL absoluta o relativa del archivo externo que contiene las variables que se van a cargar. Si el archivo SWF que realiza esta llamada se ejecuta en un navegador Web, el valor `url` debe pertenecer al mismo dominio que el archivo SWF; para ver más detalles, consulte la descripción siguiente.

method:String [opcional] - Especifica un método HTTP para enviar variables. El parámetro debe ser la cadena GET o POST. Si no se envía ninguna variable, omita este parámetro. El método GET añade las variables al final de la URL y se utiliza para un número reducido de variables. El método POST envía las variables en un encabezado HTTP independiente y se utiliza para enviar cadenas de variables largas.

Ejemplo

El ejemplo siguiente carga información de un archivo de texto denominado params.txt en el clip de película target_mc que se ha creado con createEmptyMovieClip(). Utilice la función setInterval() para comprobar el progreso de la carga. El script busca en el archivo params.txt una variable denominada done.

```
this.createEmptyMovieClip("target_mc", this.getNextHighestDepth());
target_mc.loadVariables("params.txt");
function checkParamsLoaded() {
  if (target_mc.done == undefined) {
  trace("not yet.");
  } else {
  trace("finished loading. killing interval.");
  trace("-------------");
  for (i in target_mc) {
    trace(i+": "+target_mc[i]);
  }
  trace("-------------");
  clearInterval(param_interval);
  }
}
var param_interval = setInterval(checkParamsLoaded, 100);
```

El archivo params.txt incluye el texto siguiente:

```
var1="hello"&var2="goodbye"&done="done"
```

El método MovieClip.getNextHighestDepth() utilizado en este ejemplo necesita Flash Player 7 o una versión posterior. Si el archivo SWF incluye un componente de la versión 2, utilice la clase DepthManager de los componentes de la versión 2 en lugar del método MovieClip.getNextHighestDepth().

Véase también

loadMovie (método MovieClip.loadMovie), Función loadVariablesNum, unloadMovie (método MovieClip.unloadMovie)

localToGlobal (método MovieClip.localToGlobal)

```
public localToGlobal(pt:Object) : Void
```

Convierte el objeto `pt` de coordenadas del clip de película (locales) en coordenadas del escenario (globales).

El método `MovieClip.localToGlobal()` permite convertir los valores de las coordenadas *x* e *y* dadas, relativos a la esquina superior izquierda de un clip de película especificado, en valores relativos a la esquina superior izquierda del escenario.

Primero debe crear un objeto genérico que tenga dos propiedades, `x` e `y`. Estos valores `x` e `y` (sólo pueden denominarse `x` e `y`) son las coordenadas locales debido a que son relativos a la esquina superior izquierda del clip de película. La propiedad x representa el desplazamiento horizontal con respecto a la esquina superior izquierda del clip de película. En otras palabras, representa la distancia del punto hacia la derecha. Por ejemplo, si x = 50, el punto queda 50 píxeles a la derecha de la esquina superior izquierda. La propiedad y representa el desplazamiento vertical con respecto a la esquina superior izquierda del clip de película. En otras palabras, representa la distancia del punto hacia abajo. Por ejemplo, si y = 20, el punto queda 20 píxeles por debajo de la esquina superior izquierda. El código siguiente crea un objeto genérico con estas coordenadas.

```
var myPoint:Object = new Object();
myPoint.x = 50;
myPoint.y = 20;
```

También es posible crear el objeto y asignarle simultáneamente los valores con un valor literal de objeto.

```
var myPoint:Object = {x:50, y:20};
```

Una vez creado un objeto punto con coordenadas locales, puede convertir las coordenadas en coordenadas globales. El método `localToGlobal()` no devuelve un valor porque cambia los valores de `x` e `y` del objeto genérico enviado como parámetro. Sustituye los valores relativos a un clip de película determinado (coordenadas locales) por valores relativos al escenario (coordenadas globales).

Por ejemplo, si crea un clip de película situado en el punto (_x:100, _y:100) y pasa un punto local que representa un punto próximo a la esquina superior izquierda del clip de película (x:10, y:10) al método `localToGlobal()` el método debería convertir los valores de *x* e *y* en coordenadas globales, en este caso (x:110, y:110). Esta conversión ocurre porque ahora las coordenadas *x* e *y* se expresan en relación con la esquina superior izquierda del escenario, no del clip de película.

Las coordenadas del clip de película se han expresado con _x e _y porque son las propiedades de MovieClip utilizadas para definir los valores *x* e *y* para MovieClips. Sin embargo, el objeto genérico utiliza x e y sin subrayado. El siguiente código convierte las coordenadas *x* e *y* en coordenadas globales:

```
var myPoint:Object = {x:10, y:10}; // create your generic point object
this.createEmptyMovieClip("myMovieClip", this.getNextHighestDepth());
myMovieClip._x = 100; // _x for movieclip x position
myMovieClip._y = 100; // _y for movieclip y position

myMovieClip.localToGlobal(myPoint);
trace ("x: " + myPoint.x); // output: 110
trace ("y: " + myPoint.y); // output: 110
```

Puede ampliar los métodos y los controladores de eventos de la clase MovieClip creando una subclase.

Disponibilidad: ActionScript 1.0; Flash Player 5

Parámetros

pt:Object - Nombre o identificador de un objeto creado con la clase Object que especifica las coordenadas *x* e *y* como propiedades.

Ejemplo

El ejemplo siguiente convierte las coordenadas *x* e *y* del objeto my_mc, coordenadas de clip de película (locales), en coordenadas del escenario (globales). El punto central del clip de película se refleja después de hacer clic en la instancia y arrastrarla.

```
this.createTextField("point_txt", this.getNextHighestDepth(), 0, 0, 100,
  22);
var mouseListener:Object = new Object();
mouseListener.onMouseMove = function() {
 var point:Object = {x:my_mc._width/2, y:my_mc._height/2};
 my_mc.localToGlobal(point);
 point_txt.text = "x:"+point.x+", y:"+point.y;
};
Mouse.addListener(mouseListener);
my_mc.onPress = function() {
 this.startDrag();
};
my_mc.onRelease = function() {
 this.stopDrag();
};
```

El método MovieClip.getNextHighestDepth() utilizado en este ejemplo necesita Flash Player 7 o una versión posterior. Si el archivo SWF incluye un componente de la versión 2, utilice la clase DepthManager de los componentes de la versión 2 en lugar del método MovieClip.getNextHighestDepth().

Véase también

globalToLocal (método MovieClip.globalToLocal)

_lockroot (propiedad MovieClip._lockroot)

```
public _lockroot : Boolean
```

Un valor booleano que especifica a qué hace referencia `_root` cuando se carga un archivo SWF en un clip de película. La propiedad `_lockroot` tiene el valor `undefined` de manera predeterminada. Puede establecer esta propiedad dentro del archivo SWF que se está cargando o en el controlador que está cargando el clip de película.

Por ejemplo, supongamos que tiene un documento denominado Games.fla que permite a un usuario elegir un juego y que carga dicho juego (por ejemplo, Chess.swf) en el clip de película `game_mc`. Asegúrese de que después de haber cargado Games.swf, cualquier uso de `_root` en Chess.swf hace referencia a `_root` en Chess.swf (y no a `_root` en Games.swf). Si tiene acceso a Chess.fla y lo publica con Flash Player 7 o una versión posterior, puede agregar esta sentencia en línea de tiempo principal de Chess.fla:

```
this._lockroot = true;
```

Si no dispone de acceso a Chess.fla (por ejemplo, si está cargando Chess.swf en `chess_mc` desde un sitio ajeno), podrá establecer la propiedad `_lockroot` de Chess.swf cuando la cargue. Sitúe el siguiente código ActionScript en la línea de tiempo principal de Games.fla:

```
chess_mc._lockroot = true;
```

En este caso, Chess.swf se podrá publicar para cualquier versión de Flash Player en tanto en cuanto Games.swf se publique para Flash Player 7 o posterior.

Cuando se realice una llamada al método `loadMovie()`, defina la propiedad `MovieClip._lockroot` en `true` en la película de cargador, tal como se muestra en el código siguiente. Si no establece `_lockroot` en `true` en la película de cargador, todas las referencias a `_root` que contenga la película cargada señalan a `_root` del cargador en lugar de a `_root` de la película cargada:

```
myMovieClip._lockroot = true;
```

Disponibilidad: ActionScript 1.0; Flash Player 7

Ejemplo

En el ejemplo siguiente, lockroot.fla tiene `_lockroot` aplicado en el archivo SWF principal. Si está cargado en otro documento FLA, `_root` siempre hace referencia al ámbito de lockroot.swf, lo que ayuda a prevenir conflictos. Sitúe el siguiente código ActionScript en la línea de tiempo principal de lockroot.fla:

```
this._lockroot = true;
```

```
_root.myVar = 1;
_root.myOtherVar = 2;
trace("from lockroot.swf");
for (i in _root) {
  trace(" "+i+" -> "+_root[i]);
}
trace("");
```

que averigua la siguiente información:

```
from lockroot.swf
myOtherVar -> 2
myVar -> 1
_lockroot -> true
$version -> WIN 7,0,19,0
```

El ejemplo siguiente carga dos archivos SWF, lockroot.swf y nolockroot.swf. El documento lockroot.fla contiene el código ActionScript del ejemplo anterior. El archivo nolockroot.fla tiene el siguiente código añadido al fotograma 1 de la línea de tiempo:

```
_root.myVar = 1;
_root.myOtherVar = 2;
trace("from nolockroot.swf");
for (i in _root) {
  trace(" "+i+" -> "+_root[i]);
}
trace("");
```

El archivo lockroot.swf tiene aplicado `lockroot`, no así nolockroot.swf. Una vez cargados los archivos, cada uno muestra los valores de las variables de sus ámbitos `_root`. Inserte el siguiente código ActionScript en la línea de tiempo principal del documento FLA:

```
this.createEmptyMovieClip("lockroot_mc", this.getNextHighestDepth());
lockroot_mc.loadMovie("lockroot.swf");
this.createEmptyMovieClip("nolockroot_mc", this.getNextHighestDepth());
nolockroot_mc.loadMovie("nolockroot.swf");
function dumpRoot() {
  trace("from current SWF file");
  for (i in _root) {
  trace(" "+i+" -> "+_root[i]);
  }
  trace("");
}
dumpRoot();
```

que obtiene la siguiente información:

```
 from current SWF file
 dumpRoot -> [type Function]
 $version -> WIN 7,0,19,0
 nolockroot_mc -> _level0.nolockroot_mc
 lockroot_mc -> _level0.lockroot_mc
```

```
from nolockroot.swf
myVar -> 1
i -> lockroot_mc
dumpRoot -> [type Function]
$version -> WIN 7,0,19,0
nolockroot_mc -> _level0.nolockroot_mc
lockroot_mc -> _level0.lockroot_mc

from lockroot.swf
myOtherVar -> 2
myVar -> 1
```

El archivo que no tiene `_lockroot` aplicado también contiene todas las demás variables que contiene el archivo SWF raíz. Si no tiene acceso a nolockroot.fla, puede utilizar el siguiente código ActionScript añadido a la línea de tiempo principal para cambiar `_lockroot` en el documento FLA principal precedente:

```
this.createEmptyMovieClip("nolockroot_mc", this.getNextHighestDepth());
nolockroot_mc._lockroot = true;
nolockroot_mc.loadMovie("nolockroot.swf");
```

que entonces obtendría lo siguiente:

```
from current SWF file
dumpRoot -> [type Function]
$version -> WIN 7,0,19,0
nolockroot_mc -> _level0.nolockroot_mc
lockroot_mc -> _level0.lockroot_mc

from nolockroot.swf
myOtherVar -> 2
myVar -> 1

from lockroot.swf
myOtherVar -> 2
myVar -> 1
```

Véase también

Propiedad _root, _lockroot (propiedad MovieClip._lockroot), attachMovie (método MovieClip.attachMovie), loadMovie (método MovieClip.loadMovie), onLoadInit (detector de eventos MovieClipLoader.onLoadInit)

menu (propiedad MovieClip.menu)

```
public menu : ContextMenu
```

Asocia el objeto ContextMenu especificado al clip de película. La clase ContextMenu le permite modificar el menú contextual que se muestra cuando un usuario hace clic con el botón derecho (Windows) o hace clic mientras presiona la tecla Control (Macintosh) en Flash Player.

Disponibilidad: ActionScript 1.0; Flash Player 7

Ejemplo

El ejemplo siguiente asigna el objeto ContextMenu `menu_cm` al clip de película `image_mc`. El objeto ContextMenu contiene un elemento de menú personalizado con la etiqueta "View Image in Browser..." que lleva asociada una función denominada `viewImage()`.

```
var menu_cm:ContextMenu = new ContextMenu();
menu_cm.customItems.push(new ContextMenuItem("View Image in Browser...",
  viewImage));
this.createEmptyMovieClip("image_mc", this.getNextHighestDepth());
var mclListener:Object = new Object();
mclListener.onLoadInit = function(target_mc:MovieClip) {
  target_mc.menu = menu_cm;
};
var image_mcl:MovieClipLoader = new MovieClipLoader();
image_mcl.addListener(mclListener);
image_mcl.loadClip("photo1.jpg", image_mc);

function viewImage(target_mc:MovieClip, obj:Object) {
  getURL(target_mc._url, "_blank");
}
```

Tras hacer clic en la imagen con el botón derecho del ratón (Windows) o con la tecla Control presionada (Macintosh) en tiempo de ejecución, seleccione View Image in Browser en el menú contextual para abrir una imagen en una ventana del navegador.

Véase también

`menu (propiedad Button.menu)`, `ContextMenu`, `ContextMenuItem`, `menu (propiedad TextField.menu)`

moveTo (método MovieClip.moveTo)

```
public moveTo(x:Number, y:Number) : Void
```

Mueve la posición de dibujo actual a (x, y). Si falta alguno de los parámetros, este método falla y la posición de dibujo actual no cambia.

Puede ampliar los métodos y los controladores de eventos de la clase MovieClip creando una subclase.

Disponibilidad: ActionScript 1.0; Flash Player 6

Parámetros

x:Number - Entero que indica la posición horizontal con respecto al punto de registro del clip de película principal.

y:Number - Entero que indica la posición vertical con respecto al punto de registro del clip de película principal.

Ejemplo

El ejemplo siguiente dibuja un triángulo con una línea continua magenta de 5 píxeles y un relleno azul parcialmente transparente:

```
this.createEmptyMovieClip("triangle_mc", 1);
triangle_mc.beginFill(0x0000FF, 30);
triangle_mc.lineStyle(5, 0xFF00FF, 100);
triangle_mc.moveTo(200, 200);
triangle_mc.lineTo(300, 300);
triangle_mc.lineTo(100, 300);
triangle_mc.lineTo(200, 200);
triangle_mc.endFill();
```

Véase también

createEmptyMovieClip (método MovieClip.createEmptyMovieClip), lineStyle (método MovieClip.lineStyle), lineTo (método MovieClip.lineTo)

_name (propiedad MovieClip._name)

```
public _name : String
```

Nombre de instancia del clip de película.

Disponibilidad: ActionScript 1.0; Flash Player 4

Ejemplo

El ejemplo siguiente permite hacer clic con el botón derecho del ratón (Windows) o con la tecla Control presionada (Macintosh) en un clip de película del escenario y seleccionar `Info` en el menú contextual para ver información sobre esa instancia. Añada varios clips de película con nombres de instancia y a continuación añada el siguiente código ActionScript al archivo AS o FLA:

```
var menu_cm:ContextMenu = new ContextMenu();
```

```
menu_cm.customItems.push(new ContextMenuItem("Info...", getMCInfo));
function getMCInfo(target_mc:MovieClip, obj:Object) {
  trace("You clicked on the movie clip '"+target_mc._name+"'.");
  trace("\t width:"+target_mc._width+", height:"+target_mc._height);
  trace("");
}
for (var i in this) {
  if (typeof (this[i]) == 'movieclip') {
  this[i].menu = menu_cm;
  }
}
```

Véase también

_name (propiedad Button._name)

nextFrame (método MovieClip.nextFrame)

```
public nextFrame() : Void
```

Traslada la cabeza lectora al siguiente fotograma y la detiene en dicho punto.

Puede ampliar los métodos y los controladores de eventos de la clase MovieClip creando una subclase.

Disponibilidad: ActionScript 1.0; Flash Player 5

Ejemplo

El ejemplo siguiente utiliza `_framesloaded` y `nextFrame()` para cargar contenido en un archivo SWF. No añada ningún código al fotograma 1, pero añada el siguiente código ActionScript en el fotograma 2 de la línea de tiempo:

```
if (this._framesloaded >= 3) {
  this.nextFrame();
} else {
  this.gotoAndPlay(1);
}
```

A continuación, añada el código siguiente (y el contenido que desee cargar) al fotograma 3:

```
stop();
```

Véase también

Función nextFrame, Función prevFrame, prevFrame (método MovieClip.prevFrame)

onData (controlador MovieClip.onData)

```
onData = function() {}
```

Se invoca cuando un clip de película recibe datos de una llamada `MovieClip.loadVariables()` o `MovieClip.loadMovie()`. Debe definir la función que ha de ejecutarse cuando se invoca el controlador de eventos. Puede definir la función en la línea de tiempo o en un archivo de clase que amplíe la clase MovieClip y esté vinculado a un símbolo de la biblioteca.

Sólo puede utilizar este controlador con el método `MovieClip.loadVariables()` o la función global `loadVariables()`. Si desea que se invoque el controlador de eventos con el método `MovieClip.loadMovie()` o la función `loadMovie()`, deberá utilizar `onClipEvent(data)` en lugar de este controlador.

Disponibilidad: ActionScript 1.0; Flash Player 6

Ejemplo

El ejemplo siguiente ilustra cómo utilizar correctamente el método `MovieClip.onData()`. Carga un archivo denominado *OnData.txt* el mismo directorio que el del archivo FLA. Cuando los datos del archivo se cargan en el objeto `MovieClip`, se ejecuta `onData()` y se obtienen los datos.

```
var mc:MovieClip = this.createEmptyMovieClip("mc",
  this.getNextHighestDepth());

mc.onData = function() {
  for(var i in this) {
    trace(">> " + i + ": " + this[i]);
  }
}

mc.loadVariables("OnData.txt");
```

Véase también

Controlador onClipEvent, loadVariables (método MovieClip.loadVariables)

onDragOut (controlador MovieClip.onDragOut)

```
onDragOut = function() {}
```

Se invoca cuando se presiona el botón del ratón y el puntero se desplaza fuera del objeto. Debe definir la función que ha de ejecutarse cuando se invoca el controlador de eventos. Puede definir la función en la línea de tiempo o en un archivo de clase que amplíe la clase MovieClip o esté vinculado a un símbolo de la biblioteca.

Disponibilidad: ActionScript 1.0; Flash Player 6

Ejemplo

El ejemplo siguiente define una función para el método `onDragOut` que envía una acción `trace()` al panel Salida:

```
my_mc.onDragOut = function () {
  trace ("onDragOut called");
}
```

Véase también

`onDragOver (controlador MovieClip.onDragOver)`

onDragOver (controlador MovieClip.onDragOver)

```
onDragOver = function() {}
```

Se invoca cuando el puntero se arrastra fuera del clip de película y luego se pasa por encima de éste. Debe definir la función que ha de ejecutarse cuando se invoca el controlador de eventos. Puede definir la función en la línea de tiempo o en un archivo de clase que amplíe la clase MovieClip o esté vinculado a un símbolo de la biblioteca.

Disponibilidad: ActionScript 1.0; Flash Player 6

Ejemplo

El ejemplo siguiente define una función para el método `onDragOver` que envía una acción `trace()` al panel Salida:

```
my_mc.onDragOver = function () {
  trace ("onDragOver called");
}
```

Véase también

`onDragOut (controlador MovieClip.onDragOut)`

onEnterFrame (controlador MovieClip.onEnterFrame)

```
onEnterFrame = function() {}
```

Se invoca de manera repetida con la velocidad de fotogramas del archivo SWF. La función que se asigne al controlador de eventos `onEnterFrame` se procesa antes que cualquier otro código ActionScript que este asignado a los fotogramas afectados.

Debe definir la función que ha de ejecutarse cuando se invoca el controlador de eventos. Puede definir la función en la línea de tiempo o en un archivo de clase que amplíe la clase MovieClip o esté vinculado a un símbolo de la biblioteca.

Disponibilidad: ActionScript 1.0; Flash Player 6

Ejemplo

El ejemplo siguiente define una función para el controlador de eventos `onEnterFrame` que envía una acción `trace()` al panel Salida:

```
my_mc.onEnterFrame = function () {
  trace ("onEnterFrame called");
}
```

onKeyDown (controlador MovieClip.onKeyDown)

```
onKeyDown = function() {}
```

Se invoca cuando un clip de película está resaltado y se presiona una tecla. El controlador de eventos `onKeyDown` se invoca sin parámetros. Puede utilizar los métodos `Key.getAscii()` y `Key.getCode()` para determinar qué tecla ha presionado el usuario. Debe definir la función que ha de ejecutarse cuando se invoca el controlador de eventos. Puede definir la función en la línea de tiempo o en un archivo de clase que amplíe la clase MovieClip o esté vinculado a un símbolo de la biblioteca.

El controlador de eventos `onKeyDown` sólo funciona si el clip de película tiene la selección de entrada activada y establecida. En primer lugar, la propiedad `MovieClip.focusEnabled` debe establecerse como `true` para el clip de película. Seguidamente, el clip debe recibir la selección de entrada. Esto puede hacerse utilizando `Selection.setFocus()` o estableciendo la tecla Tabulador para que se desplace al clip.

Si utiliza `Selection.setFocus()`, debe pasarse la ruta de acceso al clip de película a `Selection.setFocus()`. Es fácil que otros elementos reciban la selección de entrada después de que un usuario mueva el ratón.

Disponibilidad: ActionScript 1.0; Flash Player 6

Ejemplo

El ejemplo siguiente define una función para el método `onKeyDown()` que envía una acción `trace()` al panel Salida. Cree un clip de película denominado my_mc y añada el siguiente código ActionScript al archivo FLA o AS:

```
my_mc.onKeyDown = function () {
  trace ("key was pressed");
}
```

El clip de película debe estar seleccionado para que el controlador de eventos `onKeyDown` funcione. Añada el siguiente código ActionScript para establecer la selección de entrada:

```
my_mc.tabEnabled = true;
my_mc.focusEnabled = true;
Selection.setFocus(my_mc);
```

Cuando el usuario pulsa una tecla, `key was pressed` aparece en el panel Salida. Sin embargo, esto no ocurre si se desplaza el ratón, porque el clip de película deja de estar seleccionado. Por lo tanto, debería utilizar `Key.onKeyDown` en la mayoría de los casos.

Véase también

`getAscii (método Key.getAscii)`, `getCode (método Key.getCode)`, `onKeyDown (detector de eventos Key.onKeyDown)`, `focusEnabled (propiedad MovieClip.focusEnabled)`, `onKeyUp (controlador MovieClip.onKeyUp)`, `setFocus (método Selection.setFocus)`

onKeyUp (controlador MovieClip.onKeyUp)

```
onKeyUp = function() {}
```

Se invoca cuando se suelta una tecla. El controlador de eventos `onKeyUp` se invoca sin parámetros. Puede utilizar los métodos `Key.getAscii()` y `Key.getCode()` para determinar qué tecla ha presionado el usuario. Debe definir la función que ha de ejecutarse cuando se invoca el controlador de eventos. Puede definir la función en la línea de tiempo o en un archivo de clase que amplíe la clase MovieClip o esté vinculado a un símbolo de la biblioteca.

El controlador de eventos `onKeyUp` sólo funciona si el clip de película tiene la selección de entrada activada y establecida. En primer lugar, la propiedad `MovieClip.focusEnabled` debe establecerse como `true` para el clip de película. Seguidamente, el clip de película debe recibir la selección de entrada. Esto puede hacerse utilizando `Selection.setFocus()` o estableciendo la tecla Tabulador para que se desplace al clip.

Si utiliza `Selection.setFocus()`, debe pasarse la ruta de acceso al clip de película a `Selection.setFocus()`. Es fácil que otros elementos reciban la selección de entrada después de que un usuario mueva el ratón.

Disponibilidad: ActionScript 1.0; Flash Player 6

Ejemplo

El ejemplo siguiente define una función para el método `onKeyUp` que envía una acción `trace()` al panel Salida:

```
my_mc.onKeyUp = function () {
  trace ("onKey called");
}
```

El ejemplo siguiente define la selección de entrada:

```
my_mc.focusEnabled = true;
Selection.setFocus(my_mc);
```

Véase también

`getAscii (método Key.getAscii)`, `getCode (método Key.getCode)`, `onKeyDown (detector de eventos Key.onKeyDown)`, `focusEnabled (propiedad MovieClip.focusEnabled)`, `onKeyDown (controlador MovieClip.onKeyDown)`, `setFocus (método Selection.setFocus)`

onKillFocus (controlador MovieClip.onKillFocus)

```
onKillFocus = function(newFocus:Object) {}
```

Se invoca cuando un clip de película deja de estar seleccionado con el teclado. El método `onKillFocus` recibe un parámetro, `newFocus`, que es un objeto que representa al nuevo objeto seleccionado. Si no hay ningún objeto seleccionado con el teclado, `newFocus` contendrá el valor `null` (nulo).

Debe definir la función que ha de ejecutarse cuando se invoca el controlador de eventos. Puede definir la función en la línea de tiempo o en un archivo de clase que amplíe la clase MovieClip o esté vinculado a un símbolo de la biblioteca.

Disponibilidad: ActionScript 1.0; Flash Player 6

Parámetros

`newFocus``:Object` - El objeto que recibe la selección de teclado.

Ejemplo

El ejemplo siguiente muestra información sobre el clip de película que deja de estar seleccionado y la instancia que está seleccionada actualmente. En el escenario hay dos clips de película denominados `my_mc` y `other_mc`. Añada el siguiente código ActionScript al documento FLA o AS:

```
my_mc.onRelease = Void;
other_mc.onRelease = Void;
my_mc.onKillFocus = function(newFocus) {
  trace("onKillFocus called, new focus is: "+newFocus);
};
```

Pulse la tecla Tabulador entre las dos instancias y aparecerá información en el panel Salida.

Véase también

`onSetFocus (controlador MovieClip.onSetFocus)`

onLoad (controlador MovieClip.onLoad)

```
onLoad = function() {}
```

Se invoca cuando se crea una instancia del clip de película y ésta aparece en la línea de tiempo. Debe definir la función que ha de ejecutarse cuando se invoca el controlador de eventos. Puede definir la función en la línea de tiempo o en un archivo de clase que amplíe la clase MovieClip o esté vinculado a un símbolo de la biblioteca.

Este controlador sólo puede se utilizar con clips de película para los que disponga de símbolo en la biblioteca asociada a la clase. Si desea que se invoque un controlador de eventos al cargarse un determinado clip de película, deberá utilizar `onClipEvent(load)` o la clase MovieClipLoader en lugar de este controlador, por ejemplo, cuando utilice `MovieClip.loadMovie()` para cargar dinámicamente un archivo SWF. A diferencia de `MovieClip.onLoad`, los demás controladores se invocan cuando se carga cualquier clip de película.

Disponibilidad: ActionScript 1.0; Flash Player 6

Ejemplo

Este ejemplo muestra cómo utilizar el controlador de eventos `onLoad` en una definición de clase de ActionScript 2.0 que amplía la clase MovieClip. En primer lugar, cree un archivo de clase denominado Oval.as y defina un método de clase denominado `onLoad()`. Después, asegúrese de que el archivo de clase está situado en la ruta de clases correcta, como en el ejemplo siguiente:

```
// contents of Oval.as
class Oval extends MovieClip{
  public function onLoad () {
    trace ("onLoad called");
  }
}
```

En segundo lugar, cree un símbolo de clip de película en la biblioteca y denomínelo Oval. Haga clic en cualquier punto del símbolo en el panel Biblioteca (normalmente con el botón derecho del ratón) y seleccione Vinculación... en el menú emergente. Haga clic en la casilla Exportar para ActionScript e introduzca `Oval` en los campos de clase e identificador de ActionScript. Verifique que la opción "Exportar en primer fotograma" está seleccionada y haga clic en Aceptar.

En tercer lugar, desplácese al primer fotograma del archivo e introduzca el código siguiente en el panel Acciones:

```
var myOval:Oval = Oval(attachMovie("Oval","Oval_1",1));
```

Finalmente, realice una prueba de la película; debería ver el texto de salida "onLoad called".

Véase también

`loadMovie (método MovieClip.loadMovie)`, `Controlador onClipEvent`, `MovieClipLoader`

onMouseDown (controlador MovieClip.onMouseDown)

```
onMouseDown = function() {}
```

Se invoca cuando se presiona el botón del ratón. Debe definir la función que ha de ejecutarse cuando se invoca el controlador de eventos. Puede definir la función en la línea de tiempo o en un archivo de clase que amplíe la clase MovieClip o esté vinculado a un símbolo de la biblioteca.

Disponibilidad: ActionScript 1.0; Flash Player 6

Ejemplo

El ejemplo siguiente define una función para el método `onMouseDown()` que envía una acción `trace()` al panel Salida:

```
my_mc.onMouseDown = function () {
  trace ("onMouseDown called");
}
```

onMouseMove (controlador MovieClip.onMouseMove)

```
onMouseMove = function() {}
```

Se invoca cuando se mueve el ratón. Debe definir la función que ha de ejecutarse cuando se invoca el controlador de eventos. Puede definir la función en la línea de tiempo o en un archivo de clase que amplíe la clase MovieClip o esté vinculado a un símbolo de la biblioteca.

Disponibilidad: ActionScript 1.0; Flash Player 6

Ejemplo

El ejemplo siguiente define una función para el método `onMouseMove()` que envía una acción `trace()` al panel Salida:

```
my_mc.onMouseMove = function () {
  trace ("onMouseMove called");
}
```

onMouseUp (controlador MovieClip.onMouseUp)

```
onMouseUp = function() {}
```

Se invoca cuando se suelta el botón del ratón. Debe definir la función que ha de ejecutarse cuando se invoca el controlador de eventos. Puede definir la función en la línea de tiempo o en un archivo de clase que amplíe la clase MovieClip o esté vinculado a un símbolo de la biblioteca.

Disponibilidad: ActionScript 1.0; Flash Player 6

Ejemplo

El ejemplo siguiente define una función para el método `onMouseUp()` que envía una acción `trace()` al panel Salida:

```
my_mc.onMouseUp = function () {
  trace ("onMouseUp called");
}
```

onPress (controlador MovieClip.onPress)

```
onPress = function() {}
```

Se invoca cuando el usuario hace clic en el ratón mientras el puntero se encuentra sobre el clip de película. Debe definir la función que ha de ejecutarse cuando se invoca el controlador de eventos. Puede definir la función en la biblioteca.

Disponibilidad: ActionScript 1.0; Flash Player 6

Ejemplo

El ejemplo siguiente define una función para el método `onPress()` que envía una acción `trace()` al panel Salida:

```
my_mc.onPress = function () {
  trace ("onPress called");
}
```

onRelease (controlador MovieClip.onRelease)

```
onRelease = function() {}
```

Se invoca cuando se suelta el botón del ratón sobre un clip de película. Debe definir la función que ha de ejecutarse cuando se invoca el controlador de eventos. Puede definir la función en la línea de tiempo o en un archivo de clase que amplíe la clase MovieClip o esté vinculado a un símbolo de la biblioteca.

Disponibilidad: ActionScript 1.0; Flash Player 6

Ejemplo

El ejemplo siguiente define una función para el método `onRelease()` que envía una acción `trace()` al panel Salida:

```
my_mc.onRelease = function () {
  trace ("onRelease called");
}
```

onReleaseOutside (controlador MovieClip.onReleaseOutside)

```
onReleaseOutside = function() {}
```

Se invoca después de que se presione el botón del ratón dentro del área de un clip de película y luego se suelte fuera del área del clip de película.

Debe definir la función que ha de ejecutarse cuando se invoca el controlador de eventos. Puede definir la función en la línea de tiempo o en un archivo de clase que amplíe la clase MovieClip o esté vinculado a un símbolo de la biblioteca.

Disponibilidad: ActionScript 1.0; Flash Player 6

Ejemplo

El ejemplo siguiente define una función para el método `onReleaseOutside()` que envía una acción `trace()` al panel Salida:

```
my_mc.onReleaseOutside = function () {
  trace ("onReleaseOutside called");
}
```

onRollOut (controlador MovieClip.onRollOut)

```
onRollOut = function() {}
```

Se invoca cuando el puntero se desplaza fuera del área de un clip de película.

Debe definir la función que ha de ejecutarse cuando se invoca el controlador de eventos. Puede definir la función en la línea de tiempo o en un archivo de clase que amplíe la clase MovieClip o esté vinculado a un símbolo de la biblioteca.

Disponibilidad: ActionScript 1.0; Flash Player 6

Ejemplo

El ejemplo siguiente define una función para el método `onRollOut()` que envía una acción `trace()` al panel Salida:

```
my_mc.onRollOut = function () {
  trace ("onRollOut called");
}
```

onRollOver (controlador MovieClip.onRollOver)

```
onRollOver = function() {}
```

Se invoca cuando el puntero se desplaza sobre un área de un clip de película.

Debe definir la función que ha de ejecutarse cuando se invoca el controlador de eventos. Puede definir la función en la línea de tiempo o en un archivo de clase que amplíe la clase MovieClip o esté vinculado a un símbolo de la biblioteca.

Disponibilidad: ActionScript 1.0; Flash Player 6

Ejemplo

El ejemplo siguiente define una función para el método `onRollOver()` que envía una acción `trace()` al panel Salida:

```
my_mc.onRollOver = function () {
  trace ("onRollOver called");
}
```

onSetFocus (controlador MovieClip.onSetFocus)

```
onSetFocus = function(oldFocus:Object) {}
```

Se invoca cuando un clip de película se selecciona con el teclado. El parámetro `oldFocus` es el objeto que deja de estar seleccionado. Por ejemplo, si el usuario presiona la tecla Tabulador para desplazar la selección del teclado de un clip de película a un campo de texto, `oldFocus` contendrá la instancia del clip de película.

Si anteriormente no había ningún objeto seleccionado con el teclado, `oldFocus` contendrá un valor null (nulo).

Debe definir la función que ha de ejecutarse cuando se invoca el controlador de eventos. Puede definir la función en la línea de tiempo o en un archivo de clase que amplíe la clase MovieClip o esté vinculado a un símbolo de la biblioteca.

Disponibilidad: ActionScript 1.0; Flash Player 6

Parámetros

`oldFocus``:Object` - El objeto que deja de estar seleccionado.

Ejemplo

El ejemplo siguiente muestra información sobre el clip de película que se selecciona con el teclado y la instancia que estaba seleccionada anteriormente. En el escenario hay dos clips de película denominados `my_mc` y `other_mc`. Añada el siguiente código ActionScript al documento FLA o AS:

```
my_mc.onRelease = Void;
other_mc.onRelease = Void;
my_mc.onSetFocus = function(oldFocus) {
  trace("onSetFocus called, previous focus was: "+oldFocus);
}
```

Pulse la tecla Tabulador entre las dos instancias y aparecerá información en el panel Salida.

Véase también

`onKillFocus (controlador MovieClip.onKillFocus)`

onUnload (controlador MovieClip.onUnload)

```
onUnload = function() {}
```

Se invoca en el primer fotograma después de que el clip de película se elimine de la línea de tiempo. Flash procesa las acciones asociadas al controlador de eventos `onUnload` antes de asociar acciones al fotograma afectado. Debe definir la función que ha de ejecutarse cuando se invoca el controlador de eventos. Puede definir la función en la línea de tiempo o en un archivo de clase que amplíe la clase MovieClip o esté vinculado a un símbolo de la biblioteca.

Disponibilidad: ActionScript 1.0; Flash Player 6

Ejemplo

El ejemplo siguiente define una función para el método `MovieClip.onUnload()` que envía una acción `trace()` al panel Salida:

```
my_mc.onUnload = function () {
  trace ("onUnload called");
}
```

opaqueBackground (propiedad MovieClip.opaqueBackground)

```
public opaqueBackground : Number
```

Color del fondo opaco (no transparente) del clip de película del color especificado por el número (valor RGB hexadecimal). Si tiene el valor `null` o `undefined`, no hay fondo opaco. En clips de película donde la propiedad `cacheAsBitmap` está configurada en `true`, la definición de `opaqueBackground` puede mejorar el rendimiento de representación.

El aumento de rendimiento es palpable con clips de película que, de no estar definido `opaqueBackground`, tendrían muchas regiones transparentes.

*Nota:*La región de fondo opaco no halla correspondencias en un método `hitTest()` con el parámetro `shapeFlag` definido en `true`.

Disponibilidad: ActionScript 1.0; Flash Player 8

Ejemplo

En el ejemplo siguiente se crea un contorno triangular y la propiedad `opaqueBackground` se establece en un color específico.

```
var triangle:MovieClip = this.createEmptyMovieClip("triangle",
  this.getNextHighestDepth());
triangle._x = triangle._y = 50;
triangle.lineStyle(3, 0xFFCC00);
triangle.lineTo(0, 30);
triangle.lineTo(50, 0);
triangle.lineTo(0, 0);
triangle.endFill();
triangle.opaqueBackground = 0xCCCCCC;
```

Si el archivo SWF incluye un componente de la versión 2, utilice la clase DepthManager de los componentes de la versión 2 en lugar del método `MovieClip.getNextHighestDepth()` utilizado en este ejemplo.

Véase también

`cacheAsBitmap (propiedad MovieClip.cacheAsBitmap)`, `hitTest (método MovieClip.hitTest)`

_parent (propiedad MovieClip._parent)

```
public _parent : MovieClip
```

Referencia al clip de película u objeto que contiene el clip de película u objeto actual. El parámetro `_parent` es la referencia del objeto actual. Utilice la propiedad `_parent` para especificar una ruta de acceso relativa a los clips de película u objetos que se encuentran por encima del clip de película u objeto actual.

Puede utilizar `_parent` para subir múltiples niveles en la lista de visualización, como se muestra en el código siguiente:

```
this._parent._parent._alpha = 20;
```

Disponibilidad: ActionScript 1.0; Flash Player 5

Ejemplo

El ejemplo siguiente obtiene la referencia a un clip de película y su relación con la línea de tiempo de nivel superior. Cree un clip de película con el nombre de instancia `my_mc` y añádalo a la línea de tiempo principal. Añada el siguiente código ActionScript al archivo FLA o AS:

```
my_mc.onRelease = function() {
  trace("You clicked the movie clip: "+this);
  trace("The parent of "+this._name+" is: "+this._parent);
}
```

Al hacer clic en el clip de película, aparecerá la siguiente información en el panel Salida:

```
You clicked the movie clip: _level0.my_mc
The parent of my_mc is: _level0
```

Véase también

`_parent (propiedad Button._parent)`, `Propiedad _root`, `Función targetPath`, `_parent (propiedad TextField._parent)`

play (método MovieClip.play)

```
public play() : Void
```

Mueve la cabeza lectora por la línea de tiempo del clip de película.

Puede ampliar los métodos y los controladores de eventos de la clase MovieClip creando una subclase.

Disponibilidad: ActionScript 1.0; Flash Player 5

Ejemplo

Utilice el siguiente código ActionScript para reproducir la línea de tiempo principal de un archivo SWF. Este código ActionScript es para un botón de clip de película denominado `my_mc` de la línea de tiempo principal:

```
stop();
my_mc.onRelease = function() {
  this._parent.play();
};
```

Utilice el siguiente código ActionScript para reproducir la línea de tiempo de un clip de película en un archivo SWF. Este código ActionScript es para un botón denominado `my_btn` de la línea de tiempo principal que reproduce un clip de película llamado `animation_mc`:

```
animation_mc.stop();
my_btn.onRelease = function(){
  animation_mc.play();
};
```

Véase también

`Función play`, `gotoAndPlay (método MovieClip.gotoAndPlay)`, `Función gotoAndPlay`

prevFrame (método MovieClip.prevFrame)

```
public prevFrame() : Void
```

Traslada la cabeza lectora al fotograma anterior y la detiene en dicho punto.

Puede ampliar los métodos y los controladores de eventos de la clase MovieClip creando una subclase.

Disponibilidad: ActionScript 1.0; Flash Player 5

Ejemplo

En el ejemplo siguiente, dos botones de clip de película controlan la línea de tiempo. El botón `prev_mc` desplaza la cabeza lectora al fotograma anterior, mientras que el botón `next_mc` la desplaza al fotograma siguiente. Añada contenido a una serie de fotogramas de la línea de tiempo y añada el siguiente código ActionScript al fotograma 1 de la línea de tiempo:

```
stop();
prev_mc.onRelease = function() {
  var parent_mc:MovieClip = this._parent;
  if (parent_mc._currentframe>1) {
  parent_mc.prevFrame();
  } else {
  parent_mc.gotoAndStop(parent_mc._totalframes);
  }
};
```

```
next_mc.onRelease = function() {
  var parent_mc:MovieClip = this._parent;
  if (parent_mc._currentframe<parent_mc._totalframes) {
  parent_mc.nextFrame();
  } else {
  parent_mc.gotoAndStop(1);
  }
};
```

Véase también

Función prevFrame

_quality (propiedad MovieClip._quality)

```
public _quality : String
```

Establece o recupera la calidad de representación que se utiliza para un archivo SWF. Las fuentes de dispositivo siempre se muestran dentadas y, por consiguiente, no se ven afectadas por la propiedad `_quality`.

Puede establecer `_quality` con los valores siguientes:

Valor	Descripción	Suavizado de gráficos	Suavizado de mapa de bits
`"LOW"`	Calidad de representación baja.	Los gráficos no se suavizan.	Los mapas de bits no se suavizan.
`"MEDIUM"`	Calidad de representación media. Esta ajuste resulta adecuado para películas que no contengan texto.	Los gráficos se suavizan empleando una cuadrícula de 2 x 2 píxeles.	Flash Player 8: Los mapas de bits se suavizan en función del parámetro `smoothing` utilizado en las llamadas a `MovieClip.attachBitmap()` y `MovieClip.beginBitmapFill()`. Flash Player 6 y 7: Los mapas de bits no se suavizan.

Valor	Descripción	Suavizado de gráficos	Suavizado de mapa de bits
"HIGH"	Calidad de representación alta. Esta es la calidad de representación predeterminada de Flash.	Los gráficos se suavizan empleando una cuadrícula de 4 x 4 píxeles.	Flash Player 8: Los mapas de bits se suavizan en función del parámetro smoothing utilizado en las llamadas a MovieClip.attachBitmap() y MovieClip.beginBitmapFill(). Flash Player 6 y 7: Los mapas de bits se suavizan si el clip de película es estático.
"BEST"	Calidad de representación muy alta.	Los gráficos se suavizan empleando una cuadrícula de 4 x 4 píxeles.	Flash Player 8: Los mapas de bits se suavizan en función del parámetro smoothing utilizado en las llamadas a MovieClip.attachBitmap() y MovieClip.beginBitmapFill(). Cuando se define smoothing, el resultado se representa con una calidad muy alta cuando el clip de película se reduce, utilizando un algoritmo de valores medios. Esto puede ralentizar la representación, pero puede permitir aplicaciones como miniaturas de alta calidad de imágenes de gran tamaño. Flash Player 6 y 7: Los mapas de bits se suavizan siempre.

Nota: Aunque puede especificar esta propiedad para un objeto MovieClip, se trata en realidad de una propiedad global, por lo que puede especificar su valor simplemente como `_quality`.

Disponibilidad: ActionScript 1.0; Flash Player 6

Ejemplo

Este ejemplo establece en `LOW` la calidad de representación de un clip de película denominado `my_mc`:

```
my_mc._quality = "LOW";
```

Véase también

`Propiedad _quality`

removeMovieClip (método MovieClip.removeMovieClip)

```
public removeMovieClip() : Void
```

Elimina una instancia de clip de película creada por `duplicateMovieClip()`, `MovieClip.duplicateMovieClip()`, `MovieClip.createEmptyMovieClip()` o `MovieClip.attachMovie()`.

Este método no elimina clips de película asignados a valores de profundidad negativos. Los clips de película creados en la herramienta del entorno de edición tienen asignados valores de profundidad negativos de forma predeterminada. Para eliminar un clip de película que tiene asignado un valor negativo de profundidad, en primer lugar utilice `MovieClip.swapDepths()` para desplazar el clip de película a un valor de profundidad positivo.

Nota: Si utiliza componentes de la versión 2, no use este método. Si coloca un componente de la versión 2 en el escenario o la biblioteca, el método `getNextHighestDepth()` puede devolver una profundidad de 1048676, valor que queda fuera del rango válido. Si utiliza componentes de la versión 2, debe usar siempre la clase DepthManager de los componentes de la versión 2.

Nota: Si utiliza componentes de la versión 2 y emplea `MovieClip.getNextHighestDepth()` en lugar de la clase DepthManager para asignar valores de profundidad, quizá `removeMovieClip()` falle sin ninguna indicación. Cuando se utilizan componentes de la versión 2, la clase DepthManager reserva automáticamente las profundidades máxima (1048575) y mínima (-16383) disponibles para cursores y sugerencias. Las siguientes llamadas a `getNextHighestDepth()` devuelven 1048576, que queda fuera del rango válido. El método `removeMovieClip()` falla si encuentra un valor de profundidad fuera del rango válido. Si necesita utilizar `getNextHighestDepth()` con componentes de la versión 2, puede utilizar `swapDepths()` para asignar un valor de profundidad válido o utilizar `MovieClip.unloadMovie()` para eliminar el contenido del clip de película. También puede emplear la clase DepthManager para asignar valores de profundidad pertenecientes al rango válido.

Puede ampliar los métodos y los controladores de eventos de la clase MovieClip creando una subclase.

Disponibilidad: ActionScript 1.0; Flash Player 5

Ejemplo

Cada vez que haga clic en un botón en el ejemplo siguiente, asociará una instancia de clip de película al escenario en una posición aleatoria. Al hacer clic en una instancia de clip de película, eliminará esa instancia del archivo SWF.

```
function randRange(min:Number, max:Number):Number {
  var randNum:Number = Math.round(Math.random()*(max-min))+min;
  return randNum;
}
var bugNum:Number = 0;
addBug_btn.onRelease = addBug;
function addBug() {
  var thisBug:MovieClip = this._parent.attachMovie("bug_id",
  "bug"+bugNum+"_mc", bugNum,
{_x:randRange(50, 500), _y:randRange(50, 350)});
  thisBug.onRelease = function() {
  this.removeMovieClip();
  };
  bugNum++;
}
```

Véase también

Función `duplicateMovieClip`, `createEmptyMovieClip (método MovieClip.createEmptyMovieClip)`, `duplicateMovieClip (método MovieClip.duplicateMovieClip)`, `attachMovie (método MovieClip.attachMovie)`, `swapDepths (método MovieClip.swapDepths)`

_rotation (propiedad MovieClip._rotation)

```
public _rotation : Number
```

Especifica el giro del clip de película, expresado en grados, con respecto a su orientación original. Los valores comprendidos entre 0 y 180 representan un giro en el sentido de las agujas del reloj, mientras que los comprendidos entre 0 y -180 representan un giro en sentido contrario al de las agujas del reloj. Los valores situados fuera de este rango se suman o restan de 360 para obtener un valor que sí esté comprendido en el rango, por ejemplo, la sentencia `my_mc._rotation = 450` equivale a `my_mc._rotation = 90`.

Disponibilidad: ActionScript 1.0; Flash Player 4

Ejemplo

El ejemplo siguiente crea dinámicamente un clip de película denominado `triangle`. Cuando ejecute el archivo SWF, haga clic en el clip de película para girarlo.

```
this.createEmptyMovieClip("triangle", this.getNextHighestDepth());

triangle.beginFill(0x0000FF, 100);
triangle.moveTo(100, 100);
triangle.lineTo(100, 150);
triangle.lineTo(150, 100);
triangle.lineTo(100, 100);

triangle.onMouseUp= function() {
  this._rotation += 15;
};
```

El método `MovieClip.getNextHighestDepth()` utilizado en este ejemplo necesita Flash Player 7 o una versión posterior. Si el archivo SWF incluye un componente de la versión 2, utilice la clase DepthManager de los componentes de la versión 2 en lugar del método `MovieClip.getNextHighestDepth()`.

Véase también

`_rotation (propiedad Button._rotation)`, `_rotation (propiedad TextField._rotation)`

scale9Grid (propiedad MovieClip.scale9Grid)

`public scale9Grid : Rectangle`

Región rectangular que define las nueve regiones de escala del clip de película. Si se configura en `null`, el clip de película entero cambia de escala con normalidad cuando se aplica una transformación de escala.

Cuando un clip de película tiene definido`scale9Grid`, se divide en una cuadrícula de nueve regiones en función del rectángulo de `scale9Grid`, que constituye la región central de la cuadrícula. La cuadrícula tiene ocho regiones más:

- El área de la esquina superior izquierda fuera del rectángulo
- El área situada por encima del rectángulo
- El área de la esquina superior derecha fuera del rectángulo
- El área a la izquierda del rectángulo
- El área a la derecha del rectángulo
- El área de la esquina inferior izquierda fuera del rectángulo
- El área situada por debajo del rectángulo
- El área de la esquina inferior derecha fuera del rectángulo

Puede considerar las ocho regiones externas al centro (que define el rectángulo) como si fueran el marco de un cuadro con reglas especiales cuando se escala el clip de película.

Cuando está definida la propiedad `scale9Grid` y se cambia la escala de un clip de película, todo el texto y los clips de película secundarios cambian de escala con normalidad, independientemente de en qué regiones se encuentren con respecto a la cuadrícula scale9; sin embargo, con otro tipo de objetos se aplican las siguientes normas:

- Todo el contenido de la región central cambia de escala con normalidad.
- El contenido de las esquinas sólo cambia de escala cuando la escala de la región central cambia a 0.
- El contenido de las regiones superior e inferior sólo cambia de escala horizontalmente. El contenido de las regiones izquierda y derecha sólo cambia de escala verticalmente.
- Todos los rellenos (incluidos mapas de bits, vídeo y degradados) se expanden para ajustarse a su forma y todos los rellenos (incluidos mapas de bits, vídeo y degradados) se expanden para ajustarse a su forma.

Si se gira un clip de película, todos los cambios de escala siguientes serán normales (y se omitirá la propiedad `scale9Grid`).

Por ejemplo, imagine el siguiente clip de película y un rectángulo que se aplica como la propiedad `scale9Grid` del clip:

 Clip de película.	 El rectángulo rojo muestra la cuadrícula `scale9Grid`.

Cuando el clip de película cambia la escala o se expande, los objetos del interior del rectángulo cambian de escala con normalidad, pero los objetos del exterior cambian de escala según las normas de `scale9Grid`:

Cambio de escala al 75%:	
Cambio de escala al 50%:	
Cambio de escala al 25%:	
Expandido horizontalmente un 150%:	

Normalmente, `scale9Grid` se define cuando se configura un componente en el que las líneas del borde conservan la misma anchura si el componente cambia de escala.

En el entorno de edición de Macromedia Flash, es posible activar las días para la *escala de 9 divisiones* en el caso de un símbolo de clip de película de una biblioteca. Esto permite determinar de forma gráfica el valor de `scale9grid` para el objeto. Cuando se define la escala de 9 divisiones para un símbolo, la propiedad `scale9grid` de cualquier instancia de dicho símbolo se define automáticamente. En el caso de un símbolo que tenga activada la escala de nueve divisiones, al crear el archivo SWF, cualquier curva que se extienda en más de una región de la cuadrícula de escala de 9 divisiones se dividen en curvas independientes en cada región de la cuadrícula. Por ejemplo, imagine una curva en un símbolo de clip de película que tenga activada la escala de 9 divisiones y la misma curva en un símbolo de clip de película en la que dicha escala *no* esté activada:

Símbolo con escala en 9 divisiones activada:	
Símbolo sin escala en 9 divisiones activada:	

Cuando Flash crea el archivo SWF, la curva en el primer clip de película de la ilustración se divide en tres curvas. Este no es el caso del segundo clip de película, para el que no se ha activado la escala de 9 divisiones. Incluso si establece `scale9Grid` para el segundo clip de película que coincida con el valor de `scale9Grid` del primer clip de película, al cambiar la escala de estos clips, el resultado será diferente debido a la forma en que Flash divide las curvas en el primer clip de película:

Símbolo con escala en 9 divisiones activada	
Símbolo sin escala en 9 divisiones activada	

Disponibilidad: ActionScript 1.0; Flash Player 8

Ejemplo

El siguiente ejemplo crea un clip de película que contiene una línea de 20 píxeles (que forma un borde) y un relleno con degradado. El clip de película cambia de escala en función de la posición del ratón y, dado que en el clip de película se ha configurado una cuadrícula `scale9Grid`, el grosor de la línea de 20 píxeles no varía cuando el clip cambia de escala (pero el degradado del clip *sí* cambia de escala):

```
import flash.geom.Rectangle;
import flash.geom.Matrix;

this.createEmptyMovieClip("my_mc", this.getNextHighestDepth());
```

```
var grid:Rectangle = new Rectangle(20, 20, 260, 260);
my_mc.scale9Grid = grid ;

my_mc._x = 50;
my_mc._y = 50;

function onMouseMove()
{
  my_mc._width = _xmouse;
  my_mc._height = _ymouse;
}

my_mc.lineStyle(20, 0xff3333, 100);
var gradient_matrix:Matrix = new Matrix();
gradient_matrix.createGradientBox(15, 15, Math.PI, 10, 10);
my_mc.beginGradientFill("radial", [0xffff00, 0x0000ff],
       [100, 100], [0, 0xFF], gradient_matrix,
       "reflect", "RGB", 0.9);
my_mc.moveTo(0, 0);
my_mc.lineTo(0, 300);
my_mc.lineTo(300, 300);
my_mc.lineTo(300, 0);
my_mc.lineTo(0, 0);
my_mc.endFill();
```

Véase también

Rectangle (flash.geom.Rectangle)

scrollRect (propiedad MovieClip.scrollRect)

```
public scrollRect : Object
```

La propiedad `scrollRect` permite desplazar con rapidez el contenido del clip de película y abrir una ventana que muestre mayor cantidad de contenido. Los campos de texto y el contenido complejo se desplazan con mayor rapidez porque, para recorrer los datos, se utiliza la copia de píxeles en lugar de regenerar todo el clip de película a partir de datos vectoriales. Para ver la ganancia del rendimiento, utilice `scrollRect` junto con un clip de película que tenga `cacheAsBitmap` definido en `true`.

El clip de película se recorta y recorre la pantalla con medidas específicas de anchura, altura y desplazamiento. Las propiedades `scrollRect` se almacenan en el espacio de coordenadas del clip de película y cambian de escala al igual que el resto del clip. Los límites de esquina de la ventana recortada del clip de película en desplazamiento son el origen del clip de película (0, 0) y el punto (`scrollWidth`, `scrollHeight`). Estos puntos no están centrados en torno al origen, pero sí utilizan el origen en la esquina superior izquierda. Un clip de película desplazado siempre se desplaza en incrementos de píxeles completos. Si el clip de película se gira 90 grados y lo desplaza a izquierda y derecha (mediante la propiedad `scrollRect.x`), el clip se desplaza hacia arriba y hacia abajo.

Si se configura como objeto flash.geom.Rectangle, el clip de película se recorta con un tamaño determinado y se desplaza.

Disponibilidad: ActionScript 1.0; Flash Player 8

Ejemplo

El ejemplo siguiente establece una jerarquía de MovieClip (realizando una llamada a la función `setUpContainer()`) y después configura un nuevo rectángulo como propiedad `scrollRect`.

```
import flash.geom.Rectangle;
var container:MovieClip = setUpContainer();
var window:Rectangle = new Rectangle(0, 0, 100, 40);
container.scrollRect = window;

function setUpContainer():MovieClip {
  var mc:MovieClip = this.createEmptyMovieClip("container",
  this.getNextHighestDepth());
  mc._x = 50;
  mc._y = 50;
  mc.opaqueBackground = 0xCCCCCC;

  var content:MovieClip = mc.createEmptyMovieClip("content",
  mc.getNextHighestDepth());
  var colors:Array = [0xFF0000, 0x0000FF];
  var alphas:Array = [100, 100];
  var ratios:Array = [0, 0xFF];
  var matrix:Object = {a:150, b:0, c:0, d:0, e:150, f:0, g:150, h:150,
  i:1};
  content.beginGradientFill("linear", colors, alphas, ratios, matrix);
  content.lineTo(300, 0);
  content.lineTo(300, 300);
  content.lineTo(0, 300);
  content.lineTo(0, 0);
  content.endFill();
  content._rotation = -90;
```

```
  mc.onEnterFrame = function() {
    this.content._y += 1;
  }

  return mc;
}
```

La función `setUpContainer()` sigue estos pasos:

- Crea un objeto MovieClip denominado `container`
- Crea un objeto MovieClip denominado `content` dentro de `container`
- Dibuja una forma con degradado dentro del objeto MovieClip `content`
- Devuelve una referencia al MovieClip `container`

Si el archivo SWF incluye un componente de la versión 2, utilice la clase DepthManager de los componentes de la versión 2 en lugar del método `MovieClip.getNextHighestDepth()` utilizado en este ejemplo.

setMask (método MovieClip.setMask)

```
public setMask(mc:Object) : Void
```

Convierte el clip de película del parámetro `mc` en una máscara que revela el clip de película especificado por el clip de película.

El método `setMask()` permite a los clips de película de múltiples fotogramas con contenido complejo en múltiples capas actuar como máscaras (lo cual es posible utilizando capas de máscara). Si hay fuentes de dispositivo en un clip de película con máscara, éstas se dibujarán pero no se enmascararán. No es posible configurar un clip de película de forma que sea su propia máscara (por ejemplo, `my_mc.setMask(my_mc)`).

Si crea una capa de máscara que contiene un clip de película y luego le aplica el método `setMask()`, la llamada a `setMask()` tiene prioridad y es irreversible. Por ejemplo, puede tener un clip de película en una capa de máscara denominada `UIMask` que enmascare otra capa que contiene otro clip de película denominado `UIMaskee`. Si, durante la reproducción del archivo SWF, efectúa una llamada a `UIMask.setMask(UIMaskee)`, `UIMask` quedará enmascarada por `UIMaskee` a partir de ese punto.

Para cancelar una máscara creada con ActionScript, pase el valor `null` al método `setMask()`. El siguiente código cancela la máscara sin que ello afecte a la capa de máscara en la línea de tiempo.

```
UIMask.setMask(null);
```

Puede ampliar los métodos y los controladores de eventos de la clase MovieClip creando una subclase.

Disponibilidad: ActionScript 1.0; Flash Player 6

Parámetros

mc:Object - Nombre de instancia de un clip de película que se va a enmascarar. Puede ser una cadena o un objeto MovieClip.

Ejemplo

En este ejemplo se utiliza un clip de película `circleMask_mc` para enmascarar al clip de película `theMaskee_mc`.

```
theMaskee_mc.setMask(circleMask_mc);
```

_soundbuftime (propiedad MovieClip._soundbuftime)

```
public _soundbuftime : Number
```

Especifica el número de segundos que un sonido debe acumularse previamente en el búfer antes de que comience a reproducirse sin interrupción.

Nota: Aunque puede especificar esta propiedad para un objeto MovieClip, se trata en realidad de una propiedad global que se aplica a todos los sonidos cargados, por lo que puede especificar su valor simplemente como `_soundbuftime`. Al establecer esta propiedad para un objeto MovieClip, en realidad se define la propiedad global.

Disponibilidad: ActionScript 1.0; Flash Player 6

Véase también

`Propiedad _soundbuftime`

startDrag (método MovieClip.startDrag)

```
public startDrag([lockCenter:Boolean], [left:Number], [top:Number],
  [right:Number], [bottom:Number]) : Void
```

Permite al usuario arrastrar el clip de película especificado. Es posible continuar arrastrando el clip de película hasta que se detenga explícitamente mediante una llamada a `MovieClip.stopDrag()` o hasta que otro clip de película pueda arrastrarse. Sólo es posible arrastrar clips de película de uno en uno.

Puede ampliar los métodos y los controladores de eventos de la clase MovieClip creando una subclase.

Disponibilidad: ActionScript 1.0; Flash Player 5

Parámetros

lockCenter:Boolean [opcional] - Valor booleano que especifica si el clip de película arrastrable está bloqueado en el centro de la posición del ratón (true) o en el punto donde el usuario hizo clic por primera vez en el clip de película (false).

left:Number [opcional] - Valor relativo a las coordenadas del elemento principal del clip de película, que especifican un rectángulo limitado para el clip de película.

top:Number [opcional] - Valor relativo a las coordenadas del elemento principal del clip de película, que especifican un rectángulo limitado para el clip de película.

right:Number [opcional] - Valor relativo a las coordenadas del elemento principal del clip de película, que especifican un rectángulo limitado para el clip de película.

bottom:Number [opcional] - Valor relativo a las coordenadas del elemento principal del clip de película, que especifican un rectángulo limitado para el clip de película.

Ejemplo

El ejemplo siguiente crea una instancia de clip de película que puede arrastrarse denominada mc_1.

```
this.createEmptyMovieClip("mc_1", 1);

with (mc_1) {
  lineStyle(1, 0xCCCCCC);
  beginFill(0x4827CF);
  moveTo(0, 0);
  lineTo(80, 0);
  lineTo(80, 60);
  lineTo(0, 60);
  lineTo(0, 0);
  endFill();
}

mc_1.onPress = function() {
  this.startDrag();
};
mc_1.onRelease = function() {
  this.stopDrag();
};
```

Véase también

_droptarget (propiedad MovieClip._droptarget), Función startDrag, stopDrag (método MovieClip.stopDrag)

stop (método MovieClip.stop)

```
public stop() : Void
```

Detiene el clip de película que se está reproduciendo actualmente.

Puede ampliar los métodos y los controladores de eventos de la clase MovieClip creando una subclase.

Disponibilidad: ActionScript 1.0; Flash Player 5

Ejemplo

El ejemplo siguiente muestra cómo detener un clip de película denominado `aMovieClip`:

```
aMovieClip.stop();
```

Véase también

Función stop

stopDrag (método MovieClip.stopDrag)

```
public stopDrag() : Void
```

Finaliza un método `MovieClip.startDrag()`. Un clip de película que puede arrastrarse mediante dicho método podrá arrastrarse hasta que se añada un método `stopDrag()` o hasta que se pueda arrastrar otro clip de película. Sólo es posible arrastrar clips de película de uno en uno.

Puede ampliar los métodos y los controladores de eventos de la clase MovieClip creando una subclase.

Disponibilidad: ActionScript 1.0; Flash Player 5

Ejemplo

El ejemplo siguiente crea una instancia de clip de película que puede arrastrarse denominada `mc_1`.

```
this.createEmptyMovieClip("mc_1", 1);

with (mc_1) {
  lineStyle(1, 0xCCCCCC);
  beginFill(0x4827CF);
  moveTo(0, 0);
  lineTo(80, 0);
  lineTo(80, 60);
  lineTo(0, 60);
  lineTo(0, 0);
  endFill();
```

```
}

mc_1.onPress = function() {
  this.startDrag();
};
mc_1.onRelease = function() {
  this.stopDrag();
};
```

Véase también

`_droptarget (propiedad MovieClip._droptarget)`, `startDrag (método MovieClip.startDrag)`, `Función stopDrag`

swapDepths (método MovieClip.swapDepths)

```
public swapDepths(target:Object) : Void
```

Intercambia el apilamiento o nivel de profundidad (z-order), de este clip de película con el clip de película especificado por el parámetro `target` o con el clip de película que ocupa actualmente el nivel de profundidad especificado en el parámetro `target`. Ambos clips de película deben tener el mismo clip de película principal. El intercambio del nivel de profundidad de los clips de película provoca que un clip de película se sitúe delante o detrás del otro. Si se está interpolando un clip de película cuando se llama a este método, la interpolación se detendrá.

Puede ampliar los métodos y los controladores de eventos de la clase MovieClip creando una subclase.

Disponibilidad: ActionScript 1.0; Flash Player 5

Parámetros

`target``:Object` - Este parámetro puede tener una de las dos formas siguientes:

- Un entero exclusivo que especifica la profundidad a la que debe colocarse el nuevo clip de película.
- Una cadena que especifica la instancia de clip de película cuya profundidad se intercambiará con el clip de película al que se aplicará el método. Ambos clips de película deben tener el mismo clip de película principal.

Ejemplo

El ejemplo siguiente intercambia el orden de apilamiento de dos instancias de clip de película. Superponga en el escenario dos instancias de clip de película denominadas `myMC1_mc` y `myMC2_mc` y después añada el siguiente código ActionScript a la línea de tiempo principal:

```
myMC1_mc.onRelease = function() {
  this.swapDepths(myMC2_mc);
};
myMC2_mc.onRelease = function() {
  this.swapDepths(myMC1_mc);
};
```

Véase también

`Propiedad _level`, `getDepth (método MovieClip.getDepth)`, `getInstanceAtDepth (método MovieClip.getInstanceAtDepth)`, `getNextHighestDepth (método MovieClip.getNextHighestDepth)`

tabChildren (propiedad MovieClip.tabChildren)

```
public tabChildren : Boolean
```

Determina si los elementos secundarios de un clip de película se incluyen en el orden de tabulación automático. Si la propiedad `tabChildren` tiene el valor `undefined` o `true`, los elementos secundarios de un clip de película se incluirán en el orden de tabulación automático. Si el valor de `tabChildren` es `false`, los elementos secundarios de un clip de película no se incluirán en el orden de tabulación automático. El valor predeterminado es `undefined`.

Disponibilidad: ActionScript 1.0; Flash Player 6

Ejemplo

El artículo cuadro de lista de la interfaz de usuario creado como un clip de película contiene varios elementos. El usuario puede hacer clic en cada elemento para seleccionarlo, por lo que cada elemento aparece como un botón. Sin embargo, sólo el cuadro de lista propiamente dicho debe ser una tabulación. Los elementos del interior del cuadro de lista deben quedar excluidos del orden de tabulación. Para ello, la propiedad `tabChildren` del cuadro de lista debe configurarse como `false`.

La propiedad `tabChildren` no tiene ningún efecto si se utiliza la propiedad `tabIndex`; la propiedad `tabChildren` sólo afecta al orden de tabulación automático.

El ejemplo siguiente desactiva la tabulación de todos los clips de película secundarios del interior de un clip de película principal denominado `menu_mc`:

```
menu_mc.onRelease = function(){};
menu_mc.menu1_mc.onRelease = function(){};
menu_mc.menu2_mc.onRelease = function(){};
menu_mc.menu3_mc.onRelease = function(){};
menu_mc.menu4_mc.onRelease = function(){};

menu_mc.tabChildren = false;
```

Cambie la última línea de código por lo siguiente para incluir las instancias de clip de película secundarias de `menu_mc` en el orden de tabulación automático:

```
menu_mc.tabChildren = true;
```

Véase también

`tabIndex (propiedad Button.tabIndex)`, `tabEnabled (propiedad MovieClip.tabEnabled)`, `tabIndex (propiedad MovieClip.tabIndex)`, `tabIndex (propiedad TextField.tabIndex)`

tabEnabled (propiedad MovieClip.tabEnabled)

```
public tabEnabled : Boolean
```

Especifica si el clip de película se incluye en el orden de tabulación automático. De manera predeterminada, tiene el valor `undefined`.

Si la propiedad `tabEnabled` es `undefined`, el objeto sólo se incluirá en el orden de tabulación automático si define al menos un controlador de clip de película, como `MovieClip.onRelease`. Si `tabEnabled` se establece con el valor `true`, el objeto se incluirá en el orden de tabulación automático. Si la propiedad `tabIndex` también está configurada con un valor, el objeto se incluirá también en el orden de tabulación personalizado.

Si `tabEnabled` tiene el valor `false`, el objeto no se incluirá en el orden de tabulación automático ni en el personalizado aunque se establezca la propiedad `tabIndex`. No obstante, si `MovieClip.tabChildren` tiene el valor `true`, los elementos secundarios del clip de película podrán incluirse en el orden de tabulación automático aunque `tabEnabled` tenga el valor `false`.

Disponibilidad: ActionScript 1.0; Flash Player 6

Ejemplo

El ejemplo siguiente no incluye `myMC2_mc` en el orden de tabulación automático:

```
myMC1_mc.onRelease = function() {};
myMC2_mc.onRelease = function() {};
myMC3_mc.onRelease = function() {};
myMC2_mc.tabEnabled = false;
```

Véase también

`onRelease (controlador MovieClip.onRelease)`, `tabEnabled (propiedad Button.tabEnabled)`, `tabChildren (propiedad MovieClip.tabChildren)`, `tabIndex (propiedad MovieClip.tabIndex)`, `tabEnabled (propiedad TextField.tabEnabled)`

tabIndex (propiedad MovieClip.tabIndex)

```
public tabIndex : Number
```

Permite personalizar el orden de tabulación de los objetos de una película. La propiedad `tabIndex` tiene el valor `undefined` de manera predeterminada. Puede establecer `tabIndex` en una instancia de botón, clip de película o campo de texto.

Si un objeto de un archivo SWF contiene una propiedad `tabIndex`, se desactivará el orden de tabulación automático y el orden de tabulación se calculará a partir de las propiedades `tabIndex` de los objetos del archivo SWF. El orden de tabulación personalizado sólo incluye objetos que tienen propiedades `tabIndex`.

La propiedad `tabIndex` debe ser un entero positivo. Los objetos se ordenan conforme a lo que indiquen las propiedades `tabIndex` y en orden ascendente. Un objeto cuyo valor de `tabIndex` sea `1` precederá a un objeto cuyo valor de `tabIndex` sea `2`. El orden de tabulación personalizado no tiene en cuenta las relaciones jerárquicas de los objetos de un archivo SWF. Todos los objetos del archivo SWF que tengan propiedades `tabIndex` se incluirán en el orden de tabulación. No utilice el mismo valor de `tabIndex` para varios objetos.

Disponibilidad: ActionScript 1.0; Flash Player 6

Ejemplo

El siguiente código ActionScript establece un orden de tabulación personalizado para tres instancias de clip de película.

```
myMC1_mc.onRelease = function() {};
myMC2_mc.onRelease = function() {};
myMC3_mc.onRelease = function() {};
myMC1_mc.tabIndex = 2;
myMC2_mc.tabIndex = 1;
myMC3_mc.tabIndex = 3;
```

Véase también

tabIndex (propiedad Button.tabIndex), tabIndex (propiedad TextField.tabIndex)

_target (propiedad MovieClip._target)

```
public _target : String [read-only]
```

Devuelve la ruta de destino de la instancia de clip de película en notación con barras. Utilice la función `eval()` para convertir la ruta de acceso de destino a notación con punto.

Disponibilidad: ActionScript 1.0; Flash Player 4

Ejemplo

El ejemplo siguiente muestra las rutas de destino de instancias de clip de película de un archivo SWF en notación con barras y con punto.

```
for (var i in this) {
  if (typeof (this[i]) == "movieclip") {
  trace("name: " + this[i]._name + ",\t target: " + this[i]._target + ",\t
  target(2):"
       + eval(this[i]._target));
  }
}
```

_totalframes (propiedad MovieClip._totalframes)

```
public _totalframes : Number [read-only]
```

Devuelve el número total de fotogramas de la instancia de clip de película especificada en el parámetro `MovieClip`.

Disponibilidad: ActionScript 1.0; Flash Player 4

Ejemplo

En el ejemplo siguiente, dos botones de clip de película controlan la línea de tiempo. El botón `prev_mc` desplaza la cabeza lectora al fotograma anterior, mientras que el botón `next_mc` la desplaza al fotograma siguiente. Añada contenido a una serie de fotogramas de la línea de tiempo y añada el siguiente código ActionScript al fotograma 1 de la línea de tiempo:

```
stop();
prev_mc.onRelease = function() {
  var parent_mc:MovieClip = this._parent;
  if (parent_mc._currentframe>1) {
  parent_mc.prevFrame();
  } else {
  parent_mc.gotoAndStop(parent_mc._totalframes);
```

```
  }
};
next_mc.onRelease = function() {
  var parent_mc:MovieClip = this._parent;
  if (parent_mc._currentframe<parent_mc._totalframes) {
  parent_mc.nextFrame();
  } else {
  parent_mc.gotoAndStop(1);
  }
};
```

trackAsMenu (propiedad MovieClip.trackAsMenu)

```
public trackAsMenu : Boolean
```

Valor booleano que indica si otros botones o clips de película pueden recibir eventos de liberación del botón del ratón. La propiedad `trackAsMenu` permite crear menús. Puede establecer la propiedad `trackAsMenu` en cualquier botón u objeto del clip de película. Si la propiedad `trackAsMenu` no existe, el comportamiento predeterminado será `false`.

Puede cambiar la propiedad `trackAsMenu` en cualquier momento; el clip de película modificado adoptará de inmediato el nuevo comportamiento.

Disponibilidad: ActionScript 1.0; Flash Player 6

Ejemplo

El ejemplo siguiente define la propiedad `trackAsMenu` para tres clips de película del escenario. Haga clic en un clip de película y suelte el botón del ratón sobre un segundo clip de película para ver qué instancia recibe el evento.

```
myMC1_mc.trackAsMenu = true;
myMC2_mc.trackAsMenu = true;
myMC3_mc.trackAsMenu = false;

myMC1_mc.onRelease = clickMC;
myMC2_mc.onRelease = clickMC;
myMC3_mc.onRelease = clickMC;

function clickMC() {
  trace("you clicked the "+this._name+" movie clip.");
};
```

Véase también

`trackAsMenu (propiedad Button.trackAsMenu)`

transform (propiedad MovieClip.transform)

```
public transform : Transform
```

Objeto con propiedades pertenecientes a una matriz, transformación de color y límites de píxel de un clip de película. Las propiedades específicas --matrix, colorTransform y tres propiedades de sólo lectura (`concatenatedMatrix`, `concatenatedColorTransform` y `pixelBounds`)-- se describen en la entrada de la clase Transform.

Cada una de las propiedades de transformación de objetos en un objeto en sí. Este aspecto es importante porque el único modo de definir nuevos valores para objetos matrix o colorTransform es crear un objeto y copiarlo en la propiedad transform.matrix o transform.colorTransform.

Por ejemplo, para aumentar el valor `tx` de la matriz de un clip de película, debe realizar una copia del objeto matrix completo, modificar la propiedad `tx` del nuevo objeto y después copiar el objeto nuevo en la propiedad matrix del objeto transform:

```
var myMatrix:Object = myDisplayObject.transform.matrix;
myMatrix.tx += 10;
myDisplayObject.transform.matrix = myMatrix;
```

La propiedad `tx` no puede configurarse directamente. El código siguiente no tiene efecto en `myDisplayObject`: `myDisplayObject.transform.matrix.tx += 10;`

También puede copiar un objeto transform completo y asignarlo a la propiedad transform de otro clip de película. Por ejemplo, el código siguiente copia el objeto transform completo `myOldDisplayObj` en `myNewDisplayObj`:

```
myNewDisplayObj.transform = myOldDisplayObj.transform;
```

El nuevo clip de película, `myNewDisplayObj`, tiene ahora los mismos valores de matriz, transformación de color y límites de píxel que el antiguo clip de película, `myOldDisplayObj`.

Disponibilidad: ActionScript 1.0; Flash Player 8

Ejemplo

El ejemplo siguiente muestra cómo utilizar la propiedad `transform` del clip de película para acceder y modificar la posición del clip utilizando la posición de Matrix.

```
import flash.geom.Matrix;

var rect:MovieClip = createRectangle(20, 80, 0xFF0000);

var translateMatrix:Matrix = new Matrix();
translateMatrix.translate(10, 0);

rect.onPress = function() {
  var tmpMatrix:Matrix = this.transform.matrix;
  tmpMatrix.concat(translateMatrix);
```

```
  this.transform.matrix = tmpMatrix;
}

function createRectangle(width:Number, height:Number, color:Number,
  scope:MovieClip):MovieClip {
  scope = (scope == undefined) ? this : scope;
  var depth:Number = scope.getNextHighestDepth();
  var mc:MovieClip = scope.createEmptyMovieClip("mc_" + depth, depth);
  mc.beginFill(color);
  mc.lineTo(0, height);
  mc.lineTo(width, height);
  mc.lineTo(width, 0);
  mc.lineTo(0, 0);
  return mc;
}
```

Si el archivo SWF incluye un componente de la versión 2, utilice la clase DepthManager de los componentes de la versión 2 en lugar del método `MovieClip.getNextHighestDepth()` utilizado en este ejemplo.

Véase también

`Transform (flash.geom.Transform)`

unloadMovie (método MovieClip.unloadMovie)

```
public unloadMovie() : Void
```

Elimina el contenido de una instancia de clip de película. Se conservan las propiedades de instancia y los controladores de clip.

Para eliminar la instancia, incluidas sus propiedades y controladores de clip, utilice `MovieClip.removeMovieClip()`.

Puede ampliar los métodos y los controladores de eventos de la clase MovieClip creando una subclase.

Disponibilidad: ActionScript 1.0; Flash Player 5

Ejemplo

El ejemplo siguiente descarga una instancia de clip de película denominada `box` cuando un usuario hace clic en el clip de película `box`.

```
this.createEmptyMovieClip("box", 1);

with (box) {
  lineStyle(1, 0xCCCCCC);
  beginFill(0x4827CF);
  moveTo(0, 0);
```

```
  lineTo(80, 0);
  lineTo(80, 60);
  lineTo(0, 60);
  lineTo(0, 0);
  endFill();
}

box.onRelease = function() {
    box.unloadMovie();
};
```

Véase también

removeMovieClip (método MovieClip.removeMovieClip), attachMovie (método MovieClip.attachMovie), loadMovie (método MovieClip.loadMovie), Función unloadMovie, Función unloadMovieNum

_url (propiedad MovieClip._url)

```
public _url : String [read-only]
```

Recupera la URL del archivo SWF, JPEG, GIF o PNG del que se descargó el clip de película.

Disponibilidad: ActionScript 1.0; Flash Player 4

Ejemplo

El ejemplo siguiente muestra la URL de la imagen que se ha cargado en la instancia `image_mc` en el panel Salida.

```
this.createEmptyMovieClip("image_mc", 1);
var mclListener:Object = new Object();
mclListener.onLoadInit = function(target_mc:MovieClip) {
  trace("_url: "+target_mc._url);
};
var image_mcl:MovieClipLoader = new MovieClipLoader();
image_mcl.addListener(mclListener);
image_mcl.loadClip("http://www.macromedia.com/images/shared/product_boxes/
  112x112/box_studio_112x112.jpg", image_mc);
```

El ejemplo siguiente asigna el objeto ContextMenu `menu_cm` al clip de película `image_mc`. El objeto `menu_cm` contiene un elemento de menú personalizado con la etiqueta `View Image in Browser`que lleva asociada una función denominada `viewImage()`.

```
var menu_cm:ContextMenu = new ContextMenu();
menu_cm.customItems.push(new ContextMenuItem("View Image in Browser...",
  viewImage));
this.createEmptyMovieClip("image_mc", this.getNextHighestDepth());
var mclListener:Object = new Object();
mclListener.onLoadInit = function(target_mc:MovieClip) {
```

```
  target_mc.menu = menu_cm;
};
var image_mcl:MovieClipLoader = new MovieClipLoader();
image_mcl.addListener(mclListener);
image_mcl.loadClip("photo1.jpg", image_mc);

function viewImage(target_mc:MovieClip, obj:Object) {
  getURL(target_mc._url, "_blank");
}
```

Tras hacer clic en la imagen con el botón derecho del ratón (Windows) o con la tecla Control presionada (Macintosh) en tiempo de ejecución, seleccione View Image in Browser en el menú contextual para abrir una imagen en una ventana del navegador.

La clase MovieClipLoader utilizada en estos ejemplos necesita Flash Player 7 o una versión posterior. El método `MovieClip.getNextHighestDepth()` utilizado en estos ejemplos necesita Flash Player 7 o una versión posterior. Si el archivo SWF incluye un componente de la versión 2, utilice la clase DepthManager de los componentes de la versión 2 en lugar del método `MovieClip.getNextHighestDepth()`.

useHandCursor (propiedad MovieClip.useHandCursor)

```
public useHandCursor : Boolean
```

Valor booleano que indica si debe aparecer el puntero de mano (cursor de mano) cuando el ratón pasa por encima de un clip de película. El valor predeterminado de la propiedad `useHandCursor` es `true`. Si `useHandCursor` se establece con el valor `true`, la mano que señala empleada para los botones se mostrará cuando el ratón pase por encima de un clip de película de botón. Si `useHandCursor` se establece con el valor `false`, se utilizará el puntero de flecha.

Puede cambiar la propiedad `useHandCursor` en cualquier momento; el clip de película modificado adoptará de inmediato el nuevo comportamiento de cursor. La propiedad `useHandCursor` puede leerse de un objeto prototipo.

Disponibilidad: ActionScript 1.0; Flash Player 6

Ejemplo

El ejemplo siguiente establece la propiedad `useHandCursor` en dos clips de película denominados `myMC1_mc` y `myMC2_mc`. La propiedad se configura como `true` para una instancia y como `false` para la otra. Observe que ambas instancias pueden seguir recibiendo eventos.

```
myMC1_mc.onRelease = traceMC;
myMC2_mc.onRelease = traceMC;
myMC2_mc.useHandCursor = false;
```

```
function traceMC() {
  trace("you clicked: "+this._name);
};
```

_visible (propiedad MovieClip._visible)

```
public _visible : Boolean
```

Valor booleano que indica si un clip de película está visible. Los clips de película no visibles (que tienen la propiedad `_visible` configurada como `false`) se desactivan. Por ejemplo, no es posible hacer clic en un botón de un clip de película con `_visible` configurado con el valor `false`.

Disponibilidad: ActionScript 1.0; Flash Player 4

Ejemplo

El ejemplo siguiente establece la propiedad `_visible` en dos clips de película denominados `myMC1_mc` y `myMC2_mc`. La propiedad se configura como `true` para una instancia y como `false` para la otra. Observe que no es posible hacer clic en la instancia `myMC1_mc` tras definir la propiedad `_visible` en `false`.

```
myMC1_mc.onRelease = function() {
  trace(this._name+"._visible = false");
  this._visible = false;
};
myMC2_mc.onRelease = function() {
  trace(this._name+"._alpha = 0");
  this._alpha = 0;
};
```

Véase también

`_visible (propiedad Button._visible)`, `_visible (propiedad TextField._visible)`

_width (propiedad MovieClip._width)

```
public _width : Number
```

Anchura del clip de película, expresada en píxeles.

Disponibilidad: ActionScript 1.0; Flash Player 4

Ejemplo

El siguiente ejemplo de código muestra la altura y anchura de un clip de película en el panel Salida:

```
this.createEmptyMovieClip("triangle", this.getNextHighestDepth());

triangle.beginFill(0x0000FF, 100);
triangle.moveTo(100, 100);
triangle.lineTo(100, 150);
triangle.lineTo(150, 100);
triangle.lineTo(100, 100);

trace(triangle._name + " = " + triangle._width + " X " + triangle._height +
  " pixels");
```

El método `MovieClip.getNextHighestDepth()` utilizado en este ejemplo necesita Flash Player 7 o una versión posterior. Si el archivo SWF incluye un componente de la versión 2, utilice la clase DepthManager de los componentes de la versión 2 en lugar del método `MovieClip.getNextHighestDepth()`.

Véase también

`_height (propiedad MovieClip._height)`

_x (propiedad MovieClip._x)

```
public _x : Number
```

Entero que establece la coordenada *x* de un clip de película con respecto a las coordenadas locales del clip de película principal. Si un clip de película se encuentra en la línea de tiempo principal, su sistema de coordenadas hará referencia a la esquina superior izquierda del escenario como (0, 0). Si el clip de película está dentro de otro clip de película que incluye transformaciones, el clip de película estará en el sistema de coordenadas local del clip de película en el que está contenido. Por consiguiente, en el caso de un clip de película con un giro de 90° en sentido contrario al de las agujas del reloj, los clips de película secundarios de este clip de película heredarán un sistema de coordenadas con un giro de 90° en sentido contrario al de las agujas del reloj. Las coordenadas del clip de película hacen referencia a la posición del punto de registro.

Disponibilidad: ActionScript 1.0; Flash Player 3

Ejemplo

El ejemplo siguiente asocia un clip de película con el identificador de vinculación `cursor_id` a un archivo SWF. El clip de película se denomina `cursor_mc`, y se utiliza para sustituir el puntero predeterminado del ratón. El siguiente código ActionScript establece las coordenadas actuales de la instancia de clip de película en la posición del puntero del ratón:

```
this.attachMovie("cursor_id", "cursor_mc", this.getNextHighestDepth(),
  {_x:_xmouse, _y:_ymouse});
Mouse.hide();
var mouseListener:Object = new Object();
mouseListener.onMouseMove = function() {
  cursor_mc._x = _xmouse;
  cursor_mc._y = _ymouse;
  updateAfterEvent();
};
Mouse.addListener(mouseListener);
```

El método `MovieClip.getNextHighestDepth()` utilizado en este ejemplo necesita Flash Player 7 o una versión posterior. Si el archivo SWF incluye un componente de la versión 2, utilice la clase DepthManager de los componentes de la versión 2 en lugar del método `MovieClip.getNextHighestDepth()`.

Véase también

`_xscale (propiedad MovieClip._xscale)`, `_y (propiedad MovieClip._y)`, `_yscale (propiedad MovieClip._yscale)`

_xmouse (propiedad MovieClip._xmouse)

```
public _xmouse : Number [read-only]
```

Devuelve la coordenada *x* de la posición del ratón.

Disponibilidad: ActionScript 1.0; Flash Player 5

Ejemplo

El ejemplo siguiente devuelve las coordenadas *x* e *y* actuales del ratón en el escenario (`_level0`) y con respecto a un clip de película del escenario llamado `my_mc`.

```
this.createTextField("mouse_txt", this.getNextHighestDepth(), 0, 0, 150,
  66);
mouse_txt.html = true;
mouse_txt.multiline = true;
var row1_str:String = " \t<b>_xmouse\t</b><b>_ymouse</b>";
my_mc.onMouseMove = function() {
  mouse_txt.htmlText = "<textformat tabStops='[50,100]'>";
  mouse_txt.htmlText += row1_str;
  mouse_txt.htmlText += "<b>_level0</b>\t"+_xmouse+"\t"+_ymouse;
```

```
  mouse_txt.htmlText += "<b>my_mc</b>\t"+this._xmouse+"\t"+this._ymouse;
  mouse_txt.htmlText += "</textformat>";
};
```

El método `MovieClip.getNextHighestDepth()` utilizado en este ejemplo necesita Flash Player 7 o una versión posterior. Si el archivo SWF incluye un componente de la versión 2, utilice la clase DepthManager de los componentes de la versión 2 en lugar del método `MovieClip.getNextHighestDepth()`.

Véase también

`Mouse`, `_ymouse (propiedad MovieClip._ymouse)`

_xscale (propiedad MovieClip._xscale)

```
public _xscale : Number
```

Determina la escala horizontal (*percentage*) del clip de película aplicada desde el punto de registro del clip de película. El punto de registro predeterminado es (0,0).

La escala del sistema de coordenadas local afecta a la configuración de las propiedades `_x` e `_y`, que se definen en píxeles completos. Por ejemplo, si se aplica una escala del 50% al clip de película principal, la configuración de la propiedad `_x` desplazará un objeto situado en el clip de película la mitad de píxeles que si la película tuviera una escala del 100%.

Disponibilidad: ActionScript 1.0; Flash Player 4

Ejemplo

El ejemplo siguiente crea un clip de película denominado `box_mc` en tiempo de ejecución. La API de dibujo se utiliza para dibujar un recuadro en esta instancia y, cuando el ratón se desplaza sobre el recuadro, se aplica un cambio de escala horizontal y vertical al clip de película. Cuando el ratón se desplaza fuera de la instancia, ésta recupera la escala anterior.

```
this.createEmptyMovieClip("box_mc", 1);
box_mc._x = 100;
box_mc._y = 100;
with (box_mc) {
  lineStyle(1, 0xCCCCCC);
  beginFill(0xEEEEEE);
  moveTo(0, 0);
  lineTo(80, 0);
  lineTo(80, 60);
  lineTo(0, 60);
  lineTo(0, 0);
  endFill();
};
box_mc.onRollOver = function() {
  this._x -= this._width/2;
```

```
  this._y -= this._height/2;
  this._xscale = 200;
  this._yscale = 200;
};
box_mc.onRollOut = function() {
  this._xscale = 100;
  this._yscale = 100;
  this._x += this._width/2;
  this._y += this._height/2;
};
```

Véase también

_width (propiedad MovieClip._width), _x (propiedad MovieClip._x), _y (propiedad MovieClip._y), _yscale (propiedad MovieClip._yscale)

_y (propiedad MovieClip._y)

```
public _y : Number
```

Establece la coordenada *y* de un clip de película con respecto a las coordenadas locales del clip de película principal. Si un clip de película se encuentra en la línea de tiempo principal, su sistema de coordenadas hará referencia a la esquina superior izquierda del escenario como (0,0). Si el clip de película está dentro de otro clip de película que incluye transformaciones, el clip de película estará en el sistema de coordenadas local del clip de película en el que está contenido. Por consiguiente, en el caso de un clip de película con un giro de 90° en sentido contrario al de las agujas del reloj, los clips de película secundarios de este clip de película heredarán un sistema de coordenadas con un giro de 90° en sentido contrario al de las agujas del reloj. Las coordenadas del clip de película hacen referencia a la posición del punto de registro.

Disponibilidad: ActionScript 1.0; Flash Player 3

Ejemplo

El ejemplo siguiente asocia un clip de película con el identificador de vinculación `cursor_id` a un archivo SWF. El clip de película se denomina `cursor_mc` y se utiliza para sustituir el puntero predeterminado del ratón. El siguiente código ActionScript establece las coordenadas actuales de la instancia de clip de película en la posición del puntero del ratón:

```
this.attachMovie("cursor_id", "cursor_mc", this.getNextHighestDepth(),
  {_x:_xmouse, _y:_ymouse});
Mouse.hide();
var mouseListener:Object = new Object();
mouseListener.onMouseMove = function() {
  cursor_mc._x = _xmouse;
  cursor_mc._y = _ymouse;
```

```
  updateAfterEvent();
};
Mouse.addListener(mouseListener);
```

El método `MovieClip.getNextHighestDepth()` utilizado en este ejemplo necesita Flash Player 7 o una versión posterior. Si el archivo SWF incluye un componente de la versión 2, utilice la clase DepthManager de los componentes de la versión 2 en lugar del método `MovieClip.getNextHighestDepth()`.

Véase también

`_x (propiedad MovieClip._x)`, `_xscale (propiedad MovieClip._xscale)`, `_yscale (propiedad MovieClip._yscale)`

_ymouse (propiedad MovieClip._ymouse)

```
public _ymouse : Number [read-only]
```

Indica la coordenada *y* de la posición del ratón.

Disponibilidad: ActionScript 1.0; Flash Player 5

Ejemplo

El ejemplo siguiente devuelve las coordenadas *x* e *y* actuales del ratón en el escenario (`_level0`) y con respecto a un clip de película del escenario llamado `my_mc`.

```
this.createTextField("mouse_txt", this.getNextHighestDepth(), 0, 0, 150,
  66);
mouse_txt.html = true;
mouse_txt.multiline = true;
var row1_str:String = " \t<b>_xmouse\t</b><b>_ymouse</b>";
my_mc.onMouseMove = function() {
  mouse_txt.htmlText = "<textformat tabStops='[50,100]'>";
  mouse_txt.htmlText += row1_str;
  mouse_txt.htmlText += "<b>_level0</b>\t"+_xmouse+"\t"+_ymouse;
  mouse_txt.htmlText += "<b>my_mc</b>\t"+this._xmouse+"\t"+this._ymouse;
  mouse_txt.htmlText += "</textformat>";
};
```

El método `MovieClip.getNextHighestDepth()` utilizado en este ejemplo necesita Flash Player 7 o una versión posterior. Si el archivo SWF incluye un componente de la versión 2, utilice la clase DepthManager de los componentes de la versión 2 en lugar del método `MovieClip.getNextHighestDepth()`.

Véase también

`Mouse`, `_xmouse (propiedad MovieClip._xmouse)`

_yscale (propiedad MovieClip._yscale)

```
public _yscale : Number
```

Establece la escala vertical (*percentage*) del clip de película aplicada desde el punto de registro del clip de película. El punto de registro predeterminado es (0,0).

La escala del sistema de coordenadas local afecta a la configuración de las propiedades _x e _y, que se definen en píxeles completos. Por ejemplo, si se aplica una escala del 50% al clip de película principal, la configuración de la propiedad _x desplazará un objeto situado en el clip de película la mitad de píxeles que si la película tuviera una escala del 100%.

Disponibilidad: ActionScript 1.0; Flash Player 4

Ejemplo

El ejemplo siguiente crea un clip de película denominado box_mc en tiempo de ejecución. La API de dibujo se utiliza para dibujar un recuadro en esta instancia y, cuando el ratón se desplaza sobre el recuadro, se aplica un cambio de escala horizontal y vertical al clip de película. Cuando el ratón se desplaza fuera de la instancia, ésta recupera la escala anterior.

```
this.createEmptyMovieClip("box_mc", 1);
box_mc._x = 100;
box_mc._y = 100;
with (box_mc) {
  lineStyle(1, 0xCCCCCC);
  beginFill(0xEEEEEE);
  moveTo(0, 0);
  lineTo(80, 0);
  lineTo(80, 60);
  lineTo(0, 60);
  lineTo(0, 0);
  endFill();
};
box_mc.onRollOver = function() {
  this._x -= this._width/2;
  this._y -= this._height/2;
  this._xscale = 200;
  this._yscale = 200;
};
box_mc.onRollOut = function() {
  this._xscale = 100;
  this._yscale = 100;
  this._x += this._width/2;
  this._y += this._height/2;
};
```

Véase también

`_height (propiedad MovieClip._height)`, `_x (propiedad MovieClip._x)`, `_xscale (propiedad MovieClip._xscale)`, `_y (propiedad MovieClip._y)`

MovieClipLoader

```
Object
  |
  +-MovieClipLoader
```

`public class` ***MovieClipLoader***
`extends Object`

Esta clase le permite implementar funciones callback de detector que proporcionan información de estado mientras se están cargando archivos SWF, JPEG, GIF o PNG en clips de película. Para utilizar las funciones MovieClipLoader, utilice `MovieClipLoader.loadClip()` en lugar de `loadMovie()` o `MovieClip.loadMovie()` para cargar archivos SWF.

Una vez que envíe el comando `MovieClipLoader.loadClip()`, tendrán lugar los siguientes eventos en el orden en que se enumeran:

- Cuando se hayan grabado en el disco los primeros bytes del archivo descargado, se invocará el detector `MovieClipLoader.onLoadStart`.
- Si ha implementado el detector `MovieClipLoader.onLoadProgress`, éste se invocará durante el proceso de carga.
  Nota: Puede efectuar una llamada a `MovieClipLoader.getProgress()` en cualquier momento durante el proceso de carga.
- Cuando se haya grabado en el disco el archivo descargado completo, se invocará el detector `MovieClipLoader.onLoadComplete`.
- Una vez que se hayan ejecutado las acciones del primer fotograma del archivo descargado, se invocará el detector `MovieClipLoader.onLoadInit`

Una vez que se haya invocado `MovieClipLoader.onLoadInit`, podrá establecer propiedades, utilizar métodos e interactuar con la película descargada.

Si el archivo no se carga completamente, se invocará el detector `MovieClipLoader.onLoadError`.

Disponibilidad: ActionScript 1.0; Flash Player 7

Resumen de propiedades

Propiedades heredadas de la clase Object

constructor (propiedad Object.constructor), __proto__ (propiedad Object.__proto__), prototype (propiedad Object.prototype), __resolve (propiedad Object.__resolve)

Resumen de eventos

Evento	Descripción
`onLoadComplete = function([target_mc:MovieClip], [httpStatus:Number]) {}`	Se invoca cuando un archivo cargado con `MovieClipLoader.loadClip()` se ha descargado completamente.
`onLoadError = function(target_mc:MovieClip, errorCode:String, [httpStatus:Number]) {}`	Se invoca cuando un archivo cargado con `MovieClipLoader.loadClip()` no se carga correctamente.
`onLoadInit = function([target_mc:MovieClip]) {}`	Se invoca cuando se han ejecutado las acciones del primer fotograma del clip cargado.
`onLoadProgress = function([target_mc:MovieClip], loadedBytes:Number, totalBytes:Number) {}`	Se invoca cada vez que se graba en el disco duro el contenido durante el proceso de carga (es decir, entre `MovieClipLoader.onLoadStart` y `MovieClipLoader.onLoadComplete`).
`onLoadStart = function([target_mc:MovieClip]) {}`	Se invoca cuando una llamada a `MovieClipLoader.loadClip()` ha iniciado correctamente la descarga de un archivo.

Resumen de constructores

Firma	Descripción
`MovieClipLoader()`	Crea un objeto MovieClipLoader que puede utilizar para implementar una serie de detectores que respondan a eventos mientras se está descargando un archivo SWF, JPEG, GIF o PNG.

Resumen de métodos

Modificadores	Firma	Descripción
	`addListener(listener:Object) : Boolean`	Registra un objeto para recibir notificación de la invocación de un controlador de eventos `MovieClipLoader`.
	`getProgress(target:Object) : Object`	Devuelve el número de bytes cargados y el número total de bytes de un archivo que se está cargando mediante `MovieClipLoader.loadClip()`; en el caso de películas comprimidas, refleja el número de bytes comprimidos.
	`loadClip(url:String, target:Object) : Boolean`	Carga un archivo SWF, JPEG, JPEG progresivo, GIF sin animar o PNG en un clip de película en Flash Player mientras se reproduce el archivo original.
	`removeListener(listener:Object) : Boolean`	Elimina el detector que se ha utilizado para recibir notificación de la invocación del controlador de eventos `MovieClipLoader`.
	`unloadClip(target:Object) : Boolean`	Elimina un clip de película que se ha cargado mediante `MovieClipLoader.loadClip()`.

Métodos heredados de la clase Object

`addProperty (método Object.addProperty), hasOwnProperty (método Object.hasOwnProperty), isPropertyEnumerable (método Object.isPropertyEnumerable), isPrototypeOf (método Object.isPrototypeOf), registerClass (método Object.registerClass), toString (método Object.toString), unwatch (método Object.unwatch), valueOf (método Object.valueOf), watch (método Object.watch)`

addListener (método MovieClipLoader.addListener)

`public addListener(listener:Object) : Boolean`

Registra un objeto para recibir notificación de la invocación de un controlador de eventos `MovieClipLoader`.

Disponibilidad: ActionScript 1.0; Flash Player 7

Parámetros

`listener``:Object` - Un objeto que detecta una notificación callback de los controladores de eventos `MovieClipLoader`.

Valor devuelto

`Boolean` - Valor booleano. Si el detector se ha establecido correctamente, devuelve `true`, en caso contrario devuelve `false`.

Ejemplo

El ejemplo siguiente carga una imagen en un clip de película llamado `image_mc`. La instancia de clip de película se gira y se centra en el escenario, y tanto el escenario como el clip de película tienen un trazo dibujado alrededor de sus parámetros.

```
this.createEmptyMovieClip("image_mc", this.getNextHighestDepth());
var mclListener:Object = new Object();
mclListener.onLoadInit = function(target_mc:MovieClip) {
  target_mc._x = Stage.width/2-target_mc._width/2;
  target_mc._y = Stage.height/2-target_mc._width/2;
  var w:Number = target_mc._width;
  var h:Number = target_mc._height;
  target_mc.lineStyle(4, 0x000000);
  target_mc.moveTo(0, 0);
  target_mc.lineTo(w, 0);
  target_mc.lineTo(w, h);
  target_mc.lineTo(0, h);
  target_mc.lineTo(0, 0);
  target_mc._rotation = 3;
};
var image_mcl:MovieClipLoader = new MovieClipLoader();
image_mcl.addListener(mclListener);
image_mcl.loadClip("http://www.helpexamples.com/flash/images/image1.jpg",
  image_mc);
```

Si el archivo SWF incluye un componente de la versión 2, utilice la clase DepthManager de los componentes de la versión 2 en lugar del método `MovieClip.getNextHighestDepth()` utilizado en este ejemplo.

Véase también

onLoadComplete (detector de eventos MovieClipLoader.onLoadComplete), onLoadError (detector de eventos MovieClipLoader.onLoadError), onLoadInit (detector de eventos MovieClipLoader.onLoadInit), onLoadProgress (detector de eventos MovieClipLoader.onLoadProgress), onLoadStart (detector de eventos MovieClipLoader.onLoadStart), removeListener (método MovieClipLoader.removeListener)

getProgress (método MovieClipLoader.getProgress)

```
public getProgress(target:Object) : Object
```

Devuelve el número de bytes cargados y el número total de bytes de un archivo que se está cargando mediante `MovieClipLoader.loadClip()`; en el caso de películas comprimidas, refleja el número de bytes comprimidos. El método `getProgress` permite solicitar explícitamente esta información en lugar de (o además de) escribir una función de detector `MovieClipLoader.onLoadProgress`.

Disponibilidad: ActionScript 1.0; Flash Player 7

Parámetros

`target``:Object` - Un archivo SWF, JPEG, GIF o PNG que se carga mediante `MovieClipLoader.loadClip()`.

Valor devuelto

`Object` - Un objeto que dispone de dos propiedades de enteros: `bytesLoaded` y `bytesTotal`.

Ejemplo

El ejemplo siguiente demuestra cómo utilizar correctamente el método `getProgress()`. En lugar de utilizar este método, normalmente se crea un objeto detector para el evento `onLoadProgress`. Además la primera llamada sincrónica a `getProgress()` puede devolver el número de bytes cargados y el número total de bytes del *container* y no los valores del objeto solicitado externamente.

```
var container:MovieClip = this.createEmptyMovieClip("container",
  this.getNextHighestDepth());
var image:MovieClip = container.createEmptyMovieClip("image",
  container.getNextHighestDepth());

var mcLoader:MovieClipLoader = new MovieClipLoader();
var listener:Object = new Object();
listener.onLoadProgress = function(target:MovieClip, bytesLoaded:Number,
  bytesTotal:Number):Void {
```

```
  trace(target + ".onLoadProgress with " + bytesLoaded + " bytes of " +
  bytesTotal);
}
mcLoader.addListener(listener);
mcLoader.loadClip("http://www.w3.org/Icons/w3c_main.png", image);

var interval:Object = new Object();
interval.id = setInterval(checkProgress, 100, mcLoader, image, interval);

function checkProgress(mcLoader:MovieClipLoader, image:MovieClip,
  interval:Object):Void {
  trace(">> checking progress now with : " + interval.id);
  var progress:Object = mcLoader.getProgress(image);
  trace("bytesLoaded: " + progress.bytesLoaded + " bytesTotal: " +
  progress.bytesTotal);
  if(progress.bytesLoaded == progress.bytesTotal) {
    clearInterval(interval.id);
  }
}
```

Si el archivo SWF incluye un componente de la versión 2, utilice la clase DepthManager de los componentes de la versión 2 en lugar del método `MovieClip.getNextHighestDepth()` utilizado en este ejemplo.

Véase también

`loadClip (método MovieClipLoader.loadClip)`, `onLoadProgress (detector de eventos MovieClipLoader.onLoadProgress)`

loadClip (método MovieClipLoader.loadClip)

```
public loadClip(url:String, target:Object) : Boolean
```

Carga un archivo SWF, JPEG, JPEG progresivo, GIF sin animar o PNG en un clip de película en Flash Player mientras se reproduce el archivo original. Si carga un archivo GIF animado, sólo aparece el primer fotograma. Mediante este método es posible mostrar varios archivos SWF a la vez y cambiar entre archivos SWF sin cargar otro documento HTML.

La utilización del método `loadClip()` en lugar de `loadMovie()` o `MovieClip.loadMovie()` ofrece diversas ventajas. Los siguientes controladores se implementan mediante el uso de un objeto detector. Para activar el detector basta con registrarlo con la clase MovieClipLoader mediante `MovieClipLoader.addListener(listenerObject)`.

- El controlador `MovieClipLoader.onLoadStart` se invoca cuando comienza la carga.
- El controlador `MovieClipLoader.onLoadError` se invoca si no es posible cargar el clip.
- El controlador `MovieClipLoader.onLoadProgress` se invoca mientras progresa el proceso de carga.

- El controlador `MovieClipLoader.onLoadComplete` se invoca cuando finaliza la descarga de un archivo, pero antes de que estén disponibles los métodos y propiedades del clip de película cargado. Se llama a este controlador antes que al controlador `onLoadInit`.
- El controlador `MovieClipLoader.onLoadInit` se invoca después de que se hayan ejecutado las acciones del primer fotograma del clip para que pueda manipular el clip cargado. Se llama a este controlador antes que al controlador `onLoadComplete`. En la mayoría de los casos, utilice el controlador `onLoadInit`.

Un archivo SWF o imagen cargada en un clip de película hereda las propiedades de posición, giro y escala del clip de película. Puede utilizar la ruta de destino del clip de película para emplear como destino la película cargada.

Puede utilizar el método `loadClip()` para cargar uno o varios archivos en un solo clip de película o nivel; los objetos detectores MovieClipLoader se pasan a la instancia de clip de película de destino que se está cargando como parámetros. Como alternativa, puede crear un objeto MovieClipLoader diferente para cada archivo que cargue.

Utilice `MovieClipLoader.unloadClip()` para eliminar películas o imágenes cargadas con este método o cancelar una operación de carga que se encuentra en curso.

`MovieClipLoader.getProgress()` y `MovieClipLoaderListener.onLoadProgress` no informan de los valores reales de `bytesLoaded` y `bytesTotal` en el reproductor de edición si los archivos son locales. Si utiliza la función Visor de anchos de banda del entorno de edición, `MovieClipLoader.getProgress()` y `MovieClipLoaderListener.onLoadProgress` informarán de la descarga con la velocidad de descarga real, no con la velocidad de ancho de banda reducido que ofrece el Visor de anchos de banda.

Cuando utilice este método puede usar el modelo de seguridad de Flash Player.

En Flash Player 8:

- No se admite la carga si el clip de película que realiza la llamada se encuentra en la libre configuración local con sistema de archivos y el clip de película se encuentra en una red.
- No se admite la carga si el archivo SWF que realiza la llamada se encuentra en la libre configuración de red y el clip de película que debe cargarse está en un sistema local.
- Para acceder al entorno limitado de red desde el entorno limitado local de confianza o local con acceso a la red se necesitan permisos del sitio Web a través de un archivo de política de varios dominios.
- Es posible que los clips de película del entorno limitado local con sistema de archivos no puedan crear scripts de clips de película en el entorno limitado local con acceso a la red (y viceversa).

En Flash Player 7 y versiones anteriores:

- Los sitios Web pueden permitir el acceso entre dominios a un recurso mediante un archivo de política de varios dominios.
- La creación de scripts entre archivos SWF queda restringida en función del dominio de origen de los archivos SWF. Utilice el método `System.security.allowDomain()` para ajustar estas restricciones.

Para más información, consulte las referencias siguientes:

- Capitulo 17, "Aspectos básicos de la seguridad", de *Aprendizaje de ActionScript 2.0 en Flash*
- El documento técnico sobre la seguridad de Flash Player 8 en http://www.macromedia.com/go/fp8_security
- El documento técnico sobre interfaces API relativas a seguridad de Flash Player 8 en http://www.macromedia.com/go/fp8_security_apis

Disponibilidad: ActionScript 1.0; Flash Player 7

Parámetros

url:`String` - URL absoluta o relativa del archivo SWF, JPEG, GIF o PNG que va a cargarse. Una ruta relativa debe ser relativa al archivo SWF en el nivel 0. Las URL absolutas deben incluir la referencia al protocolo, como http:// o file:///. Los nombres de archivo no pueden incluir especificaciones de unidad de disco.

target:`Object` - Ruta de destino de un clip de película, o un entero que especifica el nivel en Flash Player en el que se cargará la película. El clip de película de destino se sustituye por el archivo SWF o la imagen que se carga.

Valor devuelto

`Boolean` - Valor booleano. Si la petición de URL se ha enviado correctamente, devuelve `true`, en caso contrario devuelve `false`.

Ejemplo

Los ejemplos siguientes muestran cómo utilizar el método `MovieClipLoader.loadClip()` creando un controlador para el evento `onLoadInit` y realizando a continuación la petición.

El siguiente código debe situarse directamente en una acción de fotograma en una línea de tiempo o pegarse en una clase que amplíe MovieClip. Este código también espera que exista una imagen denominada YourImage.jpg en el mismo directorio que el archivo SWF compilado.

```
var container:MovieClip = createEmptyMovieClip("container",
  getNextHighestDepth());
```

```
var mcLoader:MovieClipLoader = new MovieClipLoader();
mcLoader.addListener(this);
mcLoader.loadClip("YourImage.jpg", container);

function onLoadInit(mc:MovieClip) {
  trace("onLoadInit: " + mc);
}
```

Si el archivo SWF incluye un componente de la versión 2, utilice la clase DepthManager de los componentes de la versión 2 en lugar del método `MovieClip.getNextHighestDepth()` utilizado en este ejemplo.

Véase también

`onLoadInit (detector de eventos MovieClipLoader.onLoadInit)`

MovieClipLoader, constructor

```
public MovieClipLoader()
```

Crea un objeto MovieClipLoader que puede utilizar para implementar una serie de detectores que respondan a eventos mientras se está descargando un archivo SWF, JPEG, GIF o PNG.

Disponibilidad: ActionScript 1.0; Flash Player 7

Ejemplo

Véase `MovieClipLoader.loadClip()`.

Véase también

`addListener (método MovieClipLoader.addListener)`, `loadClip (método MovieClipLoader.loadClip)`

onLoadComplete (detector de eventos MovieClipLoader.onLoadComplete)

```
onLoadComplete = function([target_mc:MovieClip], [httpStatus:Number]) {}
```

Se invoca cuando un archivo cargado con `MovieClipLoader.loadClip()` se ha descargado completamente. Realice una llamada a este detector sobre un objeto agregado con `MovieClipLoader.addListener()`. Flash Player pasa el detector de eventos `onLoadComplete` al código, aunque no tiene que implementar todos los parámetros de la función de detector. El valor de `target_mc` identifica al clip de película para el que se ha realizado esta llamada. Esto resulta útil si se están cargando varios archivos con el mismo conjunto de detectores.

En Flash Player 8, este detector puede devolver un código de estado HTTP. Si Flash Player no puede obtener el código de estado del servidor o si Flash Player no puede comunicarse con el servidor, se pasa el valor predeterminado 0 al código ActionScript. Es posible generar un valor 0 en cualquier reproductor (por ejemplo, si se solicita una dirección URL incorrecta) y el complemento Flash Player siempre genera un valor 0 cuando se ejecuta en los navegadores siguientes, ya que no pueden pasar códigos de estado HTTP del servidor a Flash Player: Netscape, Mozilla, Safari, Opera e Internet Explorer para Macintosh.

Es importante entender la diferencia entre `MovieClipLoader.onLoadComplete` y `MovieClipLoader.onLoadInit`. La llamada al evento `onLoadComplete` se produce después de que se cargue el archivo SWF, JPEG, GIF o PNG, pero antes de que la aplicación se haya inicializado. Llegado este punto, es imposible acceder a los métodos y propiedades del clip de película cargado, por lo que no podrá llamar a ninguna función, trasladarse a un fotograma concreto, etc. En la mayoría de los casos, es preferible utilizar el evento `onLoadInit`, al que se llama después de que el contenido se haya cargado y esté totalmente inicializado.

Disponibilidad: ActionScript 1.0; Flash Player 7

Parámetros

`target_mc``:MovieClip` [opcional] - Clip de película cargado mediante el método `MovieClipLoader.loadClip()`.

`httpStatus``:Number` [opcional] - (sólo Flash Player 8) El código de estado HTTP devuelto por el servidor. Por ejemplo, un código de estado de 404 indica que el servidor no encontró nada que coincida con el URI solicitado. Para más información sobre los códigos de estado HTTP, consulte las secciones 10.4 y 10.5 de la especificación HTTP en ftp://ftp.isi.edu/in-notes/rfc2616.txt.

Ejemplo

El ejemplo siguiente crea un clip de película, una nueva instancia `MovieClipLoader` y un detector anónimo para los eventos `onLoadComplete`, espera que se produzca un evento `onLoadInit` para interactuar con las propiedades del elemento cargado.

```
var loadListener:Object = new Object();

loadListener.onLoadComplete = function(target_mc:MovieClip,
  httpStatus:Number):Void {
  trace(">> loadListener.onLoadComplete()");
  trace(">> ============================");
  trace(">> target_mc._width: " + target_mc._width); // 0
  trace(">> httpStatus: " + httpStatus);
}

loadListener.onLoadInit = function(target_mc:MovieClip):Void {
```

```
  trace(">> loadListener.onLoadInit()");
  trace(">> ============================");
  trace(">> target_mc._width: " + target_mc._width); // 315
}

var mcLoader:MovieClipLoader = new MovieClipLoader();
mcLoader.addListener(loadListener);

var mc:MovieClip = this.createEmptyMovieClip("mc",
  this.getNextHighestDepth());
mcLoader.loadClip("http://www.w3.org/Icons/w3c_main.png", mc);
```

Si el archivo SWF incluye un componente de la versión 2, utilice la clase DepthManager de los componentes de la versión 2 en lugar del método `MovieClip.getNextHighestDepth()` utilizado en este ejemplo.

Véase también

`addListener (método MovieClipLoader.addListener)`, `loadClip (método MovieClipLoader.loadClip)`, `onLoadStart (detector de eventos MovieClipLoader.onLoadStart)`, `onLoadError (detector de eventos MovieClipLoader.onLoadError)`, `onLoadInit (detector de eventos MovieClipLoader.onLoadInit)`

onLoadError (detector de eventos MovieClipLoader.onLoadError)

```
onLoadError = function(target_mc:MovieClip, errorCode:String,
  [httpStatus:Number]) {}
```

Se invoca cuando un archivo cargado con `MovieClipLoader.loadClip()` no se carga correctamente. Este detector se puede invocar por varios motivos; por ejemplo, si el servidor no funciona, no se puede localizar el archivo o se produce un problema de seguridad.

Realice una llamada a este detector sobre un objeto agregado con `MovieClipLoader.addListener()`.

El valor de `target_mc` identifica al clip de película para el que se ha realizado esta llamada. Este parámetro es útil cuando se cargan varios archivos con el mismo conjunto de detectores.

En el caso del parámetro `errorCode`, se devuelve la cadena `"URLNotFound"` si no se ha realizado ninguna llamada a los detectores `MovieClipLoader.onLoadStart` o `MovieClipLoader.onLoadComplete`; por ejemplo, si el servidor no funciona, no se puede localizar el archivo. Se devuelve la cadena `"LoadNeverCompleted"` si se llamó a `MovieClipLoader.onLoadStart` sin realizar una llamada a `MovieClipLoader.onLoadComplete`; por ejemplo, si se interrumpió la descarga debido una sobrecarga o una caída del servidor, etc.

En Flash Player 8, este detector puede devolver un código de estado HTTP en el parámetro `httpStatus`. Si Flash Player no puede obtener el código de estado del servidor o si Flash Player no puede comunicarse con el servidor, se pasa el valor predeterminado 0 al código ActionScript. Es posible generar un valor 0 en cualquier reproductor (por ejemplo, si se solicita una dirección URL incorrecta) y el complemento Flash Player siempre genera un valor 0 cuando se ejecuta en los navegadores siguientes, ya que no pueden pasar códigos de estado HTTP del servidor a Flash Player: Netscape, Mozilla, Safari, Opera e Internet Explorer para Macintosh. También es posible generar un valor 0 si el reproductor no intentó realizar la petición URL para la operación de carga. Esto puede ocurrir debido a que la petición quebranta las reglas de seguridad del entorno limitado para el archivo SWF.

Disponibilidad: ActionScript 1.0; Flash Player 7

Parámetros

`target_mc``:MovieClip` - Clip de película cargado mediante el método `MovieClipLoader.loadClip()`.

`errorCode``:String` - Una cadena que explica el motivo del fallo, puede ser `"URLNotFound"` o `"LoadNeverCompleted"`.

`httpStatus``:Number` [opcional] - (sólo Flash Player 8) El código de estado HTTP devuelto por el servidor. Por ejemplo, el código de estado 404 indica que el servidor no encontró nada que coincida con el URI solicitado. Para más información sobre los códigos de estado HTTP, consulte las secciones 10.4 y 10.5 de la especificación HTTP en ftp://ftp.isi.edu/in-notes/rfc2616.txt.

Ejemplo

El ejemplo siguiente muestra información en el panel Salida cuando se produce un error al cargar la imagen. La dirección URL utilizada en este ejemplo es para demostración; sustitúyala con una dirección URL válida.

```
var loadListener:Object = new Object();

loadListener.onLoadError = function(target_mc:MovieClip, errorCode:String,
  httpStatus:Number) {
```

```
  trace(">> loadListener.onLoadError()");
  trace(">> ========================");
  trace(">> errorCode: " + errorCode);
  trace(">> httpStatus: " + httpStatus);
}

var mcLoader:MovieClipLoader = new MovieClipLoader();
mcLoader.addListener(loadListener);

var mc:MovieClip = this.createEmptyMovieClip("mc",
  this.getNextHighestDepth());
mcLoader.loadClip("http://www.fakedomain.com/images/bad_hair_day.jpg", mc);
```

Si el archivo SWF incluye un componente de la versión 2, utilice la clase DepthManager de los componentes de la versión 2 en lugar del método `MovieClip.getNextHighestDepth()` utilizado en este ejemplo.

Véase también

`addListener (método MovieClipLoader.addListener)`, `loadClip (método MovieClipLoader.loadClip)`, `onLoadStart (detector de eventos MovieClipLoader.onLoadStart)`, `onLoadComplete (detector de eventos MovieClipLoader.onLoadComplete)`

onLoadInit (detector de eventos MovieClipLoader.onLoadInit)

```
onLoadInit = function([target_mc:MovieClip]) {}
```

Se invoca cuando se han ejecutado las acciones del primer fotograma del clip cargado. Una vez que se haya invocado este detector, podrá establecer propiedades, utilizar métodos e interactuar con la película cargada. Realice una llamada a este detector sobre un objeto agregado con `MovieClipLoader.addListener()`.

El valor de `target_mc` identifica al clip de película para el que se ha realizado esta llamada. Este parámetro es útil cuando se cargan varios archivos con el mismo conjunto de detectores.

Disponibilidad: ActionScript 1.0; Flash Player 7

Parámetros

`target_mc``:MovieClip` [opcional] - Clip de película cargado mediante el método `MovieClipLoader.loadClip()`.

Ejemplo

El ejemplo siguiente carga una imagen en una instancia de clip de película denominada `image_mc`. Los eventos `onLoadInit` y `onLoadComplete` se utilizan para determinar el tiempo que tarda en cargarse la imagen. La información se muestra en un campo de texto llamado `timer_txt`.

```
this.createEmptyMovieClip("image_mc", this.getNextHighestDepth());
var mclListener:Object = new Object();
mclListener.onLoadStart = function(target_mc:MovieClip) {
  target_mc.startTimer = getTimer();
};
mclListener.onLoadComplete = function(target_mc:MovieClip) {
  target_mc.completeTimer = getTimer();
};
mclListener.onLoadInit = function(target_mc:MovieClip) {
  var timerMS:Number = target_mc.completeTimer-target_mc.startTimer;
  target_mc.createTextField("timer_txt", target_mc.getNextHighestDepth(),
  0, target_mc._height,
target_mc._width, 22);
  target_mc.timer_txt.text = "loaded in "+timerMS+" ms.";
};
var image_mcl:MovieClipLoader = new MovieClipLoader();
image_mcl.addListener(mclListener);
image_mcl.loadClip("http://www.helpexamples.com/flash/images/image1.jpg",
  image_mc);
```

El ejemplo siguiente comprueba si se ha cargado una película en un clip de película creado en tiempo de ejecución. La dirección URL utilizada en este ejemplo es para demostración; sustitúyala con una dirección URL válida.

```
this.createEmptyMovieClip("tester_mc", 1);
var mclListener:Object = new Object();
mclListener.onLoadInit = function(target_mc:MovieClip) {
  trace("movie loaded");
}
var image_mcl:MovieClipLoader = new MovieClipLoader();
image_mcl.addListener(mclListener);
image_mcl.loadClip("http://www.yourserver.com/your_movie.swf", tester_mc);
```

Si el archivo SWF incluye un componente de la versión 2, utilice la clase DepthManager de los componentes de la versión 2 en lugar del método `MovieClip.getNextHighestDepth()` utilizado en este ejemplo.

Véase también

`addListener (método MovieClipLoader.addListener)`, `loadClip (método MovieClipLoader.loadClip)`, `onLoadStart (detector de eventos MovieClipLoader.onLoadStart)`

onLoadProgress (detector de eventos MovieClipLoader.onLoadProgress)

```
onLoadProgress = function([target_mc:MovieClip], loadedBytes:Number,
  totalBytes:Number) {}
```

Se invoca cada vez que se graba en el disco duro el contenido durante el proceso de carga (es decir, entre `MovieClipLoader.onLoadStart` y `MovieClipLoader.onLoadComplete`). Realice una llamada a este detector sobre un objeto agregado con `MovieClipLoader.addListener()`. Puede utilizar este método para mostrar información sobre el progreso de la descarga utilizando los parámetros `loadedBytes` y `totalBytes`.

El valor de `target_mc` identifica al clip de película para el que se ha realizado esta llamada. Esto resulta útil si se están cargando varios archivos con el mismo conjunto de detectores.

Nota: Si intenta utilizar `onLoadProgress` en modo de prueba con un archivo local que reside en el disco duro, éste no funcionará correctamente porque, en dicho modo, Flash Player carga los archivos locales íntegramente.

Disponibilidad: ActionScript 1.0; Flash Player 7

Parámetros

`target_mc``:MovieClip` [opcional] - Clip de película cargado mediante el método `MovieClipLoader.loadClip()`.

`loadedBytes``:Number` - El número de bytes que se habían cargado al invocar al detector.

`totalBytes``:Number` - El número total de bytes del archivo que se está cargando.

Ejemplo

El ejemplo siguiente crea un clip de película, una nueva instancia `MovieClipLoader` y detector de eventos anónimo. De forma periódica, muestra el progreso de un proceso de carga y notifica cuando ha finalizado la carga y el recurso queda disponible para ActionScript.

```
var container:MovieClip = this.createEmptyMovieClip("container",
  this.getNextHighestDepth());
var mcLoader:MovieClipLoader = new MovieClipLoader();
var listener:Object = new Object();
listener.onLoadProgress = function(target:MovieClip, bytesLoaded:Number,
  bytesTotal:Number):Void {
  trace(target + ".onLoadProgress with " + bytesLoaded + " bytes of " +
  bytesTotal);
}
listener.onLoadInit = function(target:MovieClip):Void {
  trace(target + ".onLoadInit");
}
mcLoader.addListener(listener);
```

```
mcLoader.loadClip("http://www.w3.org/Icons/w3c_main.png", container);
```

Si el archivo SWF incluye un componente de la versión 2, utilice la clase DepthManager de los componentes de la versión 2 en lugar del método `MovieClip.getNextHighestDepth()` utilizado en este ejemplo.

Véase también

`addListener (método MovieClipLoader.addListener)`, `loadClip (método MovieClipLoader.loadClip)`, `getProgress (método MovieClipLoader.getProgress)`

onLoadStart (detector de eventos MovieClipLoader.onLoadStart)

```
onLoadStart = function([target_mc:MovieClip]) {}
```

Se invoca cuando una llamada a `MovieClipLoader.loadClip()` ha iniciado correctamente la descarga de un archivo. Realice una llamada a este detector sobre un objeto agregado con `MovieClipLoader.addListener()`.

El valor de `target_mc` identifica al clip de película para el que se ha realizado esta llamada. Este parámetro es útil cuando se cargan varios archivos con el mismo conjunto de detectores.

Disponibilidad: ActionScript 1.0; Flash Player 7

Parámetros

`target_mc``:MovieClip` [opcional] - Clip de película cargado mediante el método `MovieClipLoader.loadClip()`.

Ejemplo

El ejemplo siguiente carga una imagen en una instancia de clip de película denominada `image_mc`. Los eventos `onLoadInit` y `onLoadComplete` se utilizan para determinar el tiempo que tarda en cargarse la imagen. La información se muestra en un campo de texto llamado `timer_txt`.

```
this.createEmptyMovieClip("image_mc", this.getNextHighestDepth());
var mclListener:Object = new Object();
mclListener.onLoadStart = function(target_mc:MovieClip) {
  target_mc.startTimer = getTimer();
};
mclListener.onLoadComplete = function(target_mc:MovieClip) {
  target_mc.completeTimer = getTimer();
};
mclListener.onLoadInit = function(target_mc:MovieClip) {
  var timerMS:Number = target_mc.completeTimer-target_mc.startTimer;
```

```
  target_mc.createTextField("timer_txt", target_mc.getNextHighestDepth(),
  0, target_mc._height,
target_mc._width, 22);
  target_mc.timer_txt.text = "loaded in "+timerMS+" ms.";
};
var image_mcl:MovieClipLoader = new MovieClipLoader();
image_mcl.addListener(mclListener);
image_mcl.loadClip("http://www.helpexamples.com/flash/images/image1.jpg",
  image_mc);
```

Si el archivo SWF incluye un componente de la versión 2, utilice la clase DepthManager de los componentes de la versión 2 en lugar del método `MovieClip.getNextHighestDepth()` utilizado en este ejemplo.

Véase también

`addListener (método MovieClipLoader.addListener)`, `loadClip (método MovieClipLoader.loadClip)`, `onLoadError (detector de eventos MovieClipLoader.onLoadError)`, `onLoadInit (detector de eventos MovieClipLoader.onLoadInit)`, `onLoadComplete (detector de eventos MovieClipLoader.onLoadComplete)`

removeListener (método MovieClipLoader.removeListener)

```
public removeListener(listener:Object) : Boolean
```

Elimina el detector que se ha utilizado para recibir notificación de la invocación del controlador de eventos `MovieClipLoader`. No se recibirán más mensajes de carga.

Disponibilidad: ActionScript 1.0; Flash Player 7

Parámetros

`listener``:Object` - Un objeto detector que se añadió utilizando `MovieClipLoader.addListener()`.

Valor devuelto

`Boolean` - Valor booleano. Si el detector se ha eliminado correctamente, devuelve `true`, en caso contrario devuelve `false`.

Ejemplo

El ejemplo siguiente carga una imagen en un clip de película y permite al usuario iniciar y detener el proceso de carga mediante dos botones llamados `start_button` y `stop_button`. Cuando el usuario inicia o detiene el proceso, se muestra información en el panel Salida.

```
this.createEmptyMovieClip("image_mc", this.getNextHighestDepth());
var mclListener:Object = new Object();
mclListener.onLoadStart = function(target_mc:MovieClip) {
  trace("\t onLoadStart");
};
mclListener.onLoadComplete = function(target_mc:MovieClip) {
  trace("\t onLoadComplete");
};
mclListener.onLoadError = function(target_mc:MovieClip, errorCode:String) {
  trace("\t onLoadError: "+errorCode);
};
mclListener.onLoadInit = function(target_mc:MovieClip) {
  trace("\t onLoadInit");
  start_button.enabled = true;
  stop_button.enabled = false;
};
var image_mcl:MovieClipLoader = new MovieClipLoader();
//
start_button.clickHandler = function() {
  trace("Starting...");
  start_button.enabled = false;
  stop_button.enabled = true;
  //
  image_mcl.addListener(mclListener);
  image_mcl.loadClip("http://www.helpexamples.com/flash/images/
  image1.jpg", image_mc);
};
stop_button.clickHandler = function() {
  trace("Stopping...");
  start_button.enabled = true;
  stop_button.enabled = false;
  //
  image_mcl.removeListener(mclListener);
};
stop_button.enabled = false;
```

Si el archivo SWF incluye un componente de la versión 2, utilice la clase DepthManager de los componentes de la versión 2 en lugar del método `MovieClip.getNextHighestDepth()` utilizado en este ejemplo.

Véase también

`addListener (método MovieClipLoader.addListener)`

unloadClip (método MovieClipLoader.unloadClip)

```
public unloadClip(target:Object) : Boolean
```

Elimina un clip de película que se ha cargado mediante `MovieClipLoader.loadClip()`. Si envía este comando mientras se está cargando una película, se invocará `MovieClipLoader.onLoadError`.

Disponibilidad: ActionScript 1.0; Flash Player 7

Parámetros

`target``:Object` - Cadena o entero que se pasa a la llamada correspondiente a `my_mcl.loadClip()`.

Valor devuelto

`Boolean` - Valor booleano. Si el clip de película se ha eliminado correctamente, devuelve `true`, en caso contrario devuelve `false`.

Ejemplo

El ejemplo siguiente carga una imagen en un clip de película llamado `image_mc`. Si hace clic en el clip de película, éste se elimina y se muestra información en el panel Salida.

```
this.createEmptyMovieClip("image_mc", this.getNextHighestDepth());
var mclListener:Object = new Object();
mclListener.onLoadInit = function(target_mc:MovieClip) {
  target_mc._x = 100;
  target_mc._y = 100;
  target_mc.onRelease = function() {
  trace("Unloading clip...");
  trace("\t name: "+target_mc._name);
  trace("\t url: "+target_mc._url);
  image_mcl.unloadClip(target_mc);
  };
};
var image_mcl:MovieClipLoader = new MovieClipLoader();
image_mcl.addListener(mclListener);
image_mcl.loadClip("http://www.helpexamples.com/flash/images/image1.jpg",
  image_mc);
```

Si el archivo SWF incluye un componente de la versión 2, utilice la clase DepthManager de los componentes de la versión 2 en lugar del método `MovieClip.getNextHighestDepth()` utilizado en este ejemplo.

Véase también

`loadClip (método MovieClipLoader.loadClip)`, `onLoadError (detector de eventos MovieClipLoader.onLoadError)`

NetConnection

```
Object
  |
  +-NetConnection
```

```
public dynamic class NetConnection
extends Object
```

La clase NetConnection proporciona un medio para reproducir archivos FLV sin interrupción desde una unidad local o dirección HTTP.

Nota: Esta clase también es compatible con Flash Player 6 cuando se utiliza con Flash Communication Server. Para más información, consulte la documentación de Flash Communication Server.

Disponibilidad: ActionScript 1.0; Flash Player 7

Resumen de propiedades

Propiedades heredadas de la clase Object

`constructor (propiedad Object.constructor), __proto__ (propiedad Object.__proto__), prototype (propiedad Object.prototype), __resolve (propiedad Object.__resolve)`

Resumen de constructores

Firma	Descripción
`NetConnection()`	Crea un objeto NetConnection que puede utilizar conjuntamente con un objeto NetStream para reproducir archivos de vídeo (FLV) locales sin interrupción.

Resumen de métodos

Modificadores	Firma	Descripción
	`connect(targetURI:String) : Boolean`	Abre una conexión local a través de la cual puede reproducir archivos de vídeo (FLV) de una dirección HTTP o un sistema de archivos local.

addProperty (método Object.addProperty), hasOwnProperty (método Object.hasOwnProperty), isPropertyEnumerable (método Object.isPropertyEnumerable), isPrototypeOf (método Object.isPrototypeOf), registerClass (método Object.registerClass), toString (método Object.toString), unwatch (método Object.unwatch), valueOf (método Object.valueOf), watch (método Object.watch)

connect (método NetConnection.connect)

```
public connect(targetURI:String) : Boolean
```

Abre una conexión local a través de la cual puede reproducir archivos de vídeo (FLV) de una dirección HTTP o un sistema de archivos local.

Cuando utilice este método puede usar el modelo de seguridad de Flash Player y tener en cuenta lo siguiente:

- El valor predeterminado es denegar el acceso entre entornos limitados. El sitio Web puede activar el acceso a un recurso utilizando un archivo de política interdominios.
- Un sitio Web puede denegar el acceso a un recurso añadiendo lógica de aplicación ActionScript en Flash Communication Server.
- En el caso de Flash Player 8, NetConnection.connect() no se admite si el archivo SWF que realiza la llamada se encuentra en el entorno limitado local con sistema de archivos.

Para más información, consulte las referencias siguientes:

- Capitulo 17, "Aspectos básicos de la seguridad", de *Aprendizaje de ActionScript 2.0 en Flash*
- El documento técnico sobre la seguridad de Flash Player 8 en http://www.macromedia.com/go/fp8_security
- El documento técnico sobre interfaces API relativas a seguridad de Flash Player 8 en http://www.macromedia.com/go/fp8_security_apis

Disponibilidad: ActionScript 1.0; Flash Player 7

Parámetros

targetURI:String - Para este parámetro, debe pasar null.

Valor devuelto

Boolean - En caso de obtenerse false, la conexión ha fallado y no se puede utilizar. Si el valor devuelto es true, la conexión no ha fallado en el momento en que se llamó al método connect(), aunque esto no garantiza que sea correcta.

Ejemplo

El ejemplo siguiente abre una conexión para reproducir el archivo video2.flv. Seleccione Nuevo vídeo en el menú de opciones del panel Biblioteca para crear un objeto de nuevo vídeo y asígnele el nombre de instancia `my_video`.

```
var connection_nc:NetConnection = new NetConnection();
connection_nc.connect(null);
var stream_ns:NetStream = new NetStream(connection_nc);
my_video.attachVideo(stream_ns);
stream_ns.play("video2.flv");
```

Véase también

`NetStream`

NetConnection, constructor

```
public NetConnection()
```

Crea un objeto NetConnection que puede utilizar conjuntamente con un objeto NetStream para reproducir archivos de vídeo (FLV) locales sin interrupción. Tras crear el objeto NetConnection, utilice `NetConnection.connect()` para establecer la conexión en sí.

La reproducción de archivos FLV externos ofrece varias ventajas frente a la incorporación de vídeo en un documento de Flash, como por ejemplo mejor rendimiento y administración de la memoria y velocidades de fotogramas de vídeo y Flash independientes. La clase NetConnection proporciona un medio para reproducir archivos FLV sin interrupción desde una unidad local o dirección HTTP.

Disponibilidad: ActionScript 1.0; Flash Player 7

Ejemplo

Consulte el ejemplo de `NetConnection.connect()`.

Véase también

`connect (método NetConnection.connect)`, `attachVideo (método Video.attachVideo)`, `NetStream`

NetStream

```
Object
  |
  +-NetStream
```

```
public dynamic class NetStream
extends Object
```

La clase NetStream proporciona métodos y propiedades para la reproducción de archivos de vídeo Flash (FLV) del sistema de archivos local o de una dirección HTTP. Deberá utilizar un objeto NetStream para reproducir vídeo sin interrupción a través de un objeto NetConnection. La reproducción de archivos FLV externos ofrece varias ventajas frente a la incorporación de vídeo en un documento de Flash, como por ejemplo mejor rendimiento y administración de la memoria y velocidades de fotogramas de vídeo y Flash independientes. Esta clase proporciona una serie de métodos y propiedades que puede utilizar para controlar el progreso del archivo conforme se carga y reproduce, así como para proporcionar al usuario control sobre la reproducción (detener, pausa, etc.).

Disponibilidad: ActionScript 1.0; Flash Player 7

Resumen de propiedades

Modificadores	Propiedad	Descripción
	`bufferLength:Number` [read-only]	El número de segundos de datos almacenados actualmente en el búfer.
	`bufferTime:Number` [read-only]	El número de segundos asignados al búfer por `NetStream.setBufferTime()`.
	`bytesLoaded:Number` [read-only]	El número de bytes de datos que se han cargado en el reproductor.
	`bytesTotal:Number` [read-only]	El tamaño total en bytes del archivo que se está cargando en el reproductor.
	`currentFps:Number` [read-only]	El número de fotogramas por segundo que se muestran.
	`time:Number` [read-only]	La posición, en segundos, de la cabeza lectora.

Propiedades heredadas de la clase Object

`constructor (propiedad Object.constructor), __proto__ (propiedad Object.__proto__), prototype (propiedad Object.prototype), __resolve (propiedad Object.__resolve)`

Resumen de eventos

Evento	Descripción
`onMetaData = function(infoObject:Object) {}`	Se invoca cuando Flash Player recibe la información incorporada en el archivo FLV que se está reproduciendo.
`onStatus = function(infoObject:Object) {}`	Se invoca cada vez que se publica un cambio de estado o error para el objeto NetStream.

Resumen de constructores

Firma	Descripción
`NetStream(connection:NetConnection)`	Crea un flujo que puede utilizarse para reproducir archivos FLV a través del objeto NetConnection especificado.

Resumen de métodos

Modificadores	Firma	Descripción
	`close() : Void`	Detiene la reproducción de todos los datos del flujo, establece la propiedad `NetStream.time` con el valor 0 y pone el flujo a disposición de otro usuario.
	`pause([flag:Boolean]) : Void`	Realiza una pausa o reanuda la reproducción de un flujo.
	`play(name:Object, start:Number, len:Number, reset:Object) : Void`	Comienza la reproducción de un archivo de vídeo externo (FLV).
	`seek(offset:Number) : Void`	Busca el fotograma clave más cercano al número de segundos especificado desde el comienzo del flujo.
	`setBufferTime(bufferTime:Number) : Void`	Especifica el tiempo que deben almacenarse en el búfer los mensajes antes de comenzar a mostrar el flujo.

Métodos heredados de la clase Object

`addProperty (método Object.addProperty), hasOwnProperty (método Object.hasOwnProperty), isPropertyEnumerable (método Object.isPropertyEnumerable), isPrototypeOf (método Object.isPrototypeOf), registerClass (método Object.registerClass), toString (método Object.toString), unwatch (método Object.unwatch), valueOf (método Object.valueOf), watch (método Object.watch)`

bufferLength (propiedad NetStream.bufferLength)

```
public bufferLength : Number [read-only]
```

El número de segundos de datos almacenados actualmente en el búfer. Puede utilizar esta propiedad conjuntamente con `NetStream.bufferTime` para estimar cuánto le falta al búfer para llenarse (por ejemplo, para mostrar información al usuario que está esperando a que los datos se carguen en el búfer).

Disponibilidad: ActionScript 1.0; Flash Player 7

Ejemplo

El ejemplo siguiente crea de forma dinámica un campo de texto que muestra información sobre el número de segundos que hay actualmente en el búfer. El campo de texto muestra también la longitud del búfer a la que está definida el vídeo, así como el porcentaje del búfer que está lleno.

```
this.createTextField("buffer_txt", this.getNextHighestDepth(), 10, 10, 300,
  22);
buffer_txt.html = true;

var connection_nc:NetConnection = new NetConnection();
connection_nc.connect(null);
var stream_ns:NetStream = new NetStream(connection_nc);
stream_ns.setBufferTime(3);
my_video.attachVideo(stream_ns);
stream_ns.play("video1.flv");

var buffer_interval:Number = setInterval(checkBufferTime, 100, stream_ns);
function checkBufferTime(my_ns:NetStream):Void {
  var bufferPct:Number = Math.min(Math.round(my_ns.bufferLength/
  my_ns.bufferTime 100), 100);
  var output_str:String = "<textformat tabStops='[100,200]'>";
  output_str += "Length: "+my_ns.bufferLength+"\t"+"Time:
  "+my_ns.bufferTime+"\t"+"Buffer:"+bufferPct+"%";
  output_str += "</textformat>";
  buffer_txt.htmlText = output_str;
}
```

Si el archivo SWF incluye un componente de la versión 2, utilice la clase DepthManager de los componentes de la versión 2 en lugar del método `MovieClip.getNextHighestDepth()` utilizado en este ejemplo.

Véase también

bufferTime (propiedad NetStream.bufferTime), bytesLoaded (propiedad NetStream.bytesLoaded)

bufferTime (propiedad NetStream.bufferTime)

```
public bufferTime : Number [read-only]
```

El número de segundos asignados al búfer por `NetStream.setBufferTime()`. El valor predeterminado es .1 (una décima de segundo). Para determinar el número de segundos que hay actualmente en el búfer, utilice `NetStream.bufferLength`.

Disponibilidad: ActionScript 1.0; Flash Player 7

Ejemplo

El ejemplo siguiente crea de forma dinámica un campo de texto que muestra información sobre el número de segundos que hay actualmente en el búfer. El campo de texto muestra también la longitud del búfer a la que está definida el vídeo, así como el porcentaje del búfer que está lleno.

```
this.createTextField("buffer_txt", this.getNextHighestDepth(), 10, 10, 300,
  22);
buffer_txt.html = true;

var connection_nc:NetConnection = new NetConnection();
connection_nc.connect(null);
var stream_ns:NetStream = new NetStream(connection_nc);
stream_ns.setBufferTime(3);
my_video.attachVideo(stream_ns);
stream_ns.play("video1.flv");

var buffer_interval:Number = setInterval(checkBufferTime, 100, stream_ns);
function checkBufferTime(my_ns:NetStream):Void {
  var bufferPct:Number = Math.min(Math.round(my_ns.bufferLength/
  my_ns.bufferTime 100), 100);
  var output_str:String = "<textformat tabStops='[100,200]'>";
  output_str += "Length: "+my_ns.bufferLength+"\t"+"Time:
  "+my_ns.bufferTime+"\t"+"Buffer:"+bufferPct+"%";
  output_str += "</textformat>";
  buffer_txt.htmlText = output_str;
}
```

Si el archivo SWF incluye un componente de la versión 2, utilice la clase DepthManager de los componentes de la versión 2 en lugar del método `MovieClip.getNextHighestDepth()` utilizado en este ejemplo.

Véase también

`setBufferTime (método NetStream.setBufferTime)`, `time (propiedad NetStream.time)`, `bufferLength (propiedad NetStream.bufferLength)`

bytesLoaded (propiedad NetStream.bytesLoaded)

```
public bytesLoaded : Number [read-only]
```

El número de bytes de datos que se han cargado en el reproductor. Puede utilizar este método conjuntamente con `NetStream.bytesTotal` para estimar cuánto le falta al búfer para llenarse (por ejemplo, para mostrar información al usuario que está esperando a que los datos se carguen en el búfer).

Disponibilidad: ActionScript 1.0; Flash Player 7

Ejemplo

El ejemplo siguiente crea una barra de progreso utilizando la interfaz API de dibujo y las propiedades `bytesLoaded` y `bytesTotal`, que muestra el progreso de carga de video1.flv en la instancia de objeto de vídeo llamada `my_video`. Asimismo se crea de forma dinámica un campo de texto llamado `loaded_txt` para ver información sobre el proceso de carga.

```
var connection_nc:NetConnection = new NetConnection();
connection_nc.connect(null);
var stream_ns:NetStream = new NetStream(connection_nc);
my_video.attachVideo(stream_ns);
stream_ns.play("video1.flv");

this.createTextField("loaded_txt", this.getNextHighestDepth(), 10, 10, 160,
  22);
this.createEmptyMovieClip("progressBar_mc", this.getNextHighestDepth());
progressBar_mc.createEmptyMovieClip("bar_mc",
  progressBar_mc.getNextHighestDepth());
with (progressBar_mc.bar_mc) {
  beginFill(0xFF0000);
  moveTo(0, 0);
  lineTo(100, 0);
  lineTo(100, 10);
  lineTo(0, 10);
  lineTo(0, 0);
  endFill();
  _xscale = 0;
}
progressBar_mc.createEmptyMovieClip("stroke_mc",
  progressBar_mc.getNextHighestDepth());
with (progressBar_mc.stroke_mc) {
  lineStyle(0, 0x000000);
  moveTo(0, 0);
  lineTo(100, 0);
  lineTo(100, 10);
  lineTo(0, 10);
  lineTo(0, 0);
}
```

```
var loaded_interval:Number = setInterval(checkBytesLoaded, 500, stream_ns);
function checkBytesLoaded(my_ns:NetStream) {
  var pctLoaded:Number = Math.round(my_ns.bytesLoaded/my_ns.bytesTotal
  100);
  loaded_txt.text = Math.round(my_ns.bytesLoaded/1000)+" of
  "+Math.round(my_ns.bytesTotal/1000)+" KB loaded ("+pctLoaded+"%)";
  progressBar_mc.bar_mc._xscale = pctLoaded;
  if (pctLoaded>=100) {
  clearInterval(loaded_interval);
  }
}
```

Si el archivo SWF incluye un componente de la versión 2, utilice la clase DepthManager de los componentes de la versión 2 en lugar del método `MovieClip.getNextHighestDepth()` utilizado en este ejemplo.

Véase también

`bytesTotal (propiedad NetStream.bytesTotal)`, `bufferLength (propiedad NetStream.bufferLength)`

bytesTotal (propiedad NetStream.bytesTotal)

```
public bytesTotal : Number [read-only]
```

El tamaño total en bytes del archivo que se está cargando en el reproductor.

Disponibilidad: ActionScript 1.0; Flash Player 7

Ejemplo

El ejemplo siguiente crea una barra de progreso utilizando la interfaz API de dibujo y las propiedades `bytesLoaded` y `bytesTotal`, que muestra el progreso de carga de video1.flv en la instancia de objeto de vídeo llamada `my_video`. Asimismo se crea de forma dinámica un campo de texto llamado `loaded_txt` para ver información sobre el proceso de carga.

```
var connection_nc:NetConnection = new NetConnection();
connection_nc.connect(null);
var stream_ns:NetStream = new NetStream(connection_nc);
my_video.attachVideo(stream_ns);
stream_ns.play("video1.flv");

this.createTextField("loaded_txt", this.getNextHighestDepth(), 10, 10, 160,
  22);
this.createEmptyMovieClip("progressBar_mc", this.getNextHighestDepth());
progressBar_mc.createEmptyMovieClip("bar_mc",
  progressBar_mc.getNextHighestDepth());
with (progressBar_mc.bar_mc) {
  beginFill(0xFF0000);
```

```
  moveTo(0, 0);
  lineTo(100, 0);
  lineTo(100, 10);
  lineTo(0, 10);
  lineTo(0, 0);
  endFill();
  _xscale = 0;
}
progressBar_mc.createEmptyMovieClip("stroke_mc",
  progressBar_mc.getNextHighestDepth());
with (progressBar_mc.stroke_mc) {
  lineStyle(0, 0x000000);
  moveTo(0, 0);
  lineTo(100, 0);
  lineTo(100, 10);
  lineTo(0, 10);
  lineTo(0, 0);
}

var loaded_interval:Number = setInterval(checkBytesLoaded, 500, stream_ns);
function checkBytesLoaded(my_ns:NetStream) {
  var pctLoaded:Number = Math.round(my_ns.bytesLoaded/my_ns.bytesTotal
  100);
  loaded_txt.text = Math.round(my_ns.bytesLoaded/1000)+" of
  "+Math.round(my_ns.bytesTotal/1000)+" KB loaded ("+pctLoaded+"%)";
  progressBar_mc.bar_mc._xscale = pctLoaded;
  if (pctLoaded>=100) {
  clearInterval(loaded_interval);
  }
}
```

Si el archivo SWF incluye un componente de la versión 2, utilice la clase DepthManager de los componentes de la versión 2 en lugar del método `MovieClip.getNextHighestDepth()` utilizado en este ejemplo.

Véase también

`bytesLoaded (propiedad NetStream.bytesLoaded)`, `bufferTime (propiedad NetStream.bufferTime)`

close (método NetStream.close)

```
public close() : Void
```

Detiene la reproducción de todos los datos del flujo, establece la propiedad `NetStream.time` con el valor 0 y pone el flujo a disposición de otro usuario. Este comando también elimina la copia local de un archivo FLV descargado mediante HTTP. Aunque Flash Player elimina la copia local que crea del archivo FLV, es posible que permanezca una copia del vídeo en el directorio de memoria caché del navegador. Si es preciso impedir totalmente la creación de un caché o almacenamiento local del archivo FLV, utilice Flash Communication Server MX.

Disponibilidad: ActionScript 1.0; Flash Player 7

Ejemplo

La función `onDisconnect()` siguiente cierra una conexión y elimina la copia temporal de video1.flv que se almacenó en el disco local al hacer clic en el botón llamado `close_btn`:

```
var connection_nc:NetConnection = new NetConnection();
connection_nc.connect(null);
var stream_ns:NetStream = new NetStream(connection_nc);
my_video.attachVideo(stream_ns);
stream_ns.play("video1.flv");

close_btn.onRelease = function(){
  stream_ns.close();
};
```

Véase también

`pause (método NetStream.pause)`, `play (método NetStream.play)`

currentFps (propiedad NetStream.currentFps)

```
public currentFps : Number [read-only]
```

El número de fotogramas por segundo que se muestran. Si está exportando archivos FLV que van a reproducirse en diversos sistemas, puede comprobar este valor durante la realización de pruebas para que le resulte más fácil determinar la cantidad de compresión que debe aplicarse al exportar el archivo.

Disponibilidad: ActionScript 1.0; Flash Player 7

Ejemplo

El ejemplo siguiente crea un campo de texto que muestra el número actual de fotogramas por segundo que muestra video1.flv.

```
var connection_nc:NetConnection = new NetConnection();
```

```
connection_nc.connect(null);
var stream_ns:NetStream = new NetStream(connection_nc);
my_video.attachVideo(stream_ns);
stream_ns.play("video1.flv");

this.createTextField("fps_txt", this.getNextHighestDepth(), 10, 10, 50,
  22);
fps_txt.autoSize = true;
var fps_interval:Number = setInterval(displayFPS, 500, stream_ns);
function displayFPS(my_ns:NetStream) {
  fps_txt.text = "currentFps (frames per second):
  "+Math.floor(my_ns.currentFps);
}
```

Si el archivo SWF incluye un componente de la versión 2, utilice la clase DepthManager de los componentes de la versión 2 en lugar del método `MovieClip.getNextHighestDepth()` utilizado en este ejemplo.

NetStream, constructor

```
public NetStream(connection:NetConnection)
```

Crea un flujo que puede utilizarse para reproducir archivos FLV a través del objeto NetConnection especificado.

Disponibilidad: ActionScript 1.0; Flash Player 7

Parámetros

`connection``:NetConnection` - Un objeto NetConnection.

Ejemplo

El código siguiente construye en primer lugar un nuevo objeto NetConnection, `connection_nc`, y lo utiliza para construir un nuevo objeto NetStream llamado `stream_ns`. Seleccione Nuevo vídeo en el menú de opciones del panel Biblioteca para crear una instancia de objeto de vídeo y asígnele el nombre de instancia `my_video`.

```
var connection_nc:NetConnection = new NetConnection();
connection_nc.connect(null);
var stream_ns:NetStream = new NetStream(connection_nc);
my_video.attachVideo(stream_ns);
stream_ns.play("video1.flv");
```

Véase también

`NetConnection`, `attachVideo (método Video.attachVideo)`

onMetaData (controlador NetStream.onMetaData)

```
onMetaData = function(infoObject:Object) {}
```

Se invoca cuando Flash Player recibe la información incorporada en el archivo FLV que se está reproduciendo.

La utilidad Flash Video Exporter (versión 1.1 o posterior) incorpora la duración del vídeo, la fecha de creación, velocidad de datos y otra información en el propio archivo de vídeo. Los distintos codificadores de vídeo incorporan diferentes conjuntos de metadatos.

Este controlador se activa después de una llamada al método `NetStream.play()`, pero antes que avance la cabeza lectora de vídeo.

En muchos casos el valor de duración incorporado en los metadatos del archivo FLV es sólo un valor aproximado sin llegar a ser exacto. Es decir, no coincide siempre con el valor de la propiedad `NetStream.time` cuando la cabeza lectora se encuentra al final del archivo de vídeo.

Disponibilidad: ActionScript 1.0; Flash Player 7

Parámetros

`infoObject``:Object` - Un objeto que contiene una propiedad para cada elemento de metadatos.

Ejemplo

El código en este ejemplo comienza creando nuevos objetos NetConnection y NetStream. Después define el controlador `onMetaData()` para el objeto NetStream. El controlador utiliza cada propiedad con nombre en el objeto infoObject que ha recibido e imprime el nombre y el valor de la propiedad.

```
var nc:NetConnection = new NetConnection();
nc.connect(null);
var ns:NetStream = new NetStream(nc);

ns.onMetaData = function(infoObject:Object) {
  for (var propName:String in infoObject) {
    trace(propName + " = " + infoObject[propName]);
  }
};

ns.play("http://www.helpexamples.com/flash/video/water.flv");
```

De este modo se puede ver la información siguiente:

```
canSeekToEnd = true
 videocodecid = 4
 framerate = 15
 videodatarate = 400
```

```
height = 215
width = 320
duration = 7.347
```

La lista de propiedades variará en función del software utilizado para codificar el archivo FLV.

Véase también

time (propiedad NetStream.time), play (método NetStream.play), NetConnection

onStatus (controlador NetStream.onStatus)

```
onStatus = function(infoObject:Object) {}
```

Se invoca cada vez que se publica un cambio de estado o error para el objeto NetStream. Si desea responder a este controlador de eventos, deberá crear una función que procese el objeto de información.

El objeto de información tiene una propiedad code que contiene una cadena que describe el resultado del controlador onStatus, así como una propiedad level que contiene una cadena que puede indicar el estado o un error.

Además de este controlador onStatus, Flash también ofrece una función "súper" denominada `System.onStatus`. Si se invoca onStatus para un objeto concreto y no hay ninguna función asignada para responder a él, Flash procesará una función asignada a `System.onStatus` en el caso de que exista.

Los siguientes eventos le notifican cuándo tienen lugar determinadas actividades NetStream.

Propiedad code	Propiedad level	Significado
`NetStream.Buffer.Empty`	status	No se están recibiendo datos con la velocidad suficiente para llenar el búfer. Se interrumpirá el flujo de datos hasta que se vuelva a llenar el búfer, momento en el cual se enviará un mensaje `NetStream.Buffer.Full` y se iniciará de nuevo la reproducción del flujo.
`NetStream.Buffer.Full`	status	El búfer está lleno y va a comenzar la reproducción del flujo.
`NetStream.Buffer.Flush`	status	Se ha finalizado el envío de datos y se vaciará el búfer.
`NetStream.Play.Start`	status	Se ha iniciado la reproducción.
`NetStream.Play.Stop`	status	Se ha detenido la reproducción.

Propiedad code	Propiedad level	Significado
`NetStream.Play.StreamNotFound`	error	No se encuentra el archivo FLV pasado al método `play()`.
`NetStream.Seek.InvalidTime`	error	En el caso de en vídeo descargado de forma progresiva, el usuario ha intentado buscar o reproducir un punto posterior a los datos de vídeo descargados hasta ese momento o más allá del final del vídeo después de haber descargado todo el archivo. La propiedad `message.details` contiene un código de tiempo que indica la última posición válida que puede buscar el usuario.
`NetStream.Seek.Notify`	status	La operación de búsqueda ha finalizado.

Si siempre aparecen errores relacionados con el búfer, deberá intentar cambiar el búfer utilizando el método `NetStream.setBufferTime()`.

Disponibilidad: ActionScript 1.0; Flash Player 6

Parámetros

`infoObject``:Object` - Un parámetro definido de acuerdo con el mensaje de estado o mensaje de error.

Ejemplo

El ejemplo siguiente muestra datos sobre el flujo en el panel Salida:

```
var connection_nc:NetConnection = new NetConnection();
connection_nc.connect(null);
var stream_ns:NetStream = new NetStream(connection_nc);
my_video.attachVideo(stream_ns);
stream_ns.play("video1.flv");
stream_ns.onStatus = function(infoObject:Object) {
     trace("NetStream.onStatus called: ("+getTimer()+" ms)");
     for (var prop in infoObject) {
       trace("\t"+prop+":\t"+infoObject[prop]);
     }
     trace("");
};
```

Véase también

setBufferTime (método NetStream.setBufferTime), onStatus (controlador System.onStatus)

pause (método NetStream.pause)

```
public pause([flag:Boolean]) : Void
```

Realiza una pausa o reanuda la reproducción de un flujo.

La primera vez que llame a este método (sin enviar ningún parámetro), se realizará una pausa en la reproducción; la siguiente vez, se reanudará la reproducción. Puede que desee asociar este método a un botón de manera que el usuario pueda presionarlo para hacer una pausa o reanudar la reproducción.

Disponibilidad: ActionScript 1.0; Flash Player 7

Parámetros

flag:Boolean [opcional] - Valor booleano que especifica si hace una pausa en la reproducción (true) o se reanuda (false). Si omite este parámetros, NetStream.pause() actúa como un conmutador: la primera vez que se llama en un flujo determinado, hace una pausa en la reproducción, y la siguiente vez, la reanuda.

Ejemplo

El ejemplo siguiente ilustra algunos usos de este método:

```
my_ns.pause(); // pauses play first time issued
my_ns.pause(); // resumes play
my_ns.pause(false); // no effect, play continues
my_ns.pause(); // pauses play
```

Véase también

close (método NetStream.close), play (método NetStream.play)

play (método NetStream.play)

```
public play(name:Object, start:Number, len:Number, reset:Object) : Void
```

Comienza la reproducción de un archivo de vídeo externo (FLV). Para ver datos de vídeo, deberá llamar a un método Video.attachVideo(); el sonido que se reproduzca sin interrupción junto al vídeo, o un archivo FLV que contenga sólo audio, se reproducirá automáticamente.

Si desea controlar el audio asociado a un archivo FLV, puede utilizar `MovieClip.attachAudio()` para dirigir el audio a un clip de película; posteriormente, podrá crear un objeto Sound para controlar algunos aspectos del audio. Para más información, consulte `MovieClip.attachAudio()`.

Si no se encuentra el archivo FLV, se invocará el controlador de eventos `NetStream.onStatus`. Si desea detener un flujo que se está reproduciendo actualmente, utilice `NetStream.close()`.

Puede reproducir archivos FLV locales almacenados en el mismo directorio que el archivo SWF o en un subdirectorio; no podrá desplazarse a un directorio de nivel superior. Por ejemplo, si el archivo SWF está situado en un directorio denominado /training y desea reproducir un vídeo almacenado en el directorio /training/videos, deberá utilizar la siguiente sintaxis:

```
my_ns.play("videos/videoName.flv");
```

Para reproducir un vídeo almacenado en el directorio /training, deberá utilizar la siguiente sintaxis:

```
my_ns.play("videoName.flv");
```

Cuando utilice este método puede usar el modelo de seguridad de Flash Player.

En Flash Player 8:

- No se admite NetStream.play() si el archivo SWF que realiza la llamada se encuentra en el entorno limitado local con sistema de archivos y el recurso no es local.
- Para acceder al entorno limitado de red desde el entorno limitado local de confianza o local con acceso a la red se necesitan permisos del sitio Web a través de un archivo de política de varios dominios.

Para más información, consulte las referencias siguientes:

- Capitulo 17, "Aspectos básicos de la seguridad", de *Aprendizaje de ActionScript 2.0 en Flash*
- El documento técnico sobre la seguridad de Flash Player 8 en http://www.macromedia.com/go/fp8_security
- El documento técnico sobre interfaces API relativas a seguridad de Flash Player 8 en http://www.macromedia.com/go/fp8_security_apis

Disponibilidad: ActionScript 1.0; Flash Player 7

Parámetros

name:Object - El nombre, entre comillas, del archivo FLV que se va a reproducir. Se admiten los formatos http:// y file://; la ubicación file:// es siempre relativa al ubicación del archivo SWF.

start:Number -

len:Number -

reset:Object -

Ejemplo

El ejemplo siguiente ilustra algunos modos de utilizar el comando `NetStream.play()`. Es posible reproducir un archivo que se encuentre en el equipo de otro usuario. El directorio joe_user es un subdirectorio de la ubicación en que está almacenado el archivo SWF. Además, puede reproducir un archivo que se encuentre en un servidor:

```
// Play a file that is on the user's computer.
my_ns.play("file://joe_user/flash/videos/lectureJune26.flv");

// Play a file on a server.
my_ns.play("http://someServer.someDomain.com/flash/video/orientation.flv");
```

Véase también

`attachAudio (método MovieClip.attachAudio)`, `close (método NetStream.close)`, `onStatus (controlador NetStream.onStatus)`, `pause (método NetStream.pause)`, `attachVideo (método Video.attachVideo)`

seek (método NetStream.seek)

```
public seek(offset:Number) : Void
```

Busca el fotograma clave más cercano al número de segundos especificado desde el comienzo del flujo. La reproducción del flujo se reanuda cuando llega a la ubicación especificada del flujo.

Disponibilidad: ActionScript 1.0; Flash Player 7

Parámetros

offset:Number - El valor de tiempo aproximado, en segundos, de desplazamiento en un archivo FLV. La cabeza lectora se mueve al número de fotogramas clave del vídeo más cercano a *numberOfSeconds* .

- Para volver al principio del flujo, pase 0 para *numberOfSeconds* .
- Para hacer una búsqueda hacia delante desde el principio del flujo, pase el número de segundos que desea avanzar. Por ejemplo, para situar la cabeza lectora a 15 segundos del principio, utilice *my_ns* .seek(15).
- Para hacer una búsqueda relativa a la posición actual, pase *my_ns* .time + *n* o *my_ns* .time - *n* para buscar *n* segundos hacia adelante o hacia atrás, respectivamente, desde la posición actual. Por ejemplo, para rebobinar 20 segundos desde la posición actual, utilice my_ns.seek(my_ns.time - 20)..

La ubicación exacta en la que busca un vídeo cambiará en función de la configuración de fotogramas por segundo a que se exportó. Por tanto, si el mismo vídeo se exporta a 6 fps y 30 fps, buscará en dos ubicaciones distintas si utiliza, por ejemplo, my_ns.seek(15) para ambos objetos de vídeo.

Ejemplo

El ejemplo siguiente ilustra algunos modos de utilizar el comando NetStream.seek(). Es posible regresar al comienzo del vídeo, desplazarse hasta un punto ubicado a 30 segundos del comienzo del vídeo y ir hacia atrás 3 minutos a partir del punto actual:

```
// Return to the beginning of the stream
my_ns.seek(0);

// Move to a location 30 seconds from the beginning of the stream
my_ns.seek(30);

// Move backwards three minutes from current location
my_ns.seek(my_ns.time - 180);
```

Véase también

, time (propiedad NetStream.time)

setBufferTime (método NetStream.setBufferTime)

```
public setBufferTime(bufferTime:Number) : Void
```

Especifica el tiempo que deben almacenarse en el búfer los mensajes antes de comenzar a mostrar el flujo. Por ejemplo, si desea asegurarse de que los 15 primeros segundos del flujo se reproducen sin interrupción, establezca *numberOfSeconds* con el valor 15; Flash comenzará a reproducir el flujo sólo después de que se acumulen 15 segundos de datos en el búfer.

Disponibilidad: ActionScript 1.0; Flash Player 7

Parámetros

bufferTime:Number - El número de segundos de los datos que se van a almacenar en el búfer antes de que Flash los muestre. El valor predeterminado es 0,1 (una décima de segundo).

Ejemplo

Consulte el ejemplo de NetStream.bufferLength.

Véase también

bufferLength (propiedad NetStream.bufferLength), bufferTime (propiedad NetStream.bufferTime)

time (propiedad NetStream.time)

```
public time : Number [read-only]
```

La posición, en segundos, de la cabeza lectora.

Disponibilidad: ActionScript 1.0; Flash Player 7

Ejemplo

El ejemplo siguiente muestra la posición actual de la cabeza lectora en un campo de texto creado de forma dinámica llamado time_txt. Seleccione Nuevo vídeo en el menú de opciones del panel Biblioteca para crear una instancia de objeto de vídeo y asígnele el nombre de instancia my_video. Cree un nuevo objeto de vídeo llamado my_video. Añada el siguiente código ActionScript al archivo FLA o AS:

```
var connection_nc:NetConnection = new NetConnection();
connection_nc.connect(null);
var stream_ns:NetStream = new NetStream(connection_nc);
my_video.attachVideo(stream_ns);
stream_ns.play("video1.flv");
//
stream_ns.onStatus = function(infoObject:Object) {
  statusCode_txt.text = infoObject.code;
```

```
};

this.createTextField("time_txt", this.getNextHighestDepth(), 10, 10, 100,
  22);
time_txt.text = "LOADING";

var time_interval:Number = setInterval(checkTime, 500, stream_ns);
function checkTime(my_ns:NetStream) {
  var ns_seconds:Number = my_ns.time;
  var minutes:Number = Math.floor(ns_seconds/60);
  var seconds = Math.floor(ns_seconds%60);
  if (seconds<10) {
  seconds = "0"+seconds;
  }
  time_txt.text = minutes+":"+seconds;
}
```

Si el archivo SWF incluye un componente de la versión 2, utilice la clase DepthManager de los componentes de la versión 2 en lugar del método `MovieClip.getNextHighestDepth()` utilizado en este ejemplo.

Véase también

`bufferLength (propiedad NetStream.bufferLength)`, `bytesLoaded (propiedad NetStream.bytesLoaded)`

Number

```
Object
  |
  +-Number
```

`public class` ***Number***
`extends Object`

La clase Number es un objeto envolvente sencillo para el tipo de datos Number. Puede manipular valores numéricos simples utilizando los métodos y propiedades asociados a la clase Number. Esta clase es idéntica a la clase Number de JavaScript.

Las propiedades de la clase Number son estáticas, lo que significa que no necesita un objeto para utilizarlas, por lo que no es preciso que utilice el constructor.

En el siguiente ejemplo, se llama al método `toString()` de la clase Number, que devuelve la cadena `1234`:

```
var myNumber:Number = new Number(1234);
myNumber.toString();
```

El siguiente ejemplo asigna el valor de la propiedad `MIN_VALUE` a una variable declarada sin utilizar el constructor:

```
var smallest:Number = Number.MIN_VALUE;
```

Disponibilidad: ActionScript 1.0; Flash Player 5

Resumen de propiedades

Modificadores	Propiedad	Descripción
`static`	`MAX_VALUE:Number`	El número más grande que puede representarse (doble precisión IEEE-754).
`static`	`MIN_VALUE:Number`	El número más pequeño que puede representarse (doble precisión IEEE-754).
`static`	`NaN:Number`	El valor IEEE-754 que representa a Not A Number (`NaN`, no es un número).
`static`	`NEGATIVE_INFINITY:Number`	Especifica el valor IEEE-754 que representa el infinito negativo.
`static`	`POSITIVE_INFINITY:Number`	Especifica el valor IEEE-754 que representa el infinito positivo.

Propiedades heredadas de la clase Object

`constructor (propiedad Object.constructor), __proto__ (propiedad Object.__proto__), prototype (propiedad Object.prototype), __resolve (propiedad Object.__resolve)`

Resumen de constructores

Firma	Descripción
`Number(num:Object)`	Crea un nuevo objeto Number.

Resumen de métodos

Modificadores	Firma	Descripción
	`toString(radix:Number) : String`	Devuelve la representación de cadena del objeto Number especificado (*`myNumber`*).
	`valueOf() : Number`	Devuelve el tipo de valor simple del objeto Number especificado.

addProperty (método Object.addProperty), hasOwnProperty (método Object.hasOwnProperty), isPropertyEnumerable (método Object.isPropertyEnumerable), isPrototypeOf (método Object.isPrototypeOf), registerClass (método Object.registerClass), toString (método Object.toString), unwatch (método Object.unwatch), valueOf (método Object.valueOf), watch (método Object.watch)

MAX_VALUE (propiedad Number.MAX_VALUE)

```
public static MAX_VALUE : Number
```

El número más grande que puede representarse (doble precisión IEEE-754). Este número es aproximadamente 1.79e+308.

Disponibilidad: ActionScript 1.0; Flash Player 5

Ejemplo

El siguiente código ActionScript muestra el número más grande y el más pequeño que puede representarse en el panel Salida.

```
trace("Number.MIN_VALUE = "+Number.MIN_VALUE);
trace("Number.MAX_VALUE = "+Number.MAX_VALUE);
```

Este código muestra los siguientes valores:

```
Number.MIN_VALUE = 4.94065645841247e-324
Number.MAX_VALUE = 1.79769313486232e+308
```

MIN_VALUE (propiedad Number.MIN_VALUE)

```
public static MIN_VALUE : Number
```

El número más pequeño que puede representarse (doble precisión IEEE-754). Este número es aproximadamente 5e-324.

Disponibilidad: ActionScript 1.0; Flash Player 5

Ejemplo

El siguiente código ActionScript muestra el número más grande y el más pequeño que puede representarse en el panel Salida.

```
trace("Number.MIN_VALUE = "+Number.MIN_VALUE);
trace("Number.MAX_VALUE = "+Number.MAX_VALUE);
```

Este código muestra los siguientes valores:

```
Number.MIN_VALUE = 4.94065645841247e-324
Number.MAX_VALUE = 1.79769313486232e+308
```

NaN (propiedad Number.NaN)

```
public static NaN : Number
```

El valor IEEE-754 que representa a Not A Number (`NaN`, no es un número).

Disponibilidad: ActionScript 1.0; Flash Player 5

Véase también

NEGATIVE_INFINITY (propiedad Number.NEGATIVE_INFINITY)

```
public static NEGATIVE_INFINITY : Number
```

Especifica el valor IEEE-754 que representa el infinito negativo. El valor de esta propiedad es el mismo que el de la constante `-Infinity`.

Infinito negativo es un valor numérico especial que se devuelve cuando una operación o función matemática devuelve un valor negativo mayor de lo que es posible representar.

Disponibilidad: ActionScript 1.0; Flash Player 5

Ejemplo

Este ejemplo compara el resultado de dividir los siguientes valores.

```
var posResult:Number = 1/0;
if (posResult == Number.POSITIVE_INFINITY) {
  trace("posResult = "+posResult); // output: posResult = Infinity
}
var negResult:Number = -1/0;
if (negResult == Number.NEGATIVE_INFINITY) {
  trace("negResult = "+negResult); // output: negResult = -Infinity
```

Constructor Number

```
public Number(num:Object)
```

Crea un nuevo objeto Number. El nuevo constructor `new Number` se utiliza principalmente como marcador de posición. Un objeto Number no es lo mismo que la función `Number()` que convierte un parámetro en un valor simple.

Disponibilidad: ActionScript 1.0; Flash Player 5

Parámetros

num:Object - El valor numérico del objeto Number que se crea o un valor para convertir a un número. Si no se especifica `value`, el valor predeterminado es 0.

Ejemplo

El código siguiente construye nuevos objetos Number:

```
var n1:Number = new Number(3.4);
var n2:Number = new Number(-10);
```

Véase también

`toString (método Number.toString)`, `valueOf (método Number.valueOf)`

POSITIVE_INFINITY (propiedad Number.POSITIVE_INFINITY)

```
public static POSITIVE_INFINITY : Number
```

Especifica el valor IEEE-754 que representa el infinito positivo. El valor de esta propiedad es el mismo que el de la constante `Infinity`.

Infinito positivo es un valor numérico especial que se devuelve cuando una operación o función matemática devuelve un valor positivo mayor de lo que es posible representar.

Disponibilidad: ActionScript 1.0; Flash Player 5

Ejemplo

Este ejemplo compara el resultado de dividir los siguientes valores.

```
var posResult:Number = 1/0;
if (posResult == Number.POSITIVE_INFINITY) {
  trace("posResult = "+posResult); // output: posResult = Infinity
}
var negResult:Number = -1/0;
if (negResult == Number.NEGATIVE_INFINITY) {
  trace("negResult = "+negResult); // output: negResult = -Infinity
```

toString (método Number.toString)

```
public toString(radix:Number) : String
```

Devuelve la representación de cadena del objeto Number especificado (*myNumber*).

Disponibilidad: ActionScript 1.0; Flash Player 5

Parámetros

radix:Number - Especifica la base numérica (de 2 a 36) para utilizar en la conversión número a cadena. Si no especifica el parámetro radix, el valor predeterminado es 10.

Valor devuelto

String - Una cadena.

Ejemplo

El ejemplo siguiente utiliza 2 y 8 para el parámetro radix y devuelve una cadena que contiene la representación correspondiente del número 9:

```
var myNumber:Number = new Number(9);
trace(myNumber.toString(2)); // output: 1001
trace(myNumber.toString(8)); // output: 11
```

El ejemplo siguiente da como resultado un valor hexadecimal.

```
var r:Number = new Number(250);
var g:Number = new Number(128);
var b:Number = new Number(114);
var rgb:String = "0x"+ r.toString(16)+g.toString(16)+b.toString(16);
trace(rgb);
// output: rgb:0xFA8072 (Hexadecimal equivalent of the color 'salmon')
```

valueOf (método Number.valueOf)

```
public valueOf() : Number
```

Devuelve el tipo de valor simple del objeto Number especificado.

Disponibilidad: ActionScript 1.0; Flash Player 5

Valor devuelto

Number - Una cadena.

Ejemplo

El ejemplo siguiente da como resultado un valor simple del objeto numSocks.

```
var numSocks = new Number(2);
trace(numSocks.valueOf()); // output: 2
```

Object

`Object`

`public class` *Object*

La clase Object se encuentra en la raíz de la jerarquía de clases de ActionScript. Esta clase contiene un pequeño subconjunto de las funciones proporcionadas por la clase Object de JavaScript.

Disponibilidad: ActionScript 1.0; Flash Player 5

Resumen de propiedades

Modificadores	Propiedad	Descripción
	`constructor:Object`	Referencia a la función constructora para una instancia de objeto determinada.
	`__proto__:Object`	Hace referencia a la propiedad `prototype` de la clase (ActionScript 2.0) o función constructora (ActionScript 1.0) utilizada para crear el objeto.
`static`	`prototype:Object`	Una referencia a la superclase de una clase u objeto de función.
	`__resolve:Object`	Una referencia a una función definida por el usuario que se invoca si el código ActionScript hace referencia a una propiedad o un método no definido.

Resumen de constructores

Firma	Descripción
`Object()`	Crea un objeto Object y almacena una referencia al método constructor del objeto en la propiedad `constructor` del objeto.

Resumen de métodos

Modificadores	Firma	Descripción
	`addProperty(name:String, getter:Function, setter:Function) : Boolean`	Crea una propiedad getter/setter.
	`hasOwnProperty(name:String) : Boolean`	Indica si un objeto tiene definida una propiedad especificada.

Modificadores	Firma	Descripción
	`isPropertyEnumerable (name:String) : Boolean`	Indica si existe la propiedad especificada y si es enumerable.
	`isPrototypeOf(theClass:Object) : Boolean`	Indica si hay una instancia de la clase Object en la cadena de prototipo del objeto especificado como un argumento.
`static`	`registerClass(name:String, theClass:Function) : Boolean`	Asocia un símbolo de clip de película a una clase Object de ActionScript.
	`toString() : String`	Convierte el objeto especificado en una cadena y devuelve dicha cadena.
	`unwatch(name:String) : Boolean`	Elimina un punto de observación creado por `Object.watch()`.
	`valueOf() : Object`	Devuelve el valor simple del objeto especificado.
	`watch(name:String, callback:Function, [userData:Object]) : Boolean`	Registra un controlador de eventos que debe invocarse cuando cambie una propiedad especificada de un objeto de ActionScript.

addProperty (método Object.addProperty)

`public addProperty(name:String, getter:Function, setter:Function) : Boolean`

Crea una propiedad getter/setter. Cuando Flash lee una propiedad getter/setter (captador/definidor), invoca la función `get` y el valor devuelto por la función se convierte en un valor de `name`. Cuando Flash escribe una propiedad getter/setter (captador/definidor), invoca la función `set` y le pasa el nuevo valor como parámetro. Si existe una propiedad con ese nombre concreto, la nueva propiedad lo sobrescribe.

La función "get" es una función que carece de parámetros. El valor devuelto puede ser de cualquier tipo. El tipo de valor puede cambiar según la invocación. El valor devuelto se trata como el valor actual de la propiedad.

La función "set" es una función que acepta un parámetro: el nuevo valor de la propiedad. Por ejemplo, si la propiedad `x` se asigna mediante la sentencia `x = 1`, la función set recibirá el parámetro `1` del tipo Number (número). El valor devuelto por la función set no se tiene en cuenta.

Puede añadir propiedades de captador/definidor a los objetos prototipo. Si añade una propiedad de captador/definidor a un objeto prototipo, todas las instancias del objeto que heredan el objeto prototipo heredarán la propiedad de captador/definidor. Esto permite añadir una propiedad de captador/definidor a una ubicación, el objeto prototipo, y aplicarla a todas las instancias de una clase (como si se agregaran métodos a objetos prototipo). Si se invoca una función get/set para una propiedad de getter/setter de un objeto prototipo heredado, la referencia que se pasa a la función getter/setter será el objeto referenciado originalmente, no el objeto prototipo.

Si se invoca incorrectamente, puede que `Object.addProperty()` falle y muestre un error. En la siguiente tabla se describen los errores que pueden tener lugar:

Situación de error	Qué ocurre
`name` no es un nombre de propiedad válido; por ejemplo, una cadena vacía.	Devuelve `false` y la propiedad no se añade.
`getter` no es un objeto de función válido.	Devuelve `false` y la propiedad no se añade.
`setter` no es un objeto de función válido.	Devuelve `false` y la propiedad no se añade.

Disponibilidad: ActionScript 1.0; Flash Player 6

Parámetros

`name``:String` - Una cadena; el nombre de la propiedad del objeto que se va a crear.

`getter``:Function` - La función que se invoca para recuperar el valor de la propiedad; este parámetro es un objeto Function.

`setter``:Function` - La función que se invoca para establecer el valor de la propiedad; este parámetro es un objeto Function. Si se pasa el valor `null` para este parámetro, la propiedad es de sólo lectura.

Valor devuelto

`Boolean` - Valor booleano: `true` si la propiedad se crea correctamente; `false` en caso contrario.

Ejemplo

En el ejemplo siguiente, un objeto tiene dos métodos internos: `setQuantity()` y `getQuantity()`. Cuando se define o recupera, se puede utilizar una propiedad, `bookcount`, para invocar estos métodos. Un tercer método interno, `getTitle()`, devuelve un valor de sólo lectura que está asociado con la propiedad `bookname`. Cuando un script recupera el valor de `myBook.bookcount`, el intérprete de ActionScript invoca automáticamente `myBook.getQuantity()`. Si un script modifica el valor de `myBook.bookcount`, el intérprete invoca `myObject.setQuantity()`. Como la propiedad `bookname` no especifica una función `set`, los intentos de modificar la propiedad `bookname` no se tienen en cuenta.

```
function Book() {
  this.setQuantity = function(numBooks:Number):Void {
  this.books = numBooks;
  };
  this.getQuantity = function():Number {
  return this.books;
  };
  this.getTitle = function():String {
  return "Catcher in the Rye";
  };
  this.addProperty("bookcount", this.getQuantity, this.setQuantity);
  this.addProperty("bookname", this.getTitle, null);
}
var myBook = new Book();
myBook.bookcount = 5;
trace("You ordered "+myBook.bookcount+" copies of "+myBook.bookname);
// output: You ordered 5 copies of Catcher in the Rye
```

Aunque el ejemplo anterior funciona, las propiedades `bookcount` y `bookname` se añaden a cada instancia del objeto `Book`, lo que exige disponer de dos propiedades para cada instancia del objeto. La inclusión de muchas propiedades, como `bookcount` y `bookname`,, en una clase puede conllevar un gran consumo de memoria. Como solución puede agregar las propiedades a `Book.prototype` de modo que las propiedades `bookcount` y `bookname` sólo existan en un lugar. Sin embargo, produce el mismo efecto que el código del ejemplo, que añadió `bookcount` y `bookname` directamente a cada instancia. Si se intenta acceder a alguna de estas propiedades en una instancia Book, la ausencia de la propiedad hará que se ascienda por la cadena de prototipo hasta que se encuentren las versiones definidas en `Book.prototype`. En el ejemplo siguiente se muestra la forma de añadir propiedades a `Book.prototype`:

```
function Book() {}
Book.prototype.setQuantity = function(numBooks:Number):Void {
  this.books = numBooks;
};
Book.prototype.getQuantity = function():Number {
  return this.books;
};
```

```
Book.prototype.getTitle = function():String {
  return "Catcher in the Rye";
};
Book.prototype.addProperty("bookcount", Book.prototype.getQuantity,
  Book.prototype.setQuantity);
Book.prototype.addProperty("bookname", Book.prototype.getTitle, null);
var myBook = new Book();
myBook.bookcount = 5;
trace("You ordered "+myBook.bookcount+" copies of "+myBook.bookname);
```

En el ejemplo siguiente se muestra cómo utilizar las funciones getter y setter implícitas que están disponibles en ActionScript 2.0. En lugar de definir la función `Book` y editar `Book.prototype`, la clase `Book` se define en un archivo externo denominado Book.as. El código siguiente debe encontrarse en un archivo externo separado, denominado Book.as, que contiene esta definición de clase exclusivamente y reside en la ruta de clases de la aplicación Flash:

```
class Book {
  var books:Number;
  function set bookcount(numBooks:Number):Void {
  this.books = numBooks;
  }
  function get bookcount():Number {
  return this.books;
  }
  function get bookname():String {
  return "Catcher in the Rye";
  }
}
```

Luego se puede incluir el código siguiente en un archivo FLA y seguir funcionando de la misma manera que en los ejemplos anteriores:

```
var myBook:Book = new Book();
myBook.bookcount = 5;
trace("You ordered "+myBook.bookcount+" copies of "+myBook.bookname);
```

Véase también

`Sentencia get`, `Sentencia set`

constructor (propiedad Object.constructor)

```
public constructor : Object
```

Referencia a la función constructora para una instancia de objeto determinada. La propiedad `constructor` se asigna automáticamente a todos los objetos cuando se crean utilizando el constructor de la clase Object.

Disponibilidad: ActionScript 1.0; Flash Player 5

Ejemplo

El ejemplo siguiente es una referencia a la función constructora del objeto `myObject`.

```
var my_str:String = new String("sven");
trace(my_str.constructor == String); //output: true
```

Si utiliza el operador `instanceof`, también puede determinar si un objeto pertenece a una clase especificada:

```
var my_str:String = new String("sven");
trace(my_str instanceof String); //output: true
```

Sin embargo, en el ejemplo siguiente la propiedad `Object.constructor` convierte los tipos de datos simples (como el literal de cadena que aparece aquí) en objetos envolventes. El operador `instanceof` no realiza ninguna conversión, como se aprecia en el ejemplo siguiente:

```
var my_str:String = "sven";
trace(my_str.constructor == String); //output: true
trace(my_str instanceof String); //output: false
```

Véase también

`Operador instanceof`

hasOwnProperty (método Object.hasOwnProperty)

```
public hasOwnProperty(name:String) : Boolean
```

Indica si un objeto tiene definida una propiedad especificada. Este método devuelve `true` si el objeto de destino tiene una propiedad que coincide con la cadena especificada por el parámetro `name`, y `false` en caso contrario. Este método no comprueba la cadena de prototipo del objeto y devuelve `true` solamente cuando la propiedad existe en el propio objeto.

Disponibilidad: ActionScript 1.0; Flash Player 6

Parámetros

`name``:String` -

Valor devuelto

`Boolean` - Valor booleano: `true` si el objeto de destino tiene la propiedad especificada por el parámetro `name`, y `false` en caso contrario.

isPropertyEnumerable (método Object.isPropertyEnumerable)

```
public isPropertyEnumerable(name:String) : Boolean
```

Indica si existe la propiedad especificada y si es enumerable. Si su valor es `true`, la propiedad existe y puede enumerarse en un bucle for..in. La propiedad debe existir en el objeto de destino porque este método no comprueba la cadena de prototipo del objeto de destino.

Aunque las propiedades que se crean son enumerables, las propiedades incorporadas no suelen ser enumerables.

Disponibilidad: ActionScript 1.0; Flash Player 6

Parámetros

name:String -

Valor devuelto

`Boolean` - Valor booleano: `true` si la propiedad especificada por el parámetro `name` es enumerable.

Ejemplo

En el ejemplo siguiente se crea un objeto genérico, se le añade una propiedad y luego se comprueba si el objeto es enumerable. A modo de comparación, en el ejemplo también se muestra que una propiedad incorporada, `Array.length`, no es enumerable.

```
var myObj:Object = new Object();
myObj.prop1 = "hello";
trace(myObj.isPropertyEnumerable("prop1")); // Output: true

var myArray = new Array();
trace(myArray.isPropertyEnumerable("length")); // Output: false
```

Véase también

`Sentencia for..in`

isPrototypeOf (método Object.isPrototypeOf)

```
public isPrototypeOf(theClass:Object) : Boolean
```

Indica si hay una instancia de la clase Object en la cadena de prototipo del objeto especificado como un argumento. Este método devuelve `true` si el objeto está presente en la cadena de prototipo del objeto especificado por el parámetro `theClass`. Devuelve `false` no sólo cuando el objeto de destino no está presente en la cadena de prototipo del objeto `theClass`, sino también cuando el argumento `theClass` no es un objeto.

Disponibilidad: ActionScript 1.0; Flash Player 6

Parámetros

`theClass``:Object` -

Valor devuelto

`Boolean` - Valor booleano: `true` si el objeto está presente en la cadena de prototipo del objeto especificado por el parámetro `theClass`, y `false` en caso contrario.

Object, constructor

```
public Object()
```

Crea un objeto Object y almacena una referencia al método constructor del objeto en la propiedad `constructor` del objeto.

Disponibilidad: ActionScript 1.0; Flash Player 5

Ejemplo

En el ejemplo siguiente se crea un objeto genérico denominado myObject:

```
var myObject:Object = new Object();
```

__proto__ (propiedad Object.__proto__)

```
public __proto__ : Object
```

Hace referencia a la propiedad `prototype` de la clase (ActionScript 2.0) o función constructora (ActionScript 1.0) utilizada para crear el objeto. La propiedad `__proto__` se asigna automáticamente a todos los objetos cuando se crean. El intérprete de ActionScript utiliza la propiedad `__proto__` para acceder a la propiedad `prototype` de la clase o función constructor del objeto con el fin de averiguar las propiedades y métodos que hereda el objeto de su superclase.

Disponibilidad: ActionScript 1.0; Flash Player 5

Ejemplo

En el ejemplo siguiente se crea una clase denominada Shape y una subclase de Shape con el nombre Circle.

```
// Shape class defined in external file named Shape.as
class Shape {
  function Shape() {}
}

// Circle class defined in external file named Circle.as
class Circle extends Shape{
  function Circle() {}
}
```

La clase Circle se puede utilizar para crear dos instancias de Circle:

```
var oneCircle:Circle = new Circle();
var twoCircle:Circle = new Circle();
```

En las siguientes sentencias trace se muestra que la propiedad `__proto_` de ambas instancias hace referencia a la propiedad `prototype` de la clase Circle.

```
trace(Circle.prototype == oneCircle.__proto__); // Output: true
trace(Circle.prototype == twoCircle.__proto__); // Output: true
```

Véase también

`prototype (propiedad Object.prototype)`

prototype (propiedad Object.prototype)

```
public static prototype : Object
```

Una referencia a la superclase de una clase u objeto de función. La propiedad `prototype` se crea y asocia automáticamente a cualquier clase u objeto de función creado. Esta propiedad se considera estática porque es específica de la clase o función que ha creado. Por ejemplo, si crea una clase personalizada, todas las instancias de la clase compartirán el valor de la propiedad `prototype` y sólo será accesible como propiedad de clase. Aunque las instancias de la clase personalizada no tienen acceso directo a la propiedad `prototype`, pueden acceder a ella a través de la propiedad `__proto__`.

Disponibilidad: ActionScript 1.0; Flash Player 6

Ejemplo

En el ejemplo siguiente se crea una clase denominada Shape y una subclase de Shape con el nombre Circle.

```
// Shape class defined in external file named Shape.as
class Shape {
  function Shape() {}
}

// Circle class defined in external file named Circle.as
class Circle extends Shape{
  function Circle() {}
}
```

La clase Circle se puede utilizar para crear dos instancias de Circle:

```
var oneCircle:Circle = new Circle();
var twoCircle:Circle = new Circle();
```

En la siguiente sentencia trace se muestra que la propiedad `prototype` de la clase Circle señala a su superclase Shape. El identificador `Shape` hace referencia a la función constructora de la clase Shape.

```
trace(Circle.prototype.constructor == Shape); // Output: true
```

La siguiente sentencia trace muestra cómo utilizar juntas las propiedades `prototype` y `__proto__` para subir dos niveles en la jerarquía de herencia (o cadena de prototipo). La propiedad `Circle.prototype.__proto__` contiene una referencia a la superclase de la clase Shape.

```
trace(Circle.prototype.__proto__ == Shape.prototype); // Output: true
```

Véase también

`__proto__ (propiedad Object.__proto__)`

registerClass (método Object.registerClass)

```
public static registerClass(name:String, theClass:Function) : Boolean
```

Asocia un símbolo de clip de película a una clase Object de ActionScript. Si un símbolo no existe, Flash crea una asociación entre un identificador de cadena y una clase de objeto.

Cuando se coloca una instancia del símbolo de clip de película especificado en la línea de tiempo, ésta se registra en la clase especificada por el parámetro `theClass`, no en la clase MovieClip.

Cuando se crea una instancia del símbolo de clip de película especificado mediante `MovieClip.attachMovie()` o `MovieClip.duplicateMovieClip()`, ésta se registra en la clase especificada por `theClass`, no en la clase MovieClip. Si `theClass` tiene el valor `null`, este método elimina cualquier definición de clase de ActionScript asociada al símbolo de clip de película o identificador de clase especificado. En el caso de símbolos de clip de película, las instancias existentes del clip de película permanecerán sin cambios, aunque las nuevas instancias del símbolo se asociarán a la clase predeterminada MovieClip.

Si ya hay un símbolo registrado en una clase, este método lo reemplazará por el nuevo registro.

Si la línea de tiempo coloca una instancia de clip de película o ésta se crea utilizando `attachMovie()` o `duplicateMovieClip()`, ActionScript invocará el constructor de la clase correspondiente con la palabra clave `this` apuntando al objeto. La función constructora se invoca sin parámetros.

Si utiliza este método para registrar un clip de película en una clase de ActionScript distinta de MovieClip, el símbolo de clip de película no heredará los métodos, propiedades y eventos de la clase MovieClip incorporada, a no ser que incluya la clase MovieClip en una cadena de prototipo de la nueva clase. El siguiente código crea una nueva clase de ActionScript denominada `theClass` que hereda las propiedades de la clase MovieClip:

```
theClass.prototype = new MovieClip();
```

Disponibilidad: ActionScript 1.0; Flash Player 6

Parámetros

`name``:String` - Cadena; el identificador de vinculación del símbolo del clip de película o el identificador de cadena de la clase ActionScript.

`theClass``:Function` - Una referencia a la función constructora de la clase ActionScript o `null` para anular el registro del símbolo.

Valor devuelto

`Boolean` - Valor booleano: si el registro de la clase se realiza correctamente, se devuelve el valor `true`; `false` en caso contrario.

Véase también

`attachMovie (método MovieClip.attachMovie)`, `duplicateMovieClip (método MovieClip.duplicateMovieClip)`

__resolve (propiedad Object.__resolve)

```
public __resolve : Object
```

Una referencia a una función definida por el usuario que se invoca si el código ActionScript hace referencia a una propiedad o un método no definido. Si el código ActionScript hace referencia a una propiedad o método no definido de un objeto, Flash Player determinará si se ha definido la propiedad `__resolve` del objeto. Si se ha definido `__resolve`, se ejecutará la función a la que hace referencia y se pasará a ella el nombre de la propiedad o método no definido. Esto permite proporcionar de manera programática valores para propiedades no definidas y sentencias para métodos no definidos y hacer que parezca como si las propiedades o los métodos estuvieran realmente definidos. Esta propiedad resulta útil para permitir una comunicación altamente transparente entre el cliente y el servidor y es la forma recomendada de invocar métodos de servidor.

Disponibilidad: ActionScript 1.0; Flash Player 6

Ejemplo

Los ejemplos siguientes forman una secuencia basada en el primer ejemplo e ilustran cinco usos distintos de la propiedad `__resolve`. Para facilitar la comprensión, las sentencias clave que difieren del uso anterior aparecen en negrita.

Sintaxis 1: en el ejemplo siguiente, `__resolve` se utiliza para crear un objeto en el que cada propiedad no definida devuelve el valor `"Hello, world!"`.

```
// instantiate a new object
var myObject:Object = new Object();

// define the __resolve function
myObject.__resolve = function (name) {
  return "Hello, world!";
};
trace (myObject.property1); // output: Hello, world!
trace (myObject.property2); // output: Hello, world!
```

Sintaxis 2: en el ejemplo siguiente se utiliza `__resolve` como *functor*; es decir, como función que genera funciones. Mediante `__resolve`, las llamadas de método no definido se redirigen a una función genérica denominada `myFunction`.

```
// instantiate a new object
var myObject:Object = new Object();

// define a function for __resolve to call
myObject.myFunction = function (name) {
  trace("Method " + name + " was called");
};

// define the __resolve function
```

```
myObject.__resolve = function (name) {
    return function () { this.myFunction(name); };
};

// test __resolve using undefined method names
myObject.someMethod(); // output: Method someMethod was called
myObject.someOtherMethod(); //output: Method someOtherMethod was called
```

Sintaxis 3: el ejemplo siguiente se basa en el anterior, pero ofrece la posibilidad de guardar en el caché los métodos resueltos. Cuando los métodos se guardan en el caché, sólo se llama a `__resolve` una vez por cada método de interés. Esto permite la *construcción diferida* de métodos de objeto. La construcción diferida es una técnica de optimización que aplaza la creación, o *construcción*, de métodos hasta el momento en que se utiliza un método por primera vez.

```
// instantiate a new object
var myObject:Object = new Object();
// define a function for __resolve to call
myObject.myFunction = function(name) {
  trace("Method "+name+" was called");
};
// define the __resolve function
myObject.__resolve = function(name) {
  trace("Resolve called for "+name); // to check when __resolve is called
  // Not only call the function, but also save a reference to it
  var f:Function = function () {
    this.myFunction(name);
  };
  // create a new object method and assign it the reference
  this[name] = f;
  // return the reference
  return f;
};
// test __resolve using undefined method names
// __resolve will only be called once for each method name
myObject.someMethod(); // calls __resolve
myObject.someMethod(); // does not call __resolve because it is now defined
myObject.someOtherMethod(); // calls __resolve
myObject.someOtherMethod(); // does not call __resolve, no longer undefined
```

Sintaxis 4: en el ejemplo siguiente, basado en el anterior, se reserva un nombre de método, `onStatus()`, para uso local con el fin de evitar que se resuelva de la misma forma que otras propiedades no definidas. El código que se añade aparece en negrita.

```
// instantiate a new object
var myObject:Object = new Object();
// define a function for __resolve to call
myObject.myFunction = function(name) {
  trace("Method "+name+" was called");
};
```

```
// define the __resolve function
myObject.__resolve = function(name) {
  // reserve the name "onStatus" for local use
  if (name == "onStatus") {
    return undefined;
  }
  trace("Resolve called for "+name); // to check when __resolve is called
  // Not only call the function, but also save a reference to it
  var f:Function = function () {
    this.myFunction(name);
  };
  // create a new object method and assign it the reference
  this[name] = f;
  // return the reference
  return f;
};
// test __resolve using the method name "onStatus"
trace(myObject.onStatus("hello"));
// output: undefined
```

Sintaxis 5: el ejemplo siguiente está basado en el anterior, pero en este caso se crea un functor que acepta parámetros. Además de hacer uso extensivo del objeto arguments, en este ejemplo se utilizan varios métodos de la clase Array.

```
// instantiate a new object
var myObject:Object = new Object();

// define a generic function for __resolve to call
myObject.myFunction = function (name) {
  arguments.shift();
  trace("Method " + name + " was called with arguments: " +
  arguments.join(','));
};

// define the __resolve function
myObject.__resolve = function (name) {
  // reserve the name "onStatus" for local use
  if (name == "onStatus") {
    return undefined;
  }
  var f:Function = function () {
    arguments.unshift(name);
    this.myFunction.apply(this, arguments);
  };
  // create a new object method and assign it the reference
  this[name] = f;
  // return the reference to the function
  return f;
};

// test __resolve using undefined method names with parameters
```

```
myObject.someMethod("hello");
// output: Method someMethod was called with arguments: hello

myObject.someOtherMethod("hello","world");
// output: Method someOtherMethod was called with arguments: hello,world
```

Véase también

`arguments`, `Array`

toString (método Object.toString)

```
public toString() : String
```

Convierte el objeto especificado en una cadena y devuelve dicha cadena.

Disponibilidad: ActionScript 1.0; Flash Player 5

Valor devuelto

`String` - Una cadena.

Ejemplo

En este ejemplo se muestra el valor que se devuelve para toString() en un objeto genérico:

```
var myObject:Object = new Object();
trace(myObject.toString()); // output: [object Object]
```

Este método se puede sustituir para que se devuelva un valor más significativo. En los ejemplos siguientes se muestra que este método se ha sustituido para las clases incorporadas Date, Array y Number:

```
// Date.toString() returns the current date and time
var myDate:Date = new Date();
trace(myDate.toString()); // output: [current date and time]

// Array.toString() returns the array contents as a comma-delimited string
var myArray:Array = new Array("one", "two");
trace(myArray.toString()); // output: one,two

// Number.toString() returns the number value as a string
// Because trace() won't tell us whether the value is a string or number
// we will also use typeof() to test whether toString() works.
var myNumber:Number = 5;
trace(typeof (myNumber)); // output: number
trace(myNumber.toString()); // output: 5
trace(typeof (myNumber.toString())); // output: string
```

En el ejemplo siguiente se muestra cómo sustituir `toString()` en una clase personalizada. Primero cree un archivo de texto denominado *Vehicle.as* que sólo contenga la definición de la clase Vehicle y luego colóquelo en la carpeta Classes dentro de la carpeta Configuration.

```
// contents of Vehicle.as
class Vehicle {
  var numDoors:Number;
  var color:String;
  function Vehicle(param_numDoors:Number, param_color:String) {
  this.numDoors = param_numDoors;
  this.color = param_color;
  }
  function toString():String {
  var doors:String = "door";
  if (this.numDoors > 1) {
    doors += "s";
  }
  return ("A vehicle that is " + this.color + " and has " + this.numDoors +
  " " + doors);
  }
}

// code to place into a FLA file
var myVehicle:Vehicle = new Vehicle(2, "red");
trace(myVehicle.toString());
// output: A vehicle that is red and has 2 doors

// for comparison purposes, this is a call to valueOf()
// there is no primitive value of myVehicle, so the object is returned
// giving the same output as toString().
trace(myVehicle.valueOf());
// output: A vehicle that is red and has 2 doors
```

unwatch (método Object.unwatch)

```
public unwatch(name:String) : Boolean
```

Elimina un punto de observación creado por `Object.watch()`. Este método devuelve el valor `true` si el punto de observación se elimina correctamente, y `false` en caso contrario.

Disponibilidad: ActionScript 1.0; Flash Player 6

Parámetros

`name``:String` - Una cadena; el nombre de la propiedad del objeto que debería dejar de observarse.

Valor devuelto

`Boolean` - Valor booleano: `true` si el punto de observación se elimina correctamente; `false` en caso contrario.

Ejemplo

Consulte el ejemplo de `Object.watch()`.

Véase también

`watch (método Object.watch)`, `addProperty (método Object.addProperty)`

valueOf (método Object.valueOf)

```
public valueOf() : Object
```

Devuelve el valor simple del objeto especificado. Si el objeto no tiene un valor simple, se devuelve el objeto.

Disponibilidad: ActionScript 1.0; Flash Player 5

Valor devuelto

`Object` - El valor simple del objeto especificado o el propio objeto.

Ejemplo

En el ejemplo siguiente se muestra el valor devuelto de valueOf() para un objeto genérico (que no tiene un valor simple) y se compara con el valor devuelto de toString(). En primer lugar, cree un objeto genérico: Después, cree un nuevo objeto Date establecido en February 1, 2004, 8:15 AM. El método toString() devuelve la hora actual con un formato legible. El método valueOf() devuelve el valor simple en milisegundos. Finalmente, cree un nuevo objeto Array que contenga dos elementos sencillos. Tanto toString() como valueOf() devuelven el mismo valor: one,two:

```
// Create a generic object
var myObject:Object = new Object();
trace(myObject.valueOf()); // output: [object Object]
trace(myObject.toString()); // output: [object Object]
```

En los ejemplos siguientes se muestran los valores devueltos para las clases incorporadas Date y Array, y se comparan con los valores devueltos de `Object.toString()`:

```
// Create a new Date object set to February 1, 2004, 8:15 AM
// The toString() method returns the current time in human-readable form
// The valueOf() method returns the primitive value in milliseconds
var myDate:Date = new Date(2004,01,01,8,15);
trace(myDate.toString()); // output: Sun Feb 1 08:15:00 GMT-0800 2004
```

```
trace(myDate.valueOf()); // output: 1075652100000

// Create a new Array object containing two simple elements
// In this case both toString() and valueOf() return the same value: one,two
var myArray:Array = new Array("one", "two");
trace(myArray.toString()); // output: one,two
trace(myArray.valueOf()); // output: one,two
```

En el ejemplo de `Object.toString()` puede encontrar un ejemplo del valor devuelto de `Object.valueOf()` para una clase personalizada que sustituye a `toString()`.

Véase también

`toString (método Object.toString)`

watch (método Object.watch)

```
public watch(name:String, callback:Function, [userData:Object]) : Boolean
```

Registra un controlador de eventos que debe invocarse cuando cambie una propiedad especificada de un objeto de ActionScript. Cuando cambia la propiedad, se invoca el controlador de eventos con `myObject` como objeto contenedor.

Puede utilizar la sentencia `return` de la definición del método `callback` para modificar el valor de la propiedad que está observando. El valor devuelto por el método `callback` se asigna a la propiedad del objeto observado. El valor de devolución que elija dependerá de si desea controlar, modificar o evitar cambios en la propiedad:

- Si simplemente está controlando la propiedad, devuelva el parámetro `newVal`.
- Si está modificando el valor de la propiedad, devuelva su propio valor.
- Si desea evitar cambios en la propiedad, devuelva el parámetro `oldVal`.

Si el método `callback` que usted define no tiene una sentencia `return`, se asignará el valor `undefined.` a la propiedad del objeto observado.

Un punto de observación puede filtrar (o convertir en nula) la asignación de valor mediante la devolución de un `newval` modificado (u `oldval`). Si elimina una propiedad para la que se ha establecido un punto de observación, el punto de observación no desaparecerá. Si posteriormente vuelve a crear la propiedad, el punto de observación seguirá surtiendo efecto. Para eliminar un punto de observación, utilice el método `Object.unwatch`.

Sólo puede registrarse un punto de observación en una propiedad. Las posteriores llamadas a `Object.watch()` en la misma propiedad reemplazan al punto de observación original.

El método `Object.watch()` se comporta de manera similar a la función `Object.watch()` en JavaScript 1.2 y posterior. La diferencia principal es el parámetro `userData` que Flash añade a `Object.watch()` y que Netscape Navigator no admite. Puede pasar el parámetro `userData` al controlador de eventos y utilizarlo en el controlador de eventos.

El método `Object.watch()` no puede observar propiedades getter/setter. Las propiedades getter/setter operan mediante *evaluación diferida* - el valor de la propiedad no se determina hasta que se consulta realmente la propiedad. La evaluación diferida suele ser eficaz porque la propiedad no se actualiza constantemente; por contra, se evalúa cuando es necesaria. No obstante, `Object.watch()` necesita evaluar una propiedad para determinar si debe invocarse la función `callback`. Para trabajar con una propiedad getter/setter, `Object.watch()` necesita evaluar la propiedad constantemente, lo cual no resulta eficaz.

Generalmente, las propiedades predefinidas de ActionScript, como `_x`, `_y`, `_width` y `_height`, son propiedades getter/setter que no pueden observarse con `Object.watch()`.

Disponibilidad: ActionScript 1.0; Flash Player 6

Parámetros

`name``:String`: Una cadena; el nombre de la propiedad del objeto que se va a observar.

`callback``:Function`: La función que se invoca cuando la propiedad observada cambia. Este parámetro es un objeto de función, en lugar de un nombre de función como una cadena. El formato de `callback` es `callback(prop, oldVal, newVal, userData)`.

`userData``:Object` [opcional]: Una parte cualquiera de los datos ActionScript que se pasa al método `callback`. Si se omite el parámetro `userData`, se pasa `undefined` al método callback.

Valor devuelto

`Boolean` : Valor booleano: `true` si el punto de observación se crea correctamente; `false` en caso contrario.

Ejemplo

En el ejemplo siguiente se emplea `watch()` para determinar si la propiedad `speed` supera el límite de velocidad:

```
// Create a new object
var myObject:Object = new Object();

// Add a property that tracks speed
myObject.speed = 0;

// Write the callback function to be executed if the speed property changes
var speedWatcher:Function = function(prop, oldVal, newVal, speedLimit) {
  // Check whether speed is above the limit
```

```
  if (newVal > speedLimit) {
  trace ("You are speeding.");
  }
  else {
  trace ("You are not speeding.");
  }

  // Return the value of newVal.
  return newVal;
}
// Use watch() to register the event handler, passing as parameters:
// - the name of the property to watch: "speed"
// - a reference to the callback function speedWatcher
// - the speedLimit of 55 as the userData parameter
myObject.watch("speed", speedWatcher, 55);

// set the speed property to 54, then to 57
myObject.speed = 54; // output: You are not speeding
myObject.speed = 57; // output: You are speeding

// unwatch the object
myObject.unwatch("speed");
myObject.speed = 54; // there should be no output
```

Véase también

addProperty (método Object.addProperty), unwatch (método Object.unwatch)

Point (flash.geom.Point)

```
Object
  |
  +-flash.geom.Point
```

public class *Point*
extends Object

La clase Point representa una ubicación en un sistema de coordenadas bidimensional, donde *x* corresponde al eje horizontal e *y* al eje vertical.

Mediante el código siguiente se crea un punto en las coordenadas (0,0):

```
var myPoint:Point = new Point();
```

Disponibilidad: ActionScript 1.0; Flash Player 8

Resumen de propiedades

Modificadores	Propiedad	Descripción
	`length:Number`	Longitud del segmento de línea desde (0,0) hasta este punto.
	`x:Number`	Coordenada horizontal del punto.
	`y:Number`	Coordenada vertical del punto.

Propiedades heredadas de la clase Object

`constructor (propiedad Object.constructor), __proto__ (propiedad Object.__proto__), prototype (propiedad Object.prototype), __resolve (propiedad Object.__resolve)`

Resumen de constructores

Firma	Descripción
`Point(x:Number, y:Number)`	Crea un nuevo punto.

Resumen de métodos

Modificadores	Firma	Descripción
	`add(v:Point) : Point`	Añade las coordenadas de otro punto a las coordenadas de éste para crear un nuevo punto.
	`clone() : Point`	Crea una copia de este objeto Point.
`static`	`distance(pt1:Point, pt2:Point) : Number`	Devuelve la distancia entre pt1 y pt2.
	`equals(toCompare:Object) : Boolean`	Determina si dos puntos son iguales.
`static`	`interpolate(pt1:Point, pt2:Point, f:Number) : Point`	Determina un punto entre dos puntos especificados.
	`normalize(length:Number) : Void`	Ajusta a una longitud establecida el tamaño del segmento de línea entre (0,0) y el punto actual.
	`offset(dx:Number, dy:Number) : Void`	Desplaza el objeto Point en la cantidad especificada.
`static`	`polar(len:Number, angle:Number) : Point`	Convierte un par de coordenadas polares en una coordenada cartesiana.

Modificadores	Firma	Descripción
	`subtract(v:Point) : Point`	Resta las coordenadas de otro punto de las coordenadas de éste para crear un nuevo punto.
	`toString() : String`	Devuelve una cadena que contiene los valores de las coordenadas *x* e *y*.

Métodos heredados de la clase Object

`addProperty (método Object.addProperty), hasOwnProperty (método Object.hasOwnProperty), isPropertyEnumerable (método Object.isPropertyEnumerable), isPrototypeOf (método Object.isPrototypeOf), registerClass (método Object.registerClass), toString (método Object.toString), unwatch (método Object.unwatch), valueOf (método Object.valueOf), watch (método Object.watch)`

add (método Point.add)

```
public add(v:Point) : Point
```

Añade las coordenadas de otro punto a las coordenadas de éste para crear un nuevo punto.

Disponibilidad: ActionScript 1.0; Flash Player 8

Parámetros

v:`flash.geom.Point`: El punto que se va a añadir.

Valor devuelto

`flash.geom.Point` - El nuevo punto.

Ejemplo

En el ejemplo siguiente se crea un objeto Point `resultPoint` sumando `point_2` a `point_1`:

```
import flash.geom.Point;
var point_1:Point = new Point(4, 8);
var point_2:Point = new Point(1, 2);
var resultPoint:Point = point_1.add(point_2);
trace(resultPoint.toString()); // (x=5, y=10)
```

clone (método Point.clone)

```
public close() : Point
```

Crea una copia de este objeto Point.

Disponibilidad: ActionScript 1.0; Flash Player 8

Valor devuelto

`flash.geom.Point` - El nuevo objeto Point.

Ejemplo

El ejemplo siguiente crea una copia del objeto Point denominada `clonedPoint` a partir de los valores del objeto `myPoint`. Aunque el objeto `clonedPoint` contiene todos los valores de `myPoint`, no es el mismo objeto.

```
import flash.geom.Point;
var myPoint:Point = new Point(1, 2);
var clonedPoint:Point = myPoint.clone();
trace(clonedPoint.x); // 1
trace(clonedPoint.y); // 2
trace(myPoint.equals(clonedPoint)); // true
trace(myPoint === clonedPoint); // false
```

distance (método Point.distance)

```
public static distance(pt1:Point, pt2:Point) : Number
```

Devuelve la distancia entre pt1 y pt2.

Disponibilidad: ActionScript 1.0; Flash Player 8

Parámetros

pt1:`flash.geom.Point` - El primer punto.

pt2:`flash.geom.Point` - El segundo punto.

Valor devuelto

`Number` - La distancia entre el primer y el segundo punto.

Ejemplo

En el ejemplo se crea `point_1` y `point_2` y luego se determina la distancia entre ellos (`distanceBetween`).

```
import flash.geom.Point;
var point_1:Point = new Point(-5, 0);
var point_2:Point = new Point(5, 0);
var distanceBetween:Number = Point.distance(point_1, point_2);
trace(distanceBetween); // 10
```

equals (método Point.equals)

```
public equals(toCompare:Object) : Boolean
```

Determina si dos puntos son iguales. Para que se consideren iguales, los dos puntos deben tener los mismos valores *x* e *y*.

Disponibilidad: ActionScript 1.0; Flash Player 8

Parámetros

`toCompare``:Object` - El punto que se va a comparar.

Valor devuelto

`Boolean` - Si el objeto equivale a este objeto Point, `true`; si no es igual, `false`.

Ejemplo

En el ejemplo siguiente se determina si los valores de un punto coinciden con los del otro. Si los objetos son iguales, `equals()` no devuelve el mismo resultado que el operador de igualdad (===) más estricto.

```
import flash.geom.Point;
var point_1:Point = new Point(1, 2);
var point_2:Point = new Point(1, 2);
var point_3:Point = new Point(4, 8);
trace(point_1.equals(point_2)); // true
trace(point_1.equals(point_3)); // false
trace(point_1 === point_2); // false
trace(point_1 === point_3); // false
```

interpolate (método Point.interpolate)

```
public static interpolate(pt1:Point, pt2:Point, f:Number) : Point
```

Determina un punto entre dos puntos especificados.

Disponibilidad: ActionScript 1.0; Flash Player 8

Parámetros

pt1:`flash.geom.Point` - El primer punto.

pt2:`flash.geom.Point` - El segundo punto.

f:`Number` - El nivel de interpolación entre dos puntos. Indica la posición del nuevo punto a lo largo de la línea entre pt1 y pt2. Si f=0, se devuelve pt1; si f=1, se devuelve pt2.

Valor devuelto

`flash.geom.Point` - El nuevo punto interpolado.

Ejemplo

En el siguiente ejemplo, el objeto interpolatedPoint (`interpolatedPoint`) se sitúa a mitad de recorrido (50%) entre `point_1` y `point_2`.

```
import flash.geom.Point;
var point_1:Point = new Point(-100, -100);
var point_2:Point = new Point(50, 50);
var interpolatedPoint:Point = Point.interpolate(point_1, point_2, .5);
trace(interpolatedPoint.toString()); // (x=-25, y=-25)
```

length (propiedad Point.length)

```
public length : Number
```

Longitud del segmento de línea desde (0,0) hasta este punto.

Disponibilidad: ActionScript 1.0; Flash Player 8

Ejemplo

En el siguiente ejemplo se crea un objeto Point, `myPoint`, y se determina la longitud de una línea desde (0, 0) hasta ese punto.

```
import flash.geom.Point;
var myPoint:Point = new Point(3,4);
trace(myPoint.length); // 5
```

Véase también

`polar (método Point.polar)`

normalize (método Point.normalize)

```
public normalize(length:Number) : Void
```

Ajusta a una longitud establecida el tamaño del segmento de línea entre (0,0) y el punto actual.

Disponibilidad: ActionScript 1.0; Flash Player 8

Parámetros

length:Number - El valor de ajuste a escala. Por ejemplo, si el punto actual se encuentra en las coordenadas (0,5) y se estandariza a 1, el valor que se devuelve es (0,1).

Ejemplo

En el ejemplo siguiente, la longitud del objeto `normalizedPoint` se extiende de 5 a 10.

```
import flash.geom.Point;
var normalizedPoint:Point = new Point(3, 4);
trace(normalizedPoint.length); // 5
trace(normalizedPoint.toString()); // (x=3, y=4)
normalizedPoint.normalize(10);
trace(normalizedPoint.length); // 10
trace(normalizedPoint.toString()); // (x=6, y=8)
```

Véase también

length (propiedad Point.length)

offset (método Point.offset)

```
public offset(dx:Number, dy:Number) : Void
```

Desplaza el objeto Point en la cantidad especificada. El valor de `dx` se suma al valor original de *x* para crear un valor *x* nuevo. El valor de `dy` se suma al valor original de *y* para crear un valor *y* nuevo.

Disponibilidad: ActionScript 1.0; Flash Player 8

Parámetros

dx:Number - Cantidad que se desplaza la coordenada horizontal, *x*.

dy:Number - Cantidad que se desplaza la coordenada vertical, *y*.

Ejemplo

En el ejemplo siguiente se desplaza la posición de un punto según las cantidades *x* e *y* especificadas.

```
import flash.geom.Point;
var myPoint:Point = new Point(1, 2);
trace(myPoint.toString()); // (x=1, y=2)
myPoint.offset(4, 8);
trace(myPoint.toString()); // (x=5, y=10)
```

Véase también

`add (método Point.add)`

Point, constructor

```
public Point(x:Number, y:Number)
```

Crea un nuevo punto. Si no se pasan parámetros a este método, se crea un punto en las coordenadas (0,0).

Disponibilidad: ActionScript 1.0; Flash Player 8

Parámetros

`x:Number` - Coordenada horizontal. El valor predeterminado es 0.

`y:Number` - Coordenada vertical. El valor predeterminado es 0.

Ejemplo

En el primer ejemplo se crea un objeto Point `point_1` con el constructor predeterminado.

```
import flash.geom.Point;
var point_1:Point = new Point();
trace(point_1.x); // 0
trace(point_1.y); // 0
```

En el segundo ejemplo se crea un objeto Point `point_2` con las coordenadas *x* = 1 e *y* = 2.

```
import flash.geom.Point;
var point_2:Point = new Point(1, 2);
trace(point_2.x); // 1
trace(point_2.y); // 2
```

polar (método Point.polar)

```
public static polar(len:Number, angle:Number) : Point
```

Convierte un par de coordenadas polares en una coordenada cartesiana.

Disponibilidad: ActionScript 1.0; Flash Player 8

Parámetros

len:Number - La coordenada del par de coordenadas polares que determina la longitud.

angle:Number - El ángulo, en radianes, del par de coordenadas polares.

Valor devuelto

flash.geom.Point - La coordenada cartesiana.

Ejemplo

En el ejemplo siguiente se crea un objeto Point cartesianPoint a partir del valor de angleInRadians y de una longitud de línea de 5. Se utiliza un valor de angleInRadians equivalente a Math.atan(3/4) dadas las características de los triángulos rectángulos con relación 3:4:5 entre los lados.

```
import flash.geom.Point;
var len:Number = 5;
var angleInRadians:Number = Math.atan(3/4);
var cartesianPoint:Point = Point.polar(len, angleInRadians);
trace(cartesianPoint.toString()); // (x=4, y=3)
```

Cuando los ordenadores trabajan con números trascendentes como pi, se producen algunos errores de redondeo debido a que los cálculos en coma flotante sólo tienen una precisión finita. Cuando utilice Math.PI, es recomendable utilizar Math.round(), como se muestra en el siguiente ejemplo.

```
import flash.geom.Point;
var len:Number = 10;
var angleInRadians:Number = Math.PI;
var cartesianPoint:Point = Point.polar(len, angleInRadians);
trace(cartesianPoint.toString()); // should be (x=-10, y=0), but is (x=-10,
  y=1.22460635382238e-15)
trace(Math.round(cartesianPoint.y)); // 0
```

Véase también

length (propiedad Point.length), round (método Math.round)

subtract (método Point.subtract)

```
public subtract(v:Point) : Point
```

Resta las coordenadas de otro punto de las coordenadas de éste para crear un nuevo punto.

Disponibilidad: ActionScript 1.0; Flash Player 8

Parámetros

v:`flash.geom.Point` - El punto que se va a restar.

Valor devuelto

`flash.geom.Point` - El nuevo punto.

Ejemplo

En el siguiente ejemplo se crea `point_3` restando `point_2` de `point_1`.

```
import flash.geom.Point;
var point_1:Point = new Point(4, 8);
var point_2:Point = new Point(1, 2);
var resultPoint:Point = point_1.subtract(point_2);
trace(resultPoint.toString()); // (x=3, y=6)
```

toString (método Point.toString)

```
public toString() : String
```

Devuelve una cadena que contiene los valores de las coordenadas *x* e *y*. Dado que se expresa como (*x*, *y*), un objeto Point que se encuentre en 23,17 se devolverá como "(*x*=23, *y*=17)".

Disponibilidad: ActionScript 1.0; Flash Player 8

Valor devuelto

`String` - Una cadena.

Ejemplo

En el ejemplo siguiente se crea un punto y sus valores se convierten en una cadena con formato (x=x, y=y).

```
import flash.geom.Point;
var myPoint:Point = new Point(1, 2);
trace("myPoint: " + myPoint.toString()); // (x=1, y=2)
```

x (propiedad Point.x)

```
public x : Number
```

Coordenada horizontal del punto. El valor predeterminado es 0.

Disponibilidad: ActionScript 1.0; Flash Player 8

Ejemplo

En el ejemplo siguiente se crea un objeto Point `myPoint` y se define la coordenada *x*.

```
import flash.geom.Point;
var myPoint:Point = new Point();
trace(myPoint.x); // 0
myPoint.x = 5;
trace(myPoint.x); // 5
```

y (propiedad Point.y)

```
public y : Number
```

Coordenada vertical del punto. El valor predeterminado es 0.

Disponibilidad: ActionScript 1.0; Flash Player 8

Ejemplo

En el ejemplo siguiente se crea un objeto Point `myPoint` y se define la coordenada *y*.

```
import flash.geom.Point;
var myPoint:Point = new Point();
trace(myPoint.y); // 0
myPoint.y = 5;
trace(myPoint.y); // 5
```

PrintJob

```
Object
  |
  +-PrintJob
```

```
public class PrintJob
extends Object
```

La clase PrintJob le permite crear contenido e imprimirlo en una o varias páginas. Esta clase, además de ofrecer mejoras para la funcionalidad de impresión del método `print()`, le permite representar contenido dinámico fuera de la pantalla, mostrar a los usuarios un único cuadro de diálogo de impresión e imprimir un documento sin redimensionar con proporciones que se ajustan a las proporciones del contenido. Esta capacidad es especialmente útil para representar e imprimir contenido dinámico, como contenido de base de datos y texto dinámico.

Asimismo, con las propiedades que rellena `PrintJob.start()`, el documento puede leer la configuración de la impresora del usuario (altura, anchura y orientación de página) y se puede configurar el documento para dar al contenido de Flash un formato apropiado de forma dinámica para la configuración de la impresora. Las propiedades de diseño de usuario son de sólo lectura, por lo que Flash Player no puede modificarlas.

Disponibilidad: ActionScript 1.0; Flash Player 7

Resumen de propiedades

Modificadores	Propiedad	Descripción
	`orientation:String` [read-only]	La orientación con que se imprime la página.
	`pageHeight:Number` [read-only]	La altura del área real de la página que se puede imprimir, en puntos.
	`pageWidth:Number` [read-only]	La anchura del área real de la página que se puede imprimir, en puntos.
	`paperHeight:Number` [read-only]	La altura total del papel en puntos.
	`paperWidth:Number` [read-only]	La anchura total del papel en puntos.

Propiedades heredadas de la clase Object

```
constructor (propiedad Object.constructor), __proto__ (propiedad
Object.__proto__), prototype (propiedad Object.prototype), __resolve
(propiedad Object.__resolve)
```

Resumen de constructores

Firma	Descripción
`PrintJob()`	Crea un objeto PrintJob que puede utilizar para imprimir una o varias páginas.

Resumen de métodos

Modificadores	Firma	Descripción
	`addPage(target:Object, [printArea:Object], [options:Object], [frameNum:Number]) : Boolean`	Envía el nivel o clip de película especificado como página individual a la cola de impresión.
	`send() : Void`	Se utiliza después de `PrintJob.start()` y `PrintJob.addPage()` para enviar a la impresora páginas que se encuentran en la cola de impresión.
	`start() : Boolean`	Muestra los cuadros de diálogo de impresión del sistema operativo e inicia el envío a la cola de impresión.

Métodos heredados de la clase Object

`addProperty (método Object.addProperty), hasOwnProperty (método Object.hasOwnProperty), isPropertyEnumerable (método Object.isPropertyEnumerable), isPrototypeOf (método Object.isPrototypeOf), registerClass (método Object.registerClass), toString (método Object.toString), unwatch (método Object.unwatch), valueOf (método Object.valueOf), watch (método Object.watch)`

addPage (método PrintJob.addPage)

```
public addPage(target:Object, [printArea:Object], [options:Object],
  [frameNum:Number]) : Boolean
```

Envía el nivel o clip de película especificado como página individual a la cola de impresión. Antes de utilizar este método, deberá utilizar `PrintJob.start()`; tras llamar a `PrintJob.addPage()` una o varias veces para un trabajo de impresión, deberá utilizar `PrintJob.send()` para enviar las páginas existentes en la cola a la impresora.

Si este método devuelve `false` (por ejemplo, si no ha llamado a `PrintJob.start()` o el usuario ha cancelado el trabajo de impresión), las siguientes llamadas a `PrintJob.addPage()` fallarán. No obstante, si las llamadas anteriores a `PrintJob.addPage()` se han realizado correctamente, el comando final `PrintJob.send()` enviará las páginas que se han situado correctamente en la cola a la impresora.

Si ha pasado un valor para `printArea`, las coordenadas `xMin` e `yMin` corresponderán a la esquina superior izquierda (coordenadas 0,0) del área imprimible de la página. El área imprimible del usuario se describe mediante las propiedades de sólo lectura `pageHeight` y `pageWidth` establecidas por `PrintJob.start()`. Dado que la impresión se alinea con la esquina superior izquierda del área imprimible de la página, la impresión se recortará por la derecha y/o por abajo si el área definida en `printArea` es mayor que el área imprimible de la página. Si no ha pasado un valor para `printArea` y el escenario es mayor que el área imprimible, se producirá el mismo tipo de recorte.

Si desea redimensionar un clip de película antes de imprimirlo, establezca las propiedades `MovieClip._xscale` y `MovieClip._yscale` antes de llamar a este método y vuelva a establecer sus valores originales posteriormente. La escala de un clip de película no tiene relación con `printArea`. Es decir, si especifica la impresión de un área de un tamaño de 50 x 50 píxeles, se imprimirán 2500 píxeles. Si ha redimensionado el clip de película, se seguirán imprimiendo 2500 píxeles, pero el clip de película se imprimirá con el tamaño redimensionado.

La función de impresión de Flash Player admite las impresoras PostScript y otras. Las impresoras que no son PostScript convierten los vectores en mapas de bits.

Disponibilidad: ActionScript 1.0; Flash Player 7

Parámetros

`target``:Object` - Un número o cadena; el nivel o el nombre de la instancia del clip de película que se va a imprimir. Pase un número para especificar un nivel (por ejemplo, 0 es la película `_root`) o una cadena (entre comillas [""]) para especificar el nombre de la instancia de un clip de película.

`printArea``:Object` [opcional] - Un objeto que especifica el área de impresión con el siguiente formato:

`{xMin:`*topLeft*, xMax:*topRight*, yMin:*bottomLeft*, yMax:*bottomRight*`}`

Las coordenadas que especifica para `printArea` representan píxeles de la pantalla relacionados con el punto de registro del clip de película `_root` (si `target` = 0), o bien el nivel o clip de película especificado por `target`. Es preciso introducir las cuatro coordenadas. La anchura (`xMax-xMin`) y la altura (`yMax-yMin`) deben ser mayores de 0.

Los puntos son unidades de medida de la impresión, mientras que los píxeles son unidades de medida de la pantalla. Los puntos tienen un tamaño físico fijo (1/72 pulgadas), mientras que el tamaño de un píxel depende de la resolución de cada pantalla. Puede utilizar las equivalencias siguientes para convertir los valores en pulgadas o centímetros en twips o puntos (un twip es 1/20 de un punto):

- 1 punto = 1/72 pulgadas =20 twips
- 1 pulgada = 72 puntos = 1440 twips
- 1 cm = 567 twips

La conversión entre píxeles y puntos no es muy fiable, ya que la relación de conversión depende de la pantalla y su resolución. Por ejemplo, cuando la pantalla está configurada para mostrar 72 píxeles por pulgada, un punto equivale a un píxel.

Nota: Si antes ha utilizado `print()`, `printAsBitmap()`, `printAsBitmapNum()` o `printNum()` para imprimir desde Flash, es posible que haya empleado una etiqueta de fotograma `#b` para especificar el área de impresión. Cuando utiliza el método `addPage()`, necesita el parámetro `printArea` para especificar el área de impresión; las etiquetas de fotograma `#b` no se tienen en cuenta.

Si omite o pasa de forma incorrecta el parámetro `printArea`, se imprime el área completa del escenario de `target`. Si no desea especificar un valor para `printArea`, pero sí para `options` o `frameNumber`, pase el valor `null` para `printArea`.

`options``:Object` [opcional] - Un parámetro que especifica si se imprime como gráfico vectorial o mapa de bits, en el formato siguiente:

`{printAsBitmap:`*`Boolean`*`}`

El valor predeterminado es `false`, que representa una solicitud de impresión vectorial. Para imprimir `target` como un mapa de bits, pase el valor `true` para printAsBitmap. Recuerde las sugerencias siguientes al determinar qué valor se utiliza:

- Si el contenido que está imprimiendo incluye una imagen de mapa de bits, utilice `{printAsBitmap:true}` para incluir transparencias y efectos de color.
- Si el contenido no incluye imágenes de mapa de bits, omita este parámetro o utilice `{printAsBitmap:false}` para imprimir el contenido en formato vectorial de calidad superior.

Si omite o pasa de forma incorrecta `options`, se utiliza la impresión vectorial. Si no desea especificar un valor para `options`, pero sí para `frameNumber` pase el valor `null` para `options`.

`frameNum``:Number` [opcional] - Un número que permite especificar el fotograma que se va a imprimir. Si pasa un `frameNumber`, no se invoca ActionScript en ese fotograma. Si omite este parámetro, se imprime el fotograma actual en `target`.

Nota: Si antes ha utilizado `print()`, `printAsBitmap()`, `printAsBitmapNum()` o `printNum()` para imprimir desde Flash, es posible que haya empleado una etiqueta de fotograma `#p` para especificar las páginas que se van a imprimir. Si desea utilizar `PrintJob.addPage()` para imprimir varios fotogramas, debe enviar un comando `PrintJob.addPage()` por cada fotograma; las etiquetas de fotograma `#p` no se tienen en cuenta. Para averiguar cómo se realiza mediante programación, consulte el apartado Ejemplo.

Valor devuelto

`Boolean` - Valor booleano: `true` si la página se ha enviado correctamente a la cola de impresión; `false` en caso contrario.

Ejemplo

En el ejemplo siguiente se muestran varias formas de enviar el comando `addPage()`:

```
my_btn.onRelease = function()
{
  var pageCount:Number = 0;

  var my_pj:PrintJob = new PrintJob();

  if (my_pj.start())
  {
  // Print entire current frame of the _root movie in vector format
  if (my_pj.addPage(0)){
    pageCount++;

    // Starting at 0,0, print an area 400 pixels wide and 500 pixels high
    // of the current frame of the _root movie in vector format
    if (my_pj.addPage(0, {xMin:0,xMax:400,yMin:0,yMax:500})){
    pageCount++;

    // Starting at 0,0, print an area 400 pixels wide and 500 pixels high
    // of frame 1 of the _root movie in bitmap format
    if (my_pj.addPage(0, {xMin:0,xMax:400,yMin:0,yMax:500},
      {printAsBitmap:true}, 1)){
      pageCount++;

      // Starting 50 pixels to the right of 0,0 and 70 pixels down,
      // print an area 500 pixels wide and 600 pixels high
      // of frame 4 of level 5 in vector format
      if (my_pj.addPage(5, {xMin:50,xMax:550,yMin:70,yMax:670},null, 4)){
      pageCount++;

      // Starting at 0,0, print an area 400 pixels wide
      // and 400 pixels high of frame 3 of the "dance_mc" movie clip
      // in bitmap format
      if (my_pj.addPage("dance_mc",
```

```
        {xMin:0,xMax:400,yMin:0,yMax:400},{printAsBitmap:true}, 3)){
        pageCount++;

        // Starting at 0,0, print an area 400 pixels wide
        // and 600 pixels high of frame 3 of the "dance_mc" movie clip
        // in vector format at 50% of its actual size
        var x:Number = dance_mc._xscale;
        var y:Number = dance_mc._yscale;
        dance_mc._xscale = 50;
        dance_mc._yscale = 50;

        if (my_pj.addPage("dance_mc",
          {xMin:0,xMax:400,yMin:0,yMax:600},null, 3)){
        pageCount++;
        }
        dance_mc._xscale = x;
        dance_mc._yscale = y;
      }
      }
    }
    }
  }
  }

  // If addPage() was successful at least once, print the spooled pages.
  if (pageCount > 0){
  my_pj.send();
  }
  delete my_pj;
}
```

Véase también

send (método PrintJob.send), start (método PrintJob.start)

orientation (propiedad PrintJob.orientation)

```
public orientation : String [read-only]
```

La orientación con que se imprime la página. Esta propiedad puede ser `"landscape"` o `"portrait"`. Esta propiedad sólo está disponible tras una llamada al método `PrintJob.start()`.

Disponibilidad: ActionScript 1.0; Flash Player 7

pageHeight (propiedad PrintJob.pageHeight)

```
public pageHeight : Number [read-only]
```

La altura del área real de la página que se puede imprimir, en puntos. Los márgenes establecidos por el usuario no se tienen en cuenta. Esta propiedad sólo está disponible tras una llamada al método `PrintJob.start()`.

Disponibilidad: ActionScript 1.0; Flash Player 7

pageWidth (propiedad PrintJob.pageWidth)

```
public pageWidth : Number [read-only]
```

La anchura del área real de la página que se puede imprimir, en puntos. Los márgenes establecidos por el usuario no se tienen en cuenta. Esta propiedad sólo está disponible tras una llamada al método `PrintJob.start()`.

Disponibilidad: ActionScript 1.0; Flash Player 7

paperHeight (propiedad PrintJob.paperHeight)

```
public paperHeight : Number [read-only]
```

La altura total del papel en puntos. Esta propiedad sólo está disponible tras una llamada al método `PrintJob.start()`.

Disponibilidad: ActionScript 1.0; Flash Player 7

paperWidth (propiedad PrintJob.paperWidth)

```
public paperWidth : Number [read-only]
```

La anchura total del papel en puntos. Esta propiedad sólo está disponible tras una llamada al método `PrintJob.start()`.

Disponibilidad: ActionScript 1.0; Flash Player 7

PrintJob, constructor

```
public PrintJob()
```

Crea un objeto PrintJob que puede utilizar para imprimir una o varias páginas.

Para implementar un trabajo de impresión, utilice los siguientes métodos en el orden indicado. Debe colocar todos los comandos relativos a un trabajo de impresión concreto en el mismo fotograma, desde el constructor a través de `PrintJob.send()` y eliminar. Reemplace los `[params]` de las llamadas al método `my_pj.addPage()` por sus parámetros personalizados.

```
// create PrintJob object
var my_pj:PrintJob = new PrintJob();

// display print dialog box, but only initiate the print job
// if start returns successfully.
if (my_pj.start()) {

  // use a variable to track successful calls to addPage
  var pagesToPrint:Number = 0;

  // add specified area to print job
  // repeat once for each page to be printed
  if (my_pj.addPage([params])) {
  pagesToPrint++;
  }
  if (my_pj.addPage([params])) {
  pagesToPrint++;
  }
  if (my_pj.addPage([params])) {
  pagesToPrint++;
  }

  // send pages from the spooler to the printer, but only if one or more
  // calls to addPage() was successful. You should always check for
  successful
  // calls to start() and addPage() before calling send().
  if (pagesToPrint > 0) {
  my_pj.send();  // print page(s)
  }
}

// clean up
delete my_pj;  // delete object
```

No puede crear un segundo objeto PrintJob mientras el primero continúa activo. No puede crear un segundo objeto PrintJob (llamando a `new PrintJob()`); mientras continúe activo el primer objeto PrintJob, el segundo objeto PrintJob no se creará.

Disponibilidad: ActionScript 1.0; Flash Player 7

Ejemplo

Véase `PrintJob.addPage()`.

Véase también

`addPage (método PrintJob.addPage)`, `send (método PrintJob.send)`, `start (método PrintJob.start)`

send (método PrintJob.send)

```
public send() : Void
```

Se utiliza después de `PrintJob.start()` y `PrintJob.addPage()` para enviar a la impresora páginas que se encuentran en la cola de impresión. Dado que las llamadas a `PrintJob.send()` no se realizarán correctamente si han fallado las correspondientes llamadas a `PrintJob.start()` y `PrintJob.addpage()`, deberá comprobar que las llamadas a `PrintJob.addpage()` y `PrintJob.start()` han sido correctas antes de llamar a `PrintJob.send()`:

```
var my_pj:PrintJob = new PrintJob();
if (my_pj.start()) {
  if (my_pj.addPage(this)) {
  my_pj.send();
  }
}
delete my_pj;
```

Disponibilidad: ActionScript 1.0; Flash Player 7

Ejemplo

Véase `PrintJob.addPage()` y `PrintJob.start()`.

Véase también

`addPage (método PrintJob.addPage)`, `start (método PrintJob.start)`

start (método PrintJob.start)

```
public start() : Boolean
```

Muestra los cuadros de diálogo de impresión del sistema operativo e inicia el envío a la cola de impresión. Los cuadros de diálogo de impresión permiten al usuario cambiar la configuración de impresión. Si el método `PrintJob.start()` devuelve un valor correctamente, las siguientes propiedades de sólo lectura se llenarán con la configuración de impresión del usuario:

Propiedad	Type	Unidades	Notas
PrintJob.paperHeight	Number	Puntos	Altura del papel total.
PrintJob.paperWidth	Number	Puntos	Anchura total del papel.
PrintJob.pageHeight	Number	Puntos	Altura del área imprimible de la página; los márgenes establecidos por el usuario no se tendrán en cuenta.
PrintJob.pageWidth	Number	Puntos	Anchura del área imprimible de la página; los márgenes establecidos por el usuario no se tendrán en cuenta.
PrintJob.orientation	Number	Puntos	"Vertical" u "horizontal."

Una vez que el usuario hace clic en Aceptar en el cuadro de diálogo Imprimir, el reproductor comienza a enviar el trabajo de impresión a la cola del sistema operativo. Deberá enviar los comandos de ActionScript que afecten a la impresión y podrá utilizar comandos `PrintJob.addPage()` para enviar páginas a la cola de impresión. Puede utilizar las propiedades de sólo lectura height, width y orientation que llena este método para dar formato a la impresión.

Dado que el usuario ve información del tipo "Imprimiendo página 1" inmediatamente después de hacer clic en Aceptar, deberá llamar a los comandos `PrintJob.addPage()` y `PrintJob.send()` lo antes posible.

Si este método devuelve `false` (por ejemplo, si el usuario hace clic en Cancelar en lugar de en Aceptar en el cuadro de diálogo Imprimir del sistema operativo), las llamadas posteriores a `PrintJob.addPage()` y `PrintJob.send()` fallarán. No obstante, aunque realice una prueba de este valor de devolución y no envíe comandos `PrintJob.addPage()` como resultado, deberá eliminar el objeto PrintJob para asegurarse de que la cola de impresión se ha vaciado, como se muestra en el siguiente ejemplo:

```
var my_pj:PrintJob = new PrintJob();
var myResult:Boolean = my_pj.start();
  if(myResult) {
  // addPage() and send() statements here
  }
delete my_pj;
```

Disponibilidad: ActionScript 1.0; Flash Player 7

Valor devuelto

`Boolean` - Valor booleano: `true` si el usuario hace clic en Aceptar cuando aparecen los cuadros de diálogo de impresión; `false` si el usuario hace clic en Cancelar o si se produce un error.

Ejemplo

En el ejemplo siguiente se muestra cómo puede utilizar el valor de la propiedad orientation para ajustar la impresión:

```
// create PrintJob object
var my_pj:PrintJob = new PrintJob();

// display print dialog box
if (my_pj.start()) {
  // boolean to track whether addPage succeeded, change this to a counter
  // if more than one call to addPage is possible
  var pageAdded:Boolean = false;

  // check the user's printer orientation setting
  // and add appropriate print area to print job
  if (my_pj.orientation == "portrait") {
  // Here, the printArea measurements are appropriate for an 8.5" x 11"
  // portrait page.
  pageAdded = my_pj.addPage(this,{xMin:0,xMax:600,yMin:0,yMax:800});
  }
  else {
  // my_pj.orientation is "landscape".
  // Now, the printArea measurements are appropriate for an 11" x 8.5"
  // landscape page.
  pageAdded = my_pj.addPage(this,{xMin:0,xMax:750,yMin:0,yMax:600});
  }

  // send pages from the spooler to the printer
```

```
  if (pageAdded) {
  my_pj.send();
  }
}

// clean up
delete my_pj;
```

Véase también

addPage (método PrintJob.addPage), send (método PrintJob.send)

Rectangle (flash.geom.Rectangle)

```
Object
  |
  +-flash.geom.Rectangle
```

public class *Rectangle*
extends Object

La clase Rectangle se utiliza para crear y modificar objetos Rectangle. Un objeto Rectangle es un área definida por su posición, según se determina en función de su ángulo superior izquierdo (*x*, *y*) y de su altura y anchura. Preste atención a estas áreas cuando diseñe, si un rectángulo tiene el ángulo superior izquierdo en 0,0, la altura en 10 y la anchura en 20, el ángulo inferior derecha estará en 9,19 debido a que la altura y la anchura se empiezan a contar en 0,0.

Como las propiedadesx, y, width y height de la clase Rectangle son independientes entre sí, el cambio del valor de una propiedad no afecta a las otras. Sin embargo, hay que tener en cuenta que las propiedades right y bottom está estrechamente relacionadas con las otras cuatro. Si cambia la propiedad right, está modificando la propiedad width y si cambia bottom modifica height, etc. Las propiedades left o *x* deben estar definidas antes de establecer width o right.

Los objetos Rectangle permiten utilizar los filtros de la clase BitmapData. También se utilizan en la propiedad MovieClip.scrollRect para permitir el recorte y desplazamiento de una instancia de MovieClip aplicando valores de anchura, altura y desplazamiento específicos.

Disponibilidad: ActionScript 1.0; Flash Player 8

Véase también

scrollRect (propiedad MovieClip.scrollRect)

Resumen de propiedades

Modificadores	Propiedad	Descripción
	`bottom:Number`	Suma de las propiedades `y` y `height`.
	`bottomRight:Point`	La posición del ángulo inferior derecho del objeto Rectangle viene determinada por los valores `x` e `y`.
	`height:Number`	La altura del rectángulo en píxeles.
	`left:Number`	La coordenada *x* del ángulo superior izquierdo del rectángulo.
	`right:Number`	Suma de las propiedades `x` y `width`.
	`size:Point`	El tamaño del objeto Rectangle, expresado como un objeto Point con valores `width` y `height`.
	`top:Number`	La coordenada *y* del ángulo superior izquierdo del rectángulo.
	`topLeft:Point`	La posición del ángulo superior izquierdo del objeto Rectangle viene determinada por los valores *x* e *y* del punto.
	`width:Number`	La anchura del rectángulo en píxeles.
	`x:Number`	La coordenada *x* del ángulo superior izquierdo del rectángulo.
	`y:Number`	La coordenada *y* del ángulo superior izquierdo del rectángulo.

Propiedades heredadas de la clase Object

`constructor (propiedad Object.constructor), __proto__ (propiedad Object.__proto__), prototype (propiedad Object.prototype), __resolve (propiedad Object.__resolve)`

Resumen de constructores

Firma	Descripción
`Rectangle(x:Number, y:Number, width:Number, height:Number)`	Crea un objeto Rectangle nuevo, cuyo ángulo superior izquierdo viene determinado por los valores de los parámetros `x` e `y`.

Resumen de métodos

Modificadores	Firma	Descripción
	`clone() : Rectangle`	Devuelve un objeto Rectangle nuevo con las mismas propiedades `x`, `y`, `width` y `height` que el objeto Rectangle original.
	`contains(x:Number, y:Number) : Boolean`	Determina si el punto especificado está incluido en la región rectangular definida por este objeto Rectangle.
	`containsPoint(pt:Point) : Boolean`	Determina si el punto especificado está incluido en la región rectangular definida por este objeto Rectangle.
	`containsRectangle(rect:Rectangle) : Boolean`	Determina si el objeto Rectangle especificado por el parámetro `rect` está contenido en este objeto Rectangle.
	`equals(toCompare:Object) : Boolean`	Determina si el objeto especificado en el parámetro `toCompare` es igual a este objeto Rectangle.
	`inflate(dx:Number, dy:Number) : Void`	Expande el objeto Rectangle según las cantidades especificadas.
	`inflatePoint(pt:Point) : Void`	Aumenta el tamaño del objeto Rectangle.
	`intersection(toIntersect:Rectangle) : Rectangle`	Si el objeto Rectangle especificado en el parámetro `toIntersect` presenta un punto de intersección con este objeto Rectangle, el método `intersection()` devuelve el área de intersección como un objeto Rectangle.
	`intersects(toIntersect:Rectangle) : Boolean`	Determina si el objeto especificado en el parámetro `toIntersect` presenta un punto de intersección con este objeto Rectangle.
	`isEmpty() : Boolean`	Determina si este objeto Rectangle está vacío o no.
	`offset(dx:Number, dy:Number) : Void`	Ajusta la posición del objeto Rectangle, definida por su ángulo superior izquierdo, según las cantidades especificadas.
	`offsetPoint(pt:Point) : Void`	Ajusta la posición del objeto Rectangle utilizando un objeto Point como parámetro.
	`setEmpty() : Void`	Establece todas las propiedades del objeto Rectangle en 0.

Modificadores	Firma	Descripción
	`toString() : String`	Crea y devuelve una cadena en la que se indican las posiciones horizontal y vertical, así como la anchura y la altura, del objeto Rectangle.
	`union(toUnion:Rectangle) : Rectangle`	Une dos rectángulos para crear un objeto Rectangle nuevo, y rellena los espacios horizontal y vertical que quedan entre los dos rectángulos.

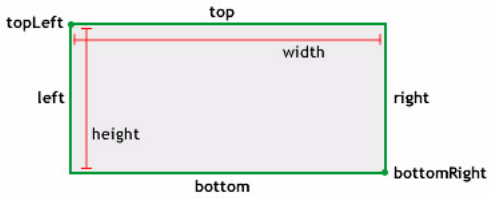
Métodos heredados de la clase Object

`addProperty (método Object.addProperty), hasOwnProperty (método Object.hasOwnProperty), isPropertyEnumerable (método Object.isPropertyEnumerable), isPrototypeOf (método Object.isPrototypeOf), registerClass (método Object.registerClass), toString (método Object.toString), unwatch (método Object.unwatch), valueOf (método Object.valueOf), watch (método Object.watch)`

bottom (propiedad Rectangle.bottom)

`public bottom : Number`

Suma de las propiedades `y` y `height`.



Disponibilidad: ActionScript 1.0; Flash Player 8

Ejemplo

El ejemplo siguiente crea un objeto Rectangle y cambia el valor de su propiedad `bottom` de 15 a 30. Tenga en cuenta que el valor de `rect.height` también se cambia de 10 a 25.

```
import flash.geom.Rectangle;

var rect:Rectangle = new Rectangle(5, 5, 10, 10);
trace(rect.height); // 10
trace(rect.bottom); // 15

rect.bottom = 30;
trace(rect.height); // 25
trace(rect.bottom); // 30
```

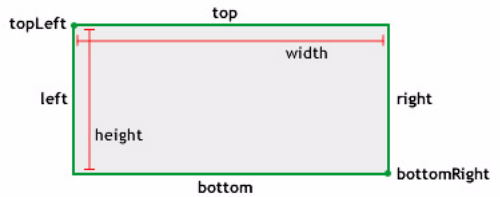
Véase también

`y (propiedad Rectangle.y)`, `height (propiedad Rectangle.height)`

bottomRight (propiedad Rectangle.bottomRight)

```
public bottomRight : Point
```

La posición del ángulo inferior derecho del objeto Rectangle viene determinada por los valores `x` e `y`.



Disponibilidad: ActionScript 1.0; Flash Player 8

Ejemplo

El ejemplo siguiente define la propiedad `bottomRight` del objeto Rectangle con los valores del objeto Point. Advierta cómo se modifican `rect.width` y `rect.height`.

```
import flash.geom.Rectangle;
import flash.geom.Point;

var rect:Rectangle = new Rectangle(1, 2, 4, 8);
trace(rect.bottom); // 10
trace(rect.right); // 5
trace(rect.height); // 8
trace(rect.width); // 4

var myBottomRight:Point = new Point(16, 32);
rect.bottomRight = myBottomRight;
trace(rect.bottom); // 32
trace(rect.right); // 16
trace(rect.height); // 30
trace(rect.width); // 15
```

Véase también

`Point (flash.geom.Point)`

clone (método Rectangle.clone)

`public close() : Rectángulo`

Devuelve un objeto Rectangle nuevo con las mismas propiedades `x`, `y`, `width` y `height` que el objeto Rectangle original.

Disponibilidad: ActionScript 1.0; Flash Player 8

Valor devuelto

`flash.geom.Rectangle` - Devuelve un objeto Rectangle nuevo con las mismas propiedades `x`, `y`, `width` y `height` que el objeto Rectangle original.

Ejemplo

El ejemplo siguiente crea tres objetos Rectangle y los compara. `rect_1` se crea utilizando el constructor Rectangle. `rect_2` se crea haciéndolo igual a `rect_1`. Y `clonedRect` se crea por clonación de `rect_1`. Tenga en cuenta que aunque `rect_2` se evalúe como igual a `rect_1`, `clonedRect` no lo hará, aunque contenga los mismos valores que `rect_1`.

```
import flash.geom.Rectangle;

var rect_1:Rectangle = new Rectangle(1, 2, 4, 8);
var rect_2:Rectangle = rect_1;
var clonedRect:Rectangle = rect_1.clone();

trace(rect_1 == rect_2); // true
trace(rect_1 == clonedFilter); // false

for(var i in rect_1) {
  trace(">> " + i + ": " + rect_1[i]);
  >> toString: [type Function]
  >> equals: [type Function]
  >> union: [type Function]
  >> intersects: [type Function]
  >> intersection: [type Function]
  >> containsRectangle: [type Function]
  >> containsPoint: [type Function]
  >> contains: [type Function]
  >> offsetPoint: [type Function]
  >> offset: [type Function]
  >> inflatePoint: [type Function]
  >> inflate: [type Function]
  >> size: (x=4, y=8)
  >> bottomRight: (x=5, y=10)
  >> topLeft: (x=1, y=2)
  >> bottom: 10
  >> top: 2
  >> right: 5
```

```
  >> left: 1
  >> isEmpty: [type Function]
  >> setEmpty: [type Function]
  >> clone: [type Function]
  >> height: 8
  >> width: 4
  >> y: 2
  >> x: 1
}

for(var i in clonedRect) {
  trace(">> " + i + ": " + clonedRect[i]);
  >> toString: [type Function]
  >> equals: [type Function]
  >> union: [type Function]
  >> intersects: [type Function]
  >> intersection: [type Function]
  >> containsRectangle: [type Function]
  >> containsPoint: [type Function]
  >> contains: [type Function]
  >> offsetPoint: [type Function]
  >> offset: [type Function]
  >> inflatePoint: [type Function]
  >> inflate: [type Function]
  >> size: (x=4, y=8)
  >> bottomRight: (x=5, y=10)
  >> topLeft: (x=1, y=2)
  >> bottom: 10
  >> top: 2
  >> right: 5
  >> left: 1
  >> isEmpty: [type Function]
  >> setEmpty: [type Function]
  >> clone: [type Function]
  >> height: 8
  >> width: 4
  >> y: 2
  >> x: 1
}
```

Como demostración de las relaciones entre `rect_1`, `rect_2` y `clonedRect`, el ejemplo siguiente modifica la propiedad `x` de `rect_1`. Al modificar `x`, el método `clone()` crea una nueva instancia basada en los valores de `rect_1` en lugar de crear referencias hacia los mismos.

```
import flash.geom.Rectangle;

var rect_1:Rectangle = new Rectangle(1, 2, 4, 8);
var rect_2:Rectangle = rect_1;
var clonedRect:Rectangle = rect_1.clone();

trace(rect_1.x); // 1
```

```
trace(rect_2.x); // 1
trace(clonedRect.x); // 1

rect_1.x = 10;

trace(rect_1.x); // 10
trace(rect_2.x); // 10
trace(clonedRect.x); // 1
```

Véase también

x (propiedad Rectangle.x), y (propiedad Rectangle.y), width (propiedad Rectangle.width), height (propiedad Rectangle.height)

contains (método Rectangle.contains)

```
public contains(x:Number, y:Number) : Boolean
```

Determina si el punto especificado está incluido en la región rectangular definida por este objeto Rectangle.

Disponibilidad: ActionScript 1.0; Flash Player 8

Parámetros

x:Number - El valor *x* (posición horizontal) del punto.

y:Number - El valor *y* (posición vertical) del punto.

Valor devuelto

Boolean - Si el punto especificado contiene el objeto Rectangle, devuelve true; en caso contrario devuelve false.

Ejemplo

El ejemplo siguiente crea un objeto Rectangle y prueba si los tres pares de coordenadas quedan entre los límites.

```
import flash.geom.Rectangle;

var rect:Rectangle = new Rectangle(10, 10, 50, 50);
trace(rect.contains(59, 59)); // true
trace(rect.contains(10, 10)); // true
trace(rect.contains(60, 60)); // false
```

Véase también

Point (flash.geom.Point)

containsPoint (método Rectangle.containsPoint)

```
public containsPoint(pt:Point) : Boolean
```

Determina si el punto especificado está incluido en la región rectangular definida por este objeto Rectangle. Este método es similar al método `Rectangle.contains()`, con la salvedad de que toma un objeto Point como parámetro.

Disponibilidad: ActionScript 1.0; Flash Player 8

Parámetros

`pt``:flash.geom.Point` - El punto, representado por sus valores *x*,*y*.

Valor devuelto

`Boolean` - Si el punto especificado contiene el objeto Rectangle, devuelve `true`; en caso contrario devuelve `false`.

Ejemplo

El ejemplo siguiente crea un objeto Rectangle y tres objetos Point y prueba si los puntos quedan entre los límites del rectángulo.

```
import flash.geom.Rectangle;
import flash.geom.Point;

var rect:Rectangle = new Rectangle(10, 10, 50, 50);
trace(rect.containsPoint(new Point(10, 10))); // true
trace(rect.containsPoint(new Point(59, 59))); // true
trace(rect.containsPoint(new Point(60, 60))); // false
```

Véase también

`contains (método Rectangle.contains)`, `Point (flash.geom.Point)`

containsRectangle (método Rectangle.containsRectangle)

```
public containsRectangle(rect:Rectangle) : Boolean
```

Determina si el objeto Rectangle especificado por el parámetro `rect` está contenido en este objeto Rectangle. Se considera que un rectángulo contiene otro si el segundo está completamente incluido dentro de los límites del primero.

Disponibilidad: ActionScript 1.0; Flash Player 8

Parámetros

rect:flash.geom.Rectangle - El objeto Rectangle que se comprueba.

Valor devuelto

Boolean - Si el objeto Rectangle especificado contiene el objeto Rectangle, devuelve true; en caso contrario devuelve false.

Ejemplo

El ejemplo siguiente crea cuatro objetos Rectangle y determina si el rectángulo A contiene el rectángulo B, C o D.

```
import flash.geom.Rectangle;

var rectA:Rectangle = new Rectangle(10, 10, 50, 50);
var rectB:Rectangle = new Rectangle(10, 10, 50, 50);
var rectC:Rectangle = new Rectangle(10, 10, 51, 51);
var rectD:Rectangle = new Rectangle(15, 15, 45, 45);

trace(rectA.containsRectangle(rectB)); // true
trace(rectA.containsRectangle(rectC)); // false
trace(rectA.containsRectangle(rectD)); // true
```

equals (método Rectangle.equals)

```
public equals(toCompare:Object) : Boolean
```

Determina si el objeto especificado en el parámetro toCompare es igual a este objeto Rectangle. Este método compara las propiedades x, y, width y height de un objeto con las de este objeto Rectangle.

Disponibilidad: ActionScript 1.0; Flash Player 8

Parámetros

toCompare:Object - El rectángulo que desea comparar con este objeto Rectangle.

Valor devuelto

Boolean - Si el objeto tiene exactamente los mismos valores en las propiedades x, y, width y height que este objeto Rectángulo, devuelve true; en caso contrario false.

Ejemplo

En el ejemplo siguiente, `rect_1` y `rect_2` son iguales, pero `rect_3` no es igual a los otros dos objetos debido a que sus propiedades `x`, `y`, `width` y `height` no son iguales a las de `rect_1` y `rect_2`.

```
import flash.geom.Rectangle;

var rect_1:Rectangle = new Rectangle(0, 0, 50, 100);
var rect_2:Rectangle = new Rectangle(0, 0, 50, 100);
var rect_3:Rectangle = new Rectangle(10, 10, 60, 110);

trace(rect_1.equals(rect_2)); // true;
trace(rect_1.equals(rect_3)); // false;
```

Incluso si la firma del método espera sólo un objeto abstracto, sólo las otras instancias Rectangle se consideran iguales.

```
import flash.geom.Rectangle;

var rect_1:Rectangle = new Rectangle(0, 0, 50, 100);
var nonRect:Object = new Object();
nonRect.x = 0;
nonRect.y = 0;
nonRect.width = 50;
nonRect.height = 100;
trace(rect_1.equals(nonRect));
```

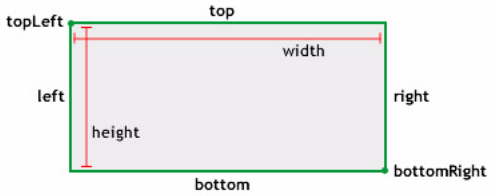
Véase también

`x (propiedad Rectangle.x)`, `y (propiedad Rectangle.y)`, `width (propiedad Rectangle.width)`, `height (propiedad Rectangle.height)`

height (propiedad Rectangle.height)

```
public height : Number
```

La altura del rectángulo en píxeles. El cambio del valor de `height` de un objeto Rectangle no afecta a las propiedades `x`, `y` o `width`.



Disponibilidad: ActionScript 1.0; Flash Player 8

Ejemplo

Cree un objeto Rectangle y cámbiele el valor de la propiedad `height` de 10 a 20. Observe que `rect.bottom` también cambia.

```
import flash.geom.Rectangle;

var rect:Rectangle = new Rectangle(5, 5, 10, 10);
trace(rect.height); // 10
trace(rect.bottom); // 15

rect.height = 20;
trace(rect.height); // 20
trace(rect.bottom); // 25
```

Véase también

`x (propiedad Rectangle.x)`, `y (propiedad Rectangle.y)`, `width (propiedad Rectangle.width)`

inflate (método Rectangle.inflate)

```
public inflate(dx:Number, dy:Number) : Void
```

Expande el objeto Rectangle según las cantidades especificadas. El punto central del objeto Rectangle permanece igual y su tamaño aumenta hacia la izquierda y la derecha en función del valor `dx` y hacia arriba y abajo según el valor `dy`.

Disponibilidad: ActionScript 1.0; Flash Player 8

Parámetros

`dx``:Number` - El valor que se añadirá a la izquierda y derecha del objeto Rectangle. La ecuación que se utiliza para calcular la nueva anchura y la posición del rectángulo es la siguiente:

```
x -= dx;
width += 2 * dx;
```

`dy``:Number` - El valor que se añadirá por arriba y por abajo del objeto Rectangle. La ecuación que se utiliza para calcular la nueva altura y la posición del rectángulo es la siguiente:

```
y -= dy;
height += 2 * dy;
```

Ejemplo

El ejemplo siguiente crea un objeto Rectangle y aumenta el valor de su propiedad `width` en 16 * 2 (32) y su propiedad `height` en 32 * 2 (64).

```
import flash.geom.Rectangle;

var rect:Rectangle = new Rectangle(1, 2, 4, 8);
trace(rect.toString()); // (x=1, y=2, w=4, h=8)

rect.inflate(16, 32);
trace(rect.toString()); // (x=-15, y=-30, w=36, h=72)
```

Véase también

`x (propiedad Rectangle.x)`, `y (propiedad Rectangle.y)`

inflatePoint (método Rectangle.inflatePoint)

```
public inflatePoint(pt:Point) : Void
```

Aumenta el tamaño del objeto Rectangle. Este método es similar al método `Rectangle.inflate()`, con la salvedad de que toma un objeto Point como parámetro.

Los siguientes dos ejemplos de código logran el mismo resultado:

```
rect1 = new flash.geom.Rectangle(0,0,2,5);
rect1.inflate(2,2)
rect1 = new flash.geom.Rectangle(0,0,2,5);
pt1 = new flash.geom.Point(2,2);
rect1.inflatePoint(pt1)
```

Disponibilidad: ActionScript 1.0; Flash Player 8

Parámetros

`pt``:flash.geom.Point` - Aumenta el tamaño del rectángulo según los valores de las coordenadas*x* e *y* del punto.

Ejemplo

El siguiente ejemplo crea un objeto Rectangle y lo expande de acuerdo con los valores *x* (horizontal) e *y* (vertical) encontrados en un punto.

```
import flash.geom.Rectangle;
import flash.geom.Point;

var rect:Rectangle = new Rectangle(0, 0, 2, 5);
trace(rect.toString()); // (x=0, y=0, w=2, h=5
```

```
var myPoint:Point = new Point(2, 2);
rect.inflatePoint(myPoint);
trace(rect.toString()); // (x=-2, y=-2, w=6, h=9)
```

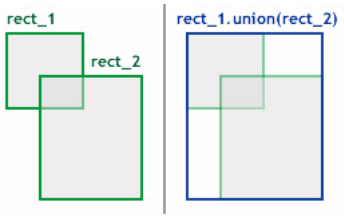
Véase también

Point (flash.geom.Point)

intersection (método Rectangle.intersection)

```
public intersection(toIntersect:Rectangle) : Rectángulo
```

Si el objeto Rectangle especificado en el parámetro `toIntersect` presenta un punto de intersección con este objeto Rectangle, el método `intersection()` devuelve el área de intersección como un objeto Rectangle. Si los rectángulos no presentan un punto de intersección, este método devuelve un objeto Rectangle vacío con las propiedades definidas en 0.



Disponibilidad: ActionScript 1.0; Flash Player 8

Parámetros

`toIntersect``:flash.geom.Rectangle` - El objeto Rectangle que quiere comparar para determinar si presenta un punto de intersección con este objeto Rectangle.

Valor devuelto

`flash.geom.Rectangle` - Un objeto Rectangle igual al área de intersección. Si los rectángulos no presentan un punto de intersección, este método devuelve un objeto Rectangle vacío con las propiedades `x`, `y`, `width` y `height` definidas en 0.

Ejemplo

El ejemplo siguiente determina el área en la que `rect_1` crea una intersección con `rect_2`.

```
import flash.geom.Rectangle;

var rect_1:Rectangle = new Rectangle(0, 0, 50, 50);
var rect_2:Rectangle = new Rectangle(25, 25, 100, 100);
var intersectingArea:Rectangle = rect_1.intersection(rect_2);
trace(intersectingArea.toString()); // (x=25, y=25, w=25, h=25)
```

intersects (método Rectangle.intersects)

```
public intersects(toIntersect:Rectangle) : Boolean
```

Determina si el objeto especificado en el parámetro `toIntersect` presenta un punto de intersección con este objeto Rectangle. Este método comprueba las propiedades `x`, `y`, `width` y `height` del objeto Rectangle especificado para determinar si presenta un punto de intersección con este objeto Rectangle.

Disponibilidad: ActionScript 1.0; Flash Player 8

Parámetros

`toIntersect``:flash.geom.Rectangle` - El objeto Rectangle que se va a comparar con este objeto Rectangle.

Valor devuelto

`Boolean` - Si el punto especificado contiene un punto de intersección con el objeto Rectangle, devuelve `true`; en caso contrario devuelve `false`.

Ejemplo

El ejemplo siguiente determina el área en la que `rectA` crea una intersección con `rectB` o `rectC`.

```
import flash.geom.Rectangle;
var rectA:Rectangle = new Rectangle(10, 10, 50, 50);
var rectB:Rectangle = new Rectangle(59, 59, 50, 50);
var rectC:Rectangle = new Rectangle(60, 60, 50, 50);
var rectAIntersectsB:Boolean = rectA.intersects(rectB);
var rectAIntersectsC:Boolean = rectA.intersects(rectC);
trace(rectAIntersectsB); // true
trace(rectAIntersectsC); // false

var firstPixel:Rectangle = new Rectangle(0, 0, 1, 1);
var adjacentPixel:Rectangle = new Rectangle(1, 1, 1, 1);
var pixelsIntersect:Boolean = firstPixel.intersects(adjacentPixel);
trace(pixelsIntersect); // false
```

Véase también

`x (propiedad Rectangle.x)`, `y (propiedad Rectangle.y)`, `width (propiedad Rectangle.width)`, `height (propiedad Rectangle.height)`

isEmpty (método Rectangle.isEmpty)

```
public isEmpty() : Boolean
```

Determina si este objeto Rectangle está vacío o no.

Disponibilidad: ActionScript 1.0; Flash Player 8

Valor devuelto

`Boolean` - Si la anchura o altura del objeto Rectangle es inferior o igual a 0, devuelve `true`; en caso contrario devuelve `false`.

Ejemplo

El ejemplo siguiente crea un objeto Rectangle vacío y comprueba que está vacío.

```
import flash.geom.*;
var rect:Rectangle = new Rectangle(1, 2, 0, 0);
trace(rect.toString()); // (x=1, y=2, w=0, h=0)
trace(rect.isEmpty()); // true
```

El ejemplo siguiente crea un objeto Rectangle no vacío y lo convierte en vacío.

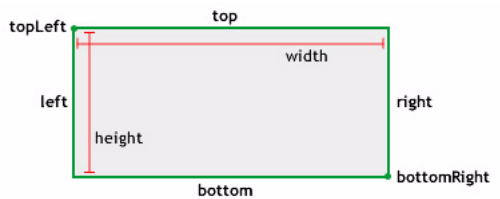
```
import flash.geom.Rectangle;

var rect:Rectangle = new Rectangle(1, 2, 4, 8);
trace(rect.isEmpty()); // false
rect.width = 0;
trace(rect.isEmpty()); // true
rect.width = 4;
trace(rect.isEmpty()); // false
rect.height = 0;
trace(rect.isEmpty()); // true
```

left (propiedad Rectangle.left)

```
public left : Number
```

La coordenada *x* del ángulo superior izquierdo del rectángulo. El cambio del valor de *x* de un objeto Rectangle no afecta a las propiedades `y`, `width` o `height`.

La propiedad `left` es igual a la propiedad `x`.



Disponibilidad: ActionScript 1.0; Flash Player 8

Ejemplo

El ejemplo siguiente cambia la propiedad `left` de 0 a 10. Observe cómo `rect.x` también cambia.

```
import flash.geom.Rectangle;

var rect:Rectangle = new Rectangle();
trace(rect.left); // 0
trace(rect.x); // 0

rect.left = 10;
trace(rect.left); // 10
trace(rect.x); // 10
```

Véase también

`x (propiedad Rectangle.x)`, `y (propiedad Rectangle.y)`, `width (propiedad Rectangle.width)`, `height (propiedad Rectangle.height)`

offset (método Rectangle.offset)

```
public offset(dx:Number, dy:Number) : Void
```

Ajusta la posición del objeto Rectangle, definida por su ángulo superior izquierdo, según las cantidades especificadas.

Disponibilidad: ActionScript 1.0; Flash Player 8

Parámetros

dx`:Number` - Desplaza el valor *x* del objeto Rectangle según esta cantidad.

dy`:Number` - Desplaza el valor *y* del objeto Rectangle según esta cantidad.

Ejemplo

El ejemplo siguiente crea un objeto Rectangle y aplica desplazamientos de 5 y 10 a los valores *x* e *y* respectivamente.

```
import flash.geom.Rectangle;

var rect:Rectangle = new Rectangle(1, 2, 4, 8);
trace(rect.toString()); // (x=1, y=2, w=4, h=8)

rect.offset(16, 32);
trace(rect.toString()); // (x=17, y=34, w=4, h=8)
```

offsetPoint (método Rectangle.offsetPoint)

```
public offsetPoint(pt:Point) : Void
```

Ajusta la posición del objeto Rectangle utilizando un objeto Point como parámetro. Este método es similar al método `Rectangle.offset()`, con la salvedad de que toma un objeto Point como parámetro.

Disponibilidad: ActionScript 1.0; Flash Player 8

Parámetros

pt`:flash.geom.Point` - El objeto Point que se va a utilizar para desplazar este objeto Rectangle.

Ejemplo

El ejemplo siguiente desplaza un objeto Rectangle utilizando los valores de un objeto Point.

```
import flash.geom.Rectangle;
import flash.geom.Point;

var rect:Rectangle = new Rectangle(1, 2, 4, 8);
trace(rect.toString()); // (x=1, y=2, w=4, h=8)

var myPoint:Point = new Point(16, 32);
rect.offsetPoint(myPoint);
trace(rect.toString()); // (x=17, y=34, w=4, h=8)
```

Véase también

`Point (flash.geom.Point)`

Rectangle, constructor

```
public Rectangle(x:Number, y:Number, width:Number, height:Number)
```

Crea un objeto Rectangle nuevo, cuyo ángulo superior izquierdo viene determinado por los valores de los parámetros `x` e `y`. Si invoca esta función constructora sin que existan parámetros, se crea un rectángulo con las propiedades `x`, `y`, `width` y `height`definidas en 0.

Disponibilidad: ActionScript 1.0; Flash Player 8

Parámetros

x`:Number` - La coordenada *x* del ángulo superior izquierdo del rectángulo.

y`:Number` - La coordenada *y* del ángulo superior izquierdo del rectángulo.

width`:Number` - La anchura del rectángulo en píxeles.

height`:Number` - La altura del rectángulo en píxeles.

Ejemplo

El ejemplo siguiente crea un objeto Rectangle con los parámetros especificados.

```
import flash.geom.Rectangle;

var rect:Rectangle = new Rectangle(5, 10, 50, 100);
trace(rect.toString()); // (x=5, y=10, w=50, h=100)
```

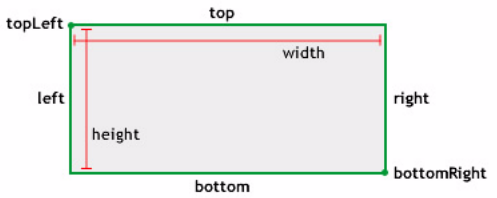
Véase también

`x (propiedad Rectangle.x)`, `y (propiedad Rectangle.y)`, `width (propiedad Rectangle.width)`, `height (propiedad Rectangle.height)`

right (propiedad Rectangle.right)

```
public right : Number
```

Suma de las propiedades `x` y `width`.



Disponibilidad: ActionScript 1.0; Flash Player 8

Ejemplo

El ejemplo siguiente crea un objeto Rectangle y cambia su propiedad `right` de 15 a 30. Observe que `rect.width` también cambia.

```
import flash.geom.Rectangle;

var rect:Rectangle = new Rectangle(5, 5, 10, 10);
trace(rect.width); // 10
trace(rect.right); // 15

rect.right = 30;
trace(rect.width); // 25
trace(rect.right); // 30
```

Véase también

`x (propiedad Rectangle.x)`, `width (propiedad Rectangle.width)`

setEmpty (método Rectangle.setEmpty)

```
public setEmpty() : Void
```

Define todas las propiedades del objeto Rectangle en 0. El objeto Rectangle está vacío si tiene una anchura o altura menor o igual que 0.

Este método establece los valores de `x`, `y`, `width` y `height` en 0.

Disponibilidad: ActionScript 1.0; Flash Player 8

Ejemplo

El ejemplo siguiente crea un objeto Rectangle no vacío y lo convierte en vacío.

```
import flash.geom.Rectangle;

var rect:Rectangle = new Rectangle(5, 10, 50, 100);
trace(rect.isEmpty()); // false
rect.setEmpty();
trace(rect.isEmpty()); // true
```

Véase también

`x (propiedad Rectangle.x)`, `y (propiedad Rectangle.y)`, `width (propiedad Rectangle.width)`, `height (propiedad Rectangle.height)`

size (propiedad Rectangle.size)

```
public size : Point
```

El tamaño del objeto Rectangle, expresado como un objeto Point con valores `width` y `height`.

Disponibilidad: ActionScript 1.0; Flash Player 8

Ejemplo

El ejemplo siguiente crea un objeto Rectangle, recupera el tamaño (`size`), cambia el tamaño (`size`) y define los nuevos valores del objeto Rectangle Es importante recordar que el objeto `Point` utilizado por la propiedad `size` utiliza los valores *x* e *y* para representar las propiedades `width` y `height` del objeto Rectangle.

```
import flash.geom.Rectangle;
import flash.geom.Point;

var rect:Rectangle = new Rectangle(1, 2, 4, 8);
var size:Point = rect.size;
trace(size.x); // 4;
trace(size.y); // 8;

size.x = 16;
```

```
size.y = 32;
rect.size = size;
trace(rect.x); // 1
trace(rect.y); // 2
trace(rect.width); // 16
trace(rect.height); // 32
```

Véase también

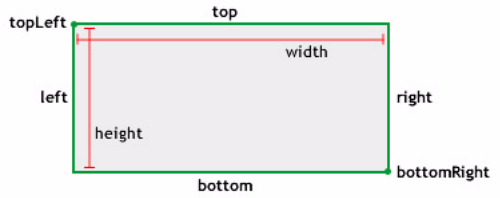
`Point (flash.geom.Point)`

top (propiedad Rectangle.top)

```
public top : Number
```

La coordenada *y* del ángulo superior izquierdo del rectángulo. El cambio del valor de la propiedad `top` de un objeto Rectangle no afecta a las propiedades `x`, `width` o `height`.

El valor de la propiedad `top` es igual a la propiedad `y`.



Disponibilidad: ActionScript 1.0; Flash Player 8

Ejemplo

El ejemplo siguiente cambia el valor de la propiedad `top` de 0 a 10. Observe cómo `rect.y` también cambia.

```
import flash.geom.Rectangle;

var rect:Rectangle = new Rectangle();
trace(rect.top); // 0
trace(rect.y); // 0

rect.top = 10;
trace(rect.top); // 10
trace(rect.y); // 10
```

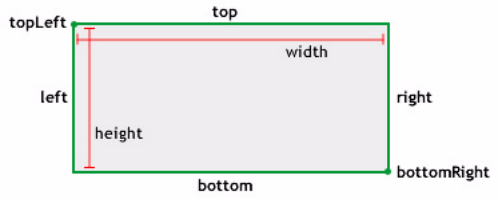
Véase también

`x (propiedad Rectangle.x)`, `y (propiedad Rectangle.y)`, `width (propiedad Rectangle.width)`, `height (propiedad Rectangle.height)`

topLeft (propiedad Rectangle.topLeft)

```
public topLeft : Point
```

La posición del ángulo superior izquierdo del objeto Rectangle viene determinada por los valores *x* e *y* del punto.



Disponibilidad: ActionScript 1.0; Flash Player 8

Ejemplo

El ejemplo siguiente define la propiedad `topLeft` del objeto Rectangle con los valores del objeto Point. Advierta cómo se modifican `rect.x` y `rect.y`.

```
import flash.geom.Rectangle;
import flash.geom.Point;

var rect:Rectangle = new Rectangle();
trace(rect.left); // 0
trace(rect.top); // 0
trace(rect.x); // 0
trace(rect.y); // 0

var myTopLeft:Point = new Point(5, 15);
rect.topLeft = myTopLeft;
trace(rect.left); // 5
trace(rect.top); // 15
trace(rect.x); // 5
trace(rect.y); // 15
```

Véase también

`Point (flash.geom.Point)`, `x (propiedad Rectangle.x)`, `y (propiedad Rectangle.y)`

toString (método Rectangle.toString)

```
public toString() : String
```

Crea y devuelve una cadena en la que se indican las posiciones horizontal y vertical, así como la anchura y la altura, del objeto Rectangle.

Disponibilidad: ActionScript 1.0; Flash Player 8

Valor devuelto

`String` - Una cadena con el valor de cada una de las siguientes propiedades del objeto Rectangle: `x`, `y`, `width` y `height`.

Ejemplo

El ejemplo siguiente concatena una representación de cadena de `rect_1` con algún texto de depuración útil.

```
import flash.geom.Rectangle;

var rect_1:Rectangle = new Rectangle(0, 0, 50, 100);
trace("Rectangle 1 : " + rect_1.toString()); // Rectangle 1 : (x=0, y=0,
  w=50, h=100)
```

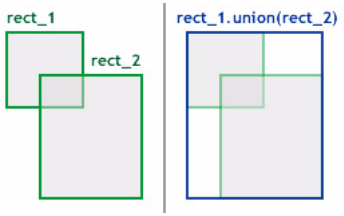
Véase también

`x (propiedad Rectangle.x)`, `y (propiedad Rectangle.y)`, `width (propiedad Rectangle.width)`, `height (propiedad Rectangle.height)`

union (método Rectangle.union)

```
public union(toUnion:Rectangle) : Rectángulo
```

Une dos rectángulos para crear un objeto Rectangle nuevo, y rellena los espacios horizontal y vertical que quedan entre los dos rectángulos.



Disponibilidad: ActionScript 1.0; Flash Player 8

Parámetros

toUnion:flash.geom.Rectangle - El objeto Rectangle que se va a añadir a este objeto Rectangle.

Valor devuelto

flash.geom.Rectangle - Un nuevo objeto Rectangle que es la unión de los dos rectángulos.

Ejemplo

El ejemplo siguiente crea un nuevo objeto Rectangle a partir de la unión de los otros dos.

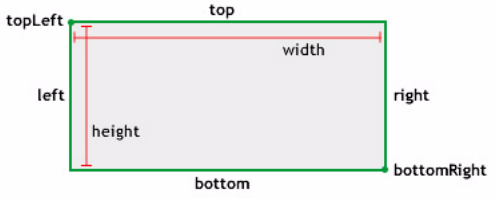
Por ejemplo, utilice un rectángulo con las propiedades x=20, y=50, width=60 y height=30 (20, 50, 60, 30) y otro con propiedades (150, 130, 50, 30). De la unión de estos dos rectángulos resulta otro de mayor tamaño que abarca los dos rectángulos y que tiene las propiedades (20, 50, 180, 110).

```
import flash.geom.Rectangle;

var rect_1:Rectangle = new Rectangle(20, 50, 60, 30);
var rect_2:Rectangle = new Rectangle(150, 130, 50, 30);
var combined:Rectangle = rect_1.union(rect_2);
trace(combined.toString()); // (x=20, y=50, w=180, h=110)
```

width (propiedad Rectangle.width)

```
public width : Number
```

La anchura del rectángulo en píxeles. El cambio del valor de la propiedad width de un objeto Rectangle no afecta a las propiedades x, y o height.



Disponibilidad: ActionScript 1.0; Flash Player 8

Ejemplo

El ejemplo siguiente crea un objeto Rectangle y cambia su propiedad `width` de 10 a 20. Observe que `rect.right` también cambia.

```
import flash.geom.Rectangle;

var rect:Rectangle = new Rectangle(5, 5, 10, 10);
trace(rect.width); // 10
trace(rect.right); // 15

rect.width = 20;
trace(rect.width); // 20
trace(rect.right); // 25
```

Véase también

`x (propiedad Rectangle.x)`, `y (propiedad Rectangle.y)`, `height (propiedad Rectangle.height)`

x (propiedad Rectangle.x)

```
public x : Number
```

La coordenada *x* del ángulo superior izquierdo del rectángulo. El cambio del valor de la propiedad *x* de un objeto Rectangle no afecta a las propiedades `y`, `width` o `height`.

La propiedad `x` es igual a la propiedad `left`.

Disponibilidad: ActionScript 1.0; Flash Player 8

Ejemplo

El ejemplo crea un objeto Rectangle y define el valor de la propiedad `x` en 10. Observe que `rect.left` también cambia.

```
import flash.geom.Rectangle;

var rect:Rectangle = new Rectangle();
trace(rect.x); // 0
trace(rect.left); // 0

rect.x = 10;
trace(rect.x); // 10
trace(rect.left); // 10
```

Véase también

`left (propiedad Rectangle.left)`

y (propiedad Rectangle.y)

```
public y : Number
```

La coordenada *y* del ángulo superior izquierdo del rectángulo. El cambio del valor de la propiedad `y` de un objeto Rectangle no afecta a las propiedades `x`, `width` o `height`.

La propiedad `y` es igual a la propiedad `top`.

Disponibilidad: ActionScript 1.0; Flash Player 8

Ejemplo

El ejemplo crea un objeto Rectangle y define el valor de la propiedad `y` en 10. Observe que `rect.top` también cambia.

```
import flash.geom.Rectangle;

var rect:Rectangle = new Rectangle();
trace(rect.y); // 0
trace(rect.top); // 0

rect.y = 10;
trace(rect.y); // 10
trace(rect.top); // 10
```

Véase también

`x (propiedad Rectangle.x)`, `width (propiedad Rectangle.width)`, `height (propiedad Rectangle.height)`, `top (propiedad Rectangle.top)`

security (System.security)

```
Object
  |
  +-System.security

public class security
extends Object
```

La clase System.security contiene métodos que especifican cómo pueden comunicarse entre sí archivos SWF de diferentes dominios.

Para más información, consulte las referencias siguientes:

- Capitulo 17, "Aspectos básicos de la seguridad", de *Aprendizaje de ActionScript 2.0 en Flash*
- El documento técnico sobre la seguridad de Flash Player 8 en http://www.macromedia.com/go/fp8_security

- El documento técnico sobre interfaces API relativas a seguridad de Flash Player 8 en http://www.macromedia.com/go/fp8_security_apis

Disponibilidad: ActionScript 1.0; Flash Player 6

Resumen de propiedades

Modificadores	Propiedad	Descripción
`static`	`sandboxType:String` [read-only]	Indica el tipo de entorno limitado de seguridad en el que se ejecuta el archivo SWF que realiza la llamada.

Propiedades heredadas de la clase Object

`constructor (propiedad Object.constructor), __proto__ (propiedad Object.__proto__), prototype (propiedad Object.prototype), __resolve (propiedad Object.__resolve)`

Resumen de métodos

Modificadores	Firma	Descripción
`static`	`allowDomain(domain1:String) : Void`	Permite a los archivos SWF y HTML de los dominios identificados acceder a los objetos y variables del archivo SWF que contiene la llamada `allowDomain()`.
`static`	`allowInsecureDomain(domain:String) : Void`	Permite a los archivos SWF y HTML de los dominios identificados acceder a los objetos y variables del archivo SWF, que se aloja mediante el protocolo HTTPS.
`static`	`loadPolicyFile(url:String) : Void`	Carga un archivo de política para distintos dominios de una ubicación especificada por el parámetro `url`.

Métodos heredados de la clase Object

`addProperty (método Object.addProperty), hasOwnProperty (método Object.hasOwnProperty), isPropertyEnumerable (método Object.isPropertyEnumerable), isPrototypeOf (método Object.isPrototypeOf), registerClass (método Object.registerClass), toString (método Object.toString), unwatch (método Object.unwatch), valueOf (método Object.valueOf), watch (método Object.watch)`

allowDomain (método security.allowDomain)

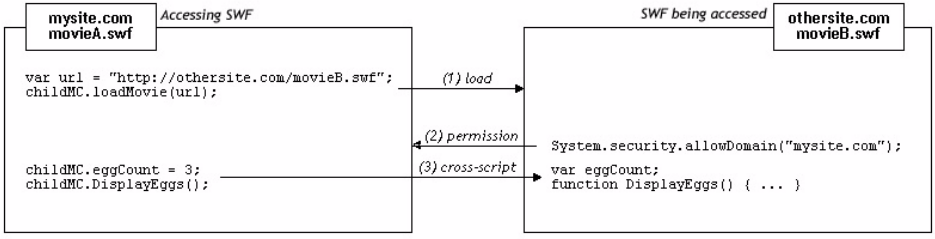
```
public static allowDomain(domain1:String) : Void
```

Permite a los archivos SWF y HTML de los dominios identificados acceder a los objetos y variables del archivo SWF que contiene la llamada `allowDomain()`.

Si hay dos archivos SWF en el mismo dominio (por ejemplo, http://mysite.com/movieA.swf y http://mysite.com/movieB.swf), el archivo movieA.swf puede examinar y modificar variables, objetos, propiedades, métodos y demás en movieB.swf, y movieB.swf puede hacer lo mismo en movieA.swf. Esto se denomina *uso de scripts entre películas*.

Si los dos archivos SWF están en dominios diferentes (por ejemplo, http://mysite.com/movieA.swf y http://othersite.com/movieB.swf), de forma predetermina Flash Player no permite a movieA.swf usar scripts en movieB.swf, ni viceversa. Un archivo SWF concede permiso a otros archivos SWF de dominios diferentes para usar scripts en él llamando a `System.security.allowDomain()`. Esto se denomina *creación de scripts en varios dominios*. Al realizar una llamada a `System.security.allowDomain("mysite.com")`, movieB.swf otorga permisos a movieA.swf para usar scripts en movieB.swf.

En cualquier operación entre dominios intervienen dos partes, por lo que es importante tener claro el origen de cada una. En esta sección, llamaremos *parte que accede* a la que usa scripts entre películas (normalmente el archivo SWF que accede a otro) y *parte a la que se accede* a la otra (por lo general, el archivo SWF al que se accede). Para continuar con este ejemplo, cuando movieA.swf usa scripts en movieB.swf, movieA.swf es la parte que accede y movieB.swf la parte a la que se accede.



Los permisos entre dominios establecidos con `System.security.allowDomain()` son asimétricos. En el ejemplo anterior, movieA.swf puede usar scripts en movieB.swf, pero movieB.swf no puede hacerlo en movieA.swf porque este último no ha llamado a `System.security.allowDomain()` con el fin de conceder permiso a othersite.com para que se usen scripts en movieA.swf. Puede establecer permisos simétricos si ambos archivos SWF llaman a System.security.allowDomain().

Flash Player protege los archivos SWF no sólo de los scripts creados en varios dominios por otros archivos SWF, sino también de los originados por archivos HTML. Los scripts de HTML se pueden usar en SWF mediante el empleo de interfaces API anteriores del navegador Flash, como `SetVariable`, o invocando las funciones callback definidas a través de `ExternalInterface.addCallback()`. Cuando el uso de scripts de HTML en SWF traspasa los límites del dominio, el archivo SWF debe llamar a `System.security.allowDomain()` para evitar que la operación falle, tanto si es la parte que accede como si es la parte a la que se accede.

Cuando se especifica una dirección IP como parámetro en `System.security.allowDomain()`, no se permite el acceso de las partes con origen en la dirección IP especificada. Sólo se permite el acceso de las partes que se cargaron especificando explícitamente esa dirección IP en la URL, en lugar del nombre de dominio asignado a la dirección IP.

Diferencias específicas entre versiones Las reglas de seguridad entre dominios de Flash Player han evolucionado de una versión a otra. En la tabla siguiente se resumen las diferencias.

<table>
<tr><th>Versión de SWF más reciente que interviene en el uso de scripts entre películas.</th><th>¿Se necesita allowDomain()?</th><th>¿Se necesita allowInsecureDomain()?</th><th>¿Qué archivo SWF debe llamar a allowDomain() o allowInsecureDomain()?</th><th>¿Qué se puede especificar en allowDomain() o allowInsecureDomain()?</th></tr>
<tr><td>5 o anterior</td><td>No</td><td>No</td><td colspan="2">N/D</td></tr>
<tr><td>6</td><td>Sí, si los superdominios no coinciden.</td><td rowspan="3">Sí, si se realiza el acceso de HHTP en HTTPS (incluso si los dominios coinciden exactamente).</td><td>El archivo SWF al que se accede o cualquier SWF con el mismo superdominio que éste.</td><td rowspan="2">■ Dominio basado en texto (mysite.com)
■ Dirección IP (192.168.1.1)</td></tr>
<tr><td>7</td><td>Sí, si los dominios no coinciden exactamente.</td><td>El archivo SWF al que se accede o cualquier SWF con el mismo dominio que éste.</td></tr>
<tr><td>8 o posterior</td><td></td><td>El archivo SWF al que se accede.</td><td>■ Dominio basado en texto (mysite.com)
■ Dirección IP (192.168.1.1)
■ Comodín (*)</td></tr>
</table>

Las versiones que controlan el comportamiento de Flash Player son las *versiones de SWF* (versión publicada de un SWF), no las versiones del propio Flash Player. Por ejemplo, cuando Flash Player 8 reproduce un archivo SWF de la versión 7, presenta un comportamiento coherente con la versión 7. Esta práctica evita que las actualizaciones del reproductor modifiquen el comportamiento de `System.security.allowDomain()` en los archivos SWF desplegados.

En la tabla anterior, la columna de la versión se reserva para la versión de SWF más reciente que interviene en el uso de scripts entre películas. Flash Player determina su comportamiento en función de la versión del archivo SWF que accede o de la versión del SWF al que se accede, las que sea más reciente.

Los párrafos siguientes ofrecen más detalles acerca de los cambios en la seguridad de Flash Player que están relacionados con `System.security.allowDomain()`.

Versión 5. Creación de scripts en varios dominios sin restricciones.

Versión 6. Se incorpora la función de seguridad para la creación de scripts entre dominios. Flash Player prohíbe la creación de scripts entre dominios de forma predeterminada; `System.security.allowDomain()` puede permitirla. Para determinar si dos archivos pertenecen al mismo dominio, Flash Player utilizar el superdominio de cada archivo, que es el nombre exacto del host de la URL del archivo, menos el primer segmento, hasta un mínimo de dos segmentos. Por ejemplo, el superdominio de www.mysite.com es simplemente mysite.com. Este ejemplo permitirá el uso de scripts entre los archivos SWF de www.mysite.com y store.mysite.com sin llamar a `System.security.allowDomain()`.

Versión 7. La coincidencia de superdominios se cambia por la coincidencia exacta de dominios. Sólo se permite el uso de scripts entre dos archivos si los nombres del host en las URL son idénticos; de lo contrario, se requiere una llamada a `System.security.allowDomain()`. De forma predeterminada, ya no se permite que los archivos cargados desde direcciones URL que no sean HTTPS usen scripts en archivos con URL HTTPS, aunque los archivos se carguen desde el mismo dominio exactamente. Esta restricción ayuda a proteger los archivos HTTPS, ya que los archivos no HTTPS son susceptibles de modificarse durante la descarga. Cualquier cambio malintencionado en un archivo no HTTPS podría dañar los archivos HTTPS, que de lo contrario serían inmunes a tales ataques. Aunque `System.security.allowInsecureDomain()` se ha introducido con el fin de permitir la desactivación voluntaria de esta restricción en los archivos SWF HTTPS a los que se accede, Macromedia desaconseja utilizar `System.security.allowInsecureDomain()`.

Versión 8. Los cambios afectan a dos áreas principalmente:

- La llamada a `System.security.allowDomain()` permite ahora operaciones de script entre películas sólo cuando el archivo SWF al que se accede es el archivo SWF que realizó la llamada a `System.security.allowDomain()`. Es decir, un archivo SWF que realiza una llamada a `System.security.allowDomain()` ahora sólo permite que se acceda al propio archivo. En versiones anteriores, la llamada a `System.security.allowDomain()` permitía operaciones de script entre películas en las que el archivo SWF al que se accedía podía ser cualquier archivo SWF del mismo dominio que el archivo que llamó a `System.security.allowDomain()`. Antes la llamada a `System.security.allowDomain()` abría todo el dominio del archivo SWF que realizaba la llamada.
- Se ha añadido la posibilidad de utilizar valores comodín con `System.security.allowDomain("*")` y `System.security.allowInsecureDomain("*")`. El valor comodín (*) permite operaciones de script entre películas en las que el archivo que accede puede ser cualquier archivo de cualquier procedencia. El comodín es un valor que otorga un permiso global. Los permisos de comodín pueden resultar útiles en general, pero en particular requieren la activación de ciertos tipos de operaciones en las nuevas reglas de seguridad de los archivos locales de Flash Player 8. En concreto, para que un archivo SWF local con permisos de acceso de red cree scripts en un archivo SWF de Internet, este último debe llamar a `System.security.allowDomain("*")`, lo que refleja que se desconoce el origen del archivo SWF local. (Si el archivo SWF de Internet al que se accede se carga desde una dirección URL HTTPS, el archivo debe llamar a `System.security.allowInsecureDomain("*")`.)

Es posible que, en ocasiones, se encuentre con la siguiente situación: El usuario carga un archivo SWF secundario de un dominio diferente y desea que el archivo SWF secundario pueda crear un script en el archivo SWF principal, pero no conoce el dominio final del que procederá el archivo SWF secundario. Esto puede ocurrir, por ejemplo, cuando se utilizan redirecciones de reparto de carga o servidores de terceros.

En esta situación, puede utilizar la propiedad `MovieClip._url` como parámetro de este método. Por ejemplo, si carga un archivo SWF en el clip de película `my_mc`, podrá llamar a `System.security.allowDomain(my_mc._url)`. Si lo hace, asegúrese de esperar a que el archivo SWF empiece a cargarse en `my_mc`, ya que la propiedad `_url` no dispone de su valor final correcto hasta este momento. La mejor forma de determinar cuándo empieza a cargarse un archivo SWF secundario es mediante `MovieClipLoader.onLoadStart`.

También puede darse la situación contraria, es decir, podría crear un archivo SWF secundario en el que quiera que su archivo principal use scripts, pero no conoce cuál será el dominio de su archivo principal. En este caso, llame a `System.security.allowDomain(_parent._url)` desde el archivo SWF secundario. No es necesario esperar a que se cargue el archivo SWF principal; el archivo principal ya estará cargado en el momento en que se cargue el archivo secundario.

Si está publicando en Flash Player 8, puede resolver estas situaciones llamando a `System.security.allowDomain("*")`. No obstante, a veces puede tratarse de una solución rápida pero peligrosa, ya que será posible acceder al archivo SWF que hace la llamada desde cualquier dominio. Por lo general resulta más seguro utilizar la propiedad `_url`.

Para más información, consulte las referencias siguientes:

- Capitulo 17, "Aspectos básicos de la seguridad", de *Aprendizaje de ActionScript 2.0 en Flash*
- El libro blanco sobre la seguridad de Flash Player 8
- El libro blanco sobre interfaces API relativas a seguridad de Flash Player 8

Disponibilidad: ActionScript 1.0; Flash Player 6

Parámetros

`domain1``:String` - Una o varias cadenas que especifican dominios que tienen acceso a objetos y variables del archivo SWF que contiene la llamada a `System.Security.allowDomain()`. Los dominios pueden presentar formatos diferentes:

- `"domain.com"`
- `"http://domain.com"`
- `"http://IPaddress"`
- (Sólo Flash Player 8) `"*"`
  Puede pasar un comodín ("*") a `System.security.allowDomain()` para permitir el acceso al archivo SWF que hace la llamada a todos los dominios, incluidos servidores locales. Antes de utilizar el comodín, asegúrese de que quiere permitir una acceso tan generalizado al archivo SWF que llama. Consulte el apartado incluido en la descripción principal de este método.

Ejemplo

El archivo SWF ubicado en www.macromedia.com/MovieA.swf contiene las líneas siguientes:

```
System.security.allowDomain("www.shockwave.com");
loadMovie("http://www.shockwave.com/MovieB.swf", my_mc);
```

Como MovieA contiene la llamada `allowDomain()`, MovieB puede acceder a los objetos y variables de MovieA. Si MovieA no contuviera esta llamada, la función de seguridad de Flash Player impediría que MovieB accediese a los objetos y variables de MovieA.

Véase también

`addCallback (método ExternalInterface.addCallback)`, `onLoadComplete (detector de eventos MovieClipLoader.onLoadComplete)`, `_parent (propiedad MovieClip._parent)`, `_url (propiedad MovieClip._url)`, `allowInsecureDomain (método security.allowInsecureDomain)`

allowInsecureDomain (método security.allowInsecureDomain)

```
public static allowInsecureDomain(domain:String) : Void
```

Permite a los archivos SWF y HTML de los dominios identificados acceder a los objetos y variables del archivo SWF, que se aloja mediante el protocolo HTTPS. Macromedia no recomienda utilizar este método; consulte "Consideraciones de seguridad" a continuación.

Este método funciona de la misma manera que `System.security.allowDomain()`, pero además admite operaciones en las que la parte que accede se carga con un protocolo distinto de HTTPS y la parte a la que se accede se carga con protocolo HTTPS. En Flash Player 7 y versiones posteriores, no se permite el uso de scripts de archivos no HTTPS en archivos HTTPS. El método `allowInsecureDomain()` elimina esta restricción cuando el archivo SWF HTTPS al que se accede lo utiliza.

Utilice `allowInsecureDomain()` solamente para permitir el uso de scripts de archivos no HTTPS en archivos HTTPS. También se puede usar para permitir la creación de scripts cuando el archivo no HTTPS que accede y el archivo HTTPS al que se accede pertenecen al mismo dominio; por ejemplo, si un archivo SWF ubicado en http://mysite.com quiere usar scripts en https://mysite.com. No utilice este método para permitir el uso de scripts entre archivos no HTTPS, entre archivos HTTPS o de archivos HTTPS en archivos no HTTPS. En esos casos debe utilizar `allowDomain()`.

Consideraciones de seguridad: Aunque Flash Player ofrece `allowInsecureDomain()` para maximizar la flexibilidad, Macromedia desaconseja su uso. Mediante la llamada a `allowInsecureDomain` se reduce la eficacia de algunas de las protecciones que proporciona a todos los usuarios la descarga de archivos a través de HTTPS. En la situación siguiente se demuestra cómo `allowInsecureDomain()` puede afectar a la seguridad si no se utiliza con mucho cuidado.

Nota: La información siguiente sólo hace referencia a una posible situación y tiene por objeto ayudar a entender `allowInsecureDomain()` mediante un ejemplo práctico de uso de scripts entre películas. No abarca todos los aspectos relacionados con la arquitectura de seguridad, y sólo debe utilizarse como información de referencia. En Macromedia Developer Center podrá encontrar información detallada sobre Flash Player y la seguridad. Para más información, consulte http://www.macromedia.com/devnet/security/.

Imagínese que está creando un sitio de comercio electrónico que consta de dos componentes: un catálogo, que no requiere seguridad porque contiene únicamente la información pública, y un componente de carrito de la compra/finalización de la compra que debe asegurar para proteger los datos financieros y personales del usuario. Supongamos que quiere que el catálogo esté disponible en http://mysite.com/catalog.swf, y el carro en https://mysite.com/cart.swf. Uno de los requisitos del sitio es evitar que un tercero pueda robar el número de la tarjeta de crédito de los usuarios aprovechándose de una carencia en la arquitectura de seguridad.

Imagine que un atacante de una parte intermedia interviene entre el servidor y los usuarios e intenta robar los números de las tarjetas de crédito de los usuarios que acceden a la aplicación del carrito de la compra. La parte intermedia puede ser, por ejemplo, un ISP sin escrúpulos que utilizan algunos de sus usuarios o un administrador malintencionado del lugar de trabajo del usuario; es decir, cualquiera que sea capaz de ver o alterar los paquetes de red que circulan por un sitio Internet público entre sus usuarios y sus servidores. Este atributo no es necesario.

Si cart.swf utiliza el protocolo HTTPS para trasmitir la información de la tarjeta de crédito a sus servidores, la parte intermedia no puede obtener directamente esta información de los paquetes de red porque las transmisiones HTTPS están cifradas. Sin embargo, el atacante puede utilizar una técnica diferente: alterar el contenido de uno de los archivos SWF cuando se envía al usuario y reemplazarlo con la versión modificada, que transmite la información del usuario a un servidor diferente, propiedad del atacante.

Entre otras cosas, el protocolo HTTPS impide que los ataques de "modificación" den resultados porque las transmisiones HTTPS, además de estar cifradas, resisten cualquier intento de ataque: si una parte intermedia altera un paquete, la parte receptora detecta la modificación y descarta el paquete. En este caso, el atacante no puede modificar el archivo cart.swf porque se transmite a través de HTTPS.

Sin embargo, supongamos que quiere contar con botones en catalog.swf, disponible en HTTP, para añadir artículos al carrito de la compra de cart.swf, disponible en HTTPS. Para conseguirlo, cart.swf llama a `allowInsecureDomain()`, que permite a catalog.swf crear scripts en cart.swf. Esta acción tiene una consecuencia no deseada: ahora el hipotético atacante puede alterar catalog.swf cuando el usuario lo descarga inicialmente, puesto que el protocolo de transmisión de catalog.swf es HTTP y no es resistente a ataques. Como cart.swf contiene una llamada a `allowInsecureDomain()`, el archivo catalog.swf modificado puede crear scripts en cart.swf. El archivo catalog.swf modificado puede utilizar ActionScript para acceder a las variables de cart.swf, y leer tanto la información de la tarjeta de crédito como otros datos confidenciales del usuario. Además, puede enviar estos datos al servidor de un atacante.

Aunque, obviamente, esta no es la solución deseada, todavía quiere permitir el uso de scripts entre películas entre los dos archivos SWF de su sitio. A continuación se exponen las dos formas posibles de rediseñar este hipotético sitio de comercio electrónico para evitar `allowInsecureDomain()`:

- Ponga a disposición todos los SWF de la aplicación a través de HTTPS. Esta es, sin duda, la solución más sencilla y fiable. En la situación descrita, tanto catalog.swf como cart.swf estarán disponibles a través de HTTPS. Puede detectar un ligero aumento del consumo del ancho de banda y de la carga de la CPU del servidor al cambiar un archivo como catalog.swf de HTTP a HTTPS. Además, es posible que los usuarios tengan que soportar una pequeña ralentización de los tiempos de carga de la aplicación. Por eso es preciso experimentar con servidores reales para determinar el alcance de estos efectos, que normalmente no suelen suponer aumentos de más del 10-20% en cada caso, y a veces ni se producen. Los resultados pueden mejorar mediante el uso de hardware o software de aceleración de HTTPS en los servidores. Una de las ventajas más importantes que ofrece la disponibilidad de los archivos SWF que operan conjuntamente a través de HTTPS es que puede utilizar una URL HTTPS como URL principal en el navegador del usuario sin que el navegador genere mensajes de advertencia de contenido mixto. Además, el icono en forma de candado del navegador queda visible, con lo que el usuario dispone de un indicador de seguridad común y fiable.

- Use scripts de HTTPS en HTTP, en lugar de a la inversa (HTTP en HTTPS). En la situación descrita, podría almacenar el contenido del carrito de la compra del usuario en catalog.swf y hacer que cart.swf gestionase el proceso de finalización de la compra solamente. En el momento de finalizar la compra, cart.swf podría recuperar el contenido del carrito desde las variables de ActionScript incluidas en catalog.swf. La restricción de uso de scripts de HTTP en HTTPS no son recíprocas; mientras que no se puede permitir que un archivo catalog.swf transmitido por HTTP use scripts en un archivo cart.swf de HTTPS sin excluir riesgos, el archivo cart.swf de HTTPS puede usar scripts en el archivo catalog.swf de HTTP. Este método está más expuesto a contingencias que el método en el que todo está en HTTPS. Dada su vulnerabilidad a los ataques, debe tener cuidado con los archivos SWF transmitidos por HTTP. Por ejemplo, cuando cart.swf recupera la variable de ActionScript que describe el contenido del carrito, el código de ActionScript incluido en cart.swf no puede garantizar que el valor de esta variable tenga el formato previsto. Por tanto, debe verificar con atención que el carrito no contenga datos no válidos que podrían inducir a cart.swf a realizar una acción no deseada. También debe asumir el riesgo de que una parte intermedia altere catalog.swf y pueda proporcionar datos válidos pero inexactos a cart.swf, por ejemplo, mediante la introducción de artículos en el carrito del usuario. El procedimiento normal de finalización de la compra reduce este riesgo en cierta medida, ya que presenta al usuario el contenido del carrito y el coste total de la compra para su aprobación. Sin embargo, el riesgo sigue existiendo.

En los navegadores Web se ha impuesto la separación entre los archivos HTTPS y no HTTPS durante años. Además, la situación descrita es un buena razón para aplicar esta restricción. Flash Player ofrece la posibilidad de omitir esta restricción de seguridad cuando es absolutamente imprescindible, pero asegurándose de considerar con atención las consecuencias antes de pasar a la acción.

Para más información, consulte las referencias siguientes:

- Capitulo 17, "Aspectos básicos de la seguridad", de *Aprendizaje de ActionScript 2.0 en Flash*
- El documento técnico sobre la seguridad de Flash Player 8 en http://www.macromedia.com/go/fp8_security
- El documento técnico sobre interfaces API relativas a seguridad de Flash Player 8 en http://www.macromedia.com/go/fp8_security_apis

Disponibilidad: ActionScript 1.0; Flash Player 7

Parámetros

domain:String - Un nombre de dominio exacto, como www.myDomainName.com o store.myDomainName.com. Sólo en Flash Player 8 puede pasar un comodín ("*") a System.security.allowInsecureDomain() para permitir el acceso al archivo SWF que hace la llamada de todos los dominios, incluidos servidores locales. No utilice el comodín a menos que esté seguro de que desea permitir el acceso al archivo SWF HTTPS de *todos* los dominios, incluidos servidores locales.

Ejemplo

En el ejemplo siguiente va a alojar un examen de matemáticas en un dominio seguro, de forma que sólo puedan acceder a él los estudiantes registrados. También va a incluir en un dominio seguro la serie de archivos SWF que ha creado, en los que se ilustran determinados conceptos. Quiere que los estudiantes accedan al examen desde el archivo SWF que contiene la información relacionada con un concepto.

```
// This SWF file is at https://myEducationSite.somewhere.com/mathTest.swf
// Concept files are at http://myEducationSite.somewhere.com
System.security.allowInsecureDomain("myEducationSite.somewhere.com");
```

Véase también

allowDomain (método security.allowDomain), exactSettings (propiedad System.exactSettings)

loadPolicyFile (método security.loadPolicyFile)

```
public static loadPolicyFile(url:String) : Void
```

Carga un archivo de política para distintos dominios de una ubicación especificada por el parámetro url. Flash Player utiliza archivos de política como mecanismo de permiso para permitir que las películas Flash carguen datos desde servidores que no sean el suyo propio.

Flash Player 7.0.14.0 buscaba archivos de política sólo en una ubicación: /crossdomain.xml en el servidor al que se había solicitado una petición de carga de datos. Para un intento de conexión XMLSocket, Flash Player 7.0.14.0 buscaba /crossdomain.xml en un servidor HTTP (puerto 80) del subdominio en el que se intentaba realizar la conexión XMLSocket. Flash Player 7.0.14.0 (y todos los reproductores anteriores) también limitaban las conexiones XMLSocket a los puertos 1024 y posteriores.

Con la incorporación de System.security.loadPolicyFile(), Flash Player 7.0.19.0 puede cargar archivos de política de ubicaciones arbitrarias, como se muestra en el siguiente ejemplo:

```
System.security.loadPolicyFile("http://foo.com/sub/dir/pf.xml");
```

De este modo, Flash Player recupera un archivo de política de la URL especificada. Los permisos otorgados por el archivo de política de dicha ubicación se aplicarán a todo el contenido del mismo nivel o de un nivel inferior de la jerarquía de directorios virtual del servidor. El siguiente código es continuación del ejemplo anterior:

```
loadVariables("http://foo.com/sub/dir/vars.txt") // allowed
loadVariables("http://foo.com/sub/dir/deep/vars2.txt") // allowed
loadVariables("http://foo.com/elsewhere/vars3.txt") // not allowed
```

Puede utilizar `loadPolicyFile()` para cargar los archivos de política que desee. Al considerar una petición que requiere un archivo de política, Flash Player siempre espera a que termine la descarga de archivos de política antes de denegar una petición. Como opción final, si no hay ningún archivo de política especificado con `loadPolicyFile()` que autorice una petición, Flash Player consulta la ubicación predeterminada original: /crossdomain.xml.

La utilización del protocolo `xmlsocket` con un número de puerto específico le permite recuperar archivos de política directamente de un servidor XMLSocket, como se muestra en el siguiente ejemplo:

```
System.security.loadPolicyFile("xmlsocket://foo.com:414");
```

De este modo, Flash Player intenta recuperar un archivo de política desde el host y el puerto especificados. Puede utilizarse cualquier puerto, no sólo los puertos 1024 y superiores. Tras establecer una conexión con el puerto especificado, Flash Player transmite `<policy-file-request />`, terminada por un byte `null`. Puede configurarse un servidor XMLSocket para que responda a peticiones de archivos de política y a conexiones XMLSocket normales en el mismo puerto; en este caso, el servidor deberá esperar `<policy-file-request />` antes de transmitir un archivo de política. También puede configurarse un servidor para que responda a peticiones de archivos de política en un puerto distinto de las conexiones estándar; en este caso, el servidor puede enviar un archivo de política en cuanto se establece una conexión en el puerto del archivo de política dedicado. El servidor debe enviar un byte nulo para terminar un archivo de política y más tarde puede cerrar la conexión; si el servidor no cierra la conexión, Flash Player la cerrará después de recibir el byte `null` de terminación.

Un archivo de política proporcionado por un servidor XMLSocket tiene la misma sintaxis que cualquier otro archivo de política, pero debe especificar también los puertos a los que se concede el acceso. Cuando un archivo de política procede de un puerto inferior a 1024, dicho archivo puede conceder acceso a cualquier puerto; cuando un archivo de política procede del puerto 1024 o superior, sólo puede conceder acceso a otros puertos 1024 y superiores. Los puertos permitidos se especifican en el atributo `"to-ports"` de la etiqueta `<allow-access-from>`. Los números de puerto únicos, los intervalos de puertos y los comodines están permitidos. El siguiente ejemplo muestra un archivo de política XMLSocket:

```
<cross-domain-policy>
<allow-access-from domain="*" to-ports="507" />
<allow-access-from domain="*.foo.com" to-ports="507,516" />
<allow-access-from domain="*.bar.com" to-ports="516-523" />
<allow-access-from domain="www.foo.com" to-ports="507,516-523" />
<allow-access-from domain="www.bar.com" to-ports="*" />
</cross-domain-policy>
```

Un archivo de política procedente de la ubicación predeterminada original (/crossdomain.xml en un servidor HTTP del puerto 80) autoriza implícitamente el acceso a todos los puertos 1024 y superiores. No hay ninguna manera de recuperar un archivo de política que autorice operaciones de XMLSocket desde cualquier otra ubicación de un servidor HTTP; las ubicaciones personalizadas para los archivos de política XMLSocket deben encontrarse en un servidor XMLSocket.

Dado que la capacidad de conectar con puertos inferiores a 1024 es una novedad, para autorizar esta conexión siempre es necesario un archivo de política cargado con `loadPolicyFile()`, incluso cuando un clip de película se conecta a su propio subdominio.

Para más información, consulte las referencias siguientes:

- Capitulo 17, "Aspectos básicos de la seguridad", de *Aprendizaje de ActionScript 2.0 en Flash*
- El documento técnico sobre la seguridad de Flash Player 8 en http://www.macromedia.com/go/fp8_security
- El documento técnico sobre interfaces API relativas a seguridad de Flash Player 8 en http://www.macromedia.com/go/fp8_security_apis

Disponibilidad: ActionScript 1.0; Flash Player 7,0,19,0

Parámetros

url`:String` - Una cadena: la URL en la que se encuentra el archivo de política de varios dominios que se va a cargar.

sandboxType (propiedad security.sandboxType)

```
public static sandboxType : String [read-only]
```

Indica el tipo de entorno limitado de seguridad en el que actúa el archivo SWF.

`System.security.sandboxType` tiene uno de los valores siguientes:

- `remote`: este archivo SWF procede de una URL de Internet y se regirá por reglas de entorno limitado basado en dominio.
- `localWithFile`: este SWF es un archivo local y no es de confianza para el usuario. No se ha publicado en una designación de red. Este archivo SWF puede leer de orígenes de datos locales, pero no puede comunicarse con Internet.
- `localWithNetwork`: este SWF es un archivo local y no es de confianza para el usuario. Se ha publicado en una designación de red. Este archivo SWF puede comunicarse con Internet, pero no puede leer de orígenes de datos locales.
- `localTrusted`: este SWF es un archivo local y el usuario ha determinado que es de confianza, mediante el Administrador de configuración o un archivo de configuración FlashPlayerTrust. Este archivo SWF puede leer de orígenes de datos locales y puede comunicarse con Internet.

Observe que esta propiedad puede examinarse desde un archivo SWF de cualquier versión, pero sólo se admite en Flash Player 8 o versiones posteriores. Esta condición especial significa que puede examinar la propiedad, por ejemplo, desde un archivo SWF de la versión 7 reproducido en Flash Player 8. El hecho de ser compatible con todas las versiones implica que si publica para una versión anterior a la 8, en el momento de la publicación no sabrá si se admitirá esta propiedad en el momento de la reproducción. Por lo tanto, en un archivo SWF de la versión 7 o anterior, puede ocurrir que esta propiedad tenga un valor undefined, lo que sólo debería pasar cuando la versión del reproductor (indicada en `System.capabilities.version`) es anterior a la 8. En este caso, puede determinar el tipo de entorno limitado según la URL de su archivo SWF, sea un archivo local o no. Si es así, puede asumir que Flash Player clasificará su SWF como `"localTrusted"` (hasta Flash Player 8, así se trataba todo el contenido local). En caso contrario, puede asumir que Flash Player clasificará su archivo SWF como `"remote"`.

Para más información, consulte lo siguiente:

- Capítulo 17, "Aspectos básicos de la seguridad," en *Aprendizaje de ActionScript 2.0 en Flash*
- El informe sobre la seguridad de Flash Player 8
- El documento técnico sobre interfaces API relativas a seguridad de Flash Player 8 en http://www.macromedia.com/go/fp8_security_apis

Disponibilidad: ActionScript 1.0; Flash Player 8

Selection

```
Object
  |
  +-Selection
```

```
public class Selection
extends Object
```

La clase Selection le permite establecer y controlar el campo de texto en el que se encuentra el punto de inserción (es decir, el campo seleccionado). Los índices de espacio de selección están basados en cero (por ejemplo, la primera posición es 0, la segunda posición es 1, etc.).

No existe función constructora para la clase Selection, ya que sólo puede haber un campo seleccionado en todo momento.

Disponibilidad: ActionScript 1.0; Flash Player 5

Resumen de propiedades

Propiedades heredadas de la clase Object

`constructor (propiedad Object.constructor), __proto__ (propiedad Object.__proto__), prototype (propiedad Object.prototype), __resolve (propiedad Object.__resolve)`

Resumen de eventos

Evento	Descripción
`onSetFocus = function([oldfocus:Object], [newfocus:Object]) {}`	Recibe una notificación cuando cambia la selección de entrada.

Resumen de métodos

Modificadores	Firma	Descripción
`static`	`addListener(listener:Object) : Void`	Registra un objeto para recibir notificaciones de cambio de selección con el teclado.
`static`	`getBeginIndex() : Number`	Devuelve el índice al comienzo del espacio de selección.
`static`	`getCaretIndex() : Number`	Devuelve el índice de la posición del punto de inserción intermitente (intercalación).

Modificadores	Firma	Descripción
static	getEndIndex() : Number	Devuelve el índice final del espacio de selección que actualmente está seleccionado con el teclado.
static	getFocus() : String	Devuelve una cadena que especifica la ruta de destino del objeto que está seleccionado con el teclado.
static	removeListener(listener:Object) : Boolean	Elimina un objeto registrado previamente con Selection.addListener().
static	setFocus(newFocus:Object) : Boolean	Selecciona con el teclado el campo de texto, botón o clip de película seleccionable (editable) especificado por el parámetro newFocus.
static	setSelection(beginIndex:Number, endIndex:Number) : Void	Establece el espacio de selección del campo de texto seleccionado con el teclado actualmente.

Métodos heredados de la clase Object

addProperty (método Object.addProperty), hasOwnProperty (método Object.hasOwnProperty), isPropertyEnumerable (método Object.isPropertyEnumerable), isPrototypeOf (método Object.isPrototypeOf), registerClass (método Object.registerClass), toString (método Object.toString), unwatch (método Object.unwatch), valueOf (método Object.valueOf), watch (método Object.watch)

addListener (método Selection.addListener)

```
public static addListener(listener:Object) : Void
```

Registra un objeto para recibir notificaciones de cambio de selección con el teclado. Cuando se selecciona con el teclado otro elemento (por ejemplo, cuando se invoca Selection.setFocus(), se invocará el método onSetFocus de cada objeto detector registrado en addListener(). El cambio de selección del teclado pueden detectarlo múltiples objetos. Si el detector especificado ya se ha registrado, no se producirá ningún cambio.

Disponibilidad: ActionScript 1.0; Flash Player 6

Parámetros

listener:Object - Un nuevo objeto con un método onSetFocus.

Ejemplo

En el ejemplo siguiente, crea dos campos de introducción de texto en tiempo de ejecución y define los bordes de cada campo de texto como `true`. Este código crea un objeto de ActionScript nuevo (genérico) denominado `focusListener`. Este objeto define para sí mismo una propiedad `onSetFocus`, a la cual asigna una función. La función está formada por dos parámetros: una referencia al campo de texto que ha dejado de estar seleccionado y una referencia al campo de texto que pasa a estar seleccionado. La función define la propiedad `border` del campo de texto que ha dejado de estar seleccionado con el valor `false` y define la propiedad border del campo que pasa a estar seleccionado con el valor `true`:

```
this.createTextField("one_txt", 99, 10, 10, 200, 20);
this.createTextField("two_txt", 100, 10, 50, 200, 20);
one_txt.border = true;
one_txt.type = "input";
two_txt.border = true;
two_txt.type = "input";

var focusListener:Object = new Object();
focusListener.onSetFocus = function(oldFocus_txt, newFocus_txt) {
  oldFocus_txt.border = false;
  newFocus_txt.border = true;
};
Selection.addListener(focusListener);
```

Cuando pruebe el archivo SWF, intente utilizar el tabulador para desplazarse entre los dos campos de texto. Compruebe que selecciona Control > Deshabilitar métodos abreviados de teclado para poder seleccionar los distintos campos mediante el tabulador.

Véase también

`setFocus (método Selection.setFocus)`

getBeginIndex (método Selection.getBeginIndex)

```
public static getBeginIndex() : Number
```

Devuelve el índice al comienzo del espacio de selección. Si no existe un índice ni un campo de texto seleccionado, el método devuelve -1. Los índices de espacio de selección están basados en cero (por ejemplo, la primera posición es 0, la segunda posición es 1, etc.).

Disponibilidad: ActionScript 1.0; Flash Player 5

Valor devuelto

`Number` - Un entero.

Ejemplo

El siguiente ejemplo crea un campo de texto en tiempo de ejecución y define sus propiedades. Se ha agregado un elemento de menú contextual que puede usarse para cambiar a mayúsculas el texto seleccionado.

```
this.createTextField("output_txt", this.getNextHighestDepth(), 0, 0, 300,
  200);
output_txt.multiline = true;
output_txt.wordWrap = true;
output_txt.border = true;
output_txt.type = "input";
output_txt.text = "Enter your text here";
var my_cm:ContextMenu = new ContextMenu();
my_cm.customItems.push(new ContextMenuItem("Uppercase...", doUppercase));
function doUppercase():Void {
  var startIndex:Number = Selection.getBeginIndex();
  var endIndex:Number = Selection.getEndIndex();
  var stringToUppercase:String = output_txt.text.substring(startIndex,
  endIndex);
  output_txt.replaceText(startIndex, endIndex,
  stringToUppercase.toUpperCase());
}
output_txt.menu = my_cm;
```

El método `MovieClip.getNextHighestDepth()` utilizado en este ejemplo necesita Flash Player 7 o una versión posterior. Si el archivo SWF incluye un componente de la versión 2, utilice la clase DepthManager de los componentes de la versión 2 en lugar del método `MovieClip.getNextHighestDepth()`.

Encontrará un ejemplo en el archivo Strings.fla en la carpeta de ejemplos de ActionScript también. Esta carpeta suele estar en rutas como las siguientes:

- Windows: *unidad de inicio*\Archivos de programa\Macromedia\Flash 8\Samples and Tutorials\Samples\ActionScript
- Macintosh: *Disco duro de Macintosh*/Applications/Macromedia Flash 8/Samples and Tutorials/Samples/ActionScript

Véase también

`getEndIndex (método Selection.getEndIndex)`

getCaretIndex (método Selection.getCaretIndex)

```
public static getCaretIndex() : Number
```

Devuelve el índice de la posición del punto de inserción intermitente (intercalación). Si no existe un punto de inserción intermitente, el método devuelve -1. Los índices de espacio de selección están basados en cero (por ejemplo, la primera posición es 0, la segunda posición es 1, etc.).

Disponibilidad: ActionScript 1.0; Flash Player 5

Valor devuelto

`Number` - Un entero.

Ejemplo

El siguiente ejemplo crea un campo de texto en tiempo de ejecución y define sus propiedades. Se utiliza el método `getCaretIndex()` para devolver el índice del símbolo de intercalación y mostrar su valor en otro campo de texto.

```
this.createTextField("pos_txt", this.getNextHighestDepth(), 50, 20, 100,
  22);
this.createTextField("content_txt", this.getNextHighestDepth(), 50, 50,
  400, 300);
content_txt.border = true;
content_txt.type = "input";
content_txt.wordWrap = true;
content_txt.multiline = true;
content_txt.onChanged = getCaretPos;

var keyListener:Object = new Object();
keyListener.onKeyUp = getCaretPos;
Key.addListener(keyListener);

var mouseListener:Object = new Object();
mouseListener.onMouseUp = getCaretPos;
Mouse.addListener(mouseListener);

function getCaretPos() {
  pos_txt.text = Selection.getCaretIndex();
}
```

El método `MovieClip.getNextHighestDepth()` utilizado en este ejemplo necesita Flash Player 7 o una versión posterior. Si el archivo SWF incluye un componente de la versión 2, utilice la clase DepthManager de los componentes de la versión 2 en lugar del método `MovieClip.getNextHighestDepth()`.

Encontrará un ejemplo en el archivo Strings.fla en la carpeta de ejemplos de ActionScript también. Esta carpeta suele estar en rutas como las siguientes:

- Windows: *unidad de inicio*\Archivos de programa\Macromedia\Flash 8\Samples and Tutorials\Samples\ActionScript
- Macintosh: *Disco duro de Macintosh*/Applications/Macromedia Flash 8/Samples and Tutorials/Samples/ActionScript

getEndIndex (método Selection.getEndIndex)

```
public static getEndIndex() : Number
```

Devuelve el índice final del espacio de selección que actualmente está seleccionado con el teclado. Si no existe un índice ni hay un espacio de selección seleccionado, el método devuelve -1. Los índices de espacio de selección están basados en cero (por ejemplo, la primera posición es 0, la segunda posición es 1, etc.).

Disponibilidad: ActionScript 1.0; Flash Player 5

Valor devuelto

`Number` - Un entero.

Ejemplo

Este ejemplo está extraído del archivo animation.fla de la carpeta de ejemplos de ActionScript.

```
// define the function which converts the selected text in an instance,
// and convert the string to upper or lower case.
function convertCase(target, menuItem) {
  var beginIndex:Number = Selection.getBeginIndex();
  var endIndex:Number = Selection.getEndIndex();
  var tempString:String;
  // make sure that text is actually selected.
  if (beginIndex>-1 && endIndex>-1) {
  // set the temporary string to the text before the selected text.
  tempString = target.text.slice(0, beginIndex);
  switch (menuItem.caption) {
  case 'Uppercase...' :
    // if the user selects the "Uppercase..." context menu item,
    // convert the selected text to upper case.
    tempString += target.text.substring(beginIndex,
  endIndex).toUpperCase();
    break;
  case 'Lowercase...' :
    tempString += target.text.substring(beginIndex,
  endIndex).toLowerCase();
    break;
  }
```

```
  // append the text after the selected text to the temporary string.
  tempString += target.text.slice(endIndex);
  // set the text in the target text field to the contents of the temporary
  string.
  target.text = tempString;
  }
}
```

Véase el archivo Strings.fla file para obtener el script completo. La carpeta de ejemplos de ActionScript suele estar en rutas como las siguientes:

- Windows: *unidad de inicio*\Archivos de programa\Macromedia\Flash 8\Samples and Tutorials\Samples\ActionScript
- Macintosh: *Disco duro de Macintosh*/Applications/Macromedia Flash 8/Samples and Tutorials/Samples/ActionScript

Véase también

`getBeginIndex (método Selection.getBeginIndex)`

getFocus (método Selection.getFocus)

```
public static getFocus() : String
```

Devuelve una cadena que especifica la ruta de destino del objeto que está seleccionado con el teclado.

- Si un objeto TextField está seleccionado con el teclado y tiene un nombre de instancia, este método devuelve la ruta de destino del objeto TextField. En caso contrario, devuelve el nombre de la variable de TextField.
- Si un objeto Button o un clip de película de botón está seleccionado con el teclado, este método devuelve la ruta de destino del objeto Button o el clip de película de botón.
- Si no hay seleccionado con el teclado un objeto TextField, un objeto Button, una instancia de componente ni un clip de película de botón, este método devuelve el valor `null`.

Disponibilidad: ActionScript 1.0; Flash Player 5

Valor devuelto

`String` - Una cadena o `null`.

Ejemplo

El siguiente ejemplo muestra la ruta de destino de selección seleccionada en una instancia de componente TextArea. Añada varias instancias de componentes o de botones, campos de texto y películas al escenario. Añada varias instancias de componentes o de botones, campos de texto y películas al archivo SWF. A continuación, añada el siguiente código ActionScript en el archivo AS o FLA.

```
var focus_ta:mx.controls.TextArea;
my_mc.onRelease = function() {};
my_btn.onRelease = function() {};

var keyListener:Object = new Object();
keyListener.onKeyDown = function() {
  if (Key.isDown(Key.SPACE)) {
  focus_ta.text = Selection.getFocus()+newline+focus_ta.text;
  }
};
Key.addListener(keyListener);
```

Pruebe el archivo SWF y utilice el tabulador para desplazarse por las instancias del escenario. Compruebe que la opción Control > Deshabilitar métodos abreviados de teclado está seleccionada en el entorno de prueba.

Véase también

onSetFocus (detector de eventos Selection.onSetFocus), setFocus (método Selection.setFocus)

onSetFocus (detector de eventos Selection.onSetFocus)

```
onSetFocus = function([oldfocus:Object], [newfocus:Object]) {}
```

Recibe una notificación cuando cambia la selección de entrada. Para utilizar este detector, debe crear un objeto detector. A continuación puede definir una función para este detector y utilizar Selection.addListener() para registrar el detector con el objeto Selection, como en el código siguiente:

```
var someListener:Object = new Object();
someListener.onSetFocus = function () {
  // statements
}
Selection.addListener(someListener);
```

Los detectores permiten que diversas partes del código cooperen, ya que varios detectores pueden recibir notificación de un solo evento.

Disponibilidad: ActionScript 1.0; Flash Player 6

Parámetros

oldfocus:Object [opcional] - El objeto que deja de estar seleccionado.

newfocus:Object [opcional] - El objeto que se selecciona.

Ejemplo

El ejemplo siguiente demuestra cómo determinar cuándo cambia la selección de entrada en un archivo SWF entre varios campos de texto creados de forma dinámica. Introduzca el siguiente código ActionScript en un archivo FLA o AS y pruebe el documento:

```
this.createTextField("one_txt", 1, 0, 0, 100, 22);
this.createTextField("two_txt", 2, 0, 25, 100, 22);
this.createTextField("three_txt", 3, 0, 50, 100, 22);
this.createTextField("four_txt", 4, 0, 75, 100, 22);

for (var i in this) {
  if (this[i] instanceof TextField) {
  this[i].border = true;
  this[i].type = "input";
  }
}

this.createTextField("status_txt", this.getNextHighestDepth(), 200, 10,
  300, 100);
status_txt.html = true;
status_txt.multiline = true;

var someListener:Object = new Object();
someListener.onSetFocus = function(oldFocus, newFocus) {
  status_txt.htmlText = "<b>setFocus triggered</b>";
  status_txt.htmlText += "<textformat tabStops='[20,80]'>";
  status_txt.htmlText += " \toldFocus:\t"+oldFocus;
  status_txt.htmlText += " \tnewFocus:\t"+newFocus;
  status_txt.htmlText += " \tgetFocus:\t"+Selection.getFocus();
  status_txt.htmlText += "</textformat>";
};
Selection.addListener(someListener);
```

El método `MovieClip.getNextHighestDepth()` utilizado en este ejemplo necesita Flash Player 7 o una versión posterior. Si el archivo SWF incluye un componente de la versión 2, utilice la clase DepthManager de los componentes de la versión 2 en lugar del método `MovieClip.getNextHighestDepth()`.

Véase también

addListener (método Selection.addListener), setFocus (método Selection.setFocus)

removeListener (método Selection.removeListener)

`public static removeListener(listener:Object) : Boolean`

Elimina un objeto registrado previamente con `Selection.addListener()`.

Disponibilidad: ActionScript 1.0; Flash Player 6

Parámetros

`listener``:Object` - El objeto dejará de recibir notificaciones de selección.

Valor devuelto

`Boolean` - Si `listener` se ha eliminado correctamente, el método devolverá el valor `true`. Si el `listener` no se ha eliminado correctamente (por ejemplo, porque `listener` no se encontraba en la lista de detectores del objeto Selection), el método devolverá el valor `false`.

Ejemplo

El código ActionScript siguiente crea varias instancias de campo de texto de forma dinámica. Al seleccionar un campo de texto, aparece la información en el panel Salida. Cuando haga clic en la instancia `remove_btn`, se elimina el detector y deja de mostrarse la información en el panel Salida.

```
this.createTextField("one_txt", 1, 0, 0, 100, 22);
this.createTextField("two_txt", 2, 0, 25, 100, 22);
this.createTextField("three_txt", 3, 0, 50, 100, 22);
this.createTextField("four_txt", 4, 0, 75, 100, 22);

for (var i in this) {
  if (this[i] instanceof TextField) {
  this[i].border = true;
  this[i].type = "input";
  }
}

var selectionListener:Object = new Object();
selectionListener.onSetFocus = function(oldFocus, newFocus) {
  trace("Focus shifted from "+oldFocus+" to "+newFocus);
};
Selection.addListener(selectionListener);

remove_btn.onRelease = function() {
  trace("removeListener invoked");
  Selection.removeListener(selectionListener);
};
```

Véase también

addListener (método Selection.addListener)

setFocus (método Selection.setFocus)

```
public static setFocus(newFocus:Object) : Boolean
```

Selecciona con el teclado el campo de texto, botón o clip de película seleccionable (editable) especificado por el parámetro `newFocus`. Si se pasa el valor `null` o `undefined`, se eliminará la selección actual de teclado.

Disponibilidad: ActionScript 1.0; Flash Player 5

Parámetros

`newFocus``:Object` - Un objeto como una instancia de botón, clip de película o campo de texto, o una cadena que especifica la ruta a una instancia de botón, clip de película o campo de texto. Si pasa un literal de cadena que especifica una ruta, encierre la ruta entre comillas (" "). Puede utilizar notación con barras o notación con puntos. Si está utilizando ActionScript 2.0, deberá utilizar notación con puntos. Puede utilizar una ruta relativa o absoluta.

Valor devuelto

`Boolean` - Un valor booleano; `true` si el intento de selección con el teclado es correcto; `false` si dicho intento fracasa.

Ejemplo

En el ejemplo siguiente, se resalta el campo de texto `username_txt` cuando se ejecuta en una ventana del navegador. Si el usuario no rellena uno de los campos de texto obligatorios (`username_txt` y `password_txt`), el cursor se sitúa automáticamente en el campo de texto al que le faltan datos. Por ejemplo, si el usuario no escribe nada en el campo de texto `username_txt` y hace clic en el botón de envío, aparece un mensaje de error y el cursor se sitúa en el campo de texto `username_txt`.

```
this.createTextField("status_txt", this.getNextHighestDepth(), 100, 70,
  100, 22);
this.createTextField("username_txt", this.getNextHighestDepth(), 100, 100,
  100, 22);
this.createTextField("password_txt", this.getNextHighestDepth(), 100, 130,
  100, 22);
this.createEmptyMovieClip("submit_mc", this.getNextHighestDepth());
submit_mc.createTextField("submit_txt", this.getNextHighestDepth(), 100,
  160, 100, 22);
submit_mc.submit_txt.autoSize = "center";
submit_mc.submit_txt.text = "Submit";
```

```
submit_mc.submit_txt.border = true;
submit_mc.onRelease = checkForm;
username_txt.border = true;
password_txt.border = true;
username_txt.type = "input";
password_txt.type = "input";
password_txt.password = true;
Selection.setFocus("username_txt");
//
function checkForm():Boolean {
  if (username_txt.text.length == 0) {
  status_txt.text = "fill in username";
  Selection.setFocus("username_txt");
  return false;
  }
  if (password_txt.text.length == 0) {
  status_txt.text = "fill in password";
  Selection.setFocus("password_txt");
  return false;
  }
  status_txt.text = "success!";
  Selection.setFocus(null);
  return true;
}
```

El método `MovieClip.getNextHighestDepth()` utilizado en este ejemplo necesita Flash Player 7 o una versión posterior. Si el archivo SWF incluye un componente de la versión 2, utilice la clase DepthManager de los componentes de la versión 2 en lugar del método `MovieClip.getNextHighestDepth()`.

Véase también

`getFocus (método Selection.getFocus)`

setSelection (método Selection.setSelection)

```
public static setSelection(beginIndex:Number, endIndex:Number) : Void
```

Establece el espacio de selección del campo de texto seleccionado con el teclado actualmente. El nuevo espacio de selección comenzará por el índice especificado en el parámetro `beginIndex` y terminará por el índice especificado en el parámetro `endIndex`. Los índices de espacio de selección están basados en cero (por ejemplo, la primera posición es 0, la segunda posición es 1, etc.). Este método no tiene ningún efecto si actualmente no hay ningún campo de texto seleccionado con el teclado.

Disponibilidad: ActionScript 1.0; Flash Player 5

Parámetros

beginIndex:Number - El primer índice del espacio de selección.

endIndex:Number - El último índice del espacio de selección.

Ejemplo

En el siguiente código ActionScript, crea un campo de texto en tiempo de ejecución y le añade una cadena. A continuación, resalta el campo de texto y selecciona un intervalo de caracteres en el campo de texto resaltado.

```
this.createTextField("myText_txt", 99, 10, 10, 200, 30);
myText_txt.text = "this is my text";
this.onEnterFrame = function () {
  Selection.setFocus("myText_txt");
  Selection.setSelection(0, 3);
  delete this.onEnterFrame;
}
```

El siguiente ejemplo ilustra cómo el parámetro `endIndex` no es inclusivo. Para seleccionar el primer carácter, debe utilizar un parámetro `endIndex` 1, no 0. Si cambia el parámetro `endIndex` a 0, no se selecciona nada.

```
this.createTextField("myText_txt", 99, 10, 10, 200, 30);
myText_txt.text = "this is my text";
this.onEnterFrame = function () {
  Selection.setFocus("myText_txt");
  Selection.setSelection(0, 1);
  delete this.onEnterFrame;
}
```

SharedObject

```
Object
  |
  +-SharedObject
```

```
public dynamic class SharedObject
extends Object
```

La clase SharedObject se emplea para leer y almacenar cantidades limitadas de datos en el equipo de un usuario. Los objetos compartidos permiten que se compartan datos en tiempo real entre objetos persistentes en el equipo del usuario. Los objetos compartidos locales son similares a las cookies del navegador.

A continuación se muestran tres usos posibles de los objetos compartidos:

- Un juego que almacena la puntuación máxima de un usuario. El juego puede proporcionar datos personalizados a los usuarios, como el nombre de usuario y la máxima puntuación, sin dedicar espacio de almacenamiento del servidor.
- Una aplicación de listín telefónico que funcione en línea y sin conexión. El listín, como una aplicación de proyector, contendría una caché de datos locales con una lista de nombres y números de teléfono introducidos por el usuario. Cuando hubiera una conexión a Internet disponible, la aplicación recuperaría información actualizada de un servidor. En caso contrario, la aplicación utilizaría los últimos datos guardados en objetos compartidos.
- Preferencias de usuario o datos de control para un sitio Web complejo, como un registro de los artículos que ha leído un usuario en un sitio de noticias. El registro de esta información le permitiría mostrar de forma distinta los artículos que ya ha leído y los que no ha leído. Si se almacena esta información en el equipo del usuario, se reduce la carga del servidor.

Los objetos compartidos locales mantienen la persistencia local. Por ejemplo, puede llamar a `SharedObject.getLocal()` para crear un objeto compartido que contenga la máxima puntuación obtenida en un juego. Dado que el objeto compartido es persistente localmente, Flash guarda sus atributos de datos en el equipo del usuario cuando se cierra el juego. La próxima vez que se abra el juego, aparecerá la máxima puntuación de la sesión anterior. También se pueden definir las propiedades del objeto compartido como `null` antes de cerrar el juego. De esta forma, la próxima vez que se ejecute el archivo SWF, el juego se abrirá sin mostrar la puntuación anterior.

Para crear un objeto local compartido, utilice la siguiente sintaxis:

```
var so:SharedObject = SharedObject.getLocal("userHighScore");
so.data.highScore = new Number();
so.flush();
```

En este ejemplo, el objeto compartido está *alineado* o copiado en el disco de forma explícita. Cuando se cierra una aplicación, los objetos compartidos se alinean automáticamente; sin embargo, aquí se muestran para demostrar el paso de escribir los datos en un disco.

Consideraciones relativas al espacio del disco local: Los objetos compartidos locales pueden ser muy útiles, pero presentan algunas limitaciones que debe considerar al diseñar su aplicación. En ocasiones es posible que los archivos SWF no puedan escribir objetos compartidos locales y algunas veces los datos almacenados en objetos compartidos locales deben eliminarse sin que el usuario lo advierta. Los usuarios de Flash Player pueden gestionar el espacio de disco disponible para dominios individuales o para todos los dominios. Si los usuarios reducen el espacio en disco disponible, es posible que se eliminen algunos objetos compartidos locales. Los usuarios de Flash Player tienen también controles de confidencialidad para impedir a dominios de terceros (distintos de los de la barra de direcciones del navegador) que lean o escriban objetos compartidos locales.

Nota: El contenido local puede escribir siempre objetos compartidos de terceros en el disco, aunque no se admita esta operación.

Macromedia recomienda que se comprueben los errores relacionados con el espacio en disco disponible y con los controles de confidencialidad del usuario. Realice estas comprobaciones cuando llame a `getLocal()` y a `flush()`:

- `SharedObject.getLocal()` - Este método devuelve `null` cuando el usuario ha desactivado objetos compartidos de terceros y el dominio del SWF no coincide con el de la barra de direcciones del navegador.
- `SharedObject.flush()` - Este método devuelve `false` cuando el usuario ha desactivado objetos compartidos para su dominio o para todos los dominios. Devuelve `"pending"` cuando se necesita más espacio de almacenamiento y el usuario debe decidir de forma interactiva si permite el aumento.

Si su SWF intenta crear o modificar objetos compartidos locales, compruebe que tiene al menos 215 píxeles de ancho y 138 píxeles de alto (las dimensiones mínimas para mostrar el cuadro de diálogo que pide a los usuarios aumentar el límite de almacenamiento de objetos compartidos locales). Si su SWF es más pequeño y necesita aumentar el límite de almacenamiento, `SharedObject.flush()` falla y devuelve `"pending"`. A partir de ese momento, al llamar al controlador `SharedObject.onStatus` se obtendrá como resultado `"SharedObject.Flush.Failed"`.

Disponibilidad: ActionScript 1.0; Flash Player 6

Véase también

`getLocal (método SharedObject.getLocal)`, `flush (método SharedObject.flush)`, `onStatus (controlador SharedObject.onStatus)`

Resumen de propiedades

Modificadores	Propiedad	Descripción
	`data:Object`	Conjunto de atributos asignados a la propiedad `data` del objeto; estos atributos pueden compartirse y/o almacenarse.

Propiedades heredadas de la clase Object

`constructor (propiedad Object.constructor), __proto__ (propiedad Object.__proto__), prototype (propiedad Object.prototype), __resolve (propiedad Object.__resolve)`

Resumen de eventos

Evento	Descripción
`onStatus = function(infoObject:Object) {}`	Se invoca cada vez que se publica un error, una advertencia o una nota informativa para un objeto compartido.

Resumen de métodos

Modificadores	Firma	Descripción
	`clear() : Void`	Elimina todos los datos del objeto compartido y elimina el objeto compartido del disco.
	`flush([minDiskSpace:Number]) : Object`	Graba inmediatamente un objeto compartido persistente localmente en un archivo local.
`static`	`getLocal(name:String, [localPath:String], [secure:Boolean]) : SharedObject`	Devuelve una referencia a un objeto compartido persistente localmente que sólo está disponible para el cliente actual.
	`getSize() : Number`	Obtiene el tamaño actual en bytes del objeto compartido.

Métodos heredados de la clase Object

`addProperty (método Object.addProperty), hasOwnProperty (método Object.hasOwnProperty), isPropertyEnumerable (método Object.isPropertyEnumerable), isPrototypeOf (método Object.isPrototypeOf), registerClass (método Object.registerClass), toString (método Object.toString), unwatch (método Object.unwatch), valueOf (método Object.valueOf), watch (método Object.watch)`

clear (método SharedObject.clear)

```
public clear() : Void
```

Elimina todos los datos del objeto compartido y elimina el objeto compartido del disco. La referencia a `my_so` continúa activa y `my_so` está ahora vacío.

Disponibilidad: ActionScript 1.0; Flash Player 7

Ejemplo

El ejemplo siguiente guarda datos en el objeto compartido y, a continuación, vacía todos los datos del objeto compartido.

```
var my_so:SharedObject = SharedObject.getLocal("superfoo");
my_so.data.name = "Hector";
trace("before my_so.clear():");
for (var prop in my_so.data) {
  trace("\t"+prop);
}
trace("");
my_so.clear();
trace("after my_so.clear():");
for (var prop in my_so.data) {
  trace("\t"+prop);
}
```

Este código ActionScript muestra el siguiente mensaje en el panel Salida:

```
before my_so.clear():
  name

after my_so.clear():
```

data (propiedad SharedObject.data)

```
public data : Object
```

Conjunto de atributos asignados a la propiedad `data` del objeto; estos atributos pueden compartirse y/o almacenarse. Cada atributo puede ser un objeto de cualquier tipo básico de ActionScript o JavaScript (Array, Number, Boolean, etc.). Por ejemplo, las siguientes líneas asignan valores a diversos aspectos de un objeto compartido:

```
var items_array:Array = new Array(101, 346, 483);
var currentUserIsAdmin:Boolean = true;
var currentUserName:String = "Ramona";

var my_so:SharedObject = SharedObject.getLocal("superfoo");
my_so.data.itemNumbers = items_array;
my_so.data.adminPrivileges = currentUserIsAdmin;
my_so.data.userName = currentUserName;
```

```
for (var prop in my_so.data) {
  trace(prop+": "+my_so.data[prop]);
}
```

Todos los atributos de la propiedad `data` de un objeto compartido se guardan si el objeto es persistente, y el objeto compartido contiene la siguiente información:

```
userName: Ramona
adminPrivileges: true
itemNumbers: 101,346,483
```

Nota: No asigne valores directamente a la propiedad `data` de un objeto compartido, como en `so.data = someValue`; Flash no tiene en cuenta estas asignaciones.

Para eliminar atributos de objetos locales compartidos, utilice código como, por ejemplo, `delete so.data.attributeName`; la configuración de un atributo con el valor `null` o `undefined` para un objeto local compartido no elimina el atributo en cuestión.

Para crear valores *private* para un objeto compartido (los valores que sólo están disponibles para la instancia de cliente mientras el objeto se está utilizando no se almacenan con el objeto cuando se cierra), cree propiedades que no sean `data` con nombre para almacenarlos, como se muestra en el siguiente ejemplo:

```
var my_so:SharedObject = SharedObject.getLocal("superfoo");
my_so.favoriteColor = "blue";
my_so.favoriteNightClub = "The Bluenote Tavern";
my_so.favoriteSong = "My World is Blue";

for (var prop in my_so) {
  trace(prop+": "+my_so[prop]);
}
```

El objeto compartido contiene los siguientes datos:

```
favoriteSong: My World is Blue
favoriteNightClub: The Bluenote Tavern
favoriteColor: blue
data: [object Object]
```

Disponibilidad: ActionScript 1.0; Flash Player 6

Ejemplo

El ejemplo siguiente guarda texto de un componente TextInput en un objeto compartido con el nombre `my_so` (para ver el ejemplo completo, consulte `SharedObject.getLocal()`:

```
// Create a listener object and function for the <enter> event.
var textListener:Object = new Object();
textListener.enter = function(eventObj:Object) {
  my_so.data.myTextSaved = eventObj.target.text;
  my_so.flush();
};
```

Véase también

flush (método SharedObject.flush)

```
public flush([minDiskSpace:Number]) : Object
```

Graba inmediatamente un objeto compartido persistente localmente en un archivo local. Si no utiliza este método, Flash grabará el objeto compartido en un archivo cuando termine la sesión del objeto compartido (es decir, cuando se cierre el archivo SWF, cuando se aplique garbage collection a un objeto compartido porque ya no tiene ninguna referencia a él o cuando llame a `SharedObject.clear()`.

Si este método devuelve `"pending"`, Flash Player mostrará un cuadro de diálogo en el que se pedirá al usuario que aumente el espacio en disco disponible para objetos de este dominio. Para permitir que crezca el espacio asignado al objeto compartido cuando vuelva a guardarse, lo que evita que se devuelva el valor `"pending"`, pase un valor para `minimumDiskSpace`. Cuando Flash intenta grabar el archivo, busca el número de bytes pasados a `minimumDiskSpace`, en lugar de buscar el espacio suficiente para guardar el objeto compartido con su tamaño actual.

Por ejemplo, si espera que un objeto compartido crezca hasta un tamaño máximo de 500 bytes, aunque es posible que su tamaño inicial sea muy inferior a éste, pase el valor 500 para `minimumDiskSpace`. Si Flash pide al usuario que asigne espacio del disco al objeto compartido, pide 500 bytes. Una vez que el usuario asigne la cantidad de espacio solicitada, Flash no tendrá que solicitar más espacio en posteriores intentos de almacenamiento del objeto (siempre y cuando su tamaño no supere 500 bytes).

Cuando el usuario responde al cuadro de diálogo, se llama de nuevo a este método y devuelve `true` o `false`; también se llama a `SharedObject.onStatus` con una propiedad `code` de `SharedObject.Flush.Success` o `SharedObject.Flush.Failed`.

Para más información, consulte "Consideraciones relativas al espacio del disco local" en la información general sobre la clase SharedObject.

Disponibilidad: ActionScript 1.0; Flash Player 6

Parámetros

`minDiskSpace``:Number` [opcional] - Un entero que especifica el número de bytes que debe asignarse a este objeto. El valor predeterminado es 0.

Valor devuelto

`Object` - Un valor booleano: `true`, `false`o el valor de cadena `"pending"`, como se describe en la siguiente lista:

- Si el usuario ha permitido el almacenamiento de información local para objetos de este dominio y la cantidad de espacio asignada es suficiente para almacenar el objeto, este método devolverá el valor `true`. (Si ha pasado un valor para `minimumDiskSpace`, la cantidad de espacio asignada deberá ser al menos igual a dicho valor para que se devuelva `true`.)
- Si el usuario ha permitido el almacenamiento de información local para objetos de este dominio pero la cantidad de espacio asignada es insuficiente para almacenar el objeto, este método devolverá el valor `"pending"`.
- Si el usuario ha denegado el almacenamiento de información local de manera permanente para objetos de este dominio, o si Flash no puede guardar el objeto por cualquier motivo, este método devolverá `false`.

Nota: El contenido local siempre puede escribir en el disco objetos compartidos de dominios de terceros (distintos del que aparece en la barra de direcciones actual del navegador), aunque no esté admitida dicha operación.

Ejemplo

La siguiente función obtiene un objeto compartido, `my_so`,, y aplica la configuración proporcionada por el usuario a las propiedades de escritura. Por último, se llama a `flush()` para guardar la configuración y asignar un mínimo de 1000 bytes de espacio en disco.

```
this.syncSettingsCore = function(soName:String, override:Boolean,
  settings:Object) {
  var my_so:SharedObject = SharedObject.getLocal(soName, "http://
  www.mydomain.com/app/sys");
  // settings list index
  var i;
  // For each specified value in settings:
  // If override is true, set the persistent setting to the provided value.
  // If override is false, fetch the persistent setting, unless there
  // isn't one, in which case, set it to the provided value.
  for (i in settings) {
  if (override || (my_so.data[i] == null)) {
    my_so.data[i] = settings[i];
  } else {
    settings[i] = my_so.data[i];
  }
  }
  my_so.flush(1000);
};
```

Véase también

clear (método SharedObject.clear), onStatus (controlador SharedObject.onStatus)

getLocal (método SharedObject.getLocal)

```
public static getLocal(name:String, [localPath:String], [secure:Boolean]) :
  SharedObject
```

Devuelve una referencia a un objeto compartido persistente localmente que sólo está disponible para el cliente actual. Si el objeto compartido no existe ya, este método crea uno. Se trata de un método estático de la clase SharedObject. Para asignar el objeto a una variable, utilice una sintaxis como la siguiente:

```
var so:SharedObject = SharedObject.getLocal("savedData")
```

Nota: Si el usuario ha optado por no permitir nunca el almacenamiento local para este dominio, el objeto no se guardará localmente aunque se especifique un valor para `localPath`. La excepción a esta regla es el contenido local. El contenido local siempre puede escribir en el disco objetos compartidos de dominios de terceros (distintos del que aparece en la barra de direcciones actual del navegador), aunque no esté admitida dicha operación.

Para evitar conflictos de nombres, Flash busca en la ubicación del archivo SWF que está creando el objeto compartido. Por ejemplo, si un archivo SWF ubicado en www.myCompany.com/apps/stockwatcher.swf crea un objeto compartido denominado `portfolio`, el objeto compartido no entre en conflicto con otro objeto denominado `portfolio` creado por un archivo SWF en www.yourCompany.com/photoshoot.swf porque los archivos SWF tienen su origen en directorios distintos.

Aunque el parámetro `localPath` es opcional, debe contemplar su utilización, particularmente si otros archivos SWF van a necesitar acceder al objeto compartido. Si los datos del objeto compartido son específicos de un archivo SWF que no se trasladará a otra ubicación, lo más recomendable será utilizar el valor predeterminado. Si otros archivos SWF necesitan acceder al objeto compartido o si el archivo SWF que crea el objeto compartido se va a trasladar posteriormente, el valor de este parámetro puede afectar a si los archivos SWF van a disponer de acceso al objeto compartido. Por ejemplo, si crea un objeto compartido con `localPath` configurado con el valor predeterminado de ruta completa del archivo SWF, ningún otro archivo SWF podrá acceder a dicho objeto compartido. Si posteriormente traslada el archivo SWF original a otra ubicación, ni tan siquiera dicho archivo SWF podrá acceder a los datos ya almacenados en el objeto compartido.

Puede reducir la probabilidad de que se restrinja accidentalmente el acceso a un archivo compartido empleando el parámetro `localpath`. La opción más permisiva consiste en establecer el parámetro `localPath` con "/", lo que pone el objeto compartido a disposición de todos los archivos SWF del dominio pero aumenta la probabilidad de que se produzcan conflictos de nombres con otros objetos compartidos del dominio. Hay disponibles opciones más restrictivas, incluso se puede anexar al parámetro `localPath` nombres de carpetas incluidos en la ruta completa al archivo SWF; por ejemplo, sus opciones de parámetros `localPath` para el objeto compartido `portfolio` que crea el archivo SWF en www.myCompany.com/apps/stockwatcher.swf son: "/"; "/apps" y "/apps/stockwatcher.swf". Deberá determinar qué opción ofrece máxima flexibilidad para su aplicación.

Cuando utilice este método, puede ser conveniente usar el modelo de seguridad de Flash Player:

- No puede acceder a objetos compartidos de otros entornos limitados.
- Los usuarios pueden limitar el acceso a objetos compartidos utilizando el cuadro de diálogo Configuración de Flash Player o el Administrador de configuración. De forma predeterminada, pueden crearse objetos compartidos de un máximo de 100 Kb de datos por dominio. Los usuarios con derechos administrativos y los usuarios pueden también limitar la capacidad de escribir en el sistema de archivos.

Si publica contenido del archivo SWF para reproducirlo como archivos locales (con archivos SWF instalados localmente o con proyectores [EXE]) y necesita acceder a un objeto compartido específico desde más de un archivo SWF local, tenga en cuenta que para archivos locales, se pueden usar dos ubicaciones distintas para almacenar objetos compartidos. El dominio utilizado depende de los permisos de seguridad concedidos al archivo local que creó el objeto compartido. Los archivos locales pueden tener tres niveles de permisos diferentes: 1) acceso al sistema de archivos local solamente, 2) acceso a la red solamente o 3) acceso a la red y al sistema de archivos local. Los archivos locales con acceso al sistema de archivos local (casos 1 o 3) almacenan sus objetos compartidos en una ubicación. Los archivos locales sin acceso al sistema de archivos local (caso 2) almacenan sus objetos compartidos en otra ubicación. Para más información, consulte lo siguiente:

- Capítulo 17, "Aspectos básicos de la seguridad," en *Aprendizaje de ActionScript 2.0 en Flash*
- El documento técnico sobre la seguridad de Flash Player 8 en http://www.macromedia.com/go/fp8_security
- El documento técnico sobre interfaces API relativas a seguridad de Flash Player 8 en http://www.macromedia.com/go/fp8_security_apis

Disponibilidad: ActionScript 1.0; Flash Player 6

Parámetros

name:String - Una cadena que representa el nombre del objeto. El nombre puede incluir barras diagonales (/); por ejemplo, work/addresses es un nombre válido. No se admiten espacios en los nombres de objetos compartidos, ni tampoco los siguientes caracteres:

~ % & \ ; : " ' , < > ? #

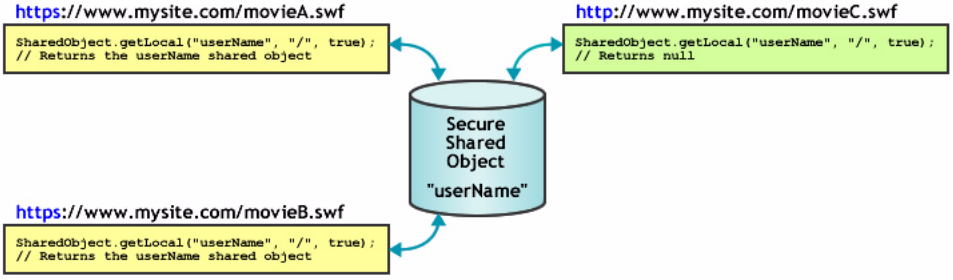
localPath:String [opcional] - Una cadena que especifica la ruta completa o parcial al archivo SWF que creó el objeto compartido y que determina dónde se almacena éste localmente. El valor predeterminado es la ruta completa.

secure:Boolean [opcional] - (sólo Flash Player 8) Determina si el acceso a este objeto compartido se limita a archivos SWF enviados a través de una conexión HTTPS. Si el archivo SWF se envía por HTTPS:

- Si este parámetro es true, Flash Player crea un nuevo objeto compartido seguro u obtiene una referencia a un objeto compartido seguro existente. Sólo pueden leer o escribir en este objeto compartido seguro archivos SWF enviados a través de HTTPS que llamen a SharedObject.getLocal() con el parámetro secure definido como true.
- Si este parámetro es false, Flash Player crea un nuevo objeto compartido u obtiene una referencia a un objeto compartido existente. Los archivos SWF enviados mediante conexiones no HTTPS pueden leer o escribir en este objeto compartido.

Si su archivo SWF se ha enviado a través de una conexión no HTTPS e intenta definir este parámetro como true, fallará la creación de un nuevo objeto compartido (o el acceso a un objeto compartido seguro creado anteriormente) y el valor devuelto será null. Independientemente del valor de este parámetro, los objetos compartidos creados aumentan el espacio en disco total permitido en el dominio. El valor predeterminado es false.

El siguiente diagrama muestra el uso del parámetro secure:



Valor devuelto

`SharedObject` - Una referencia a un objeto compartido persistente localmente que sólo está disponible para el cliente actual. Si Flash no puede crear ni localizar el objeto compartido (por ejemplo, si se ha especificado `localPath` pero dicho directorio no existe o si se ha usado el parámetro `secure` de forma incorrecta), este método devolverá el valor `null`.

Este método falla y devuelve `null` si se ha prohibido a contenido de Flash de terceros la creación y el almacenamiento de objetos compartidos persistentes (excepto en el caso de contenido local). Los usuarios pueden prohibir los objetos compartidos persistentes de terceros en el panel Configuración global de almacenamiento del Administrador de configuración.

Ejemplo

El siguiente ejemplo crea un objeto compartido que almacena el texto escrito en una instancia de componente `TextInput`. El archivo SWF resultante carga el texto guardado desde el objeto compartido cuando inicia la reproducción. Cada vez que el usuario presiona Intro, el texto del campo de texto se escribe en el objeto compartido. Para utilizar este ejemplo, arrastre un componente `TextInput` al escenario y asígnele a la instancia el nombre `myText_ti`. Copie el código siguiente en la línea de tiempo principal (haga clic en una zona vacía del escenario o presione la tecla Esc para anular la selección del componente):

```
// Create the shared object and set localpath to server root.
var my_so:SharedObject = SharedObject.getLocal("savedText", "/");
// Load saved text from the shared object into the myText_ti TextInput
  component.
myText_ti.text = my_so.data.myTextSaved;
// Assign an empty string to myText_ti if the shared object is undefined
// to prevent the text input box from displaying "undefined" when
// this script is first run.
if (myText_ti.text == undefined) {
  myText_ti.text = "";
}
// Create a listener object and function for <enter> event
var textListener:Object = new Object();
textListener.enter = function(eventObj:Object) {
  my_so.data.myTextSaved = eventObj.target.text;
  my_so.flush();
};
// Register the listener with the TextInput component instance
myText_ti.addEventListener("enter", textListener);
```

El ejemplo siguiente guarda el último fotograma introducido por un usuario en un objeto compartido local `kookie`:

```
// Get the kookie
var my_so:SharedObject = SharedObject.getLocal("kookie");
```

```
// Get the user of the kookie and go to the frame number saved for this
  user.
if (my_so.data.user != undefined) {
  this.user = my_so.data.user;
  this.gotoAndStop(my_so.data.frame);
}
```

El siguiente bloque de código se coloca en cada fotograma del archivo SWF:

```
// On each frame, call the rememberme function to save the frame number.
function rememberme() {
  my_so.data.frame=this._currentframe;
  my_so.data.user="John";
}
```

getSize (método SharedObject.getSize)

```
public getSize() : Number
```

Obtiene el tamaño actual en bytes del objeto compartido.

Flash calcula el tamaño de un objeto compartido comprobando todas sus propiedades de datos; cuantas más propiedades de datos tenga el objeto, mayor será el tiempo necesario para estimar su tamaño. La estimación del tamaño de un objeto puede consumir bastante tiempo de proceso, por lo que es posible que prefiera evitar este método si no lo necesita por un motivo concreto.

Disponibilidad: ActionScript 1.0; Flash Player 6

Valor devuelto

`Number` - Un valor numérico que especifica el tamaño en bytes del objeto compartido.

Ejemplo

El ejemplo siguiente obtiene el tamaño del objeto compartido `my_so`:

```
var items_array:Array = new Array(101, 346, 483);
var currentUserIsAdmin:Boolean = true;
var currentUserName:String = "Ramona";

var my_so:SharedObject = SharedObject.getLocal("superfoo");
my_so.data.itemNumbers = items_array;
my_so.data.adminPrivileges = currentUserIsAdmin;
my_so.data.userName = currentUserName;

var soSize:Number = my_so.getSize();
trace(soSize);
```

onStatus (controlador SharedObject.onStatus)

```
onStatus = function(infoObject:Object) {}
```

Se invoca cada vez que se publica un error, una advertencia o una nota informativa para un objeto compartido. Si desea responder a este controlador de eventos, deberá crear una función que procese el objeto de información generado por el objeto compartido.

El objeto de información tiene una propiedad code que contiene una cadena que describe el resultado del controlador `onStatus`, así como una propiedad `level` que contiene una cadena que puede indicar `"Status"` o `"Error"`.

Además de este controlador `onStatus`, Flash también proporciona una super función denominada `System.onStatus`. Si se llama a `onStatus` para un objeto concreto y no hay ninguna función asignada para responder, Flash procesa una función asignada a `System.onStatus`, si existe.

Los siguientes eventos le notifican cuándo tienen lugar determinadas actividades `SharedObject`:

Propiedad code	Propiedad level	Significado
`SharedObject.Flush.Failed`	Error	El comando `SharedObject.flush()` que ha devuelto `"pending"` ha fallado (el usuario no ha asignado espacio adicional del disco para el objeto compartido cuando Flash Player mostró el cuadro de diálogo Configuración global de almacenamiento).
`SharedObject.Flush.Success`	Status	El comando `SharedObject.flush()` que devolvió `"pending"` ha finalizado correctamente (el usuario ha asignado espacio adicional del disco para el objeto compartido).

Disponibilidad: ActionScript 1.0; Flash Player 6

Parámetros

`infoObject``:Object` - Un parámetro definido de acuerdo con el mensaje de estado.

Ejemplo

El ejemplo siguiente muestra distintos mensajes que dependen de si el usuario ha elegido permitir o denegar que la instancia de objeto SharedObject escriba en el disco.

```
var message_str:String;
this.createTextField("message_txt", this.getNextHighestDepth(), 0, 0, 300,
  22);
message_txt.html = true;
this.createTextField("status_txt", this.getNextHighestDepth(), 10, 30, 300,
  100);
status_txt.multiline = true;
status_txt.html = true;

var items_array:Array = new Array(101, 346, 483);
var currentUserIsAdmin:Boolean = true;
var currentUserName:String = "Ramona";
var my_so:SharedObject = SharedObject.getLocal("superfoo");
my_so.data.itemNumbers = items_array;
my_so.data.adminPrivileges = currentUserIsAdmin;
my_so.data.userName = currentUserName;

my_so.onStatus = function(infoObject:Object) {
  status_txt.htmlText = "<textformat tabStops='[50]'>";
  for (var i in infoObject) {
    status_txt.htmlText += "<b>"+i+"</b>"+"\t"+infoObject[i];
  }
  status_txt.htmlText += "</textformat>";
};

var flushResult = my_so.flush(1000001);
switch (flushResult) {
case 'pending' :
  message_str = "flush is pending, waiting on user interaction.";
  break;
case true :
  message_str = "flush was successful. Requested storage space approved.";
  break;
case false :
  message_str = "flush failed. User denied request for additional
  storage.";
  break;
}
message_txt.htmlText = "<a href=\"asfunction:System.showSettings,1\
"><u>"+message_str+"</u></a>";
```

El método `MovieClip.getNextHighestDepth()` utilizado en este ejemplo necesita Flash Player 7 o una versión posterior. Si el archivo SWF incluye un componente de la versión 2, utilice la clase DepthManager de los componentes de la versión 2 en lugar del método `MovieClip.getNextHighestDepth()`.

Véase también

getLocal (método SharedObject.getLocal), onStatus (controlador System.onStatus)

Sound

```
Object
  |
  +-Sound
```

public class *Sound*
extends Object

La clase Sound le permite controlar el sonido de una película. Puede añadir sonidos de la biblioteca a un clip de película mientras se reproduce la película, así como controlar dichos sonidos. Si no especifica un destino cuando al crear un nuevo objeto Sound, podrá utilizar los métodos para controlar el sonido de toda la película.

Deberá utilizar el constructor `new Sound` para crear un objeto Sound antes de llamar a los métodos de la clase Sound.

Disponibilidad: ActionScript 1.0; Flash Player 5

Resumen de propiedades

Modificadores	Propiedad	Descripción
	`duration:Number` [read-only]	La duración de un sonido, en milisegundos.
	`id3:Object` [read-only]	Proporciona acceso a los metadatos que forman parte de un archivo MP3.
	`position:Number` [read-only]	El número de milisegundos que se ha estado reproduciendo un sonido.

Propiedades heredadas de la clase Object

constructor (propiedad Object.constructor), __proto__ (propiedad Object.__proto__), prototype (propiedad Object.prototype), __resolve (propiedad Object.__resolve)

Resumen de eventos

Evento	Descripción
`onID3 = function() {}`	Se invoca cada vez que hay nuevos datos ID3 disponibles para un archivo MP3 cargado mediante `Sound.attachSound()` o `Sound.loadSound()`.
`onLoad = function(success :Boolean) {}`	Se invoca automáticamente cuando se carga un sonido.
`onSoundComplete = function() {}`	Se invoca automáticamente cuando se deja de reproducir un sonido.

Resumen de constructores

Firma	Descripción
`Sound([target:Object ])`	Crea un nuevo objeto Sound para un clip de película especificado.

Resumen de métodos

Modificadores	Firma	Descripción
	`attachSound(id:Strin g) : Void`	Asocia el sonido especificado en el parámetro `id` al objeto Sound indicado.
	`getBytesLoaded() : Number`	Devuelve el número de bytes cargados (sin interrupción) para el objeto Sound especificado.
	`getBytesTotal() : Number`	Devuelve el tamaño en bytes del objeto Sound especificado.
	`getPan() : Number`	Devuelve el nivel de desplazamiento horizontal establecido en la última llamada a `setPan()` como un entero entre -100 (izquierda) y +100 (derecha).
	`getTransform() : Object`	Devuelve la información de transformación de sonido para el objeto Sound especificado configurado con la última llamada a `Sound.setTransform()`.
	`getVolume() : Number`	Devuelve el nivel de volumen del sonido como un entero entre 0 y 100, siendo 0 desactivado y 100 volumen total.
	`loadSound(url:String , isStreaming:Boolean) : Void`	Carga un archivo MP3 en un objeto Sound.

Modificadores	Firma	Descripción
	`setPan(value:Number) : Void`	Determina cómo se reproduce el sonido en los canales (altavoces) izquierdo y derecho.
	`setTransform(transformObject:Object) : Void`	Establece la información de transformación (o balance) de sonido para un objeto Sound.
	`setVolume(value:Number) : Void`	Establece el volumen para el objeto Sound.
	`start([secondOffset:Number], [loops:Number]) : Void`	Comienza la reproducción del último sonido asociado desde el principio, si no hay ningún parámetro especificado, o a partir del punto del sonido especificado por el parámetro `secondOffset`.
	`stop([linkageID:String]) : Void`	Detiene todos los sonidos que se están reproduciendo actualmente, si no se ha especificado ningún parámetro, o sólo el sonido especificado por el parámetro `idName`.

Métodos heredados de la clase Object

`addProperty (método Object.addProperty)`, `hasOwnProperty (método Object.hasOwnProperty)`, `isPropertyEnumerable (método Object.isPropertyEnumerable)`, `isPrototypeOf (método Object.isPrototypeOf)`, `registerClass (método Object.registerClass)`, `toString (método Object.toString)`, `unwatch (método Object.unwatch)`, `valueOf (método Object.valueOf)`, `watch (método Object.watch)`

attachSound (método Sound.attachSound)

`public attachSound(id:String) : Void`

Asocia el sonido especificado en el parámetro `id` al objeto Sound indicado. El sonido debe encontrarse en la biblioteca del archivo SWF actual y su exportación debe haberse especificado en el cuadro de diálogo Propiedades de vinculación. Debe llamar a `Sound.start()` para comenzar a reproducir el sonido.

Para asegurarse de que el sonido puede controlarse desde cualquier escena del archivo SWF, coloque el sonido en la línea de tiempo principal del archivo SWF.

Disponibilidad: ActionScript 1.0; Flash Player 5

Parámetros

id:String - El identificador de un sonido exportado en la biblioteca. El identificador se encuentra en el cuadro de diálogo Propiedades de vinculación.

Ejemplo

El ejemplo siguiente asocia el `logoff_id` del sonido a `my_sound`. Un sonido en la biblioteca tiene el identificador de vinculación `logoff_id`.

```
var my_sound:Sound = new Sound();
my_sound.attachSound("logoff_id");
my_sound.start();
```

duration (propiedad Sound.duration)

```
public duration : Number [read-only]
```

La duración de un sonido, en milisegundos.

Disponibilidad: ActionScript 1.0; Flash Player 6

Ejemplo

El ejemplo siguiente carga un sonido y muestra la duración del archivo de sonido en el panel Salida. Añada el siguiente código ActionScript al archivo AS o FLA.

```
var my_sound:Sound = new Sound();
my_sound.onLoad = function(success:Boolean) {
  var totalSeconds:Number = this.duration/1000;
  trace(this.duration+" ms ("+Math.round(totalSeconds)+" seconds)");
  var minutes:Number = Math.floor(totalSeconds/60);
  var seconds = Math.floor(totalSeconds)%60;
  if (seconds<10) {
  seconds = "0"+seconds;
  }
  trace(minutes+":"+seconds);
};
my_sound.loadSound("song1.mp3", true);
```

El siguiente ejemplo carga varias canciones en un archivo SWF. Una barra de progreso, creada en la interfaz API de dibujo, muestra cómo avanza la carga. Cuando comienza y finaliza la carga de la música, aparece la información correspondiente en el panel Salida. Añada el siguiente código ActionScript al archivo AS o FLA.

```
var pb_height:Number = 10;
var pb_width:Number = 100;
var pb:MovieClip = this.createEmptyMovieClip("progressBar_mc",
  this.getNextHighestDepth());
pb.createEmptyMovieClip("bar_mc", pb.getNextHighestDepth());
pb.createEmptyMovieClip("vBar_mc", pb.getNextHighestDepth());
```

```
pb.createEmptyMovieClip("stroke_mc", pb.getNextHighestDepth());
pb.createTextField("pos_txt", pb.getNextHighestDepth(), 0, pb_height,
  pb_width, 22);

pb._x = 100;
pb._y = 100;

with (pb.bar_mc) {
  beginFill(0x00FF00);
  moveTo(0, 0);
  lineTo(pb_width, 0);
  lineTo(pb_width, pb_height);
  lineTo(0, pb_height);
  lineTo(0, 0);
  endFill();
  _xscale = 0;
}
with (pb.vBar_mc) {
  lineStyle(1, 0x000000);
  moveTo(0, 0);
  lineTo(0, pb_height);
}
with (pb.stroke_mc) {
  lineStyle(3, 0x000000);
  moveTo(0, 0);
  lineTo(pb_width, 0);
  lineTo(pb_width, pb_height);
  lineTo(0, pb_height);
  lineTo(0, 0);
}

var my_interval:Number;
var my_sound:Sound = new Sound();
my_sound.onLoad = function(success:Boolean) {
  if (success) {
  trace("sound loaded");
  }
};
my_sound.onSoundComplete = function() {
  clearInterval(my_interval);
  trace("Cleared interval");
}
my_sound.loadSound("song3.mp3", true);
my_interval = setInterval(updateProgressBar, 100, my_sound);
```

```
function updateProgressBar(the_sound:Sound):Void {
  var pos:Number = Math.round(the_sound.position/the_sound.duration 100);
  pb.bar_mc._xscale = pos;
  pb.vBar_mc._x = pb.bar_mc._width;
  pb.pos_txt.text = pos+"%";
}
```

El método `MovieClip.getNextHighestDepth()` utilizado en este ejemplo necesita Flash Player 7 o una versión posterior. Si el archivo SWF incluye un componente de la versión 2, utilice la clase DepthManager de los componentes de la versión 2 en lugar del método `MovieClip.getNextHighestDepth()`.

Véase también

`position (propiedad Sound.position)`

getBytesLoaded (método Sound.getBytesLoaded)

```
public getBytesLoaded() : Number
```

Devuelve el número de bytes cargados (sin interrupción) para el objeto Sound especificado. Puede comparar el valor de `getBytesLoaded()` con el valor de `getBytesTotal()` para determinar el porcentaje de un sonido que se ha cargado.

Disponibilidad: ActionScript 1.0; Flash Player 6

Valor devuelto

`Number` - Un entero que indica el número de bytes cargados.

Ejemplo

El ejemplo siguiente crea de forma dinámica dos campos de texto que muestran los bytes cargados y el total de bytes de un archivo de sonido que se carga en el archivo SWF. Un campo de texto muestra también un mensaje cuando termina la carga del archivo. Añada el siguiente código ActionScript al archivo FLA o AS:

```
this.createTextField("message_txt", this.getNextHighestDepth(),
  10,10,300,22)
this.createTextField("status_txt", this.getNextHighestDepth(), 10, 50, 300,
  40);
status_txt.autoSize = true;
status_txt.multiline = true;
status_txt.border = false;

var my_sound:Sound = new Sound();
my_sound.onLoad = function(success:Boolean) {
  if (success) {
  this.start();
```

```
  message_txt.text = "Finished loading";
  }
};
my_sound.onSoundComplete = function() {
  message_txt.text = "Clearing interval";
  clearInterval(my_interval);
};
my_sound.loadSound("song2.mp3", true);
var my_interval:Number;
my_interval = setInterval(checkProgress, 100, my_sound);
function checkProgress(the_sound:Sound):Void {
  var pct:Number = Math.round(the_sound.getBytesLoaded()/
  the_sound.getBytesTotal() 100);
  var pos:Number = Math.round(the_sound.position/the_sound.duration 100);
  status_txt.text = the_sound.getBytesLoaded()+" of
  "+the_sound.getBytesTotal()+" bytes ("+pct+"%)"+newline;
  status_txt.text += the_sound.position+" of "+the_sound.duration+"
  milliseconds ("+pos+"%)"+newline;
}
```

El método `MovieClip.getNextHighestDepth()` utilizado en este ejemplo necesita Flash Player 7 o una versión posterior. Si el archivo SWF incluye un componente de la versión 2, utilice la clase DepthManager de los componentes de la versión 2 en lugar del método `MovieClip.getNextHighestDepth()`.

Véase también

`getBytesTotal (método Sound.getBytesTotal)`

getBytesTotal (método Sound.getBytesTotal)

```
public getBytesTotal() : Number
```

Devuelve el tamaño en bytes del objeto Sound especificado.

Disponibilidad: ActionScript 1.0; Flash Player 6

Valor devuelto

`Number` - Un entero que indica el tamaño total en bytes del objeto Sound especificado.

Ejemplo

Véase `Sound.getBytesLoaded()` para obtener una muestra de cómo se utiliza este método.

Véase también

`getBytesLoaded (método Sound.getBytesLoaded)`

getPan (método Sound.getPan)

`public getPan() : Number`

Devuelve el nivel de desplazamiento horizontal establecido en la última llamada a `setPan()` como un entero entre -100 (izquierda) y +100 (derecha). (0 establece de la misma forma los canales izquierdo y derecho.) La configuración de desplazamiento horizontal controla el balance izquierda-derecha del sonido actual y otros sonidos futuros del archivo SWF.

Este método es acumulativo con `setVolume()` o `setTransform()`.

Disponibilidad: ActionScript 1.0; Flash Player 5

Valor devuelto

`Number` - Un entero.

Ejemplo

El ejemplo siguiente crea un control deslizante mediante la interfaz API de dibujo. Cuando el usuario arrastra el control deslizante, cambia el nivel de desplazamiento horizontal del sonido cargado. El nivel de desplazamiento horizontal actual aparece en un campo de texto que se crea de forma dinámica. Añada el siguiente código ActionScript al archivo FLA o AS:

```
var bar_width:Number = 200;
this.createEmptyMovieClip("bar_mc", this.getNextHighestDepth());
with (bar_mc) {
  lineStyle(4, 0x000000);
  moveTo(0, 0);
  lineTo(bar_width+4, 0);
  lineStyle(0, 0x000000);
  moveTo((bar_width/2)+2, -8);
  lineTo((bar_width/2)+2, 8);
}
bar_mc._x = 100;
bar_mc._y = 100;

this.createEmptyMovieClip("knob_mc", this.getNextHighestDepth());
with (knob_mc) {
  lineStyle(0, 0x000000);
  beginFill(0xCCCCCC);
  moveTo(0, 0);
  lineTo(4, 0);
  lineTo(4, 10);
  lineTo(0, 10);
  lineTo(0, 0);
  endFill();
}
knob_mc._x = bar_mc._x+(bar_width/2);
knob_mc._y = bar_mc._y-(knob_mc._height/2);
```

```
knob_mc.left = knob_mc._x-(bar_width/2);
knob_mc.right = knob_mc._x+(bar_width/2);
knob_mc.top = knob_mc._y;
knob_mc.bottom = knob_mc._y;

knob_mc.onPress = function() {
  this.startDrag(false, this.left, this.top, this.right, this.bottom);
};
knob_mc.onRelease = function() {
  this.stopDrag();
  var multiplier:Number = 100/(this.right-this.left) 2;
  var pan:Number = (this._x-this.left-(bar_width/2)) multiplier;
  my_sound.setPan(pan);
  pan_txt.text = my_sound.getPan();
};

var my_sound:Sound = new Sound();
my_sound.loadSound("song2.mp3", true);
this.createTextField("pan_txt", this.getNextHighestDepth(), knob_mc._x,
  knob_mc._y+knob_mc._height, 20, 22);
pan_txt.selectable = false;
pan_txt.autoSize = "center";
pan_txt.text = my_sound.getPan();
```

El método `MovieClip.getNextHighestDepth()` utilizado en este ejemplo necesita Flash Player 7 o una versión posterior. Si el archivo SWF incluye un componente de la versión 2, utilice la clase DepthManager de los componentes de la versión 2 en lugar del método `MovieClip.getNextHighestDepth()`.

Véase también

`setPan (método Sound.setPan)`

getTransform (método Sound.getTransform)

```
public getTransform() : Object
```

Devuelve la información de transformación de sonido para el objeto Sound especificado configurado con la última llamada a `Sound.setTransform()`.

Disponibilidad: ActionScript 1.0; Flash Player 5

Valor devuelto

`Object` - Un objeto con propiedades que contienen los valores de porcentaje de los canales del objeto Sound especificado.

Ejemplo

El ejemplo siguiente asocia cuatro clips de película desde un símbolo de la biblioteca (identificador de vinculación: `knob_id`) utilizados como controles deslizantes (o botones) para controlar el archivo de sonido que se carga en el archivo SWF. Estos controles deslizantes controlan el objeto de transformación, o balance, de un archivo de sonido. Para más información, consulte la entrada para `Sound.setTransform()`. Añada el siguiente código ActionScript al archivo FLA o AS:

```
var my_sound:Sound = new Sound();
my_sound.loadSound("song1.mp3", true);
var transform_obj:Object = my_sound.getTransform();

this.createEmptyMovieClip("transform_mc", this.getNextHighestDepth());
transform_mc.createTextField("transform_txt",
  transform_mc.getNextHighestDepth, 0, 8, 120, 22);
transform_mc.transform_txt.html = true;

var knob_ll:MovieClip = transform_mc.attachMovie("knob_id", "ll_mc",
  transform_mc.getNextHighestDepth(), {_x:0, _y:30});
var knob_lr:MovieClip = transform_mc.attachMovie("knob_id", "lr_mc",
  transform_mc.getNextHighestDepth(), {_x:30, _y:30});
var knob_rl:MovieClip = transform_mc.attachMovie("knob_id", "rl_mc",
  transform_mc.getNextHighestDepth(), {_x:60, _y:30});
var knob_rr:MovieClip = transform_mc.attachMovie("knob_id", "rr_mc",
  transform_mc.getNextHighestDepth(), {_x:90, _y:30});

knob_ll.top = knob_ll._y;
knob_ll.bottom = knob_ll._y+100;
knob_ll.left = knob_ll._x;
knob_ll.right = knob_ll._x;
knob_ll._y = knob_ll._y+(100-transform_obj['ll']);
knob_ll.onPress = pressKnob;
knob_ll.onRelease = releaseKnob;
knob_ll.onReleaseOutside = releaseKnob;

knob_lr.top = knob_lr._y;
knob_lr.bottom = knob_lr._y+100;
knob_lr.left = knob_lr._x;
knob_lr.right = knob_lr._x;
knob_lr._y = knob_lr._y+(100-transform_obj['lr']);
knob_lr.onPress = pressKnob;
knob_lr.onRelease = releaseKnob;
knob_lr.onReleaseOutside = releaseKnob;

knob_rl.top = knob_rl._y;
knob_rl.bottom = knob_rl._y+100;
knob_rl.left = knob_rl._x;
knob_rl.right = knob_rl._x;
knob_rl._y = knob_rl._y+(100-transform_obj['rl']);
```

```
knob_rl.onPress = pressKnob;
knob_rl.onRelease = releaseKnob;
knob_rl.onReleaseOutside = releaseKnob;

knob_rr.top = knob_rr._y;
knob_rr.bottom = knob_rr._y+100;
knob_rr.left = knob_rr._x;
knob_rr.right = knob_rr._x;
knob_rr._y = knob_rr._y+(100-transform_obj['rr']);
knob_rr.onPress = pressKnob;
knob_rr.onRelease = releaseKnob;

knob_rr.onReleaseOutside = releaseKnob;

updateTransformTxt();

function pressKnob() {
  this.startDrag(false, this.left, this.top, this.right, this.bottom);
}
function releaseKnob() {
  this.stopDrag();
  updateTransformTxt();
}
function updateTransformTxt() {
  var ll_num:Number = 30+100-knob_ll._y;
  var lr_num:Number = 30+100-knob_lr._y;
  var rl_num:Number = 30+100-knob_rl._y;
  var rr_num:Number = 30+100-knob_rr._y;
  my_sound.setTransform({ll:ll_num, lr:lr_num, rl:rl_num, rr:rr_num});
  transform_mc.transform_txt.htmlText = "<textformat
  tabStops='[0,30,60,90]'>";
  transform_mc.transform_txt.htmlText +=
  ll_num+"\t"+lr_num+"\t"+rl_num+"\t"+rr_num;
  transform_mc.transform_txt.htmlText += "</textformat>";
}
```

El método `MovieClip.getNextHighestDepth()` utilizado en este ejemplo necesita Flash Player 7 o una versión posterior. Si el archivo SWF incluye un componente de la versión 2, utilice la clase DepthManager de los componentes de la versión 2 en lugar del método `MovieClip.getNextHighestDepth()`.

Véase también

`setTransform (método Sound.setTransform)`

getVolume (método Sound.getVolume)

```
public getVolume() : Number
```

Devuelve el nivel de volumen del sonido como un entero entre 0 y 100, siendo 0 desactivado y 100 volumen total. El valor predeterminado es 100.

Disponibilidad: ActionScript 1.0; Flash Player 5

Valor devuelto

`Number` - Un entero.

Ejemplo

El ejemplo siguiente crea un control deslizante que utiliza la interfaz API de dibujo y un clip de película creado en tiempo de ejecución. Un campo de texto creado de forma dinámica muestra el nivel de volumen actual del sonido que se reproduce en el archivo SWF. Añada el siguiente código ActionScript al archivo AS o FLA:

```
var my_sound:Sound = new Sound();
my_sound.loadSound("song3.mp3", true);

this.createEmptyMovieClip("knob_mc", this.getNextHighestDepth());

knob_mc.left = knob_mc._x;
knob_mc.right = knob_mc.left+100;
knob_mc.top = knob_mc._y;
knob_mc.bottom = knob_mc._y;

knob_mc._x = my_sound.getVolume();

with (knob_mc) {
  lineStyle(0, 0x000000);
  beginFill(0xCCCCCC);
  moveTo(0, 0);
  lineTo(4, 0);
  lineTo(4, 18);
  lineTo(0, 18);
  lineTo(0, 0);
  endFill();
}

knob_mc.createTextField("volume_txt", knob_mc.getNextHighestDepth(),
  knob_mc._width+4, 0, 30, 22);
knob_mc.volume_txt.text = my_sound.getVolume();

knob_mc.onPress = function() {
  this.startDrag(false, this.left, this.top, this.right, this.bottom);
  this.isDragging = true;
```

```
};
knob_mc.onMouseMove = function() {
  if (this.isDragging) {
  this.volume_txt.text = this._x;
  }
}
knob_mc.onRelease = function() {
  this.stopDrag();
  this.isDragging = false;
  my_sound.setVolume(this._x);

};
```

El método `MovieClip.getNextHighestDepth()` utilizado en este ejemplo necesita Flash Player 7 o una versión posterior. Si el archivo SWF incluye un componente de la versión 2, utilice la clase DepthManager de los componentes de la versión 2 en lugar del método `MovieClip.getNextHighestDepth()`.

Véase también

`setVolume (método Sound.setVolume)`

id3 (propiedad Sound.id3)

```
public id3 : Object [read-only]
```

Proporciona acceso a los metadatos que forman parte de un archivo MP3.

Los archivos de sonido MP3 pueden contener etiquetas ID3 que proporcionan metadatos sobre el archivo. Si un sonido MP3 que usted carga utilizando `Sound.attachSound()` o `Sound.loadSound()` contiene etiquetas ID3, puede consultar estas propiedades. Sólo se admiten las etiquetas ID3 que utilizan el conjunto de caracteres UTF-8.

Flash Player 6 (6.0.40.0) y versiones posteriores utilizan la propiedad `Sound.id3` para lograr la compatibilidad con las etiquetas ID3 1.0 e ID3 1.1. Flash Player 7 añade compatibilidad con las etiquetas ID3 2.0, específicamente con las de las versiones 2.3 y 2.4. En la siguiente tabla se enumeran las etiquetas ID3 2.0 estándar y el tipo de contenido que representan las etiquetas; las consultas de estas etiquetas se realizan con el formato *`my_sound`* `.id3.COMM`, *`my_sound`* `.id3.TIME`, etc. Los archivos MP3 pueden contener etiquetas distintas de las indicadas en esta tabla; `Sound.id3` proporciona el acceso también a dichas etiquetas.

Propiedad	Descripción
TFLT	Tipo de archivo
TIME	Tiempo
TIT1	Descripción de grupo de contenido

Propiedad	Descripción
TIT2	Título/nombre de canción/descripción del contenido
TIT3	Subtítulo/descripción adicional
TKEY	Clave inicial
TLAN	Idiomas
TLEN	Longitud
TMED	Tipo de medio
TOAL	Álbum/película/título de espectáculo original
TOFN	Nombre de archivo original
TOLY	Letrista/escritor original
TOPE	Artistas/intérpretes originales
TORY	Año de publicación original
TOWN	Propietario/licenciatario del archivo
TPE1	Intérpretes/solistas principales
TPE2	Grupo/orquesta/acompañamiento
TPE3	Director/intérprete adicional
TPE4	Interpretado, remezclado o editado por
TPOS	Parte de un conjunto
TPUB	Editor
TRCK	Número de pista/posición en el conjunto
TRDA	Fechas de grabación
TRSN	Nombre de emisora de Internet
TRSO	Propietario de emisora de Internet
TSIZ	Tamaño
TSRC	ISRC (código de grabación estándar internacional)
TSSE	Software/hardware y configuración empleados para la codificación
TYER	Año
WXXX	Fotograma de vínculo de URL

Flash Player 6 admitía diversas etiquetas ID3 1.0. Si dichas etiquetas no están en el archivo MP3 pero sí lo están las correspondientes etiquetas ID3 2.0, las etiquetas ID3 2.0 se copiarán en las propiedades de ID3 1.0, como se muestra en la siguiente tabla. Este proceso ofrece compatibilidad con scripts de versiones anteriores que pueda haber escrito con anterioridad y que lean propiedades ID3 1.0.

Etiqueta ID3 2.0	Etiqueta ID3 1.0 correspondiente
COMM	Sound.id3.comment
TALB	Sound.id3.album
TCON	Sound.id3.genre
TIT2	Sound.id3.songname
TPE1	Sound.id3.artist
TRCK	Sound.id3.track
TYER	Sound.id3.year

Disponibilidad: ActionScript 1.0; Flash Player 6

Ejemplo

El ejemplo siguiente sigue la evolución de las propiedades ID3 de song.mp3 en el panel Salida:

```
var my_sound:Sound = new Sound();
my_sound.onID3 = function(){
  for( var prop in my_sound.id3 ){
  trace( prop + " : "+ my_sound.id3[prop] );
  }
}
my_sound.loadSound("song.mp3", false);
```

Véase también

`attachSound (método Sound.attachSound)`, `loadSound (método Sound.loadSound)`

loadSound (método Sound.loadSound)

```
public loadSound(url:String, isStreaming:Boolean) : Void
```

Carga un archivo MP3 en un objeto Sound. Puede utilizar el parámetro `isStreaming` para indicar si el sonido es un evento o un sonido de flujo.

Los sonidos de eventos se cargan completamente antes de reproducirse. Se administran mediante la clase Sound de ActionScript y responden a todos los métodos y propiedades de esta clase.

Los sonidos de flujo se reproducen mientras se están descargando. La reproducción comienza cuando se han recibido suficientes datos para iniciar el descompresor.

Todos los MP3 (sonidos de eventos y de flujo) cargados con este método se guardan en el caché de archivo del navegador del sistema del usuario.

Cuando utilice este método puede usar el modelo de seguridad de Flash Player.

Para Flash Player 8:

- No se admite `Sound.loadSound()` si el archivo SWF que realiza la llamada se encuentra en el entorno limitado local con sistema de archivos y el sonido está en un entorno limitado de red.
- Para acceder desde el entorno limitado local de confianza o local con acceso a la red se necesitan permisos del sitio Web a través de un archivo de política de varios dominios.

Flash Player 7 y versiones anteriores:

- Los sitios Web pueden permitir el acceso a un recurso solicitado desde distintos dominios a través de un archivo de política de varios dominios.

Para más información, consulte lo siguiente:

- Capítulo 17, "Aspectos básicos de la seguridad," en *Aprendizaje de ActionScript 2.0 en Flash*
- El informe sobre la seguridad de Flash Player 8
- El documento técnico sobre interfaces API relativas a seguridad de Flash Player 8 en http://www.macromedia.com/go/fp8_security_apis

Disponibilidad: ActionScript 1.0; Flash Player 6

Parámetros

`url``:String` - La ubicación de un archivo de sonido MP3 en un servidor.

`isStreaming``:Boolean` - Un valor booleano que indica si el sonido es un flujo de sonido (`true`) o un sonido de evento (`false`).

Ejemplo

El ejemplo siguiente carga un sonido de evento, que no puede reproducirse hasta que se ha cargado completamente:

```
var my_sound:Sound = new Sound();
my_sound.loadSound("song1.mp3", false);
```

El siguiente ejemplo carga un flujo de sonido:

```
var my_sound:Sound = new Sound();
my_sound.loadSound("song1.mp3", true);
```

Véase también

onLoad (controlador Sound.onLoad)

onID3 (controlador Sound.onID3)

```
onID3 = function() {}
```

Se invoca cada vez que hay nuevos datos ID3 disponibles para un archivo MP3 cargado mediante `Sound.attachSound()` o `Sound.loadSound()`. Este controlador proporciona acceso a datos ID3 sin realizar ningún sondeo. Si hay etiquetas ID3 1.0 y ID3 2.0 en el archivo, se llamará a este controlador dos veces.

Disponibilidad: ActionScript 1.0; Flash Player 7

Ejemplo

El ejemplo siguiente muestra las propiedades ID3 de `song1.mp3` en una instancia del componente DataGrid. Añada un DataGrid con el nombre de instancia `id3_dg` a su documento y el siguiente código ActionScript a su archivo FLA o AS:

```
import mx.controls.gridclasses.DataGridColumn;
var id3_dg:mx.controls.DataGrid;
id3_dg.move(0, 0);
id3_dg.setSize(Stage.width, Stage.height);
var property_dgc:DataGridColumn = id3_dg.addColumn(new
  DataGridColumn("property"));
property_dgc.width = 100;
property_dgc.headerText = "ID3 Property";
var value_dgc:DataGridColumn = id3_dg.addColumn(new
  DataGridColumn("value"));
value_dgc.width = id3_dg._width-property_dgc.width;
value_dgc.headerText = "ID3 Value";

var my_sound:Sound = new Sound();
my_sound.onID3 = function() {
trace("onID3 called at "+getTimer()+" ms.");
for (var prop in this.id3) {
id3_dg.addItem({property:prop, value:this.id3[prop]});
}
};
my_sound.loadSound("song1.mp3", true);
```

Véase también

attachSound (método Sound.attachSound), id3 (propiedad Sound.id3), loadSound (método Sound.loadSound)

onLoad (controlador Sound.onLoad)

```
onLoad = function(success:Boolean) {}
```

Se invoca automáticamente cuando se carga un sonido. Debe crear la función que debe ejecutarse cuando se invoque el controlador this. Puede utilizar una función anónima o una función con nombre (para ver un ejemplo de cada una, consulte `Sound.onSoundComplete`). Debe definir este controlador antes de llamar a `mySound.loadSound()`.

Disponibilidad: ActionScript 1.0; Flash Player 6

Parámetros

`success``:Boolean` - Un valor booleano `true` si `my_sound` se ha cargado correctamente, `false` en caso contrario.

Ejemplo

El ejemplo siguiente crea un nuevo objeto Sound y carga un sonido. El controlador `onLoad` gestiona la carga del sonido, por lo que puede iniciar la canción una vez que se ha cargado sin errores. Cree un nuevo archivo FLA y añada el siguiente código ActionScript al archivo FLA o AS. Para que este ejemplo funcione, debe tener un MP3 con el nombre `song1.mp3` en el mismo directorio que el archivo FLA o AS.

```
this.createTextField("status_txt", this.getNextHighestDepth(), 0,0,100,22);

// create a new Sound object
var my_sound:Sound = new Sound();
// if the sound loads, play it; if not, trace failure loading
my_sound.onLoad = function(success:Boolean) {
if (success) {
my_sound.start();
status_txt.text = "Sound loaded";
} else {
status_txt.text = "Sound failed";
}
};
// load the sound
my_sound.loadSound("song1.mp3", true);
```

El método `MovieClip.getNextHighestDepth()` utilizado en este ejemplo necesita Flash Player 7 o una versión posterior. Si el archivo SWF incluye un componente de la versión 2, utilice la clase DepthManager de los componentes de la versión 2 en lugar del método `MovieClip.getNextHighestDepth()`.

Véase también

`loadSound (método Sound.loadSound)`

onSoundComplete (controlador Sound.onSoundComplete)

```
onSoundComplete = function() {}
```

Se invoca automáticamente cuando se deja de reproducir un sonido. Puede utilizar este controlador para activar eventos de un archivo SWF cuando un sonido termine de reproducirse.

Debe crear la función que debe ejecutarse cuando se invoque este controlador. Puede utilizar una función anónima o una función con nombre.

Disponibilidad: ActionScript 1.0; Flash Player 6

Ejemplo

Sintaxis 1: El ejemplo siguiente utiliza una función anónima:

```
var my_sound:Sound = new Sound();
my_sound.attachSound("mySoundID");
my_sound.onSoundComplete = function() {
trace("mySoundID completed");
};
my_sound.start();
```

Sintaxis 2: El ejemplo siguiente utiliza una función con nombre:

```
function callback1() {
trace("mySoundID completed");
}
var my_sound:Sound = new Sound();
my_sound.attachSound("mySoundID");
my_sound.onSoundComplete = callback1;
my_sound.start();
```

Véase también

`onLoad (controlador Sound.onLoad)`

position (propiedad Sound.position)

```
public position : Number [read-only]
```

El número de milisegundos que se ha estado reproduciendo un sonido. Si el sonido se reproduce de manera indefinida, la posición se restablece en 0 al comienzo de cada nueva reproducción.

Disponibilidad: ActionScript 1.0; Flash Player 6

Ejemplo

Véase `Sound.duration` para obtener una muestra de cómo se utiliza esta propiedad.

Véase también

`duration (propiedad Sound.duration)`

setPan (método Sound.setPan)

```
public setPan(value:Number) : Void
```

Determina cómo se reproduce el sonido en los canales (altavoces) izquierdo y derecho. En el caso de sonidos mono, *`pan`* determina a través de qué altavoz (izquierdo o derecho) se reproduce el sonido.

Disponibilidad: ActionScript 1.0; Flash Player 5

Parámetros

`value``:Number` - Un entero que especifica el balance izquierda-derecha de un sonido. El intervalo de valores válidos es de -100 a 100, donde -100 utiliza sólo el canal izquierdo, 100 utiliza sólo el canal derecho y 0 reparte el sonido de forma uniforme entre los dos canales.

Ejemplo

Véase `Sound.getPan()` para obtener una muestra de cómo se utiliza este método.

Véase también

`attachSound (método Sound.attachSound)`, `getPan (método Sound.getPan)`, `setTransform (método Sound.setTransform)`, `setVolume (método Sound.setVolume)`, `start (método Sound.start)`

setTransform (método Sound.setTransform)

```
public setTransform(transformObject:Object) : Void
```

Establece la información de transformación (o balance) de sonido para un objeto Sound.

El parámetro `soundTransformObject` es un objeto que se crea utilizando el método constructor de la clase Object genérica con parámetros que especifican cómo se distribuye el sonido entre los canales (altavoces) izquierdo y derecho.

Los sonidos necesitan una cantidad considerable de espacio en disco y de memoria. Dado que los sonidos estéreo utilizan el doble de información que los sonidos mono, normalmente es preferible utilizar sonidos mono de 22-KHz de 6 bits. Puede utilizar `setTransform()` para reproducir sonidos mono como si fueran estéreo, reproducir sonidos estéreo en mono y agregar a los sonidos efectos muy atractivos.

Las propiedades de `soundTransformObject` son éstas:

`ll`: Un valor porcentual que especifica la cantidad de entrada izquierda que debe reproducirse en el altavoz izquierdo (0-100).

`lr`: Un valor porcentual que especifica la cantidad de entrada derecha que debe reproducirse en el altavoz izquierdo (0-100).

`rr`: Un valor porcentual que especifica la cantidad de entrada derecha que debe reproducirse en el altavoz derecho (0-100).

`rl`: Un valor porcentual que especifica la cantidad de entrada izquierda que debe reproducirse en el altavoz derecho (0-100).

El resultado neto de estos parámetros se representa mediante la siguiente fórmula:

```
leftOutput = left_input ~ ll + right_input ~ lr
rightOutput = right_input ~ rr + left_input ~ rl
```

Los valores de `left_input` o `right_input` se determinan mediante el tipo (estéreo o mono) de sonido del archivo SWF.

Los sonidos estéreo dividen la entrada de sonido de manera homogénea entre los altavoces izquierdo y derecho y tienen la siguiente configuración de transformación predeterminada:

```
ll = 100
lr = 0
rr = 100
rl = 0
```

Los sonidos mono reproducen toda la entrada de sonido en el altavoz izquierdo y tienen la siguiente configuración de transformación predeterminada:

```
ll = 100
lr = 100
rr = 0
rl = 0
```

Disponibilidad: ActionScript 1.0; Flash Player 5

Parámetros

transformObject:Object - Un objeto creado con el constructor de la clase Object genérica.

Ejemplo

El ejemplo siguiente muestra una configuración que puede conseguirse mediante el uso de `setTransform()`, pero que no se obtiene si se utilizan `setVolume()` o `setPan()`, aunque se hayan combinado.

El código siguiente crea un nuevo objeto `soundTransformObject` y establece sus propiedades de forma que el sonido de los dos canales se reproduzca solamente por el canal izquierdo.

```
var mySoundTransformObject:Object = new Object();
mySoundTransformObject.ll = 100;
mySoundTransformObject.lr = 100;
mySoundTransformObject.rr = 0;
mySoundTransformObject.rl = 0;
```

Para aplicar el objeto `soundTransformObject` a un objeto Sound, debe pasarlo al objeto Sound mediante `setTransform()` de la forma siguiente:

```
my_sound.setTransform(mySoundTransformObject);
```

El ejemplo siguiente reproduce un sonido estéreo como mono; el objeto `soundTransformObjectMono` tiene los siguientes parámetros:

```
var mySoundTransformObjectMono:Object = new Object();
mySoundTransformObjectMono.ll = 50;
mySoundTransformObjectMono.lr = 50;
mySoundTransformObjectMono.rr = 50;
mySoundTransformObjectMono.rl = 50;
my_sound.setTransform(mySoundTransformObjectMono);
```

Este ejemplo reproduce en el canal izquierdo a media capacidad y añade el resto al canal derecho; el objeto `soundTransformObjectHalf` tiene los siguientes parámetros:

```
var mySoundTransformObjectHalf:Object = new Object();
mySoundTransformObjectHalf.ll = 50;
mySoundTransformObjectHalf.lr = 0;
mySoundTransformObjectHalf.rr = 100;
mySoundTransformObjectHalf.rl = 50;
my_sound.setTransform(mySoundTransformObjectHalf);

var mySoundTransformObjectHalf:Object = {ll:50, lr:0, rr:100, rl:50};
```

Consulte también el ejemplo de `Sound.getTransform()`.

Véase también

`Object`, `getTransform (método Sound.getTransform)`

setVolume (método Sound.setVolume)

```
public setVolume(value:Number) : Void
```

Establece el volumen para el objeto Sound.

Disponibilidad: ActionScript 1.0; Flash Player 5

Parámetros

`value``:Number` - Un número del 0 al 100 que representa un nivel de volumen. 100 es máximo volumen y 0 sin volumen. El valor predeterminado es 100.

Ejemplo

Véase `Sound.getVolume()` para obtener una muestra de cómo se utiliza este método.

Véase también

`setPan (método Sound.setPan)`, `setTransform (método Sound.setTransform)`

Constructor Sound

```
public Sound([target:Object])
```

Crea un nuevo objeto Sound para un clip de película especificado. Si no especifica una instancia de destino, el objeto Sound controlará todos los sonidos de la película.

Disponibilidad: ActionScript 1.0; Flash Player 5

Parámetros

`target``:Object` [opcional] - La instancia de clip de película en la que se utiliza el objeto Sound.

Ejemplo

El siguiente ejemplo crea un nuevo objeto Sound con el nombre `global_sound`. La segunda línea llama a `setVolume()` y ajusta el volumen de todos los sonidos de la película al 50%.

```
var global_sound:Sound = new Sound();
global_sound.setVolume(50);
```

El siguiente ejemplo crea un objeto Sound, lo pasa al clip de película de destino *my_mc* y llama al método `start`, que reproduce todos los sonidos de *my_mc* .

```
var movie_sound:Sound = new Sound(my_mc);
movie_sound.start();
```

start (método Sound.start)

```
public start([secondOffset:Number], [loops:Number]) : Void
```

Comienza la reproducción del último sonido asociado desde el principio, si no hay ningún parámetro especificado, o a partir del punto del sonido especificado por el parámetro `secondOffset`.

Disponibilidad: ActionScript 1.0; Flash Player 5

Parámetros

secondOffset`:Number` [opcional] - Un parámetro que permite iniciar la reproducción del sonido en un punto específico. Por ejemplo, si tiene un sonido de 30 segundos y desea que comience a reproducirse a la mitad, especifique 15 para el parámetro `secondOffset`. El sonido no se retrasa 15 segundos sino que empieza a reproducirse en la marca de 15 segundos.

loops`:Number` [opcional] - Un parámetro que permite especificar el número de veces consecutivas que debe reproducirse el sonido. Este parámetro no está disponible en el caso de flujos de sonido.

Ejemplo

El ejemplo siguiente crea un nuevo objeto Sound y carga un sonido. El controlador `onLoad` gestiona la carga del sonido, por lo que puede iniciar la canción una vez que se ha cargado sin errores. A continuación, comienza a reproducirse el sonido con el método `start()`. Cree un nuevo archivo FLA y añada el siguiente código ActionScript al archivo FLA o AS. Para que este ejemplo funcione, debe tener un MP3 con el nombre song1.mp3 en el mismo directorio que el archivo FLA o AS.

```
this.createTextField("status_txt", this.getNextHighestDepth(), 0,0,100,22);

// create a new Sound object
var my_sound:Sound = new Sound();
// if the sound loads, play it; if not, trace failure loading
my_sound.onLoad = function(success:Boolean) {
  if (success) {
  my_sound.start();
  status_txt.text = "Sound loaded";
  } else {
  status_txt.text = "Sound failed";
  }
```

```
};
// load the sound
my_sound.loadSound("song1.mp3", true);
```

El método `MovieClip.getNextHighestDepth()` utilizado en este ejemplo necesita Flash Player 7 o una versión posterior. Si el archivo SWF incluye un componente de la versión 2, utilice la clase DepthManager de los componentes de la versión 2 en lugar del método `MovieClip.getNextHighestDepth()`.

Véase también

`stop (método Sound.stop)`

stop (método Sound.stop)

```
public stop([linkageID:String]) : Void
```

Detiene todos los sonidos que se están reproduciendo actualmente, si no se ha especificado ningún parámetro, o sólo el sonido especificado por el parámetro `idName`.

Disponibilidad: ActionScript 1.0; Flash Player 5

Parámetros

`linkageID``:String` [opcional] - Un parámetro que especifica que deje de reproducirse un sonido determinado. El parámetro `idName` debe ir entre comillas (" ").

Ejemplo

El ejemplo siguiente emplea dos botones, `stop_btn` y `play_btn`, para controlar la reproducción de un sonido que se carga en un archivo SWF. Añada dos botones a su documento y el siguiente código ActionScript a su archivo FLA o AS:

```
var my_sound:Sound = new Sound();
my_sound.loadSound("song1.mp3", true);

stop_btn.onRelease = function() {
  trace("sound stopped");
  my_sound.stop();
};
play_btn.onRelease = function() {
  trace("sound started");
  my_sound.start();
};
```

Véase también

`start (método Sound.start)`

Stage

```
Object
  |
  +-Stage
```

```
public class Stage
extends Object
```

La clase Stage es una clase de nivel superior a cuyos métodos, propiedades y controladores puede acceder sin emplear un constructor. Utilice los métodos y propiedades de esta clase para acceder a la información de límites de un archivo SWF y manipularla.

Disponibilidad: ActionScript 1.0; Flash Player 5

Resumen de propiedades

Modificadores	Propiedad	Descripción
`static`	`align:String`	Indica la alineación actual del archivo SWF en el reproductor o navegador.
`static`	`height:Number`	Propiedad (sólo lectura); indica la altura actual en píxeles del escenario.
`static`	`scaleMode:String`	Indica la escala actual del archivo SWF dentro de Flash Player.
`static`	`showMenu:Boolean`	Especifica si deben mostrarse u ocultarse los elementos predeterminados del menú contextual de Flash Player.
`static`	`width:Number`	Propiedad (sólo lectura); indica la anchura actual en píxeles del escenario.

Propiedades heredadas de la clase Object

`constructor (propiedad Object.constructor), __proto__ (propiedad Object.__proto__), prototype (propiedad Object.prototype), __resolve (propiedad Object.__resolve)`

Resumen de eventos

Evento	Descripción
`onResize = function() {}`	Se invoca cuando `Stage.scaleMode` se establece como noScale y se redimensiona el archivo SWF.

Resumen de métodos

Modificadores	Firma	Descripción
static	addListener(listener:Object) : Void	Detecta cuándo se redimensiona un archivo SWF (pero sólo si Stage.scaleMode = "noScale").
static	removeListener(listener:Object) : Boolean	Elimina un objeto detector creado con addListener().

Métodos heredados de la clase Object

addProperty (método Object.addProperty), hasOwnProperty (método Object.hasOwnProperty), isPropertyEnumerable (método Object.isPropertyEnumerable), isPrototypeOf (método Object.isPrototypeOf), registerClass (método Object.registerClass), toString (método Object.toString), unwatch (método Object.unwatch), valueOf (método Object.valueOf), watch (método Object.watch)

addListener (método Stage.addListener)

```
public static addListener(listener:Object) : Void
```

Detecta cuándo se redimensiona un archivo SWF (pero sólo si `Stage.scaleMode = "noScale"`). El método `addListener()` no funciona con la configuración predeterminada de escala de clip de película (`showAll`) ni con otras configuraciones de escala (`exactFit` y `noBorderr`).

Para utilizar `addListener()`, deberá crear primero un *objeto detector*. Los objetos detectores del escenario reciben notificación de `Stage.onResize`.

Disponibilidad: ActionScript 1.0; Flash Player 6

Parámetros

`listener``:Object` - Un objeto que detecta una notificación callback de los controladores de eventos `Stage.onResize`.

Ejemplo

Este ejemplo crea un nuevo objeto detector con el nombre `stageListener`. A continuación, utiliza `stageListener` para llamar a `onResize` y definir una función que se llamará cuando se active `onResize`. Por último, el código añade el objeto `stageListener` a la lista de funciones callback del objeto Stage. Los objetos detectores permiten que varios objetos detecten las notificaciones de cambio de tamaño.

```
this.createTextField("stageSize_txt", this.getNextHighestDepth(), 10, 10,
  100, 22);
var stageListener:Object = new Object();
stageListener.onResize = function() {
  stageSize_txt.text = "w:"+Stage.width+", h:"+Stage.height;
};
Stage.scaleMode = "noScale";
Stage.addListener(stageListener);
```

Véase también

`onResize (detector de eventos Stage.onResize)`, `removeListener (método Stage.removeListener)`

align (propiedad Stage.align)

```
public static align : String
```

Indica la alineación actual del archivo SWF en el reproductor o navegador.

En la siguiente tabla se enumeran los valores de la propiedad `align`. Todos aquellos valores no incluidos aquí centran el archivo SWF en el área de Flash Player o el navegador, que es la configuración predeterminada.

Valor	*Vertical*	*Horizontal*
"T"	top	center
"B"	bottom	center
"L"	center	left
"R"	center	right
"TL"	top	left
"TR"	top	right
"BL"	bottom	left
"BR"	bottom	right

Disponibilidad: ActionScript 1.0; Flash Player 6

Ejemplo

El ejemplo siguiente demuestra las distintas alineaciones del archivo SWF. Añada una instancia de ComboBox al documento con el nombre `stageAlign_cb`. Añada el siguiente código ActionScript al archivo FLA o AS:

```
var stageAlign_cb:mx.controls.ComboBox;
stageAlign_cb.dataProvider = ['T', 'B', 'L', 'R', 'TL', 'TR', 'BL', 'BR'];
var cbListener:Object = new Object();
cbListener.change = function(evt:Object) {
  var align:String = evt.target.selectedItem;
  Stage.align = align;
};
stageAlign_cb.addEventListener("change", cbListener);
Stage.scaleMode = "noScale";
```

Seleccione distintas configuraciones de alineación desde el ComboBox.

height (propiedad Stage.height)

```
public static height : Number
```

Propiedad (sólo lectura); indica la altura actual en píxeles del escenario. Cuando el valor de `Stage.scaleMode` es noScale, la propiedad height representa la altura de Flash Player. Cuando el valor de `Stage.scaleMode` no es noScale, la propiedad height representa la altura del archivo SWF.

Disponibilidad: ActionScript 1.0; Flash Player 6

Ejemplo

Este ejemplo crea un nuevo objeto detector con el nombre `stageListener`. A continuación, utiliza `myListener` para llamar a `onResize` y definir una función que se llamará cuando se active `onResize`. Por último, el código añade el objeto `myListener` a la lista de funciones callback del objeto Stage. Los objetos detectores permiten que varios objetos detecten las notificaciones de cambio de tamaño.

```
this.createTextField("stageSize_txt", this.getNextHighestDepth(), 10, 10,
  100, 22);
var stageListener:Object = new Object();
stageListener.onResize = function() {
  stageSize_txt.text = "w:"+Stage.width+", h:"+Stage.height;
};
Stage.scaleMode = "noScale";
Stage.addListener(stageListener);
```

Véase también

align (propiedad Stage.align), scaleMode (propiedad Stage.scaleMode), width (propiedad Stage.width)

onResize (detector de eventos Stage.onResize)

```
onResize = function() {}
```

Se invoca cuando `Stage.scaleMode` se establece como noScale y se redimensiona el archivo SWF. Puede utilizar este controlador de eventos para escribir una función que disponga los objetos en el escenario cuando se redimensione el archivo SWF.

```
myListener.onResize = function(){
 // your statements here
 }
```

Disponibilidad: ActionScript 1.0; Flash Player 6

Ejemplo

El siguiente ejemplo muestra un mensaje en el panel Salida cuando se cambia el tamaño del escenario:

```
Stage.scaleMode = "noScale"
var myListener:Object = new Object();
myListener.onResize = function () {
  trace("Stage size is now " + Stage.width + " by " + Stage.height);
}
Stage.addListener(myListener);
// later, call Stage.removeListener(myListener)
```

Véase también

scaleMode (propiedad Stage.scaleMode), addListener (método Stage.addListener), removeListener (método Stage.removeListener)

removeListener (método Stage.removeListener)

```
public static removeListener(listener:Object) : Boolean
```

Elimina un objeto detector creado con addListener().

Disponibilidad: ActionScript 1.0; Flash Player 6

Parámetros

listener:Object - Un objeto añadido a una lista de funciones de callback de un objeto con addListener().

Valor devuelto

Boolean - Valor booleano.

Ejemplo

El ejemplo siguiente muestra las dimensiones del escenario en un campo de texto creado de forma dinámica. Cuando se modifica el tamaño del escenario, se actualizan los valores del campo de texto. Cree un botón con el nombre de instancia remove_btn. Añada el siguiente código ActionScript al fotograma 1 de la línea de tiempo.

```
this.createTextField("stageSize_txt", this.getNextHighestDepth(), 10, 10,
  100, 22);
stageSize_txt.autoSize = true;
stageSize_txt.border = true;
var stageListener:Object = new Object();
stageListener.onResize = function() {
  stageSize_txt.text = "w:"+Stage.width+", h:"+Stage.height;
};
Stage.addListener(stageListener);

remove_btn.onRelease = function() {
  stageSize_txt.text = "Removing Stage listener...";
  Stage.removeListener(stageListener);
}
```

Seleccione Control > Probar película para probar este ejemplo. Los valores que ve en el campo de texto se actualizan cuando cambia el tamaño del entorno de prueba. Al hacer clic en remove_btn, el detector desaparece y los valores dejan de actualizarse en el campo de texto.

Véase también

addListener (método Stage.addListener)

scaleMode (propiedad Stage.scaleMode)

```
public static scaleMode : String
```

Indica la escala actual del archivo SWF dentro de Flash Player. La propiedad scaleMode fuerza al archivo SWF a adoptar un modo de escala específico. De manera predeterminada, el archivo SWF utiliza los parámetros HTML establecidos en el cuadro de diálogo Configuración de publicación.

La propiedad `scaleMode` puede utilizar los valores `"exactFit"`, `"showAll"`, `"noBorder"` y `"noScale"`. Cualquier otro valor establece la propiedad `scaleMode` con el valor predeterminado, `"showAll"`.

- `showAll` (Predeterminada) permite ver todo el contenido de Flash en el área especificada sin distorsión, al mismo tiempo que mantiene la proporción original del contenido. Es posible que aparezcan bordes a ambos lados de la aplicación.
- `noBorder` modifica la escala del contenido de Flash para que ocupe toda el área especificada, sin distorsión pero quizá con algún recorte, mientras se mantiene la proporción original de la aplicación.
- `exactFit` hace que la totalidad del contenido de Flash sea visible en el área especificada sin intentar mantener la proporción original. Puede ocurrir una distorsión.
- `noScale` hace que el tamaño del contenido de Flash sea fijo, de manera que permanezca sin cambios aunque cambie el tamaño de la ventana del reproductor. Puede producirse un recorte si la ventana del reproductor es más pequeña que el contenido de Flash.

Nota: la configuración predeterminada es showAll, salvo en el modo de probar película, en el que la configuración predeterminada es noScale

Disponibilidad: ActionScript 1.0; Flash Player 6

Ejemplo

El ejemplo siguiente prueba varias configuraciones de escala para el archivo SWF. Añada una instancia de ComboBox al documento con el nombre `scaleMode_cb`. Añada el siguiente código ActionScript al archivo FLA o AS:

```
var scaleMode_cb:mx.controls.ComboBox;
scaleMode_cb.dataProvider = ["showAll", "exactFit", "noBorder", "noScale"];
var cbListener:Object = new Object();
cbListener.change = function(evt:Object) {
  var scaleMode_str:String = evt.target.selectedItem;
  Stage.scaleMode = scaleMode_str;
};
scaleMode_cb.addEventListener("change", cbListener);
```

Para ver otro ejemplo, consulte el archivo stagesize.fla en la carpeta de ejemplos de ActionScript. La lista siguiente muestra rutas habituales a la carpeta de ejemplos de ActionScript:

- Windows: *unidad de inicio*\Archivos de programa\Macromedia\Flash 8\Samples and Tutorials\Samples\ActionScript
- Macintosh: *Disco duro de Macintosh*/Applications/Macromedia Flash 8/Samples and Tutorials/Samples/ActionScript

showMenu (propiedad Stage.showMenu)

```
public static showMenu : Boolean
```

Especifica si deben mostrarse u ocultarse los elementos predeterminados del menú contextual de Flash Player. Si `showMenu` se establece con el valor `true` (que es el valor predeterminado), aparecerán todos los elementos del menú contextual. Si `showMenu` se establece con el valor `false`, sólo aparecerán Configuración y Acerca de Macromedia Flash Player.

Disponibilidad: ActionScript 1.0; Flash Player 6

Ejemplo

El ejemplo siguiente crea un vínculo de texto que permite al usuario hacer clic para activar y desactivar el menú contextual de Flash Player.

```
this.createTextField("showMenu_txt", this.getNextHighestDepth(), 10, 10,
  100, 22);
showMenu_txt.html = true;
showMenu_txt.autoSize = true;
showMenu_txt.htmlText = "<a
  href=\"asfunction:toggleMenu\"><u>Stage.showMenu = "+Stage.showMenu+"</
  u></a>";
function toggleMenu() {
  Stage.showMenu = !Stage.showMenu;
  showMenu_txt.htmlText = "<a
  href=\"asfunction:toggleMenu\"><u>Stage.showMenu = "+Stage.showMenu+"</
  u></a>";
}
```

Véase también

`ContextMenu`, `ContextMenuItem`

width (propiedad Stage.width)

```
public static width : Number
```

Propiedad (sólo lectura); indica la anchura actual en píxeles del escenario. Cuando el valor de `Stage.scaleMode` es `"noScale"`, la propiedad `width` representa la anchura de Flash Player. Esto significa que `Stage.width` variará al redimensionar la ventana del reproductor. Cuando el valor de `Stage.scaleMode` no es `"noScale"`, `width` representa la anchura del archivo SWF configurada durante la edición del archivo en el cuadro de diálogo Propiedades del documento. Esto significa que el valor de `width` se mantiene constante al redimensionar la ventana del reproductor.

Disponibilidad: ActionScript 1.0; Flash Player 6

Ejemplo

Este ejemplo crea un nuevo objeto detector con el nombre `stageListener`. A continuación, utiliza `stageListener` para llamar a `onResize` y definir una función que se llamará cuando se active `onResize`. Por último, el código añade el objeto `stageListener` a la lista de funciones callback del objeto Stage. Los objetos detectores permiten que varios objetos detecten las notificaciones de cambio de tamaño.

```
this.createTextField("stageSize_txt", this.getNextHighestDepth(), 10, 10,
  100, 22);
var stageListener:Object = new Object();
stageListener.onResize = function() {
  stageSize_txt.text = "w:"+Stage.width+", h:"+Stage.height;
};
Stage.scaleMode = "noScale";
Stage.addListener(stageListener);
```

Véase también

`align (propiedad Stage.align)`, `height (propiedad Stage.height)`, `scaleMode (propiedad Stage.scaleMode)`

String

```
Object
  |
  +-String
```

`public class` ***String***
`extends Object`

La clase String es un envolvente para el tipo de datos primitivo de cadena y proporciona métodos y propiedades que le permiten manipular tipos de valores de cadena primitivos. Puede convertir el valor de cualquier objeto en una cadena utilizando la función `String()`.

Todos los métodos de la clase String, salvo `concat()`), `fromCharCode()`, `slice()` y `substr()`, son genéricos, lo que significa que los métodos llaman a `toString()` antes de realizar sus operaciones y puede llamar a estos métodos con otros objetos que no sean String.

Dado que todos los índices de cadenas están basados en cero, el índice del último carácter de cualquier cadena `x` es `x.length - 1`.

Puede llamar a cualquiera de los métodos de la clase String utilizando el método constructor `new String` o utilizando un nuevo valor de literal de cadena. Si especifica un literal de cadena, el intérprete de ActionScript lo convierte automáticamente en un objeto String temporal, llama al método y luego descarta el objeto String temporal. También puede utilizar la propiedad `String.length` con un literal de cadena.

No debe confundir un literal de cadena con un objeto String. En el siguiente ejemplo, la primera línea de código crea el literal de cadena `first_string`, mientras que la segunda línea de código crea el objeto String `second_string`:

```
var first_string:String = "foo"
var second_string:String = new String("foo")
```

Utilice literales de cadena a no ser que necesite utilizar un objeto String específicamente.

Disponibilidad: ActionScript 1.0; Flash Player 5

Resumen de propiedades

Modificadores	Propiedad	Descripción
	`length:Number`	Un entero que especifica el número de caracteres del objeto String especificado.

Propiedades heredadas de la clase Object

`constructor (propiedad Object.constructor), __proto__ (propiedad Object.__proto__), prototype (propiedad Object.prototype), __resolve (propiedad Object.__resolve)`

Resumen de constructores

Firma	Descripción
`String(value:String)`	Crea un nuevo objeto String.

Resumen de métodos

Modificadores	Firma	Descripción
	`charAt(index:Number) : String`	Devuelve el carácter situado en la posición especificada por el parámetro `index`.
	`charCodeAt(index:Number) : Number`	Devuelve un entero de 16 bits comprendido entre 0 y 65535 que representa el carácter especificado por `index`.
	`concat(value:Object) : String`	Combina el valor del objeto String con los parámetros y devuelve la nueva cadena formada; el valor original, `my_str`, permanece sin cambios.
`static`	`fromCharCode() : String`	Devuelve una cadena formada por los caracteres representados por los valores Unicode de los parámetros.

Modificadores	Firma	Descripción
	`indexOf(value:String , [startIndex:Number]) : Number`	Busca la cadena y devuelve la posición en la que aparece por primera vez `value` en la posición de `startIndex` o después de dicha posición dentro de la cadena que origina la llamada.
	`lastIndexOf(value:String, [startIndex:Number]) : Number`	Busca la cadena de derecha a izquierda y devuelve el índice de la última posición en que aparece `value` antes que `startIndex` dentro de la cadena que origina la llamada.
	`slice(start:Number, end:Number) : String`	Devuelve una cadena que incluye el carácter inicial (`start`) y todos los caracteres que aparecen hasta el carácter final (`end`), exclusive.
	`split(delimiter:String, [limit:Number]) : Array`	Divide un objeto String en subcadenas fragmentándolo siempre que aparece el parámetro delimitador (`delimiter`) especificado y devuelve las subcadenas en una matriz.
	`substr(start:Number, length:Number) : String`	Devuelve los caracteres contenidos en una cadena desde el índice especificado por el parámetro `start` hasta el número de caracteres especificado por el parámetro `length`.
	`substring(start:Number, end:Number) : String`	Devuelve una cadena formada por los caracteres comprendidos entre los puntos especificados por los parámetros `start` y `end`.
	`toLowerCase() : String`	Devuelve una copia del objeto `String` con todos los caracteres en mayúsculas convertidos en minúsculas.
	`toString() : String`	Devuelve las propiedades de un objeto como cadenas aunque no lo sean.
	`toUpperCase() : String`	Devuelve una copia del objeto String con todos los caracteres en minúsculas convertidos en mayúsculas.
	`valueOf() : String`	Devuelve la cadena.

Métodos heredados de la clase Object

`addProperty (método Object.addProperty), hasOwnProperty (método Object.hasOwnProperty), isPropertyEnumerable (método Object.isPropertyEnumerable), isPrototypeOf (método Object.isPrototypeOf), registerClass (método Object.registerClass), toString (método Object.toString), unwatch (método Object.unwatch), valueOf (método Object.valueOf), watch (método Object.watch)`

charAt (método String.charAt)

```
public charAt(index:Number) : String
```

Devuelve el carácter situado en la posición especificada por el parámetro `index`. Si `index` no es un número entre 0 y `string.length` - 1, se devolverá una cadena vacía.

Este método es igual que `String.charCodeAt()`, con la diferencia de que el valor devuelto es un carácter, no un código de carácter de entero de 16 bits.

Disponibilidad: ActionScript 1.0; Flash Player 5

Parámetros

`index``:Number` - Un entero que especifica la posición de un carácter en la cadena. El primer carácter se indica por `0` y el último, por `my_str .length-1`.

Valor devuelto

`String` - El carácter en el índice especificado. O una `String` vacía si el índice especificado está fuera del rango de los índices de esta `String`.

Ejemplo

En el ejemplo siguiente, este método se llama en la primera letra de la cadena `Chris`:

```
var my_str:String = "Chris";
var firstChar_str:String = my_str.charAt(0);
trace(firstChar_str); // output: C
```

Véase también

`charCodeAt (método String.charCodeAt)`

charCodeAt (método String.charCodeAt)

```
public charCodeAt(index:Number) : Number
```

Devuelve un entero de 16 bits comprendido entre 0 y 65535 que representa el carácter especificado por `index`. Si `index` no es un número entre 0 y `string.length` - 1, se devolverá `NaN`.

Este método es igual que `String.charAt()`, con la diferencia de que el valor devuelto es un código de carácter de entero de 16 bits, no un carácter.

Disponibilidad: ActionScript 1.0; Flash Player 5

Parámetros

index:Number - Un entero que especifica la posición de un carácter en la cadena. El primer carácter se indica por 0, y el último, por my_str.length - 1.

Valor devuelto

Number - Un entero que representa el carácter especificado por index.

Ejemplo

En el ejemplo siguiente, este método se llama en la primera letra de la cadena "Chris":

```
var my_str:String = "Chris";
var firstChar_num:Number = my_str.charCodeAt(0);
trace(firstChar_num); // output: 67
```

Véase también

charAt (método String.charAt)

concat (método String.concat)

```
public concat(value:Object) : String
```

Combina el valor del objeto String con los parámetros y devuelve la nueva cadena formada; el valor original, my_str, permanece sin cambios.

Disponibilidad: ActionScript 1.0; Flash Player 5

Parámetros

value:Object - value1[,...valueN] Cero o más valores para concatenar.

Valor devuelto

String - Una cadena.

Ejemplo

El ejemplo siguiente crea dos cadenas y las combina utilizado String.concat():

```
var stringA:String = "Hello";
var stringB:String = "World";
var combinedAB:String = stringA.concat(" ", stringB);
trace(combinedAB); // output: Hello World
```

fromCharCode (método String.fromCharCode)

```
public static fromCharCode() : String
```

Devuelve una cadena formada por los caracteres representados por los valores Unicode de los parámetros.

Disponibilidad: ActionScript 1.0; Flash Player 5

Valor devuelto

`String` - Una cadena de los códigos de caracteres Unicode especificados.

Ejemplo

El ejemplo siguiente utiliza `fromCharCode()` para insertar un carácter `@` en la dirección de correo electrónico:

```
var address_str:String = "dog"+String.fromCharCode(64)+"house.net";
trace(address_str); // output: dog@house.net
```

indexOf (método String.indexOf)

```
public indexOf(value:String, [startIndex:Number]) : Number
```

Busca la cadena y devuelve la posición en la que aparece por primera vez `value` en la posición de `startIndex` o después de dicha posición dentro de la cadena que origina la llamada. Este índice está basado en cero, con lo cual se considera que el primer carácter está en el índice 0, no en el índice 1. Si no se encuentra `value`, el método devuelve -1.

Disponibilidad: ActionScript 1.0; Flash Player 5

Parámetros

`value``:String` - Una cadena que especifica que texto de la búsqueda.

`startIndex``:Number` [opcional] - Un entero que especifica el índice inicial de la búsqueda.

Valor devuelto

`Number` - La posición en que aparece por primera vez la subcadena especificada o -1.

Ejemplo

Los siguientes ejemplos utilizan `indexOf()` para devolver el índice de caracteres y subcadenas:

```
var searchString:String = "Lorem ipsum dolor sit amet.";
var index:Number;

index = searchString.indexOf("L");
trace(index); // output: 0
```

```
index = searchString.indexOf("l");
trace(index); // output: 14

index = searchString.indexOf("i");
trace(index); // output: 6

index = searchString.indexOf("ipsum");
trace(index); // output: 6

index = searchString.indexOf("i", 7);
trace(index); // output: 19

index = searchString.indexOf("z");
trace(index); // output: -1
```

Véase también

lastIndexOf (método String.lastIndexOf)

lastIndexOf (método String.lastIndexOf)

```
public lastIndexOf(value:String, [startIndex:Number]) : Number
```

Busca la cadena de derecha a izquierda y devuelve el índice de la última posición en que aparece `value` antes que `startIndex` dentro de la cadena que origina la llamada. Este índice está basado en cero, con lo cual se considera que el primer carácter está en el índice 0, no en el índice 1. Si no se encuentra `value`, el método devuelve -1.

Disponibilidad: ActionScript 1.0; Flash Player 5

Parámetros

`value``:String` - Una cadena; la cadena que se busca.

`startIndex``:Number` [opcional] - Un entero que especifica dónde se inicia la búsqueda de `value`.

Valor devuelto

`Number` - La posición en que aparece por última vez la subcadena especificada o -1.

Ejemplo

El ejemplo siguiente muestra cómo utilizar `lastIndexOf()` para devolver el índice de un carácter determinado:

```
var searchString:String = "Lorem ipsum dolor sit amet.";
var index:Number;
```

```
index = searchString.lastIndexOf("L");
trace(index); // output: 0

index = searchString.lastIndexOf("l");
trace(index); // output: 14

index = searchString.lastIndexOf("i");
trace(index); // output: 19

index = searchString.lastIndexOf("ipsum");
trace(index); // output: 6

index = searchString.lastIndexOf("i", 18);
trace(index); // output: 6

index = searchString.lastIndexOf("z");
trace(index); // output: -1
```

Véase también

`indexOf (método String.indexOf)`

length (propiedad String.length)

```
public length : Number
```

Un entero que especifica el número de caracteres del objeto String especificado.

Dado que todos los índices de cadenas están basados en cero, el índice del último carácter de cualquier cadena `x` es `x.length - 1`.

Disponibilidad: ActionScript 1.0; Flash Player 5

Ejemplo

El ejemplo siguiente crea un nuevo objeto String y utiliza `String.length` para contar el número de caracteres:

```
var my_str:String = "Hello world!";
trace(my_str.length); // output: 12
```

El ejemplo siguiente pasa de 0 a `my_str.length`. El código comprueba los caracteres de una cadena y si contiene el carácter `@`, muestra `true` en el panel Salida. Si no contiene el carácter `@` aparece `false` en el panel Salida.

```
function checkAtSymbol(my_str:String):Boolean {
  for (var i = 0; i<my_str.length; i++) {
  if (my_str.charAt(i) == "@") {
    return true;
  }
  }
```

```
  return false;
}

trace(checkAtSymbol("dog@house.net")); // output: true
trace(checkAtSymbol("Chris")); // output: false
```

Encontrará también un ejemplo en el archivo Strings.fla de la carpeta de ejemplos de ActionScript. La lista siguiente muestra rutas habituales a esta carpeta:

- Windows: *unidad de inicio*\Archivos de programa\Macromedia\Flash 8\Samples and Tutorials\Samples\ActionScript
- Macintosh: *Disco duro de Macintosh*/Applications/Macromedia Flash 8/Samples and Tutorials/Samples/ActionScript

slice (método String.slice)

```
public slice(start:Number, end:Number) : String
```

Devuelve una cadena que incluye el carácter inicial (`start`) y todos los caracteres que aparecen hasta el carácter final (`end`), exclusive. El objeto String original no se modifica. Si no se especifica el parámetro `end`, el final de la subcadena será el final de la cadena. Si el carácter indexado por `start` es el mismo o está situado a la derecha del carácter indexado por `end`, el método devolverá una cadena vacía.

Disponibilidad: ActionScript 1.0; Flash Player 5

Parámetros

`start``:Number` - El índice basado en cero del punto inicial de la división. Si `start` es un número negativo, el punto de inicio se establece desde el final de la cadena, donde -1 es el último carácter.

`end``:Number` - Un entero que es un número por encima del índice del punto final de la división. El carácter indexado por el parámetro `end` no se incluye en la cadena extraída. Si se omite este parámetro, se empleará `String.length`. Si `end` es un número negativo, el punto final se establece desde el final de la cadena, donde -1 es el último carácter.

Valor devuelto

`String` - Una subcadena de la cadena especificada.

Ejemplo

El ejemplo siguiente crea una variable, `my_str`, , le asigna un valor String y llama al método `slice()` utilizando distintos valores para los parámetros `start` y `end`. Cada llamada a `slice()` se ajusta en una sentencia `trace()` que muestra la salida en el panel Salida.

```
// Index values for the string literal
// positive index: 0 1 2 3 4
// string: L o r e m
// negative index: -5 -4 -3 -2 -1

var my_str:String = "Lorem";

// slice the first character
trace("slice(0,1): "+my_str.slice(0, 1)); // output: slice(0,1): L
trace("slice(-5,1): "+my_str.slice(-5, 1)); // output: slice(-5,1): L

// slice the middle three characters
trace("slice(1,4): "+my_str.slice(1, 4)); // slice(1,4): ore
trace("slice(1,-1): "+my_str.slice(1, -1)); // slice(1,-1): ore

// slices that return empty strings because start is not to the left of end
trace("slice(1,1): "+my_str.slice(1, 1)); // slice(1,1):
trace("slice(3,2): "+my_str.slice(3, 2)); // slice(3,2):
trace("slice(-2,2): "+my_str.slice(-2, 2)); // slice(-2,2):

// slices that omit the end parameter use String.length, which equals 5
trace("slice(0): "+my_str.slice(0)); // slice(0): Lorem
trace("slice(3): "+my_str.slice(3)); // slice(3): em
```

Encontrará también un ejemplo en el archivo Strings.fla de la carpeta de ejemplos de ActionScript. La lista siguiente muestra rutas habituales a esta carpeta:

- Windows: *unidad de inicio*\Archivos de programa\Macromedia\Flash 8\Samples and Tutorials\Samples\ActionScript
- Macintosh: *Disco duro de Macintosh*/Applications/Macromedia Flash 8/Samples and Tutorials/Samples/ActionScript

Véase también

`substr (método String.substr)`, `substring (método String.substring)`

split (método String.split)

```
public split(delimiter:String, [limit:Number]) : Array
```

Divide un objeto String en subcadenas fragmentándolo siempre que aparece el parámetro delimitador (`delimiter`) especificado y devuelve las subcadenas en una matriz. Si utiliza una cadena vacía ("") como delimitador, cada carácter de la cadena se situará como elemento de la matriz.

Si el parámetro `delimiter` tiene el valor undefined, se situará la cadena completa en el primer elemento de la matriz devuelta.

Disponibilidad: ActionScript 1.0; Flash Player 5

Parámetros

`delimiter``:String` - Una cadena; el carácter o cadena donde se divide `my_str`.

`limit``:Number` [opcional] - El número de elementos que se va a incluir en la matriz.

Valor devuelto

`Array` - Una matriz que contiene las subcadenas de `my_str`.

Ejemplo

El ejemplo siguiente devuelve una matriz con cinco elementos:

```
var my_str:String = "P,A,T,S,Y";
var my_array:Array = my_str.split(",");
for (var i = 0; i<my_array.length; i++) {
  trace(my_array[i]);
}
// output:
  P
  A
  T
  S
  Y
```

El ejemplo siguiente devuelve una matriz con dos elementos, `"P"` y `"A"`:

```
var my_str:String = "P,A,T,S,Y";
var my_array:Array = my_str.split(",", 2);
trace(my_array); // output: P,A
```

El ejemplo siguiente muestra que si utiliza una cadena vacía ("") para el parámetro `delimiter`, cada carácter de la cadena se situará como elemento de la matriz:

```
var my_str:String = new String("Joe");
var my_array:Array = my_str.split("");
for (var i = 0; i<my_array.length; i++) {
```

```
  trace(my_array[i]);
}
// output:
  J
  o
  e
```

Encontrará también un ejemplo en el archivo Strings.fla de la carpeta de ejemplos de ActionScript. La lista siguiente muestra rutas habituales a esta carpeta:

- Windows: *unidad de inicio*\Archivos de programa\Macromedia\Flash 8\Samples and Tutorials\Samples\ActionScript
- Macintosh: *Disco duro de Macintosh*/Applications/Macromedia Flash 8/Samples and Tutorials/Samples/ActionScript

Véase también

`join (método Array.join)`

Constructor String

```
public String(value:String)
```

Crea un nuevo objeto String.

Nota: Dado que los literales de cadena utilizan menos espacio que los objetos String y son generalmente más fáciles de usar, deberá utilizar literales de cadena en lugar del constructor de la clase String a no ser que exista un motivo específico para utilizar un objeto String en lugar de un literal de cadena.

Disponibilidad: ActionScript 1.0; Flash Player 5

Parámetros

`value``:String` - Valor inicial del nuevo objeto String.

substr (método String.substr)

```
public substr(start:Number, length:Number) : String
```

Devuelve los caracteres contenidos en una cadena desde el índice especificado por el parámetro `start` hasta el número de caracteres especificado por el parámetro `length`. El método `substr` no cambia la cadena especificada por `my_str`, sino que devuelve una cadena nueva.

Disponibilidad: ActionScript 1.0; Flash Player 5

Parámetros

start:Number - Un entero que indica la posición del primer carácter de my_str que se va a utilizar para crear la subcadena. Si start es un número negativo, el punto de inicio se establece desde el final de la cadena, donde -1 es el último carácter.

length:Number - Número de caracteres de la subcadena que se va a crear. Si no se especifica length, la subcadena incluye todos los caracteres desde el principio al final de la cadena.

Valor devuelto

String - Una subcadena de la cadena especificada.

Ejemplo

El ejemplo siguiente crea una nueva cadena, my_str y utiliza substr() para devolver la segunda palabra de la cadena; primero utilizando un parámetro start positivo y, a continuación, con un parámetro start negativo:

```
var my_str:String = new String("Hello world");
var mySubstring:String = new String();
mySubstring = my_str.substr(6,5);
trace(mySubstring); // output: world

mySubstring = my_str.substr(-5,5);
trace(mySubstring); // output: world
```

Encontrará también un ejemplo en el archivo Strings.fla de la carpeta de ejemplos de ActionScript. La lista siguiente muestra rutas habituales a esta carpeta:

- Windows: *unidad de inicio*\Archivos de programa\Macromedia\Flash 8\Samples and Tutorials\Samples\ActionScript
- Macintosh: *Disco duro de Macintosh*/Applications/Macromedia Flash 8/Samples and Tutorials/Samples/ActionScript

substring (método String.substring)

```
public substring(start:Number, end:Number) : String
```

Devuelve una cadena formada por los caracteres comprendidos entre los puntos especificados por los parámetros start y end. Si no se especifica el parámetro end, el final de la subcadena será el final de la cadena. Si el valor de start es igual al valor de end, el método devolverá una cadena vacía. Si el valor de start es mayor que el valor de end, los parámetros se intercambiarán automáticamente antes de que se ejecute la función y el valor original permanecerá sin cambios.

Disponibilidad: ActionScript 1.0; Flash Player 5

Parámetros

start:Number - Un entero que indica la posición del primer carácter de my_str que se va a utilizar para crear la subcadena. Los valores válidos para start incluyen del 0 a String.length - 1. Si start es un número negativo, se utiliza 0.

end:Number - Un entero que es 1+ el índice del último carácter de my_str que se va a extraer. Los valores válidos para end son desde 1 hasta String.length. El carácter indexado por el parámetro end no se incluye en la cadena extraída. Si se omite este parámetro, se empleará String.length. Si este parámetro es un valor negativo, se utiliza 0.

Valor devuelto

String - Una subcadena de la cadena especificada.

Ejemplo

El ejemplo siguiente muestra cómo utilizar substring():

```
var my_str:String = "Hello world";
var mySubstring:String = my_str.substring(6,11);
trace(mySubstring); // output: world
```

El ejemplo siguiente muestra qué ocurre si se utiliza un parámetro start negativo:

```
var my_str:String = "Hello world";
var mySubstring:String = my_str.substring(-5,5);
trace(mySubstring); // output: Hello
```

Encontrará también un ejemplo en el archivo Strings.fla de la carpeta de ejemplos de ActionScript. La lista siguiente muestra rutas habituales a esta carpeta:

- Windows: *unidad de inicio*\Archivos de programa\Macromedia\Flash 8\Samples and Tutorials\Samples\ActionScript
- Macintosh: *Disco duro de Macintosh*/Applications/Macromedia Flash 8/Samples and Tutorials/Samples/ActionScript

toLowerCase (método String.toLowerCase)

```
public toLowerCase() : String
```

Devuelve una copia del objeto String con todos los caracteres en mayúsculas convertidos en minúsculas. El valor original permanece sin cambios.

Disponibilidad: ActionScript 1.0; Flash Player 5

Valor devuelto

String - Una cadena.

Ejemplo

El ejemplo siguiente crea una cadena con todos los caracteres en mayúsculas y, a continuación, crea una copia de esa cadena mediante `toLowerCase()` para convertir todos los caracteres a minúsculas:

```
var upperCase:String = "LOREM IPSUM DOLOR";
var lowerCase:String = upperCase.toLowerCase();
trace("upperCase: " + upperCase); // output: upperCase: LOREM IPSUM DOLOR
trace("lowerCase: " + lowerCase); // output: lowerCase: lorem ipsum dolor
```

Encontrará también un ejemplo en el archivo Strings.fla de la carpeta de ejemplos de ActionScript. La lista siguiente muestra rutas habituales a esta carpeta:

- Windows: *unidad de inicio*\Archivos de programa\Macromedia\Flash 8\Samples and Tutorials\Samples\ActionScript
- Macintosh: *Disco duro de Macintosh*/Applications/Macromedia Flash 8/Samples and Tutorials/Samples/ActionScript

Véase también

`toUpperCase (método String.toUpperCase)`

toString (método String.toString)

```
public toString() : String
```

Devuelve las propiedades de un objeto como cadenas aunque no lo sean.

Disponibilidad: ActionScript 1.0; Flash Player 5

Valor devuelto

`String` - La cadena.

Ejemplo

El ejemplo siguiente devuelve una cadena en mayúsculas que incluye todas las propiedades de un objeto, sean o no sean cadenas:

```
var employee:Object = new Object();
employee.name = "bob";
employee.salary = 60000;
employee.id = 284759021;
```

```
var employeeData:String = new String();
for (prop in employee)
{
  employeeData += employee[prop].toString().toUpperCase() + " ";
}
trace(employeeData);
```

Si el método `toString()` no estuviera incluido en este código y la línea del bucle `for` utilizara `employee[prop].toUpperCase()`, el resultado sería "`undefined undefined BOB`". Si se incluye el método `toString()` se obtiene el resultado esperado: "`284759021 60000 BOB`".

toUpperCase (método String.toUpperCase)

```
public toUpperCase() : String
```

Devuelve una copia del objeto String con todos los caracteres en minúsculas convertidos en mayúsculas. El valor original permanece sin cambios.

Disponibilidad: ActionScript 1.0; Flash Player 5

Valor devuelto

`String` - Una cadena.

Ejemplo

El ejemplo siguiente crea una cadena con todos los caracteres en mayúsculas y, a continuación, crea una copia de la cadena con `toUpperCase()`:

```
var lowerCase:String = "lorem ipsum dolor";
var upperCase:String = lowerCase.toUpperCase();
trace("lowerCase: " + lowerCase); // output: lowerCase: lorem ipsum dolor
trace("upperCase: " + upperCase); // output: upperCase: LOREM IPSUM DOLOR
```

Encontrará también un ejemplo en el archivo Strings.fla de la carpeta de ejemplos de ActionScript. La lista siguiente muestra rutas habituales a esta carpeta:

- Windows: *unidad de inicio*\Archivos de programa\Macromedia\Flash 8\Samples and Tutorials\Samples\ActionScript
- Macintosh: *Disco duro de Macintosh*/Applications/Macromedia Flash 8/Samples and Tutorials/Samples/ActionScript

Véase también

`toLowerCase (método String.toLowerCase)`

valueOf (método String.valueOf)

```
public valueOf() : String
```

Devuelve la cadena.

Disponibilidad: ActionScript 1.0; Flash Player 5

Valor devuelto

`String` - El valor de la cadena.

Ejemplo

El ejemplo siguiente crea una nueva instancia del objeto String y a continuación muestra que el método `valueOf` devuelve una referencia al valor *primitivo*, en lugar de una instancia del objeto.

```
var str:String = new String("Hello World");
var value:String = str.valueOf();
trace(str instanceof String); // true
trace(value instanceof String); // false
trace(str === value); // false
```

StyleSheet (TextField.StyleSheet)

```
Object
  |
  +-TextField.StyleSheet
```

```
public class StyleSheet
extends Object
```

La clase StyleSheet permite crear un objeto StyleSheet que contenga reglas de formato de texto, como tamaño de fuente, color y otros estilos. Posteriormente podrá aplicar estilos definidos mediante una hoja de estilos a un objeto TextField que contenga texto con formato HTML o XML. Se aplicará formato automáticamente al texto del objeto TextField conforme a los estilos de las etiquetas definidas por el objeto StyleSheet. Puede utilizar estilos de texto para definir nuevas etiquetas de formato, redefinir las etiquetas HTML incorporadas o crear clases de estilos para aplicarlas a determinadas etiquetas HTML.

Para aplicar estilos a un objeto TextField, asigne el objeto StyleSheet a la propiedad `styleSheet` de un objeto TextField.

Flash Player admite un subconjunto de propiedades en la especificación CSS1 original (www.w3.org/TR/REC-CSS1). En la tabla siguiente se muestran las propiedades CSS y los valores admitidos, así como los nombres de propiedad de ActionScript correspondientes. Cada nombre de propiedad de ActionScript se deriva del nombre de propiedad CSS correspondiente; si el nombre contiene un guión, éste se omite y el carácter siguiente va en mayúscula.

Propiedad CSS	Propiedad de ActionScript	Uso y valores admitidos
`color`	`color`	Sólo se admiten valores de color hexadecimales. No se admiten los nombres de los colores (como `blue`). Los colores se escriben en el siguiente formato: `#FF0000`.
`display`	`display`	Los valores admitidos son `inline`, `block` y `none`.
`font-family`	`fontFamily`	Lista de fuentes que se deben utilizar, separadas por comas, en orden descendente de conveniencia. Se puede utilizar cualquier nombre de familia de fuentes. Si especifica un nombre de fuente genérico, se convertirá a una fuente de dispositivo adecuada. Hay disponibles las siguientes conversiones de fuentes: `mono` se convierte en `_typewriter`, `sans-serif` se convierte en `_sans` y `serif` se convierte en `_serif`.
`font-size`	`fontSize`	Sólo se utiliza la parte numérica del valor. Las unidades (px, pt) no se analizan; los píxeles y los puntos son equivalentes.
`font-style`	`fontStyle`	Los valores reconocidos son `normal` y `italic`.
`font-weight`	`fontWeight`	Los valores reconocidos son `normal` y `bold`.

Propiedad CSS	Propiedad de ActionScript	Uso y valores admitidos
`kerning`	`kerning`	Los valores reconocidos son `true` y `false`. El ajuste entre caracteres sólo se admite en las fuentes incorporadas. Algunas fuentes, como Courier New, no admiten el ajuste entre caracteres. La propiedad de ajuste entre caracteres sólo puede utilizarse en archivos SWF creados en Windows; no en archivos SWF creados en Macintosh. Sin embargo, estos archivos SWF pueden reproducirse en versiones no Windows de Flash Player y se aplica el ajuste entre caracteres.
`letter-spacing`	`letterSpacing`	La cantidad de espacio distribuido uniformemente entre caracteres. El valor especifica el número de píxeles que se añaden después de cada carácter durante el avance. Un valor negativo condensa el espacio entre caracteres. Sólo se utiliza la parte numérica del valor. Las unidades (px, pt) no se analizan; los píxeles y los puntos son equivalentes.
`margin-left`	`marginLeft`	Sólo se utiliza la parte numérica del valor. Las unidades (px, pt) no se analizan; los píxeles y los puntos son equivalentes.
`margin-right`	`marginRight`	Sólo se utiliza la parte numérica del valor. Las unidades (px, pt) no se analizan; los píxeles y los puntos son equivalentes.
`text-align`	`textAlign`	Los valores reconocidos son `left`, `center`, `right`, y `justify`.
`text-decoration`	`textDecoration`	Los valores reconocidos son `none` y `underline`.
`text-indent`	`textIndent`	Sólo se utiliza la parte numérica del valor. Las unidades (px, pt) no se analizan; los píxeles y los puntos son equivalentes.

Disponibilidad: ActionScript 1.0; Flash Player 7

Resumen de propiedades

Propiedades heredadas de la clase Object

constructor (propiedad Object.constructor), __proto__ (propiedad Object.__proto__), prototype (propiedad Object.prototype), __resolve (propiedad Object.__resolve)

Resumen de eventos

Evento	Descripción
`onLoad = function(success:Boolean) {}`	Invocado cuando se ha finalizado una operación `load()`.

Resumen de constructores

Firma	Descripción
`StyleSheet()`	Crea un objeto StyleSheet.

Resumen de métodos

Modificadores	Firma	Descripción
	`clear() : Void`	Elimina todos los estilos del objeto StyleSheet especificado.
	`getStyle(name:String) : Object`	Devuelve una copia del objeto de estilo asociado al estilo especificado (`name`).
	`getStyleNames() : Array`	Devuelve una matriz que contiene los nombres (cadenas) de todos los estilos registrados en esta hoja de estilos.
	`load(url:String) : Boolean`	Inicia la carga del archivo CSS en la hoja de estilos.
	`parseCSS(cssText:String) : Boolean`	Analiza la CSS de `cssText` y carga junto a ésta la hoja de estilos.
	`setStyle(name:String, style:Object) : Void`	Añade un nuevo estilo con el nombre especificado en el objeto StyleSheet.
	`transform(style:Object) : TextFormat`	Amplía la capacidad de análisis de la CSS.

addProperty (método Object.addProperty), hasOwnProperty (método Object.hasOwnProperty), isPropertyEnumerable (método Object.isPropertyEnumerable), isPrototypeOf (método Object.isPrototypeOf), registerClass (método Object.registerClass), toString (método Object.toString), unwatch (método Object.unwatch), valueOf (método Object.valueOf), watch (método Object.watch)

clear (método StyleSheet.clear)

```
public clear() : Void
```

Elimina todos los estilos del objeto StyleSheet especificado.

Disponibilidad: ActionScript 1.0; Flash Player 7

Ejemplo

El ejemplo siguiente carga una hoja de estilos denominada styles.css en un archivo SWF y muestra los estilos que se han cargado en el panel Salida. Cuando hace clic en `clear_btn`, se eliminan todos los estilos del objeto `my_styleSheet`.

```
// Create a new StyleSheet object
import TextField.StyleSheet;
var my_styleSheet:StyleSheet = new StyleSheet();

my_styleSheet.onLoad = function(success:Boolean) {
  if (success) {
    trace("Styles loaded.");
    var styles_array:Array = my_styleSheet.getStyleNames();
    for (var i = 0; I < styles_array.length; i++) {
      trace("\t"+styles_array[i]);
    }
    trace("");
  } else {
    trace("Error loading CSS");
  }
};

// Start the loading operation
my_styleSheet.load("styles.css");
```

```
clear_btn.onRelease = function() {
  my_styleSheet.clear();
  trace("Styles cleared.");
  var styles_array:Array = my_styleSheet.getStyleNames();
  for (var i = 0; i<styles_array.length; i++) {
    trace("\t"+styles_array[i]);
  }
  trace("");
};
```

getStyle (método StyleSheet.getStyle)

```
public getStyle(name:String) : Object
```

Devuelve una copia del objeto de estilo asociado al estilo especificado (`name`). Si no hay ningún objeto de estilo asociado a `name`, se devuelve `null`.

Disponibilidad: ActionScript 1.0; Flash Player 7

Parámetros

`name``:String` - El nombre del estilo que se va a recuperar.

Valor devuelto

`Object` - Un objeto de estilo; en caso contrario, `null`.

Ejemplo

El ejemplo siguiente carga estilos de un archivo CSS, analiza la hoja de estilos y muestra los nombres de estilos y los valores de propiedades en el panel Salida. Cree un archivo de ActionScript con el nombre StyleSheetTracer.as e introduzca en él el código siguiente:

```
import TextField.StyleSheet;
class StyleSheetTracer {
  // StyleSheetTracer.displayFromURL
  // This method displays the CSS style sheet at
  // the specified URL in the Output panel.
  static function displayFromURL(url:String):Void {
    // Create a new StyleSheet object
    var my_styleSheet:StyleSheet = new StyleSheet();
    // The load operation is asynchronous, so set up
    // a callback function to display the loaded StyleSheet.
    my_styleSheet.onLoad = function(success:Boolean) {
      if (success) {
        StyleSheetTracer.display(this);
      } else {
        trace("Error loading style sheet "+url);
      }
    };
```

```
    // Start the loading operation.
    my_styleSheet.load(url);
  }
  static function display(my_styleSheet:StyleSheet):Void {
    var styleNames:Array = my_styleSheet.getStyleNames();
    if (!styleNames.length) {
      trace("This is an empty style sheet.");
    } else {
      for (var i = 0; i<styleNames.length; i++) {
        var styleName:String = styleNames[i];
        trace("Style "+styleName+":");
        var styleObject:Object = my_styleSheet.getStyle(styleName);
        for (var propName in styleObject) {
          var propValue = styleObject[propName];
          trace("\t"+propName+": "+propValue);
        }
        trace("");
      }
    }
  }
}
```

Cree un documento CSS denominado styles.css, con dos estilos llamados `.heading` y `.mainBody` que definen las propiedades de `font-family`, `font-size` y `font-weight`. Introduzca el código siguiente en el documento CSS:

```
/~ In styles.css ~/
.heading {
font-family: Arial, Helvetica, sans-serif;
font-size: 24px;
font-weight: bold;
}
.mainBody {
font-family: Arial, Helvetica, sans-serif;
font-size: 12px;
font-weight: normal;
}
```

Por último, en un archivo FLA o ActionScript, especifique el siguiente código ActionScript para cargar la hoja de estilos externa, styles.css:

```
StyleSheetTracer.displayFromURL("styles.css");
```

Este código muestra la siguiente información en el panel Salida:

```
Style .heading:
fontWeight: bold
fontSize: 24px
fontFamily: Arial, Helvetica, sans-serif
```

```
Style .mainBody:
fontWeight: normal
fontSize: 12px
fontFamily: Arial, Helvetica, sans-serif
```

Véase también

getStyleNames (método StyleSheet.getStyleNames)

```
public getStyleNames() : Array
```

Devuelve una matriz que contiene los nombres (cadenas) de todos los estilos registrados en esta hoja de estilos.

Disponibilidad: ActionScript 1.0; Flash Player 7

Valor devuelto

`Array` - Una matriz de nombres de estilos (como cadenas).

Ejemplo

Este ejemplo crea un objeto StyleSheet denominado `styleSheet` que contiene dos estilos, `heading` y `bodyText`. A continuación, invoca el método `getStyleNames()` del objeto StyleSheet, asigna el resultado a la matriz `names_array` y muestra el contenido de la matriz en el panel Salida.

```
import TextField.StyleSheet;
var my_styleSheet:StyleSheet = new StyleSheet();
my_styleSheet.setStyle("heading", {fontsize:'24px'});
my_styleSheet.setStyle("bodyText", {fontsize:'12px'});
var names_array:Array = my_styleSheet.getStyleNames();
trace(names_array.join("\n"));
```

Aparecerá la información siguiente en el panel Salida:

```
bodyText
heading
```

Véase también

`getStyle (método StyleSheet.getStyle)`

load (método StyleSheet.load)

```
public load(url:String) : Boolean
```

Inicia la carga del archivo CSS en la hoja de estilos. La operación de carga es asíncrona; utilice el controlador callback `onLoad()` para determinar cuándo termina de cargarse el archivo. El archivo CSS debe residir en el mismo dominio que el archivo SWF que lo está cargando.

Disponibilidad: ActionScript 1.0; Flash Player 7

Parámetros

`url``:String` - La URL del archivo CSS que se va a cargar. La dirección URL debe estar en el mismo dominio que la URL donde reside actualmente el archivo Flash.

Valor devuelto

`Boolean` - `false` si no se pasa ningún parámetro (`null`); `true`, en caso contrario. Utilice el controlador callback `onLoad()` para comprobar el estado de una hoja de estilos cargada.

Ejemplo

Para ver un ejemplo de hojas de estilo que se cargan de forma asíncrona con ActionScript 2.0, consulte el ejemplo de `getStyle()`.

El ejemplo siguiente carga el archivo CSS denominado styles.css en el objeto StyleSheet `my_styleSheet`. Cuando el archivo se haya cargado correctamente, se aplicará el objeto StyleSheet a un objeto TextField denominado `news_txt`.

```
import TextField.StyleSheet;
this.createTextField("news_txt", 999, 10, 10, 320, 240);
news_txt.multiline = true;
news_txt.wordWrap = true;
news_txt.html = true;

var my_styleSheet:StyleSheet = new StyleSheet();
my_styleSheet.onLoad = function(success:Boolean) {
  if (success) {
  news_txt.styleSheet = my_styleSheet;
  news_txt.htmlText = "<p class=\"heading\">Heading goes here!</p>"
    + "<p class=\"mainBody\">Lorem ipsum dolor sit amet, consectetuer "
    + "adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet
  "
    + "dolore magna aliquam erat volutpat.</p>";
  }
};
my_styleSheet.load("styles.css");
```

Para consultar el código completo de styles.css, véase el ejemplo de `getStyle()`.

Véase también

onLoad (controlador StyleSheet.onLoad), getStyle (método StyleSheet.getStyle)

onLoad (controlador StyleSheet.onLoad)

```
onLoad = function(success:Boolean) {}
```

Invocado cuando se ha finalizado una operación `load()`. Si se ha cargado correctamente la hoja de estilos, el parámetro `success` es `true`. Si no se ha recibido el documento o si se ha producido un error al recibir la respuesta del servidor, el parámetro `success` tiene el valor `false`.

Disponibilidad: ActionScript 1.0; Flash Player 7

Parámetros

`success``:Boolean` - Un valor booleano que indica si el archivo CSS se ha cargado correctamente (`true`) o no (`false`).

Ejemplo

El ejemplo siguiente carga el archivo CSS denominado styles.css en el objeto StyleSheet `my_styleSheet`. Cuando el archivo termina de cargarse correctamente, se aplica el objeto StyleSheet a un objeto TextField denominado `news_txt`.

```
import TextField.StyleSheet;
this.createTextField("news_txt", 999, 10, 10, 320, 240);
news_txt.multiline = true;
news_txt.wordWrap = true;
news_txt.html = true;

var my_styleSheet:StyleSheet = new StyleSheet();
my_styleSheet.onLoad = function(success:Boolean) {
  if (success) {
  news_txt.styleSheet = my_styleSheet;
  news_txt.htmlText = "<p class=\"heading\">Heading goes here!"
    + "</p><p class=\"mainBody\">Lorem ipsum dolor "
    + "sit amet, consectetuer adipiscing elit, sed diam nonummy "
    + "nibh euismod tincidunt ut laoreet dolore magna aliquam "
    + "erat volutpat.</p>";
  }
};
my_styleSheet.load("styles.css");
```

Para consultar el código completo de styles.css, véase el ejemplo de `getStyle()`. Para ver un ejemplo de hojas de estilo que se cargan de forma asíncrona con ActionScript 2.0, consulte el ejemplo de `getStyle()`.

Véase también

load (método StyleSheet.load), getStyle (método StyleSheet.getStyle)

parseCSS (método StyleSheet.parseCSS)

```
public parseCSS(cssText:String) : Boolean
```

Analiza la CSS de `cssText` y carga junto a ésta la hoja de estilos. Si un estilo de `cssText` ya se encuentra en StyleSheet, se conservarán las propiedades de StyleSheet y sólo las de `cssText` se añadirán o modificarán.

Para ampliar la capacidad de análisis de CSS, puede sustituir este método creando una subclase de la clase StyleSheet.

Disponibilidad: ActionScript 1.0; Flash Player 7

Parámetros

`cssText``:String` - El texto CSS que se va a analizar.

Valor devuelto

`Boolean` - Un valor booleano que indica si el texto se ha analizado correctamente (`true`) o no (`false`).

Ejemplo

El ejemplo siguiente analiza la CSS en `css_str` El script muestra información que indica si se ha analizado correctamente y, a continuación, muestra la CSS analizada en el panel Salida.

```
import TextField.StyleSheet;
var css_str:String = ".heading {font-family: Arial, Helvetica, sans-serif;
  font-size: 24px; font-weight: bold; }";
var my_styleSheet:StyleSheet = new StyleSheet();
if (my_styleSheet.parseCSS(css_str)) {
  trace("parsed successfully");
  dumpStyles(my_styleSheet);
} else {
  trace("unable to parse CSS");
}
//
function dumpStyles(styles:StyleSheet):Void {
  var styleNames_array:Array = styles.getStyleNames();
  for (var i = 0; i<styleNames_array.length; i++) {
    var styleName_str:String = styleNames_array[i];
    var styleObject:Object = styles.getStyle(styleName_str);
    trace(styleName_str);
```

```
    for (var prop in styleObject) {
      trace("\t"+prop+": "+styleObject[prop]);
    }
    trace("");
  }
}
```

setStyle (método StyleSheet.setStyle)

```
public setStyle(name:String, style:Object) : Void
```

Añade un nuevo estilo con el nombre especificado en el objeto StyleSheet. Si no hay ningún estilo con dicho nombre en la hoja de estilos, se añadirá. Si ya hay ningún estilo con este nombre en la hoja de estilos, se sustituirá. Si el parámetro `style` tiene el valor `null`, se eliminará el estilo indicado.

Flash Player crea una copia del objeto de estilo que se pasa a este método.

Para ver una lista de los estilos admitidos, consulte la tabla en la descripción de la clase StyleSheet.

Disponibilidad: ActionScript 1.0; Flash Player 7

Parámetros

`name``:String` - El nombre del estilo que se va a añadir a StyleSheet.

`style``:Object` - Un objeto que describe el estilo o `null`.

Ejemplo

El ejemplo siguiente añade un estilo con el nombre `emphasized` a la hoja de estilos `myStyleSheet`. El estilo incluye dos propiedades de estilo: `color` y `fontWeight`. El objeto de estilo se define con el operador `{}`.

```
myStyleSheet.setStyle("emphasized", {color:'#000000',fontWeight:'bold'});
```

También podría crear un objeto de estilo utilizando una instancia de la clase Object y pasar el objeto (`styleObj`) como el parámetro `style`, como muestra el ejemplo siguiente.

```
import TextField.StyleSheet;
var my_styleSheet:StyleSheet = new StyleSheet();

var styleObj:Object = new Object();
styleObj.color = "#000000";
styleObj.fontWeight = "bold";
my_styleSheet.setStyle("emphasized", styleObj);
delete styleObj;

var styleNames_array:Array = my_styleSheet.getStyleNames();
```

```
for (var i=0;i<styleNames_array.length;i++) {
  var styleName:String = styleNames_array[i];
  var thisStyle:Object = my_styleSheet.getStyle(styleName);
  trace(styleName);
  for (var prop in thisStyle) {
    trace("\t"+prop+": "+thisStyle[prop]);
  }
  trace("");
}
```

Aparecerá la información siguiente en el panel Salida:

```
 emphasized
 fontWeight: bold
 color: #000000
```

Nota: Flash Player crea una copia del objeto style pasado a `setStyle()`, por lo que el comando `delete styleObj` en el ejemplo reduce el uso de memoria eliminando el objeto style original transmitido a `setStyle()`.

Véase también

`Operador de inicializador de objeto ({})`, `StyleSheet (TextField.StyleSheet)`

Constructor StyleSheet

```
public StyleSheet()
```

Crea un objeto StyleSheet.

Disponibilidad: ActionScript 1.0; Flash Player 7

Ejemplo

El ejemplo siguiente carga una hoja de estilos y controla los estilos que se cargan en el documento. Añada el siguiente código ActionScript al archivo ActionScript o FLA:

```
import TextField.StyleSheet;
var my_styleSheet:StyleSheet = new StyleSheet();
my_styleSheet.onLoad = function(success:Boolean) {
  if (success) {
    trace("Styles loaded:");
    var styles_array:Array = my_styleSheet.getStyleNames();
    trace(styles_array.join(newline));
  } else {
    trace("Error loading CSS");
  }
};
my_styleSheet.load("styles.css");
```

El archivo styles.css contiene dos estilos denominados `.heading` y `.mainbody`, por lo que aparece la siguiente información en el panel Salida:

```
Styles loaded:
.heading
.mainBody
```

Encontrará el código completo para styles.css en el ejemplo de `getStyle()`.

Véase también

`getStyle (método StyleSheet.getStyle)`

transform (método StyleSheet.transform)

`public transform(style:Object) : TextFormat`

Amplía la capacidad de análisis de la CSS. Los desarrolladores avanzados pueden sustituir este método ampliando la clase StyleSheet.

Disponibilidad: ActionScript 1.0; Flash Player 7

Parámetros

`style``:Object` - Un objeto que describe el estilo, que contiene reglas de estilo como propiedades del objeto o `null`.

Valor devuelto

`TextFormat` - Un objeto TextFormat que contiene el resultado de la asignación de reglas de CSS a las propiedades de formato de texto.

Ejemplo

En el ejemplo siguiente se extiende el método `transform()`:

```
import TextField.StyleSheet;
class AdvancedCSS extends StyleSheet {
  public function AdvancedCSS() {
    trace("AdvancedCSS instantiated");
  }

  public function transform(styleObject):TextFormat {
    trace("tranform called");
  }
}
```

System

```
Object
  |
  +-System
```

```
public class System
extends Object
```

La clase System contiene propiedades relativas a determinadas operaciones que se desarrollan en el equipo del usuario, como las realizadas con objetos compartidos, para la configuración local de cámaras y micrófonos, y cuando se utiliza el Portapapeles. El paquete System incluye las siguientes propiedades y métodos en clases específicas: la clase capabilities, la clase security y la clase IME.

Disponibilidad: ActionScript 1.0; Flash Player 6

Véase también

`capabilities (System.capabilities)`, `security (System.security)`, `IME (System.IME)`

Resumen de propiedades

Modificadores	Propiedad	Descripción
`static`	`exactSettings:Boolean`	Valor booleano que especifica si debe utilizarse un superdominio (`false`) o un dominio exacto (`true`) que coincida con las reglas al acceder a la configuración local (por ejemplo, los permisos de acceso a una cámara o un micrófono) o a datos localmente persistentes (objetos compartidos).
`static`	`useCodepage:Boolean`	Un valor booleano que indica a Flash Player si debe utilizar Unicode o la página de códigos tradicional del sistema operativo en el que se ejecuta el reproductor para interpretar los archivos de texto externos.

Propiedades heredadas de la clase Object

`constructor (propiedad Object.constructor)`, `__proto__ (propiedad Object.__proto__)`, `prototype (propiedad Object.prototype)`, `__resolve (propiedad Object.__resolve)`

Resumen de eventos

Evento	Descripción
`onStatus = function(infoObject:Object) {}`	Controlador de eventos: ofrece un supercontrolador de eventos para determinados objetos.

Resumen de métodos

Modificadores	Firma	Descripción
`static`	`setClipboard(text:String) : Void`	Reemplaza el contenido del portapapeles por una cadena de texto especificada.
`static`	`showSettings([tabID:Number]) : Void`	Muestra el panel Configuración especificado de Flash Player.

Métodos heredados de la clase Object

`addProperty (método Object.addProperty)`, `hasOwnProperty (método Object.hasOwnProperty)`, `isPropertyEnumerable (método Object.isPropertyEnumerable)`, `isPrototypeOf (método Object.isPrototypeOf)`, `registerClass (método Object.registerClass)`, `toString (método Object.toString)`, `unwatch (método Object.unwatch)`, `valueOf (método Object.valueOf)`, `watch (método Object.watch)`

exactSettings (propiedad System.exactSettings)

```
public static exactSettings : Boolean
```

Valor booleano que especifica si debe utilizarse un superdominio (`false`) o un dominio exacto (`true`) que coincida con las reglas al acceder a la configuración local (por ejemplo, los permisos de acceso a una cámara o un micrófono) o a datos localmente persistentes (objetos compartidos). El valor predeterminado es `true` para archivos publicados para Flash Player 7 o versiones posteriores y `false` para archivos publicados para Flash Player 6.

Si el valor es `true`, la configuración y los datos de un archivo SWF albergado en here.xyz.com se almacenarán en un directorio denominado here.xyz.com, la configuración y los datos de un archivo SWF albergado en there.xyz.com se almacenarán en un directorio denominado there.xyz.com, etc. Si el valor es `false`, la configuración y los datos para los archivos SWF albergados en here.xyz.com, there.xyz.com y xyz.com están compartidos y se almacenan todos en un directorio denominado xyz.com.

Si algunos de sus archivos configuran esta propiedad con el valor `false` y otros con el valor `true`, puede que los archivos SWF situados en subdominios diferentes compartan configuración y datos. Por ejemplo, si esta propiedad se configura con el valor `false` en un archivo SWF albergado en here.xyz.com y con el valor `true` en un archivo SWF albergado en xyz.com, ambos archivos utilizarán la misma configuración y los mismos datos, concretamente los ubicados en el directorio xyz.com. Si no es éste el comportamiento deseado, asegúrese de que establece esta propiedad en cada archivo de manera que represente correctamente el lugar en el que desea almacenar la configuración y los datos.

Si desea cambiar el valor predeterminado de esta propiedad, debe hacerlo en el primer fotograma de su documento. Si desea cambiar el valor predeterminado de esta propiedad, debe hacerlo cerca del principio del script. La propiedad no puede cambiarse después de una actividad que necesite acceso a la configuración local, como `System.showSettings()` o `SharedObject.getLocal()`.

Si utiliza `loadMovie()`, `MovieClip.loadMovie()` o `MovieClipLoader.loadClip()` para cargar un archivo SWF en otro, todos los archivos publicados para Flash Player 7 comparten un solo valor para `System.exactSettings`, y todos los archivos publicados para Flash Player 6 comparten un solo valor para `System.exactSettings`. Si utiliza `MovieClip.loadMovie()` o `MovieClipLoader.loadClip()` para cargar un archivo SWF en otro, todos los archivos comparten un solo valor de `System.exactSettings`. Por lo tanto, si especifica un valor para esta propiedad en un archivo publicado para una versión concreta de Player, debe hacer lo mismo en todos los archivos que vaya a cargar. Si carga múltiples archivos, la configuración especificada en el último archivo cargado sobrescribirá a la configuración especificada previamente.

El valor predeterminado de `System.exactSettings` normalmente resultará adecuado. Muchas veces, el único requisito que desea que se cumpla es que cuando un archivo SWF guarde un objeto compartido en una sesión, el mismo archivo SWF pueda recuperar el mismo archivo compartido en una sesión posterior. Esta situación no cambiará sea cual sea el valor de `System.exactSettings`. Aunque es posible que desee cambiar la configuración predeterminada de `System.exactSettings` para que un archivo SWF publicado para Flash Player 7 o posterior pueda recuperar objetos compartidos originalmente creados por un archivo SWF publicado para Flash Player 6. Dado que el reproductor ha almacenado los objetos compartidos creados por el archivo SWF de Flash Player 6 en una carpeta específica para el superdominio de dicho archivo SWF, deberá utilizar reglas de superdominio para la recuperación de objetos compartidos en el archivo SWF de Flash Player 7. Este paso exige que se especifique `System.exactSettings = false` en el archivo SWF de Flash Player 7. También es posible que tenga archivos SWF publicados para Flash Player 6 y archivos de Flash Player 7 que compartan los mismos datos de objetos compartidos. En este caso, simplemente elija un valor para `System.exactSettings` (`true` o `false`) y utilícelo de manera coherente en los archivos SWF de Flash Player 6 y Flash Player 7.

Disponibilidad: ActionScript 1.0; Flash Player 7

Ejemplo

El ejemplo siguiente muestra cómo especificar reglas de coincidencia de superdominio:

Véase también

`loadMovie (método MovieClip.loadMovie)`, `loadClip (método MovieClipLoader.loadClip)`, `getLocal (método SharedObject.getLocal)`, `exactSettings (propiedad System.exactSettings)`

onStatus (controlador System.onStatus)

```
onStatus = function(infoObject:Object) {}
```

Controlador de eventos: ofrece un supercontrolador de eventos para determinados objetos.

Las clases LocalConnection, NetStream y SharedObject ofrecen un controlador de eventos onStatus que utiliza un objeto de información para proporcionar mensajes de información, estado o error. Para responder a este controlador de eventos, deberá crear una función que procese el objeto de información y deberá conocer el formato y el contenido del objeto de información devuelto.

Además de estos métodos onStatuts específicos, Flash también ofrece una superfunción denominada `System.onStatus` que actúa como controlador secundario de mensajes de error. Si una instancia de la clase LocalConnection, NetStream o SharedObject pasa un objeto de información con la propiedad de nivel "error" pero no ha definido una función onStatus para esa instancia concreta, Flash utilizará en su lugar la función que defina para `System.onStatus`.

Nota: Las clases Camera y Microphone también tienen controladores onStatus, pero no pasan objetos de información con la propiedad de nivel "error". Por consiguiente, no se llamará a `System.onStatus` a no ser que especifique una función para estos controladores.

Disponibilidad: ActionScript 1.0; Flash Player 6

Parámetros

`infoObject``:Object` - Un parámetro definido de acuerdo con el mensaje de estado.

Ejemplo

El ejemplo siguiente muestra cómo crear una función System.onStatus para procesar objetos de información cuando no existe una función onStatus específica de clase:

```
// Create generic function
System.onStatus = function(genericError:Object){
  // Your script would do something more meaningful here
  trace("An error has occurred. Please try again.");
}
```

El ejemplo siguiente muestra cómo crear una función onStatus para una instancia de la clase NetStream:

```
// Create function for NetStream object

videoStream_ns.onStatus = function(infoObject:Object) {
  if (infoObject.code == "NetStream.Play.StreamNotFound") {
    trace("Could not find video file.");
  }
}
```

Véase también

`onStatus (controlador Camera.onStatus)`, `onStatus (controlador LocalConnection.onStatus)`, `onStatus (controlador Microphone.onStatus)`, `onStatus (controlador NetStream.onStatus)`, `onStatus (controlador SharedObject.onStatus)`

setClipboard (método System.setClipboard)

```
public static setClipboard(text:String) : Void
```

Reemplaza el contenido del portapapeles por una cadena de texto especificada.

Nota: Por motivos de seguridad, no es posible leer el contenido del portapapeles del sistema. Dicho de otro modo, el correspondiente método `System.getClipboard()` no existe.

Disponibilidad: ActionScript 1.0; Flash Player 7

Parámetros

`text``:String` - Una cadena de texto normal que se sitúa en el Portapapeles del sistema reemplazando el contenido actual (si existe).

Ejemplo

El ejemplo siguiente coloca la frase `"Hello World"` en el Portapapeles del sistema:

```
System.setClipboard("Hello world");
```

El ejemplo siguiente crea dos campos de texto en tiempo de ejecución, `in_txt` y `out_txt`. Cuando selecciona el texto en el campo `in_txt`, puede hacer clic en `copy_btn` para copiar los datos en el Portapapeles. A continuación puede pegar el texto en el campo `out_txt`.

```
this.createTextField("in_txt", this.getNextHighestDepth(), 10, 10, 160,
  120);
in_txt.multiline = true;
in_txt.border = true;
in_txt.text = "lorum ipsum...";
this.createTextField("out_txt", this.getNextHighestDepth(), 10, 140, 160,
  120);
out_txt.multiline = true;
out_txt.border = true;
out_txt.type = "input";

copy_btn.onRelease = function() {
  System.setClipboard(in_txt.text);
  Selection.setFocus("out_txt");
};
```

Si el archivo SWF incluye un componente de la versión 2, utilice la clase DepthManager de los componentes de la versión 2 en lugar del método `MovieClip.getNextHighestDepth()` utilizado en este ejemplo.

showSettings (método System.showSettings)

```
public static showSettings([tabID:Number]) : Void
```

Muestra el panel Configuración especificado de Flash Player. El panel permite al usuario realizar cualquiera de las acciones siguientes:

- Permitir o denegar el acceso a la cámara y el micrófono
- Especificar el espacio del disco local disponible para objetos compartidos
- Seleccionar una cámara y un micrófono predeterminados
- Especificar la configuración de ganancia del micrófono y de supresión de eco

Por ejemplo, si su aplicación requiere la utilización de una cámara, puede indicar al usuario que seleccione Permitir en el panel Parámetros de privacidad y luego enviar un comando `System.showSettings(0)`. (Compruebe que el tamaño del escenario es como mínimo de 215 x 138 píxeles)

Disponibilidad: ActionScript 1.0; Flash Player 6

Parámetros

`tabID``:Number` [opcional] - Un número; un número que especifica el panel Configuración de Flash Player que se va a mostrar, como se describe en la tabla siguiente:

Valor pasado para panel para `panel`	**Panel Configuración mostrado**
Ninguno (se omite el parámetro) o un valor no admitido	El panel que se abrió la última vez que el usuario cerró el panel Configuración de Flash Player.
0	Privacidad
1	Almacenamiento local
2	Micrófono
3	Cámara

Ejemplo

El ejemplo siguiente ilustra cómo mostrar el panel Parámetros de almacenamiento local de Flash Player:

Véase también

`get (método Camera.get)`, `get (propiedad Microphone.get)`, `getLocal (método SharedObject.getLocal)`

useCodepage (propiedad System.useCodepage)

```
public static useCodepage : Boolean
```

Un valor booleano que indica a Flash Player si debe utilizar Unicode o la página de códigos tradicional del sistema operativo en el que se ejecuta el reproductor para interpretar los archivos de texto externos. El valor predeterminado es `System.useCodepage`.

- Cuando la propiedad se establece con el valor false, Flash Player interpreta los archivos de texto externos como Unicode. (Estos archivos deben codificarse como Unicode al guardarlos.)
- Si la propiedad se establece con el valor true, Flash Player interpretará los archivos de texto externos empleando la página de códigos tradicional del sistema operativo en el que se ejecuta el reproductor.

Deberá codificar como Unicode el texto que cargue como archivo externo (mediante las sentencias `loadVariables()` o `getURL()`, o bien la clase LoadVars o XML) al guardar el archivo de texto para que Flash Player lo reconozca como Unicode. Para codificar archivos externos como Unicode, guarde los archivos en una aplicación que admita Unicode, como el Bloc de notas en Windows 2000.

Si carga archivos de texto externos que no están codificados como Unicode, deberá configurar `System.useCodepage` con el valor true. Añada el código siguiente como la primera línea de código del primer fotograma del archivo SWF que carga los datos:

```
System.useCodepage = true;
```

Cuando este código está presente, Flash Player interpreta el texto externo empleando la página de códigos tradicional del sistema operativo en el que se ejecuta Flash Player. Ésta es normalmente CP1252 para el sistema operativo Windows en inglés y Shift-JIS para un sistema operativo en japonés. Si establece `System.useCodepage` con el valor true, Flash Player 6 y versiones posteriores tratarán el texto de la misma forma que Flash Player 5. (Flash Player 5 consideraba todo el texto como si estuviera en la página de códigos tradicional del sistema operativo en el que se ejecuta el reproductor.)

Si establece `System.useCodepage` con el valor true, recuerde que la página de códigos tradicional del sistema operativo en el que se ejecuta el reproductor debe incluir los caracteres utilizados en el archivo de texto externo para que se muestre el texto. Por ejemplo, si carga un archivo de texto externo que contiene caracteres chinos, dichos caracteres no se visualizarán en un sistema que utilice la página de códigos CP1252, ya que dicha página de códigos no contiene caracteres chinos.

Para garantizar que los usuarios de todas las plataformas puedan ver los archivos de texto externos que se utilizan en sus archivos SWF, debe codificar todos los archivos de texto externos como Unicode y establecer `System.useCodepage` en false de forma predeterminada. De esta forma, Flash Player 6 y versiones posteriores interpretará el texto como Unicode.

Disponibilidad: ActionScript 1.0; Flash Player 6

TextField

```
Object
  |
  +-TextField
```

```
public dynamic class TextField
extends Object
```

La clase TextField se utiliza para crear áreas para ver e introducir texto. Todos los campos dinámicos y de entrada de texto de un archivo SWF son instancias de la clase TextField. Puede asignar a un campo de texto un nombre de instancia en el inspector de propiedades y utilizar los métodos y propiedades de la clase TextField para manipularlo con ActionScript. Los nombres de instancias de TextField se muestran en el explorador de películas y en el cuadro de diálogo Insertar ruta de destino del panel Acciones.

Para crear un campo de texto de forma dinámica, no se utiliza un operador `new`. Puede usar `MovieClip.createTextField()` en su lugar.

Los métodos de la clase TextField permiten establecer, seleccionar y manipular texto de un campo de texto dinámico o de entrada que se cree durante la edición o la ejecución.

ActionScript proporciona diversas maneras de dar formato a los textos durante la ejecución. La clase TextFormat permite definir el formato de carácter y de párrafo para los objetos TextField. En Flash Player 7 y versiones posteriores, puede aplicar estilos CSS (Cascading Style Sheets, hojas de estilos en cascada) a los campos de texto mediante la propiedad TextField.styleSheet y la clase StyleSheet. Puede utilizar CSS para aplicar un estilo a las etiquetas HTML incorporadas, definir nuevas etiquetas de formato o aplicar estilos. Puede asignar texto con formato HTML, que opcionalmente puede utilizar estilos CSS, directamente a un campo de texto. En Flash Player 7 y versiones posteriores, el texto HTML que asigna a un campo de texto puede contener elementos multimedia incorporados (clips de película, archivos SWF, JPEG, GIF y PNG). El texto se ajustará alrededor del elemento multimedia incorporado, igual que los navegadores Web ajustan texto alrededor del elemento multimedia incorporado en un documento HTML.

Flash Player también admite un subconjunto de etiquetas HTML que puede utilizar para dar formato al texto.

Disponibilidad: ActionScript 1.0; Flash Player 6

Véase también

`Object`, `createTextField (método MovieClip.createTextField)`

Resumen de propiedades

Modificadores	Propiedad	Descripción
	`_alpha:Number`	Establece o recupera el valor de transparencia alfa del campo de texto.
	`antiAliasType:String`	Tipo de suavizado que se utiliza en esta instancia de TextField.
	`autoSize:Object`	Controla la asignación de tamaño y alineación automáticas de los campos de texto.
	`background:Boolean`	Especifica si el campo de texto tiene relleno de fondo.
	`backgroundColor:Number`	Color del fondo del campo de texto.
	`border:Boolean`	Especifica si el campo de texto tiene un borde.
	`borderColor:Number`	Color del borde del campo de texto.
	`bottomScroll:Number` [read-only]	Un entero (índice basado en uno) que indica la última línea visible en el campo de texto.

Modificadores	Propiedad	Descripción
	`condenseWhite:Boolean`	Valor booleano que especifica si debe eliminarse el espacio en blanco adicional (espacios, saltos de línea, etc.) de un campo de texto HTML cuando el campo se muestra en un navegador.
	`embedFonts:Boolean`	Especifica si representar utilizando contornos de fuentes incorporadas.
	`filters:Array`	Matriz indexada que contiene todos los objetos de filtro actualmente asociados al campo de texto.
	`gridFitType:String`	Tipo de ajuste de cuadrícula que se utiliza en esta instancia de TextField.
	`_height:Number`	Altura del campo de texto, expresada en píxeles.
	`_highquality:Number`	*Desfasada* desde Flash Player 7. Esta propiedad está desfasada y en su lugar debe utilizarse `TextField._quality`. Especifica el nivel de visualización suavizada que se aplica al archivo SWF actual.
	`hscroll:Number`	Indica la posición de desplazamiento horizontal actual.
	`html:Boolean`	Indicador que especifica si el campo de texto contiene una representación HTML.
	`htmlText:String`	Si el campo de texto es HTML, esta propiedad contiene la representación HTML del contenido del campo de texto.
	`length:Number` [read-only]	Indica el número de caracteres de un campo de texto.
	`maxChars:Number`	Indica el número máximo de caracteres que puede contener el campo de texto.
	`maxhscroll:Number` [read-only]	Valor máximo de `TextField.hscroll`.
	`maxscroll:Number` [read-only]	Valor máximo de `TextField.scroll`.
	`menu:ContextMenu`	Asocia el objeto ContextMenu *`contextMenu`* al campo de texto *`my_txt`* .
	`mouseWheelEnabled:Boolean`	Valor booleano que indica si Flash Player debe desplazar automáticamente campos de texto de varias líneas cuando se hace clic con el puntero del ratón en un campo de texto y el usuario hace girar la rueda del ratón.

Modificadores	Propiedad	Descripción
	`multiline:Boolean`	Indica si el campo de texto consta de varias líneas.
	`_name:String`	El nombre de instancia del campo de texto.
	`_parent:MovieClip`	Referencia al clip de película u objeto que contiene el campo de texto u objeto actual.
	`password:Boolean`	Especifica si el campo de texto es de contraseña.
	`_quality:String`	La calidad de representación utilizada para un archivo SWF.
	`restrict:String`	Indica el conjunto de caracteres que los usuarios pueden introducir en el campo de texto.
	`_rotation:Number`	Giro del campo de texto, expresado en grados, con respecto a su orientación original.
	`scroll:Number`	La posición vertical del texto de un campo de texto.
	`selectable:Boolean`	Valor booleano que indica si el campo de texto puede seleccionarse.
	`sharpness:Number`	La nitidez de los bordes de glifo en esta instancia de TextField.
	`_soundbuftime:Number`	El número de segundos que un sonido debe acumularse previamente en el búfer antes de que comience a reproducirse sin interrupción.
	`styleSheet:StyleSheet`	Asocia una hoja de estilos al campo de texto.
	`tabEnabled:Boolean`	Especifica si el campo de texto se incluye en el orden de tabulación automático.
	`tabIndex:Number`	Permite personalizar el orden de tabulación de los objetos de un archivo SWF.
	`_target:String` [read-only]	Ruta de destino de la instancia del campo de texto.
	`text:String`	Indica el texto actual del campo de texto.
	`textColor:Number`	Indica el color del texto de un campo de texto.
	`textHeight:Number`	Indica la altura del texto.
	`textWidth:Number`	Indica la anchura del texto.
	`thickness:Number`	El grosor de los bordes de glifo en esta instancia de TextField.
	`type:String`	Especifica el tipo de campo de texto.

Modificadores	Propiedad	Descripción
	`_url:String` [read-only]	Recupera el URL del archivo SWF que creó el campo de texto.
	`variable:String`	El nombre de la variable a la que está asociado el campo de texto.
	`_visible:Boolean`	Valor booleano que indica si el campo de texto *my_txt* puede seleccionarse.
	`_width:Number`	Anchura del campo de texto, expresada en píxeles.
	`wordWrap:Boolean`	Valor booleano que indica si el campo de texto tiene ajuste de texto.
	`_x:Number`	Entero que establece la coordenada x de un campo de texto como relativa a las coordenadas locales del clip de película principal.
	`_xmouse:Number` [read-only]	Devuelve la coordenada x de la posición del ratón relativa al campo de texto.
	`_xscale:Number`	Determina la escala horizontal del campo de texto aplicada desde el punto de registro del campo de texto, expresada como porcentaje.
	`_y:Number`	Establece la coordenada y de un campo de texto como relativa a las coordenadas locales del clip de película principal.
	`_ymouse:Number` [read-only]	Indica la coordenada y de la posición del ratón relativa al campo de texto.
	`_yscale:Number`	La escala vertical del campo de texto aplicada desde el punto de registro del campo de texto, expresada como porcentaje.

Propiedades heredadas de la clase Object

```
constructor (propiedad Object.constructor), __proto__ (propiedad
Object.__proto__), prototype (propiedad Object.prototype), __resolve
(propiedad Object.__resolve)
```

Resumen de eventos

Evento	Descripción
`onChanged = function(changedField:TextField) {}`	Controlador de eventos/detector; se invoca cuando cambia el contenido de un campo de texto.
`onKillFocus = function(newFocus:Object) {}`	Se invoca cuando un campo de texto deja de estar seleccionado con el teclado.
`onScroller = function(scrolledField:TextField) {}`	Controlador de eventos/detector; se invoca cuando cambia una de las propiedades de desplazamiento de un campo de texto.
`onSetFocus = function(oldFocus:Object) {}`	Se invoca cuando un campo de texto se selecciona con el teclado.

Resumen de métodos

Modificadores	Firma	Descripción
	`addListener(listener:Object) : Boolean`	Registra un objeto para recibir mensajes de notificación de eventos TextField.
	`getDepth() : Number`	Devuelve la profundidad de un campo de texto.
`static`	`getFontList() : Array`	Devuelve los nombres de las fuentes del sistema del reproductor en una matriz.
	`getNewTextFormat() : TextFormat`	Devuelve un objeto TextFormat que contiene una copia del objeto de formato de texto del campo de texto.
	`getTextFormat([beginIndex:Number], [endIndex:Number]) : TextFormat`	Devuelve un objeto TextFormat para un carácter, rango de caracteres u objeto completo TextField.
	`removeListener(listener:Object) : Boolean`	Elimina un objeto detector anteriormente registrado en una instancia de campo de texto con `TextField.addListener()`.
	`removeTextField() : Void`	Elimina el campo de texto.
	`replaceSel(newText:String) : Void`	Reemplaza la selección actual por el contenido del parámetro `newText`.

Modificadores	Firma	Descripción
	`replaceText(beginIndex:Number, endIndex:Number, newText:String) : Void`	Reemplaza el rango de caracteres especificado por los parámetros `beginIndex` y `endIndex` del campo de texto indicado por el contenido del parámetro `newText`.
	`setNewTextFormat(tf:TextFormat) : Void`	Establece el formato de texto nuevo predeterminado de un campo de texto.
	`setTextFormat([beginIndex:Number], [endIndex:Number], textFormat:TextFormat) : Void`	Aplica el formato de texto especificado por el parámetro `textFormat` a todo el texto de un campo de texto o a parte de éste.

Métodos heredados de la clase Object

`addProperty (método Object.addProperty)`, `hasOwnProperty (método Object.hasOwnProperty)`, `isPropertyEnumerable (método Object.isPropertyEnumerable)`, `isPrototypeOf (método Object.isPrototypeOf)`, `registerClass (método Object.registerClass)`, `toString (método Object.toString)`, `unwatch (método Object.unwatch)`, `valueOf (método Object.valueOf)`, `watch (método Object.watch)`

addListener (método TextField.addListener)

`public addListener(listener:Object) : Boolean`

Registra un objeto para recibir mensajes de notificación de eventos TextField. El objeto recibirá notificaciones de eventos siempre que se llame a los controladores de eventos `onChanged` y `onScroller`. Cuando un campo de texto cambia o se desplaza, se invocan los controladores de eventos `TextField.onChanged` y `TextField.onScroller`, seguidos de los controladores de eventos `onChanged` y `onScroller` de los objetos registrados como detectores. Pueden registrarse varios objetos como detectores.

Para eliminar un objeto detector de un campo de texto, llame a `TextField.removeListener()`.

La fuente del evento pasa a los controladores `onScroller` y `onChanged` una referencia a la instancia de campo de texto como parámetro. Puede capturar estos datos colocando un parámetro en el método del controlador de eventos. Por ejemplo, el siguiente código utiliza `txt` como parámetro que se pasa al controlador de eventos `onScroller`. El parámetro se utiliza posteriormente en una sentencia `trace` para enviar el nombre de instancia del campo de texto al panel Salida.

```
my_txt.onScroller = function(textfield_txt:TextField) {
  trace(textfield_txt._name+" scrolled");
};
```

Disponibilidad: ActionScript 1.0; Flash Player 6

Parámetros

`listener``:Object` - Un objeto con un controlador de eventos `onChanged` o `onScroller`.

Valor devuelto

`Boolean` -

Ejemplo

El siguiente ejemplo define un controlador `onChanged` para el campo de texto `my_txt`. A continuación, define un nuevo objeto detector `txtListener` y define un controlador `onChanged` para ese objeto. Este controlador se invocará cuando el campo de texto `my_txt` cambie. La última línea de código llama a `TextField.addListener` para registrar el objeto detector `txtListener` con el campo de texto `my_txt` para que reciba una notificación cuando cambie `my_txt`.

```
this.createTextField("my_txt", this.getNextHighestDepth(), 10, 10, 100,
  22);
my_txt.border = true;
my_txt.type = "input";

my_txt.onChanged = function(textfield_txt:TextField) {
  trace(textfield_txt._name+" changed");
};
var txtListener:Object = new Object();
txtListener.onChanged = function(textfield_txt:TextField) {
  trace(textfield_txt._name+" changed and notified myListener");
};
my_txt.addListener(txtListener);
```

El método `MovieClip.getNextHighestDepth()` utilizado en este ejemplo necesita Flash Player 7 o una versión posterior. Si el archivo SWF incluye un componente de la versión 2, utilice la clase DepthManager de los componentes de la versión 2 en lugar del método `MovieClip.getNextHighestDepth()`.

Véase también

onChanged (controlador TextField.onChanged), onScroller (controlador TextField.onScroller), removeListener (método TextField.removeListener)

_alpha (propiedad TextField._alpha)

```
public _alpha : Number
```

Establece o recupera el valor de transparencia alfa del campo de texto. Los valores válidos son los comprendidos entre 0 (totalmente transparente) y 100 (totalmente opaco). El valor predeterminado es 100. Los valores de transparencia no se admiten para campos de texto que utilizan fuentes de dispositivo. Deberá utilizar fuentes incorporadas para utilizar la propiedad de transparencia `_alpha` con un campo de texto.

Disponibilidad: ActionScript 1.0; Flash Player 6

Ejemplo

El siguiente código define la propiedad `_alpha` de un campo de texto denominado `my_txt` como 20%. Cree un nuevo símbolo de fuente en la biblioteca seleccionando Nueva Fuente en el menú de opciones Biblioteca. A continuación, establezca la vinculación de la fuente a `my font`. Defina la vinculación para un símbolo de fuente a `my font`. Añada el siguiente código ActionScript a un archivo AS o FLA.

```
var my_fmt:TextFormat = new TextFormat();
my_fmt.font = "my font";
// where 'my font' is the linkage name of a font in the Library
this.createTextField("my_txt", this.getNextHighestDepth(), 10, 10, 100,
  22);
my_txt.border = true;
my_txt.embedFonts = true;
my_txt.text = "Hello World";
my_txt.setTextFormat(my_fmt);
my_txt._alpha = 20;
```

El método `MovieClip.getNextHighestDepth()` utilizado en este ejemplo necesita Flash Player 7 o una versión posterior. Si el archivo SWF incluye un componente de la versión 2, utilice la clase DepthManager de los componentes de la versión 2 en lugar del método `MovieClip.getNextHighestDepth()`.

Véase también

_alpha (propiedad Button._alpha), _alpha (propiedad MovieClip._alpha)

antiAliasType (propiedad TextField.antiAliasType)

```
public antiAliasType : String
```

Tipo de suavizado que se utiliza en esta instancia de TextField. El suavizado avanzado sólo está disponible en Flash Player 8 y versiones posteriores. Sólo puede controlar este parámetro de configuración si la fuente está incorporada (con la propiedad `embedFonts` como `true`). Para Flash Player 8, el valor predeterminado es `"advanced"`.

Para definir los valores de esta propiedad, utilice los siguientes valores de cadena:

Valor de la cadena	Descripción
`"normal"`	Aplica el suavizado de texto regular. Equivale al tipo de suavizado que utilizaba Flash Player en la versión 7 y anteriores.
`"advanced"`	Aplica suavizado avanzado, que aumenta la legibilidad del texto. (Esta función está disponible a partir de Flash Player 8.) El suavizado avanzado proporciona máxima calidad de representación para las fuentes de tamaño pequeño. Ofrece mejores resultados con aplicaciones que tienen gran cantidad de texto pequeño. No se recomienda utilizarlo con fuentes de más de 48 puntos.

Disponibilidad: ActionScript 1.0; Flash Player 8

Ejemplo

Este ejemplo crea dos campos de texto y aplica suavizado avanzado sólo al primero. Se asume que tiene una fuente incorporada en la Biblioteca con el identificador de vinculación definido como `"Times-12"`. Para incorporar la fuente, siga estos paso:

- Abra la biblioteca.
- Haga clic en el menú de opciones Biblioteca en el ángulo superior derecho de la Biblioteca.
- Seleccione "Nueva fuente" en la lista desplegable.
- Asigne a la fuente el nombre "Times-12".
- Seleccione "Times New Roman" en la lista desplegable de fuentes.
- Presione el botón "Aceptar".
- Haga clic con el botón derecho en la fuente que ha creado y seleccione "Vinculación..."
- Active la casilla de verificación "Exportar para ActionScript".
- Acepte el identificador predeterminado "Times-12" haciendo clic en el botón "Aceptar".

```
var my_format:TextFormat = new TextFormat();
my_format.font = "Times-12";
```

```
var my_text1:TextField = this.createTextField("my_text1",
  this.getNextHighestDepth(), 10, 10, 300, 30);
my_text1.text = "This text uses advanced anti-aliasing.";
my_text1.antiAliasType = "advanced";
my_text1.border = true;
my_text1.embedFonts = true;
my_text1.setTextFormat(my_format);

var my_text2:TextField = this.createTextField("my_text2",
  this.getNextHighestDepth(), 10, 50, 300, 30);
my_text2.text = "This text uses normal anti-aliasing."
my_text2.antiAliasType = "normal";
my_text2.border = true;
my_text2.embedFonts = true;
my_text2.setTextFormat(my_format);
```

Si el archivo SWF incluye un componente de la versión 2, utilice la clase DepthManager de los componentes de la versión 2 en lugar del método `MovieClip.getNextHighestDepth()` utilizado en este ejemplo.

Véase también

`TextRenderer (flash.text.TextRenderer)`, `gridFitType (propiedad TextField.gridFitType)`, `thickness (propiedad TextField.thickness)`, `sharpness (propiedad TextField.sharpness)`

autoSize (propiedad TextField.autoSize)

`public autoSize : Object`

Controla la asignación de tamaño y alineación automáticas de los campos de texto. Los valores aceptables para autoSize son `"none"` (predeterminado), `"left"`, `"right"` y `"center"`. Cuando define la propiedad `autoSize` , `true` es un sinónimo de `"left"` y `false` es un sinónimo de `"none"`.

Los valores de `autoSize` y `TextField.wordWrap` determinan si un campo de texto se amplía o contrae hacia el lado izquierdo, el derecho o el inferior. El valor predeterminado de estas propiedades es `false`.

Si `autoSize` se establece con el valor `"none"` (predeterminado) o `false`, no se producirá cambio de tamaño.

Si `autoSize` se establece en `"left"` o `true`, se considerará que el texto está justificado a la izquierda, lo que significa que el lado izquierdo del campo de texto permanecerá fijo y el cambio de tamaño de un campo de texto de una sola línea tendrá lugar en el lado derecho. Si el texto incluye un salto de línea (por ejemplo, `"\n" or "\r"`)), también se cambiará el tamaño del lado inferior para dar cabida a la siguiente línea de texto. Si `wordWrap` también se ha configurado con el valor `true`, sólo se cambiará el tamaño del lado inferior del campo de texto, mientras que el lado derecho permanecerá fijo.

Si `autoSize` se establece en `"right"`, se considerará que el texto está justificado a la derecha, lo que significa que el lado derecho del campo de texto permanecerá fijo y el cambio de tamaño de un campo de texto de una sola línea tendrá lugar en el lado izquierdo. Si el texto incluye un salto de línea (por ejemplo, `"\n" or "\r"`)), también se cambiará el tamaño del lado inferior para dar cabida a la siguiente línea de texto. Si `wordWrap` también se ha configurado con el valor `true`, sólo se cambiará el tamaño del lado inferior del campo de texto, mientras que el lado izquierdo permanecerá fijo.

Si `autoSize` se establece en `"center"`, se considerará que el texto está centrado, lo que significa que el cambio de tamaño de un campo de texto de una sola línea se distribuirá por igual entre los lados derecho e izquierdo. Si el texto incluye un salto de línea (por ejemplo, `"\n" or "\r"`)), también se cambiará el tamaño del lado inferior para dar cabida a la siguiente línea de texto. Si `wordWrap` también se ha configurado con el valor `true`, sólo se cambiará el tamaño del lado inferior del campo de texto, mientras que los lados izquierdo y derecho permanecerán fijos.

Disponibilidad: ActionScript 1.0; Flash Player 6

Ejemplo

Puede utilizar el código siguiente y especificar otros valores para `autoSize` para ver cómo cambia el tamaño del campo cuando se modifican estos valores. Si hace clic con el ratón mientras se reproduce el archivo SWF, se reemplazará la cadena `short text"` de cada campo de texto por texto más largo utilizando distintas configuraciones de `autoSize`..

```
this.createTextField("left_txt", 997, 10, 10, 70, 30);
this.createTextField("center_txt", 998, 10, 50, 70, 30);
this.createTextField("right_txt", 999, 10, 100, 70, 30);
this.createTextField("true_txt", 1000, 10, 150, 70, 30);
this.createTextField("false_txt", 1001, 10, 200, 70, 30);

left_txt.text = "short text";
left_txt.border = true;

center_txt.text = "short text";
center_txt.border = true;
```

```
right_txt.text = "short text";
right_txt.border = true;

true_txt.text = "short text";
true_txt.border = true;

false_txt.text = "short text";
false_txt.border = true;

// create a mouse listener object to detect mouse clicks
var myMouseListener:Object = new Object();
// define a function that executes when a user clicks the mouse
myMouseListener.onMouseDown = function() {
  left_txt.autoSize = "left";
  left_txt.text = "This is much longer text";
  center_txt.autoSize = "center";
  center_txt.text = "This is much longer text";
  right_txt.autoSize = "right";
  right_txt.text = "This is much longer text";
  true_txt.autoSize = true;
  true_txt.text = "This is much longer text";
  false_txt.autoSize = false;
  false_txt.text = "This is much longer text";
};
// register the listener object with the Mouse object
Mouse.addListener(myMouseListener);
```

background (propiedad TextField.background)

```
public background : Boolean
```

Especifica si el campo de texto tiene relleno de fondo. Si es `true`, el campo de texto tiene relleno de fondo. Si es `false`, el campo de texto no tiene relleno de fondo.

Disponibilidad: ActionScript 1.0; Flash Player 6

Ejemplo

El ejemplo siguiente crea un campo de texto con un color de fondo que se activa y desactiva cuando se presiona casi cualquier tecla del teclado.

```
this.createTextField("my_txt", this.getNextHighestDepth(), 10, 10, 320,
  240);
my_txt.border = true;
my_txt.text = "Lorum ipsum";
my_txt.backgroundColor = 0xFF0000;

var keyListener:Object = new Object();
keyListener.onKeyDown = function() {
  my_txt.background = !my_txt.background;
```

```
};
Key.addListener(keyListener);
```

El método `MovieClip.getNextHighestDepth()` utilizado en este ejemplo necesita Flash Player 7 o una versión posterior. Si el archivo SWF incluye un componente de la versión 2, utilice la clase DepthManager de los componentes de la versión 2 en lugar del método `MovieClip.getNextHighestDepth()`.

backgroundColor (propiedad TextField.backgroundColor)

```
public backgroundColor : Number
```

Color del fondo del campo de texto. El valor predeterminado es `0xFFFFFF` (blanco). Esta propiedad puede recuperarse o establecerse aunque no haya actualmente ningún fondo, pero el color sólo estará visible si el campo de texto tiene un borde.

Disponibilidad: ActionScript 1.0; Flash Player 6

Ejemplo

Consulte el ejemplo de `TextField.background`.

Véase también

`background (propiedad TextField.background)`

border (propiedad TextField.border)

```
public border : Boolean
```

Especifica si el campo de texto tiene un borde. Si es `true`, especifica si el campo de texto tiene un borde. Si es `false`, el campo de texto no tiene borde.

Disponibilidad: ActionScript 1.0; Flash Player 6

Ejemplo

El ejemplo siguiente crea un campo de texto denominado `my_txt`, define la propiedad border como `true` y muestra texto en el campo.

```
this.createTextField("my_txt", this.getNextHighestDepth(), 10, 10, 320,
  240);
my_txt.border = true;
my_txt.text = "Lorum ipsum";
```

El método `MovieClip.getNextHighestDepth()` utilizado en este ejemplo necesita Flash Player 7 o una versión posterior. Si el archivo SWF incluye un componente de la versión 2, utilice la clase DepthManager de los componentes de la versión 2 en lugar del método `MovieClip.getNextHighestDepth()`.

borderColor (propiedad TextField.borderColor)

```
public borderColor : Number
```

Color del borde del campo de texto. El valor predeterminado es `0x000000` (negro). Esta propiedad puede recuperarse o establecerse aunque no haya actualmente ningún borde.

Disponibilidad: ActionScript 1.0; Flash Player 6

Ejemplo

El ejemplo siguiente crea un campo de texto denominado `my_txt`, define la propiedad border como `true` y muestra texto en el campo.

```
this.createTextField("my_txt", this.getNextHighestDepth(), 10, 10, 320,
  240);
my_txt.border = true;
my_txt.borderColor = 0x00FF00;
my_txt.text = "Lorum ipsum";
```

El método `MovieClip.getNextHighestDepth()` utilizado en este ejemplo necesita Flash Player 7 o una versión posterior. Si el archivo SWF incluye un componente de la versión 2, utilice la clase DepthManager de los componentes de la versión 2 en lugar del método `MovieClip.getNextHighestDepth()`.

Véase también

`border (propiedad TextField.border)`

bottomScroll (propiedad TextField.bottomScroll)

```
public bottomScroll : Number [read-only]
```

Un entero (índice basado en uno) que indica la última línea visible en el campo de texto. Considere el campo de texto como una ventana en un bloque de texto. La propiedad `TextField.scroll` es el índice basado en uno de la primera línea visible en la ventana.

Todo el texto existente entre las líneas `TextField.scroll` y `TextField.bottomScroll` está visible actualmente en el campo de texto.

Disponibilidad: ActionScript 1.0; Flash Player 6

Ejemplo

El ejemplo siguiente crea un campo de texto y lo llena de texto. Debe insertar un botón (con el nombre de instancia "my_btn") y cuando haga clic en él, se rastrean las propiedades `scroll` y `bottomScroll` del campo de texto para el campo `comment_txt`.

```
this.createTextField("comment_txt", this.getNextHighestDepth(), 0, 0, 160,
  120);
comment_txt.html = true;
comment_txt.selectable = true;
comment_txt.multiline = true;
comment_txt.wordWrap = true;
comment_txt.htmlText = "<b>What is hexadecimal?</b><br>"
  + "The hexadecimal color system uses six digits to represent color
  values. "
  + "Each digit has sixteen possible values or characters. The characters
  range"
  + " from 0 to 9 and then A to F. Black is represented by (#000000) and
  white, "
  + "at the opposite end of the color system, is (#FFFFFF).";
my_btn.onRelease = function() {
  trace("scroll: "+comment_txt.scroll);
  trace("bottomScroll: "+comment_txt.bottomScroll);
};
```

El método `MovieClip.getNextHighestDepth()` utilizado en este ejemplo necesita Flash Player 7 o una versión posterior. Si el archivo SWF incluye un componente de la versión 2, utilice la clase DepthManager de los componentes de la versión 2 en lugar del método `MovieClip.getNextHighestDepth()`.

condenseWhite (propiedad TextField.condenseWhite)

```
public condenseWhite : Boolean
```

Valor booleano que especifica si debe eliminarse el espacio en blanco adicional (espacios, saltos de línea, etc.) de un campo de texto HTML cuando el campo se muestra en un navegador. El valor predeterminado es `false`.

Si establece el valor `true`, deberá utilizar comandos HTML estándar, como `<BR>` y `<P>`, para incluir saltos de línea en el campo de texto.

Si el `.html` del campo de texto es `false`, esta propiedad no se tiene en cuenta.

Disponibilidad: ActionScript 1.0; Flash Player 6

Ejemplo

El ejemplo siguiente crea dos campos de texto, denominados `first_txt` y `second_txt`. El espacio blanco se elimina del segundo campo de texto. Añada el siguiente código ActionScript al archivo FLA o AS:

```
var my_str:String = "Hello\tWorld\nHow are you?\t\t\tEnd";

this.createTextField("first_txt", this.getNextHighestDepth(), 10, 10, 160,
  120);
first_txt.html = true;
first_txt.multiline = true;
first_txt.wordWrap = true;
first_txt.condenseWhite = false;
first_txt.border = true;
first_txt.htmlText = my_str;

this.createTextField("second_txt", this.getNextHighestDepth(), 180, 10,
  160, 120);
second_txt.html = true;
second_txt.multiline = true;
second_txt.wordWrap = true;
second_txt.condenseWhite = true;
second_txt.border = true;
second_txt.htmlText = my_str;
```

El método `MovieClip.getNextHighestDepth()` utilizado en este ejemplo necesita Flash Player 7 o una versión posterior. Si el archivo SWF incluye un componente de la versión 2, utilice la clase DepthManager de los componentes de la versión 2 en lugar del método `MovieClip.getNextHighestDepth()`.

Véase también

`html (propiedad TextField.html)`

embedFonts (propiedad TextField.embedFonts)

```
public embedFonts : Boolean
```

Especifica si representar utilizando contornos de fuentes incorporadas. Valor booleano que, cuando es `true`, representa el campo de texto empleando contornos de fuentes incorporadas. Si es `false`, representa el campo de texto empleando fuentes de dispositivo.

Si define `embedFonts` como `true` para un campo de texto, debe especificar una fuente para ese texto mediante la propiedad `font` de un objeto `TextFormat` aplicado al campo de texto. Si la fuente especificada *no* existe en la biblioteca (con el nombre de instancia de vinculación correspondiente), el texto no se mostrará.

Disponibilidad: ActionScript 1.0; Flash Player 6

Ejemplo

En este ejemplo, debe crear un campo de texto dinámico denominado `my_txt` y utilizar el siguiente código ActionScript para incorporar fuentes y girar el campo de texto. La referencia a `my font` lleva a un símbolo Font en la biblioteca, con vinculación a `my font`. El ejemplo asume que tiene un símbolo Font en la biblioteca llamado `my font`, con las propiedades de vinculación siguientes: el identificador definido como `my font`, y las opciones Exportar para ActionScript y Exportar en primer fotograma seleccionadas.

```
var my_fmt:TextFormat = new TextFormat();
my_fmt.font = "my font";

this.createTextField("my_txt", this.getNextHighestDepth(), 10, 10, 160,
  120);
my_txt.wordWrap = true;
my_txt.embedFonts = true;
my_txt.text = "Hello world";
my_txt.setTextFormat(my_fmt);
my_txt._rotation = 45;
```

El método `MovieClip.getNextHighestDepth()` utilizado en este ejemplo necesita Flash Player 7 o una versión posterior. Si el archivo SWF incluye un componente de la versión 2, utilice la clase DepthManager de los componentes de la versión 2 en lugar del método `MovieClip.getNextHighestDepth()`.

filters (propiedad TextField.filters)

```
public filters : Array
```

Matriz indexada que contiene todos los objetos de filtro actualmente asociados al campo de texto. El paquete flash.filters contiene varias clases que definen filtros específicos que puede utilizar.

Los filtros pueden aplicarse en tiempo de diseño en la herramienta del entorno de edición de Flash o en tiempo de ejecución con código ActionScript. Para aplicar un filtro con ActionScript, debe realizar una copia temporal de toda la matriz `TextField.filters`, modificar la matriz temporal y volver a asignar el valor de la matriz temporal a la matriz `TextField.filters`. A la matriz `TextField.filters` no pueden añadírsele directamente objetos filter nuevos. El código siguiente no afecta al clip de película de destino, denominado `myTextField`:

```
myTextField.filters[0].push(myDropShadow);
```

Para añadir un filtro con ActionScript, debe seguir estos pasos (suponiendo que el clip de película de destino se denomina `myTextField`):

- Cree un nuevo objeto filter con la función constructora de la clase de filtro que haya elegido.
- Asigne el valor de la matriz `myTextField.filters` a una matriz temporal, denominada por ejemplo `myFilters`.
- Añada el nuevo objeto filter a la matriz temporal, `myFilters`.
- Asigne el valor de la matriz temporal a la matriz `myTextField.filters`.

Si la matriz `filters` está vacía, no tendrá que utilizar una matriz temporal. En lugar de ello, podrá asignar directamente un literal de matriz que contenga uno o varios objetos filter que haya creado.

Para modificar un objeto filter existente, creado tanto en tiempo de diseño o como en tiempo de ejecución, debe utilizar la técnica de modificar una copia de la matriz `filters`, de la forma siguiente:

- Asigne el valor de la matriz `myTextField.filters` a una matriz temporal, denominada por ejemplo `myFilters`.
- Modifique la propiedad utilizando la matriz temporal, `myFilters`. Por ejemplo, si desea establecer la propiedad `quality` del primer filtro de la matriz, podría utilizar el código siguiente: `myList[0].quality = 1;`
- Asigne el valor de la matriz temporal a la matriz `myTextField.filters`.

Para borrar los filtros de un campo de texto, defina `filters` con una matriz vacía (`[]`).

Si está trabajando con una matriz `filters` que contiene varios filtros y necesita averiguar el tipo de filtro asignado a cada índice de matriz, puede conservar su propia matriz `filters` y utilizar una estructura de datos aparte para averiguar el tipo de filtro asociado a cada índice de la matriz. No hay ninguna forma sencilla de determinar el tipo de filtro asociado a cada índice de la matriz `filters`.

Disponibilidad: ActionScript 1.0; Flash Player 8

Ejemplo

El ejemplo siguiente añade un filtro de sombra a un clip de película denominado `myTextField`:

```
var myDropFilter = new flash.filters.DropShadowFilter();
var myFilters:Array = myTextField.filters;
myFilters.push(myDropFilter);
myTextField.filters = myFilters;
```

El ejemplo siguiente cambia la configuración de `quality` del primer filtro de la matriz a 15 (este ejemplo sólo funciona si el campo de texto `myTextField` tiene asociado como mínimo un objeto filter):

```
var myList:Array = myTextField.filters;
myList[0].quality = 15;
myTextField.filters = myList;
```

Véase también

getDepth (método TextField.getDepth)

```
public getDepth() : Number
```

Devuelve la profundidad de un campo de texto.

Disponibilidad: ActionScript 1.0; Flash Player 6

Valor devuelto

`Number` - Un entero que representa la profundidad del campo de texto.

Ejemplo

El ejemplo siguiente muestra campos de texto que residen a distintas profundidades. Cree un campo de texto dinámico en el escenario y añada el siguiente código ActionScript a su archivo FLA o AS. El código crea dos campos de texto en tiempo de ejecución y devuelve sus profundidades.

```
this.createTextField("first_mc", this.getNextHighestDepth(), 10, 10, 100,
  22);
this.createTextField("second_mc", this.getNextHighestDepth(), 10, 10, 100,
  22);
for (var prop in this) {
  if (this[prop] instanceof TextField) {
  var this_txt:TextField = this[prop];
  trace(this_txt._name+" is a TextField at depth: "+this_txt.getDepth());
  }
}
```

El método `MovieClip.getNextHighestDepth()` utilizado en este ejemplo necesita Flash Player 7 o una versión posterior. Si el archivo SWF incluye un componente de la versión 2, utilice la clase DepthManager de los componentes de la versión 2 en lugar del método `MovieClip.getNextHighestDepth()`.

getFontList (método TextField.getFontList)

```
public static getFontList() : Array
```

Devuelve los nombres de las fuentes del sistema del reproductor en una matriz. (Este método no devuelve los nombres de todas las fuentes existentes en los archivos SWF cargados actualmente.) Los nombres son de tipo cadena `String`. Se trata de un método estático de la clase TextField global. No se puede especificar una instancia de campo de texto cuando se llama a este método.

Disponibilidad: ActionScript 1.0; Flash Player 6

Valor devuelto

`Array` - Una matriz de nombres de fuentes.

Ejemplo

El código siguiente muestra una lista de fuentes que devuelve `getFontList()`:

```
var font_array:Array = TextField.getFontList();
font_array.sort();
trace("You have "+font_array.length+" fonts currently installed");
trace("---------------------------------------");
for (var i = 0; i<font_array.length; i++) {
  trace("Font #"+(i+1)+":\t"+font_array[i]);
}
```

getNewTextFormat (método TextField.getNewTextFormat)

```
public getNewTextFormat() : TextFormat
```

Devuelve un objeto TextFormat que contiene una copia del objeto de formato de texto del campo de texto. El objeto de formato de texto es el formato que recibe el texto insertado, como, por ejemplo, el texto insertado con el método `replaceSel()` o el texto introducido por un usuario. Cuando se invoca `getNewTextFormat()`, el objeto TextFormat devuelto tiene todas sus propiedades definidas. Ninguna de las propiedades tiene el valor `null`.

Disponibilidad: ActionScript 1.0; Flash Player 6

Valor devuelto

`TextFormat` - Un objeto TextFormat.

Ejemplo

El ejemplo siguiente muestra el objeto de formato de texto del campo de texto especificado (`my_txt`t).

```
this.createTextField("my_txt", this.getNextHighestDepth(), 10, 10, 160,
  120);
var my_fmt:TextFormat = my_txt.getNewTextFormat();
trace("TextFormat has the following properties:");
for (var prop in my_fmt) {
  trace(prop+": "+my_fmt[prop]);
}
```

El método `MovieClip.getNextHighestDepth()` utilizado en este ejemplo necesita Flash Player 7 o una versión posterior. Si el archivo SWF incluye un componente de la versión 2, utilice la clase DepthManager de los componentes de la versión 2 en lugar del método `MovieClip.getNextHighestDepth()`.

getTextFormat (método TextField.getTextFormat)

```
public getTextFormat([beginIndex:Number], [endIndex:Number]) : TextFormat
```

Devuelve un objeto TextFormat para un carácter, rango de caracteres u objeto completo TextField.

La tabla siguiente describe tres posibles usos:

Uso	Descripción
`my_textField.getTextFormat()`	Devuelve un objeto TextFormat que contiene información de formato para todo el texto de un campo de texto. En el objeto TextFormat resultante, sólo se establecen las propiedades que son comunes a todo el texto del campo de texto. Todas las propiedades que sean *mixtas*, es decir, que tengan valores diferentes en distintos puntos del texto, tendrán el valor `null`..
`my_textField.getTextFormat(beginIndex:Number)`	Devuelve un objeto TextFormat que contiene una copia del formato del texto del campo de texto en la posición de `beginIndex`.
`my_textField.getTextFormat(beginIndex:Number,endIndex:Number)`	Devuelve un objeto TextFormat que contiene información de formato para el texto comprendido entre `beginIndex` y `endIndex`. En el objeto TextFormat resultante, sólo se establecen las propiedades que son comunes a todo el texto del rango especificado. Todas las propiedades que sean mixtas (es decir, que tengan valores diferentes en distintos puntos del rango), estarán configuradas con el valor `null`.

Disponibilidad: ActionScript 1.0; Flash Player 6

Parámetros

beginIndex:Number [opcional] - Un entero que especifica un carácter en una cadena. Si no especifica beginIndex y endIndex, el objeto TextFormat devuelto afecta al TextField completo.

endIndex:Number [opcional] - Un entero que especifica la posición final de un fragmento de texto. Si especifica beginIndex pero no especifica endIndex, TextFormat afecta al único carácter que especifica beginIndex.

Valor devuelto

TextFormat - El objeto TextFormat que representa las propiedades de formato del texto especificado.

Ejemplo

El siguiente código ActionScript lleva un control de toda la información de formato de un campo de texto creado en tiempo de ejecución.

```
this.createTextField("dyn_txt", this.getNextHighestDepth(), 0, 0, 100,
  200);
dyn_txt.text = "Frank";
dyn_txt.setTextFormat(new TextFormat());
var my_fmt:TextFormat = dyn_txt.getTextFormat();
for (var prop in my_fmt) {
  trace(prop+": "+my_fmt[prop]);
}
```

El método MovieClip.getNextHighestDepth() utilizado en este ejemplo necesita Flash Player 7 o una versión posterior. Si el archivo SWF incluye un componente de la versión 2, utilice la clase DepthManager de los componentes de la versión 2 en lugar del método MovieClip.getNextHighestDepth().

Véase también

getNewTextFormat (método TextField.getNewTextFormat), setNewTextFormat (método TextField.setNewTextFormat), setTextFormat (método TextField.setTextFormat)

gridFitType (propiedad TextField.gridFitType)

```
public gridFitType : String
```

Tipo de ajuste de cuadrícula que se utiliza en esta instancia de TextField. Esta propiedad sólo se aplica si la propiedad `antiAliasType` del campo de texto se define como `"advanced"`.

Para la propiedad `gridFitType`, puede utilizar los siguientes valores de cadena:

Valor de la cadena	Descripción
`"none"`	Sin ajuste de cuadrícula. Las líneas horizontales y verticales de los glifos no se ajustan a la cuadrícula de píxeles. Esta es una buena opción para animación o para tamaños de fuente grandes.
`"pixel"`	Especifica que las líneas horizontales y verticales intensas se ajustan a la cuadrícula de píxeles. Este parámetro sólo funciona para campos de texto alineados a la izquierda. Para usar este parámetro de configuración, la propiedad `antiAliasType` del campo de texto debe estar definida como `"advanced"`. En general, es la opción que más facilita la lectura de texto con alineación a la izquierda.
`"subpixel"`	Especifica que las líneas horizontales y verticales intensas se ajusten a la cuadrícula de subpíxeles en monitores LCD. Para usar este parámetro de configuración, la propiedad `antiAliasType` del campo de texto debe estar definida como `"advanced"`. `"subpixel"` suele ser una buena opción para texto dinámico con alineación central o alineación a la derecha, y en ocasiones ofrece un buen equilibrio entre animación y calidad de texto.

Disponibilidad: ActionScript 1.0; Flash Player 8

Ejemplo

Este ejemplo muestra tres campos de texto que utilizan distintos valores de `gridFitType`. Se asume que tiene una fuente incorporada en la Biblioteca con el identificador de vinculación definido como `"Times-12"`. Para incorporar la fuente, siga estos paso:

- Abra la biblioteca.
- Haga clic en el menú de opciones Biblioteca en el ángulo superior derecho de la Biblioteca.
- Seleccione "Nueva fuente" en la lista desplegable.
- Asigne a la fuente el nombre "Times-12".
- Seleccione "Times New Roman" en la lista desplegable de fuentes.

- Presione el botón "Aceptar".
- Haga clic con el botón derecho en la fuente que ha creado y seleccione "Vinculación..."
- Active la casilla de verificación "Exportar para ActionScript".
- Acepte el identificador predeterminado "Times-12" haciendo clic en el botón "Aceptar".

```
var my_format:TextFormat = new TextFormat();
my_format.font = "Times-12";

var my_text1:TextField = this.createTextField("my_text1",
  this.getNextHighestDepth(), 9.5, 10, 400, 100);
my_text1.text = "this.gridFitType = none";
my_text1.embedFonts = true;
my_text1.antiAliasType = "advanced";
my_text1.gridFitType = "none";
my_text1.setTextFormat(my_format);

var my_text2:TextField = this.createTextField("my_text2",
  this.getNextHighestDepth(), 9.5, 40, 400, 100);
my_text2.text = "this.gridFitType = advanced";
my_text2.embedFonts = true;
my_text2.antiAliasType = "advanced";
my_text2.gridFitType = "pixel";
my_text2.setTextFormat(my_format);

var my_text3:TextField = this.createTextField("my_text3",
  this.getNextHighestDepth(), 9.5, 70, 400, 100);
my_text3.text = "this.gridFitType = subpixel";
my_text3.embedFonts = true;
my_text3.antiAliasType = "advanced";
my_text3.gridFitType = "subpixel";
my_text3.setTextFormat(my_format);
```

Si el archivo SWF incluye un componente de la versión 2, utilice la clase DepthManager de los componentes de la versión 2 en lugar del método `MovieClip.getNextHighestDepth()` utilizado en este ejemplo.

Véase también

`TextRenderer (flash.text.TextRenderer)`, `antiAliasType (propiedad TextField.antiAliasType)`, `sharpness (propiedad TextField.sharpness)`

_height (propiedad TextField._height)

```
public _height : Number
```

Altura del campo de texto, expresada en píxeles.

Disponibilidad: ActionScript 1.0; Flash Player 6

Ejemplo

El siguiente ejemplo de código establece la altura y anchura de un campo de texto:

```
my_txt._width = 200;
my_txt._height = 200;
```

_highquality (propiedad TextField._highquality)

```
public _highquality : Number
```

Desfasada desde Flash Player 7. Esta propiedad está desfasada y en su lugar debe utilizarse `TextField._quality`.

Especifica el nivel de visualización suavizada que se aplica al archivo SWF actual. Especifique 2 (mejor calidad) para aplicar alta calidad con el suavizado de mapa de bits siempre activado. Especifique 1 (alta calidad) para aplicar la visualización suavizada, que suaviza los mapas de bits si el archivo SWF no contiene animación y es el valor predeterminado. Especifique 0 (baja calidad) para evitar el suavizado.

Disponibilidad: ActionScript 1.0; Flash Player 6

Véase también

`_quality (propiedad TextField._quality)`

hscroll (propiedad TextField.hscroll)

```
public hscroll : Number
```

Indica la posición de desplazamiento horizontal actual. Si la propiedad `hscroll` es 0, el texto no se desplazará horizontalmente.

Las unidades de desplazamiento horizontal son píxeles, mientras que las de desplazamiento vertical son líneas. El desplazamiento horizontal se mide en píxeles porque la mayoría de las fuentes que utiliza normalmente tienen espaciado proporcional, lo que significa que los caracteres tienen anchuras diferentes. Flash realiza el desplazamiento vertical por líneas porque el usuario normalmente desea ver una línea de texto íntegramente y no sólo parte de ella. Aunque en una línea se utilicen varias fuentes, la altura de la línea se ajusta a la de la fuente más grande.

Nota: La propiedad hscroll está basada en cero (a diferencia de la propiedad de desplazamiento vertical `TextField.scroll`, que está basada en uno).

Disponibilidad: ActionScript 1.0; Flash Player 6

Ejemplo

El ejemplo siguiente desplaza el campo `my_txt` horizontalmente mediante dos botones llamados `scrollLeft_btn` y `scrollRight_btn`. El desplazamiento se muestra en un campo de texto denominado `scroll_txt`. Añada el siguiente código ActionScript al archivo FLA o AS:

```
this.createTextField("scroll_txt", this.getNextHighestDepth(), 10, 10, 160,
  20);
this.createTextField("my_txt", this.getNextHighestDepth(), 10, 30, 160,
  22);
my_txt.border = true;
my_txt.multiline = false;
my_txt.wordWrap = false;
my_txt.text = "Lorem ipsum dolor sit amet, consectetuer adipiscing...";

scrollLeft_btn.onRelease = function() {
  my_txt.hscroll -= 10;
  scroll_txt.text = my_txt.hscroll+" of "+my_txt.maxhscroll;
};
scrollRight_btn.onRelease = function() {
  my_txt.hscroll += 10;
  scroll_txt.text = my_txt.hscroll+" of "+my_txt.maxhscroll;
};
```

El método `MovieClip.getNextHighestDepth()` utilizado en este ejemplo necesita Flash Player 7 o una versión posterior. Si el archivo SWF incluye un componente de la versión 2, utilice la clase DepthManager de los componentes de la versión 2 en lugar del método `MovieClip.getNextHighestDepth()`.

Véase también

`maxhscroll (propiedad TextField.maxhscroll)`, `scroll (propiedad TextField.scroll)`

html (propiedad TextField.html)

```
public html : Boolean
```

Indicador que especifica si el campo de texto contiene una representación HTML. Si el valor de la propiedad `html` es `true`, el campo de texto es un campo de texto HTML. Si el valor de `html` es `false`, el campo de texto no es un campo de texto HTML.

Disponibilidad: ActionScript 1.0; Flash Player 6

Ejemplo

El ejemplo siguiente crea un campo de texto que establece la propiedad `html` como `true`. En el campo de texto aparece texto con formato HTML.

```
this.createTextField("my_txt", this.getNextHighestDepth(), 10, 10, 160,
  22);
my_txt.html = true;
my_txt.htmlText = "<b> this is bold text </b>";
```

El método `MovieClip.getNextHighestDepth()` utilizado en este ejemplo necesita Flash Player 7 o una versión posterior. Si el archivo SWF incluye un componente de la versión 2, utilice la clase DepthManager de los componentes de la versión 2 en lugar del método `MovieClip.getNextHighestDepth()`.

Véase también

`htmlText (propiedad TextField.htmlText)`

htmlText (propiedad TextField.htmlText)

```
public htmlText : String
```

Si el campo de texto es HTML, esta propiedad contiene la representación HTML del contenido del campo de texto. Si el campo de texto no es un campo de texto HTML, se comportará de forma idéntica a la propiedad `text`. Puede indicar que un campo de texto es un campo de texto HTML en el inspector de propiedades o estableciendo la propiedad `html` del campo de texto con el valor `true`.

Disponibilidad: ActionScript 1.0; Flash Player 6

Ejemplo

El ejemplo siguiente crea un campo de texto que establece la propiedad `html` como `true`. En el campo de texto aparece texto con formato HTML.

```
this.createTextField("my_txt", this.getNextHighestDepth(), 10, 10, 160,
  22);
my_txt.html = true;
my_txt.htmlText = "<b> this is bold text </b>";
```

El método `MovieClip.getNextHighestDepth()` utilizado en este ejemplo necesita Flash Player 7 o una versión posterior. Si el archivo SWF incluye un componente de la versión 2, utilice la clase DepthManager de los componentes de la versión 2 en lugar del método `MovieClip.getNextHighestDepth()`.

Véase también

`html (propiedad TextField.html)`, `Protocolo asfunction`

length (propiedad TextField.length)

```
public length : Number [read-only]
```

Indica el número de caracteres de un campo de texto. Esta propiedad devuelve el mismo valor que `text.length`, aunque es más rápida. El carácter de tabulador (`\t`) cuenta como un carácter.

Disponibilidad: ActionScript 1.0; Flash Player 6

Ejemplo

El ejemplo siguiente devuelve el número de caracteres del campo de texto `date_txt` que muestra la fecha actual.

```
var today:Date = new Date();
this.createTextField("date_txt", this.getNextHighestDepth(), 10, 10, 100,
  22);
date_txt.autoSize = true;
date_txt.text = today.toString();
trace(date_txt.length);
```

El método `MovieClip.getNextHighestDepth()` utilizado en este ejemplo necesita Flash Player 7 o una versión posterior. Si el archivo SWF incluye un componente de la versión 2, utilice la clase DepthManager de los componentes de la versión 2 en lugar del método `MovieClip.getNextHighestDepth()`.

maxChars (propiedad TextField.maxChars)

```
public maxChars : Number
```

Indica el número máximo de caracteres que puede contener el campo de texto. Un script puede insertar más texto que `maxChars`; la propiedad `maxChars` sólo indica cuánto texto puede introducir el usuario. Si el valor de esta propiedad es `null`, no hay límite en cuanto a la cantidad de texto que puede introducirse.

Disponibilidad: ActionScript 1.0; Flash Player 6

Ejemplo

El ejemplo siguiente crea un campo de texto llamado `age_txt` que sólo permite a los usuarios introducir un máximo de dos números en el campo.

```
this.createTextField("age_txt", this.getNextHighestDepth(), 10, 10, 30,
  22);
age_txt.type = "input";
age_txt.border = true;
age_txt.maxChars = 2;
age_txt.restrict = "0-9";
```

El método `MovieClip.getNextHighestDepth()` utilizado en este ejemplo necesita Flash Player 7 o una versión posterior. Si el archivo SWF incluye un componente de la versión 2, utilice la clase DepthManager de los componentes de la versión 2 en lugar del método `MovieClip.getNextHighestDepth()`.

maxhscroll (propiedad TextField.maxhscroll)

```
public maxhscroll : Number [read-only]
```

Valor máximo de `TextField.hscroll`.

Disponibilidad: ActionScript 1.0; Flash Player 6

Ejemplo

Consulte el ejemplo de `TextField.hscroll`.

maxscroll (propiedad TextField.maxscroll)

```
public maxscroll : Number [read-only]
```

Valor máximo de `TextField.scroll`.

Disponibilidad: ActionScript 1.0; Flash Player 6

Ejemplo

El ejemplo siguiente establece el valor máximo del campo de texto de desplazamiento `my_txt`. Cree dos botones, `scrollUp_btn` y `scrollDown_btn`, para desplazar el campo de texto. Añada el siguiente código ActionScript al archivo AS o FLA.

```
this.createTextField("scroll_txt", this.getNextHighestDepth(), 10, 10, 160,
  20);
this.createTextField("my_txt", this.getNextHighestDepth(), 10, 30, 320,
  240);
my_txt.multiline = true;
my_txt.wordWrap = true;
for (var i = 0; i<10; i++) {
  my_txt.text += "Lorem ipsum dolor sit amet, consectetuer adipiscing elit,
  sed diam nonummy nibh "
      + "euismod tincidunt ut laoreet dolore magna aliquam erat volutpat.";
}
scrollUp_btn.onRelease = function() {
  my_txt.scroll--;
  scroll_txt.text = my_txt.scroll+" of "+my_txt.maxscroll;
};
```

```
scrollDown_btn.onRelease = function() {
  my_txt.scroll++;
  scroll_txt.text = my_txt.scroll+" of "+my_txt.maxscroll;
};
```

El método `MovieClip.getNextHighestDepth()` utilizado en este ejemplo necesita Flash Player 7 o una versión posterior. Si el archivo SWF incluye un componente de la versión 2, utilice la clase DepthManager de los componentes de la versión 2 en lugar del método `MovieClip.getNextHighestDepth()`.

menu (propiedad TextField.menu)

```
public menu : ContextMenu
```

Asocia el objeto ContextMenu *`contextMenu`* al campo de texto *`my_txt`* . La clase ContextMenu le permite modificar el menú contextual que se muestra cuando un usuario hace clic con el botón derecho (Windows) o hace clic mientras presiona la tecla Control (Macintosh) en Flash Player.

Esta propiedad sólo funciona con campos de texto seleccionables (editables); no tiene ningún efecto sobre campos de texto no seleccionables.

Disponibilidad: ActionScript 1.0; Flash Player 7

Ejemplo

El ejemplo siguiente asigna el objeto ContextMenu `menu_cm` al campo de texto `news_txt`. El objeto ContextMenu contiene un elemento de menú personalizado "Resize" con un controlador callback asociado denominado `doResize()`, que podría utilizarse para agregar la funcionalidad de cambio de tamaño (que no se muestra):

```
this.createTextField("news_txt", this.getNextHighestDepth(), 10, 10, 320,
  240);
news_txt.border = true;
news_txt.wordWrap = true;
news_txt.multiline = true;
news_txt.text = "To see the custom context menu item, right click (PC) or ";
news_txt.text += "control click (Mac) within the text field.";
var menu_cm:ContextMenu = new ContextMenu();
menu_cm.customItems.push(new ContextMenuItem("Resize", doResize));

function doResize(obj:TextField, item:ContextMenuItem):Void {
  // "Resize" code here
  trace("you selected: "+item.caption);
}
news_txt.menu = menu_cm;
```

Al hacer clic con el botón derecho del ratón o mientras presiona Control en la zona del campo de texto, aparece el elemento de menú personalizado.

Nota: No se puede utilizar un elemento de menú que ya se ha usado en Flash. Por ejemplo, `Print...` (con tres puntos) está reservado para Flash, por lo que no puede utilizarse; sin embargo, puede usar `Print..` (con dos puntos) o cualquier otro elemento de menú que no utilice Flash.

Si el archivo SWF incluye un componente de la versión 2, utilice la clase DepthManager de los componentes de la versión 2 en lugar del método `MovieClip.getNextHighestDepth()` utilizado en este ejemplo.

Véase también

`Button`, `ContextMenu`, `ContextMenuItem`, `MovieClip`

mouseWheelEnabled (propiedad TextField.mouseWheelEnabled)

```
public mouseWheelEnabled : Boolean
```

Valor booleano que indica si Flash Player debe desplazar automáticamente campos de texto de varias líneas cuando se hace clic con el puntero del ratón en un campo de texto y el usuario hace girar la rueda del ratón. De forma predeterminada, este valor es `true`. Esta propiedad resulta útil si desea impedir que la rueda del ratón pueda desplazar campos de texto o si desea implementar su propio desplazamiento de campo de texto.

Disponibilidad: ActionScript 1.0; Flash Player 7

Ejemplo

El ejemplo siguiente crea dos campos de texto. El campo `scrollable_txt` tiene la propiedad `mouseWheelEnabled` como true, por lo que `scrollable_txt` se desplaza cuando hace clic en el campo y gira la rueda del ratón. El campo `nonscrollable_txt` no se desplaza si hace clic en el campo y gira la rueda del ratón.

```
var font_array:Array = TextField.getFontList().sort();

this.createTextField("scrollable_txt", this.getNextHighestDepth(), 10, 10,
  240, 320);
scrollable_txt.border = true;
scrollable_txt.wordWrap = true;
scrollable_txt.multiline = true;
scrollable_txt.text = font_array.join("\n");

this.createTextField("nonscrollable_txt", this.getNextHighestDepth(), 260,
  10, 240, 320);
nonscrollable_txt.border = true;
nonscrollable_txt.wordWrap = true;
nonscrollable_txt.multiline = true;
```

```
nonscrollable_txt.mouseWheelEnabled = false;
nonscrollable_txt.text = font_array.join("\n");
```

`Mouse.onMouseWheel`

Si el archivo SWF incluye un componente de la versión 2, utilice la clase DepthManager de los componentes de la versión 2 en lugar del método `MovieClip.getNextHighestDepth()` utilizado en este ejemplo.

Véase también

`mouseWheelEnabled (propiedad TextField.mouseWheelEnabled)`

multiline (propiedad TextField.multiline)

`public multiline : Boolean`

Indica si el campo de texto consta de varias líneas. Si el valor es `true`, el campo de texto consta de varias líneas; si el valor es `false`, se trata de un campo de texto de una sola línea.

Disponibilidad: ActionScript 1.0; Flash Player 6

Ejemplo

El ejemplo siguiente crea un campo de texto de varias líneas denominado `fontList_txt` que muestra una larga lista de varias líneas de fuentes.

```
var font_array:Array = TextField.getFontList().sort();

this.createTextField("fontList_txt", this.getNextHighestDepth(), 10, 10,
  240, 320);
fontList_txt.border = true;
fontList_txt.wordWrap = true;
fontList_txt.multiline = true;
fontList_txt.text = font_array.join("\n");
```

El método `MovieClip.getNextHighestDepth()` utilizado en este ejemplo necesita Flash Player 7 o una versión posterior. Si el archivo SWF incluye un componente de la versión 2, utilice la clase DepthManager de los componentes de la versión 2 en lugar del método `MovieClip.getNextHighestDepth()`.

_name (propiedad TextField._name)

`public _name : String`

El nombre de instancia del campo de texto.

Disponibilidad: ActionScript 1.0; Flash Player 6

Ejemplo

El ejemplo siguiente muestra campos de texto que residen a distintas profundidades. Cree un campo de texto dinámico en el escenario. Añada el siguiente código ActionScript al archivo FLA o AS, que crea dinámicamente dos campos de texto en tiempo de ejecución y muestra sus profundidades en el panel Salida.

```
this.createTextField("first_mc", this.getNextHighestDepth(), 10, 10, 100,
  22);
this.createTextField("second_mc", this.getNextHighestDepth(), 10, 10, 100,
  22);
for (var prop in this) {
  if (this[prop] instanceof TextField) {
  var this_txt:TextField = this[prop];
  trace(this_txt._name+" is a TextField at depth: "+this_txt.getDepth());
  }
}
```

Cuando pruebe el documento, aparecen el nombre de instancia y la profundidad en el panel Salida.

El método `MovieClip.getNextHighestDepth()` utilizado en este ejemplo necesita Flash Player 7 o una versión posterior. Si el archivo SWF incluye un componente de la versión 2, utilice la clase DepthManager de los componentes de la versión 2 en lugar del método `MovieClip.getNextHighestDepth()`.

onChanged (controlador TextField.onChanged)

```
onChanged = function(changedField:TextField) {}
```

Controlador de eventos/detector; se invoca cuando cambia el contenido de un campo de texto. De manera predeterminada, tiene el valor undefined (no definido); puede definirlo en un script.

Una referencia a la instancia de campo de texto se pasa como parámetro al controlador `onChanged`. Puede capturar estos datos colocando un parámetro en el método del controlador de eventos. Por ejemplo, el siguiente código utiliza `textfield_txt` como parámetro que se pasa al controlador de eventos `onChanged`. El parámetro se utiliza posteriormente en una sentencia `trace()` para enviar el nombre de instancia del campo de texto al panel Salida:

```
this.createTextField("myInputText_txt", 99, 10, 10, 300, 20);
myInputText_txt.border = true;
myInputText_txt.type = "input";

myInputText_txt.onChanged = function(textfield_txt:TextField) {
trace("the value of "+textfield_txt._name+" was changed. New value is:
  "+textfield_txt.text);
};
```

El controlador `onChanged` sólo se activa cuando el cambio se debe a la interacción del usuario; por ejemplo, cuando el usuario está escribiendo algo en el teclado, cambiando algo en el campo de texto con el ratón o seleccionando un elemento de un menú. Los cambios programáticos en el campo de texto no activan el evento `onChanged`, ya que el código reconoce los cambios realizados en el campo de texto.

Disponibilidad: ActionScript 1.0; Flash Player 6

Parámetros

`changedField``:TextField` - El campo que activa el evento.

Véase también

`TextFormat`, `setNewTextFormat (método TextField.setNewTextFormat)`

onKillFocus (controlador TextField.onKillFocus)

```
onKillFocus = function(newFocus:Object) {}
```

Se invoca cuando un campo de texto deja de estar seleccionado con el teclado. El método `onKillFocus` recibe un parámetro, `newFocus`, que es un objeto que representa al nuevo objeto seleccionado. Si no hay ningún objeto seleccionado con el teclado, `newFocus` contendrá el valor `null` (nulo).

Disponibilidad: ActionScript 1.0; Flash Player 6

Parámetros

`newFocus``:Object` - El objeto que recibe la selección.

Ejemplo

El ejemplo siguiente crea dos campos de texto, denominados `first_txt` y `second_txt`. Cuando se selecciona un campo de texto, aparece en el panel Salida la información acerca del campo de texto seleccionado y el que ya no está seleccionado.

```
this.createTextField("first_txt", 1, 10, 10, 300, 20);
first_txt.border = true;
first_txt.type = "input";
this.createTextField("second_txt", 2, 10, 40, 300, 20);
second_txt.border = true;
second_txt.type = "input";
first_txt.onKillFocus = function(newFocus:Object) {
  trace(this._name+" lost focus. New focus changed to: "+newFocus._name);
};
```

```
first_txt.onSetFocus = function(oldFocus:Object) {
  trace(this._name+" gained focus. Old focus changed from:
  "+oldFocus._name);
}
```

Véase también

onSetFocus (controlador TextField.onSetFocus)

onScroller (controlador TextField.onScroller)

```
onScroller = function(scrolledField:TextField) {}
```

Controlador de eventos/detector; se invoca cuando cambia una de las propiedades de desplazamiento de un campo de texto.

Una referencia a la instancia de campo de texto se pasa como parámetro al controlador `onScroller`. Puede capturar estos datos colocando un parámetro en el método del controlador de eventos. Por ejemplo, el siguiente código utiliza `my_txt` como parámetro que se pasa al controlador de eventos `onScroller`. El parámetro se utiliza posteriormente en una sentencia `trace()` para enviar el nombre de instancia del campo de texto al panel Salida.

```
myTextField.onScroller = function (my_txt:TextField) {
  trace (my_txt._name + " scrolled");
};
```

El controlador de eventos `TextField.onScroller` se utiliza normalmente para implementar barras de desplazamiento. Las barras de desplazamiento suelen tener un control deslizante u otro indicador que muestra la posición actual de desplazamiento horizontal o vertical en un campo de texto. Es posible navegar por los campos de texto utilizando el ratón y el teclado, lo que provoca que cambie la posición de desplazamiento. Es preciso notificar al código de la barra de desplazamiento si la posición de desplazamiento cambia como resultado de dicha interacción del usuario, para lo cual se utiliza `TextField.onScroller`.

La llamada a `onScroller` tiene lugar tanto si la posición de desplazamiento cambia como resultado de la interacción del usuario como si lo hace por cambios programáticos. El controlador `onChanged` sólo se activa si el cambio se debe a la interacción del usuario. Las dos opciones son necesarias, ya que es frecuente que un fragmento de código cambie la posición de desplazamiento y el código de la barra de desplazamiento no esté relacionado y desconozca que la posición de desplazamiento ha cambiado porque no se le ha notificado.

Disponibilidad: ActionScript 1.0; Flash Player 6

Parámetros

`scrolledField``:TextField` - Una referencia al objeto TextField cuya posición de desplazamiento ha cambiado.

Ejemplo

El ejemplo siguiente crea un campo de texto con el nombre `my_txt` y utiliza dos botones llamados `scrollUp_btn` y `scrollDown_btn` para desplazar el contenido del campo de texto. Cuando se llama al controlador de eventos `onScroller`, se emplea una sentencia trace para mostrar información en el panel Salida. Cree dos botones con los nombres de instancia `scrollUp_btn` y `scrollDown_btn`, y añada el siguiente código ActionScript al archivo FLA o AS:

```
this.createTextField("scroll_txt", this.getNextHighestDepth(), 10, 10, 160,
  20);
this.createTextField("my_txt", this.getNextHighestDepth(), 10, 30, 320,
  240);
my_txt.multiline = true;
my_txt.wordWrap = true;

for (var i = 0; i<10; i++) {
  my_txt.text += "Lorem ipsum dolor sit amet, consectetuer adipiscing elit,
  sed diam "
    + "nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat
  volutpat.";
}
scrollUp_btn.onRelease = function() {
  my_txt.scroll--;
};
scrollDown_btn.onRelease = function() {
  my_txt.scroll++;
};
my_txt.onScroller = function() {
    trace("onScroller called");
  scroll_txt.text = my_txt.scroll+" of "+my_txt.maxscroll;
};
```

El método `MovieClip.getNextHighestDepth()` utilizado en este ejemplo necesita Flash Player 7 o una versión posterior. Si el archivo SWF incluye un componente de la versión 2, utilice la clase DepthManager de los componentes de la versión 2 en lugar del método `MovieClip.getNextHighestDepth()`.

Véase también

`hscroll (propiedad TextField.hscroll)`, `maxhscroll (propiedad TextField.maxhscroll)`, `maxscroll (propiedad TextField.maxscroll)`, `scroll (propiedad TextField.scroll)`

onSetFocus (controlador TextField.onSetFocus)

```
onSetFocus = function(oldFocus:Object) {}
```

Se invoca cuando un campo de texto se selecciona con el teclado. El parámetro `oldFocus` es el objeto que deja de estar seleccionado. Por ejemplo, si el usuario presiona la tecla Tabulador parar desplazar la selección del teclado de un botón a un campo de texto, oldFocus contendrá la instancia del botón. Si anteriormente no había ningún objeto seleccionado con el teclado, `oldFocus` contendrá un valor null (nulo).

Disponibilidad: ActionScript 1.0; Flash Player 6

Parámetros

`oldFocus``:Object` - El objeto que deja de estar seleccionado.

Ejemplo

Consulte el ejemplo de `TextField.onKillFocus`.

Véase también

`onKillFocus (controlador TextField.onKillFocus)`

_parent (propiedad TextField._parent)

```
public _parent : MovieClip
```

Referencia al clip de película u objeto que contiene el campo de texto u objeto actual. El objeto actual es el que contiene el código ActionScript que hace referencia a `_parent`.

Utilice `_parent` para especificar una ruta de acceso relativa a los clips de película u objetos que se encuentran por encima del campo de texto actual. Puede utilizar `_parent` para subir varios niveles en la lista de visualización, como se muestra a continuación:

```
_parent._parent._alpha = 20;
```

Disponibilidad: ActionScript 1.0; Flash Player 6

Ejemplo

El siguiente código ActionScript crea dos campos de texto y devuelve información acerca del `_parent` de cada objeto. El primer campo de texto, `first_txt`, se crea en la línea de tiempo principal. El segundo, `second_txt`, se crea dentro del clip de película `holder_mc`.

```
this.createTextField("first_txt", this.getNextHighestDepth(), 10, 10, 160,
  22);
first_txt.border = true;
trace(first_txt._name+"'s _parent is: "+first_txt._parent);
```

```
this.createEmptyMovieClip("holder_mc", this.getNextHighestDepth());
holder_mc.createTextField("second_txt", holder_mc.getNextHighestDepth(),
  10, 40, 160, 22);
holder_mc.second_txt.border = true;
trace(holder_mc.second_txt._name+"'s _parent is:
  "+holder_mc.second_txt._parent);
```

El panel Salida muestra la siguiente información:

```
first_txt's _parent is: _level0
second_txt's _parent is: _level0.holder_mc
```

El método `MovieClip.getNextHighestDepth()` utilizado en este ejemplo necesita Flash Player 7 o una versión posterior. Si el archivo SWF incluye un componente de la versión 2, utilice la clase DepthManager de los componentes de la versión 2 en lugar del método `MovieClip.getNextHighestDepth()`.

Véase también

`_parent (propiedad Button._parent)`, `_parent (propiedad MovieClip._parent)`, ,

password (propiedad TextField.password)

```
public password : Boolean
```

Especifica si el campo de texto es de contraseña. Si el valor de password es `true`, el campo de texto es de contraseña y oculta la entrada de caracteres empleando asteriscos en lugar de los caracteres reales. Si su valor es `false`, el campo de texto no es de contraseña. Cuando está activado el modo de contraseña, los comandos *Cortar* y *Copiar* y sus correspondientes atajos de teclado no funcionan. Este mecanismo de seguridad impide que un usuario sin escrúpulos pueda utilizar los métodos abreviados para descubrir una contraseña en un equipo en el que no haya nadie presente.

Disponibilidad: ActionScript 1.0; Flash Player 6

Ejemplo

El ejemplo siguiente crea dos campos de texto: `username_txt` y `password_txt`. Se introduce texto en los dos campos de texto; sin embargo, `password_txt` tiene la propiedad password definida como `true`. Por lo tanto, los caracteres se muestran como asteriscos en el campo `password_txt`.

```
this.createTextField("username_txt", this.getNextHighestDepth(), 10, 10,
  100, 22);
username_txt.border = true;
username_txt.type = "input";
username_txt.maxChars = 16;
username_txt.text = "hello";
```

```
this.createTextField("password_txt", this.getNextHighestDepth(), 10, 40,
  100, 22);
password_txt.border = true;
password_txt.type = "input";
password_txt.maxChars = 16;
password_txt.password = true;
password_txt.text = "world";
```

El método `MovieClip.getNextHighestDepth()` utilizado en este ejemplo necesita Flash Player 7 o una versión posterior. Si el archivo SWF incluye un componente de la versión 2, utilice la clase DepthManager de los componentes de la versión 2 en lugar del método `MovieClip.getNextHighestDepth()`.

_quality (propiedad TextField._quality)

```
public _quality : String
```

La calidad de representación utilizada para un archivo SWF. Las fuentes de dispositivo siempre se muestran dentadas y, por consiguiente, no se ven afectadas por la propiedad `_quality`.

Nota: Aunque puede especificar esta propiedad para un objeto TextField, se trata en realidad de una propiedad global, por lo que puede especificar su valor simplemente como `_quality`. Para más información, consulte la propiedad global `_quality`.

La propiedad `_quality` puede configurarse con los siguientes valores:

- `"LOW"` Calidad de representación baja. No se suavizan ni los gráficos ni los mapas de bits.
- `"MEDIUM"` Calidad de representación media. Los gráficos se suavizan empleando una cuadrícula de 2 x 2 píxeles, pero los mapas de bits no se suavizan. Esta configuración resulta adecuada para películas que no contengan texto.
- `"HIGH"` Calidad de representación alta. Los gráficos se suavizan empleando una cuadrícula de 4 x 4 píxeles, mientras que los mapas de bits se suavizan si la película es estática. Esta es la calidad de representación predeterminada de Flash.
- `"BEST"` Calidad de representación muy alta. Los gráficos se suavizan empleando una cuadrícula de 4 x 4 píxeles y los mapas de bits se suavizan siempre.

Disponibilidad: ActionScript 1.0; Flash Player 6

Ejemplo

El ejemplo siguiente establece la calidad de representación como `LOW`:

```
my_txt._quality = "LOW";
```

Véase también

`Propiedad _quality`

removeListener (método TextField.removeListener)

`public removeListener(listener:Object) : Boolean`

Elimina un objeto detector anteriormente registrado en una instancia de campo de texto con `TextField.addListener()`.

Disponibilidad: ActionScript 1.0; Flash Player 6

Parámetros

`listener``:Object` - El objeto dejará de recibir notificaciones de `TextField.onChanged` o `TextField.onScroller`.

Valor devuelto

`Boolean` - Si `listener` se ha eliminado correctamente, el método devolverá el valor `true` . Si el `listener` no se ha eliminado correctamente (por ejemplo, porque `listener` no se encontraba en la lista de detectores del objeto TextField), el método devolverá el valor `false`.

Ejemplo

El ejemplo siguiente crea un campo de introducción de texto llamado `my_txt`. Cuando el usuario escribe en el campo, aparece información sobre el número de caracteres del campo de texto en el panel Salida. Si el usuario hace clic en la instancia `removeListener_btn`, se elimina el detector y deja de mostrarse la información.

```
this.createTextField("my_txt", this.getNextHighestDepth(), 10, 10, 160,
  20);
my_txt.border = true;
my_txt.type = "input";

var txtListener:Object = new Object();
txtListener.onChanged = function(textfield_txt:TextField) {
  trace(textfield_txt+" changed. Current length is:
  "+textfield_txt.length);
};
my_txt.addListener(txtListener);

removeListener_btn.onRelease = function() {
  trace("Removing listener...");
  if (!my_txt.removeListener(txtListener)) {
  trace("Error! Unable to remove listener");
  }
};
```

El método `MovieClip.getNextHighestDepth()` utilizado en este ejemplo necesita Flash Player 7 o una versión posterior. Si el archivo SWF incluye un componente de la versión 2, utilice la clase DepthManager de los componentes de la versión 2 en lugar del método `MovieClip.getNextHighestDepth()`.

removeTextField (método TextField.removeTextField)

```
public removeTextField() : Void
```

Elimina el campo de texto. Esta operación sólo puede realizarse en un campo de texto creado con `MovieClip.createTextField()`. Al llamar a este método, se elimina el campo de texto. Este método es similar a `MovieClip.removeMovieClip()`.

Disponibilidad: ActionScript 1.0; Flash Player 6

Ejemplo

El ejemplo siguiente crea un campo de texto que puede eliminarse del escenario al hacer clic en la instancia remove_btn. Cree un botón y llámelo `remove_btn` y, a continuación, añada el siguiente código ActionScript al archivo FLA o AS.

```
this.createTextField("my_txt", this.getNextHighestDepth(), 10, 10, 300,
  22);
my_txt.text = new Date().toString();
my_txt.border = true;

remove_btn.onRelease = function() {
  my_txt.removeTextField();
};
```

El método `MovieClip.getNextHighestDepth()` utilizado en este ejemplo necesita Flash Player 7 o una versión posterior. Si el archivo SWF incluye un componente de la versión 2, utilice la clase DepthManager de los componentes de la versión 2 en lugar del método `MovieClip.getNextHighestDepth()`.

replaceSel (método TextField.replaceSel)

```
public replaceSel(newText:String) : Void
```

Reemplaza la selección actual por el contenido del parámetro `newText`. El texto se inserta en la posición de la selección actual empleando el formato de carácter predeterminado y el formato de párrafo predeterminado actuales. El texto no se considera como HTML aunque se trate de un campo de texto HTML.

Puede utilizar el método `replaceSel()` para insertar y eliminar texto sin modificar el formato de carácter y de párrafo del resto del texto.

Debe utilizar `Selection.setFocus()` para seleccionar el campo con el teclado antes de enviar este comando.

Disponibilidad: ActionScript 1.0; Flash Player 6

Parámetros

`newText``:String` - Una cadena.

Ejemplo

El ejemplo siguiente crea en el escenario un campo de texto de varias líneas con texto. Cuando selecciona texto y hace clic con el botón derecho del ratón o con la tecla Control presionada, puede seleccionar `Enter current date` en el menú contextual. Esta opción llama a una función que reemplaza el texto seleccionado por la fecha actual.

```
this.createTextField("my_txt", this.getNextHighestDepth(), 10, 10, 320,
  240);
my_txt.border = true;
my_txt.wordWrap = true;
my_txt.multiline = true;
my_txt.type = "input";
my_txt.text = "Select some sample text from the text field and then right-
  click/control click "
    + "and select 'Enter current date' from the context menu to replace the
  "
    + "currently selected text with the current date.";

var my_cm:ContextMenu = new ContextMenu();
my_cm.customItems.push(new ContextMenuItem("Enter current date",
  enterDate));
function enterDate(obj:Object, menuItem:ContextMenuItem) {
  var today_str:String = new Date().toString();
  var date_str:String = today_str.split(" ", 3).join(" ");
  my_txt.replaceSel(date_str);
}
my_txt.menu = my_cm;
```

El método `MovieClip.getNextHighestDepth()` utilizado en este ejemplo necesita Flash Player 7 o una versión posterior. Si el archivo SWF incluye un componente de la versión 2, utilice la clase DepthManager de los componentes de la versión 2 en lugar del método `MovieClip.getNextHighestDepth()`.

Véase también

`setFocus (método Selection.setFocus)`

replaceText (método TextField.replaceText)

```
public replaceText(beginIndex:Number, endIndex:Number, newText:String) :
  Void
```

Reemplaza el rango de caracteres especificado por los parámetros `beginIndex` y `endIndex` del campo de texto indicado por el contenido del parámetro `newText`.

Disponibilidad: ActionScript 1.0; Flash Player 7

Parámetros

`beginIndex``:Number` - El valor de índice inicial para el rango de sustitución.

`endIndex``:Number` - El valor de índice final para el rango de sustitución.

`newText``:String` - El texto que va a reemplazar al rango de caracteres especificado.

Ejemplo

El ejemplo siguiente crea un campo de texto `my_txt` y le asigna una cadena de texto `dog@house.net`. Se utiliza el método `indexOf()` para encontrar la primera vez que aparece el símbolo especificado (@). Si se encuentra el símbolo, el texto especificado (entre el índice de 0 y el símbolo) se reemplaza con la cadena `bird`. Si no se encuentra el símbolo, aparece un mensaje de error en el panel Salida.

```
this.createTextField("my_txt", this.getNextHighestDepth(), 10, 10, 320,
  22);
my_txt.autoSize = true;
my_txt.text = "dog@house.net";

var symbol:String = "@";
var symbolPos:Number = my_txt.text.indexOf(symbol);
if (symbolPos>-1) {
  my_txt.replaceText(0, symbolPos, "bird");
} else {
  trace("symbol '"+symbol+"' not found.");
}
```

Si el archivo SWF incluye un componente de la versión 2, utilice la clase DepthManager de los componentes de la versión 2 en lugar del método `MovieClip.getNextHighestDepth()` utilizado en este ejemplo.

restrict (propiedad TextField.restrict)

```
public restrict : String
```

Indica el conjunto de caracteres que los usuarios pueden introducir en el campo de texto. Si el valor de la propiedad `restrict` es `null`, puede introducir cualquier carácter. Si el valor de la propiedad `restrict` es una cadena vacía, no se puede introducir ningún carácter. Si el valor de la propiedad `restrict` es una cadena de caracteres, puede introducir caracteres solamente en la cadena del campo de texto. La cadena se examina de izquierda a derecha. Puede especificarse un rango utilizando un guión (-). Esto sólo limita la interacción del usuario; un script puede introducir cualquier texto en el campo de texto. Esta propiedad no se sincroniza con las casillas de verificación Incorporar contornos de fuentes del inspector de propiedades.

Si la cadena empieza por ^, inicialmente se aceptan todos los caracteres; los caracteres posteriores de la cadena se excluyen del conjunto de caracteres aceptados. Si la cadena no empieza por ^, inicialmente no se acepta ningún carácter; los caracteres posteriores de la cadena se incluyen en el conjunto de caracteres aceptados.

Disponibilidad: ActionScript 1.0; Flash Player 6

Ejemplo

El ejemplo siguiente sólo permite que se introduzcan caracteres en mayúsculas, espacios y números en un campo de texto:

```
my_txt.restrict = "A-Z 0-9";
```

El ejemplo siguiente incluye todos los caracteres, con excepción de las letras en mayúsculas:

```
my_txt.restrict = "^a-z";
```

Puede utilizar una barra invertida para introducir un ^ o - literalmente. Las secuencias con barras invertidas aceptadas son \-, \^ o \\. La barra invertida debe ser un carácter real de la cadena, por lo que cuando se especifica en ActionScript, se debe utilizar una barra doble. Por ejemplo, el código siguiente incluye sólo el guión (-) y el símbolo de intercalación (^):

```
my_txt.restrict = "\\-\\^";
```

El carácter ^ puede utilizarse en cualquier lugar de la cadena para incluir o excluir caracteres. El código siguiente incluye sólo letras en mayúsculas, pero excluye la letra Q en mayúscula:

```
my_txt.restrict = "A-Z^Q";
```

Puede usar la secuencia de escape `\u` para construir cadenas `restrict`. El código siguiente incluye solamente los caracteres desde el ASCII 32 (espacio) al ASCII 126 (tilde).

```
my_txt.restrict = "\u0020-\u007E";
```

_rotation (propiedad TextField._rotation)

```
public _rotation : Number
```

Giro del campo de texto, expresado en grados, con respecto a su orientación original. Los valores comprendidos entre 0 y 180 representan un giro en el sentido de las agujas del reloj, mientras que los comprendidos entre 0 y -180 representan un giro en sentido contrario al de las agujas del reloj. Los valores situados fuera de este rango se suman o restan de 360 para obtener un valor que sí esté comprendido en el rango. Por ejemplo, la sentencia `my_txt._rotation = 450` es igual que `my_txt._rotation = 90`.

Los valores de giro no se admiten para campos de texto que utilizan fuentes de dispositivo. Debe utilizar fuentes incorporadas para utilizar `_rotation` con un campo de texto.

Disponibilidad: ActionScript 1.0; Flash Player 6

Ejemplo

En este ejemplo, debe crear un campo de texto dinámico denominado `my_txt` y utilizar el siguiente código ActionScript para incorporar fuentes y girar el campo de texto. La referencia a `my font` lleva a un símbolo Font en la biblioteca, con vinculación a `my font`.

```
var my_fmt:TextFormat = new TextFormat();
my_fmt.font = "my font";

this.createTextField("my_txt", this.getNextHighestDepth(), 10, 10, 160,
  120);
my_txt.wordWrap = true;
my_txt.embedFonts = true;
my_txt.text = "Hello world";
my_txt.setTextFormat(my_fmt);
my_txt._rotation = 45;
```

Aplique más formato al campo de texto utilizando la clase `TextFormat class`.

El método `MovieClip.getNextHighestDepth()` utilizado en este ejemplo necesita Flash Player 7 o una versión posterior. Si el archivo SWF incluye un componente de la versión 2, utilice la clase DepthManager de los componentes de la versión 2 en lugar del método `MovieClip.getNextHighestDepth()`.

Véase también

`_rotation (propiedad Button._rotation)`, `_rotation (propiedad MovieClip._rotation)`, `TextFormat`

scroll (propiedad TextField.scroll)

```
public scroll : Number
```

La posición vertical del texto de un campo de texto. La propiedad scroll es útil para dirigir a los usuarios a un párrafo específico en un pasaje largo, o para crear campos de texto con desplazamiento. Esta propiedad puede recuperarse y modificarse.

Las unidades de desplazamiento horizontal son píxeles, y las de desplazamiento vertical son líneas. El desplazamiento horizontal se mide en píxeles porque la mayoría de las fuentes que utiliza normalmente tienen espaciado proporcional, lo que significa que los caracteres tienen anchuras diferentes. Flash realiza el desplazamiento vertical por líneas porque el usuario normalmente desea ver una línea de texto íntegramente y no sólo parte de ella. Aunque en una línea se utilicen varias fuentes, la altura de la línea se ajusta a la de la fuente más grande.

Disponibilidad: ActionScript 1.0; Flash Player 6

Ejemplo

El ejemplo siguiente establece el valor máximo del campo de texto de desplazamiento `my_txt`. Cree dos botones, `scrollUp_btn` y `scrollDown_btn` para desplazar el campo de texto. Añada el siguiente código ActionScript a un archivo AS o FLA.

```
this.createTextField("scroll_txt", this.getNextHighestDepth(), 10, 10, 160,
  20);
this.createTextField("my_txt", this.getNextHighestDepth(), 10, 30, 320,
  240);
my_txt.multiline = true;
my_txt.wordWrap = true;
for (var i = 0; i<10; i++) {
  my_txt.text += "Lorem ipsum dolor sit amet, consectetuer adipiscing elit,
  sed diam nonummy "
    + "nibh euismod tincidunt ut laoreet dolore magna aliquam erat
  volutpat.";
}
scrollUp_btn.onRelease = function() {
  my_txt.scroll--;
  scroll_txt.text = my_txt.scroll+" of "+my_txt.maxscroll;
};
scrollDown_btn.onRelease = function() {
  my_txt.scroll++;
  scroll_txt.text = my_txt.scroll+" of "+my_txt.maxscroll;
};
```

El método `MovieClip.getNextHighestDepth()` utilizado en este ejemplo necesita Flash Player 7 o una versión posterior. Si el archivo SWF incluye un componente de la versión 2, utilice la clase DepthManager de los componentes de la versión 2 en lugar del método `MovieClip.getNextHighestDepth()`.

Véase también

hscroll (propiedad TextField.hscroll), maxscroll (propiedad TextField.maxscroll)

selectable (propiedad TextField.selectable)

```
public selectable : Boolean
```

Valor booleano que indica si el campo de texto puede seleccionarse. El valor `true` indica que el texto es seleccionable. La propiedad `selectable` controla si un campo de texto es seleccionable y no si un campo de texto es editable. Un campo dinámico puede seleccionarse aunque no sea editable. Si un campo de texto dinámico no es seleccionable, ello significa que no puede seleccionar su texto.

Si selectable se establece con el valor `false`, el texto del campo de texto no responderá a los comandos de selección del ratón o el teclado y el texto no se podrá copiar utilizando el comando Copiar. Si selectable se establece con el valor `true`, el texto del campo de texto podrá seleccionarse utilizando el ratón o el teclado. Puede seleccionar texto de esta forma aunque el campo de texto sea dinámico en lugar de un campo de entrada de texto. El texto puede copiarse utilizando el comando Copiar.

Disponibilidad: ActionScript 1.0; Flash Player 6

Ejemplo

El ejemplo siguiente crea un campo de texto seleccionable que actualiza constantemente la fecha y la hora actual.

```
this.createTextField("date_txt", this.getNextHighestDepth(), 10, 10, 100,
  22);
date_txt.autoSize = true;
date_txt.selectable = true;

var date_interval:Number = setInterval(updateTime, 500, date_txt);
function updateTime(my_txt:TextField) {
  my_txt.text = new Date().toString();
}
```

El método `MovieClip.getNextHighestDepth()` utilizado en este ejemplo necesita Flash Player 7 o una versión posterior. Si el archivo SWF incluye un componente de la versión 2, utilice la clase DepthManager de los componentes de la versión 2 en lugar del método `MovieClip.getNextHighestDepth()`.

setNewTextFormat (método TextField.setNewTextFormat)

```
public setNewTextFormat(tf:TextFormat) : Void
```

Establece el formato de texto nuevo predeterminado de un campo de texto. El nuevo formato de texto predeterminado es el formato que recibe el texto insertado, como, por ejemplo, el texto insertado con el método `replaceSel()` o el texto introducido por un usuario. Cuando se inserta texto, se asigna el formato predeterminado de nuevo texto al nuevo texto insertado.

El formato de texto predeterminado se especifica mediante `textFormat`, que es un objeto TextFormat.

Disponibilidad: ActionScript 1.0; Flash Player 6

Parámetros

`tf``:TextFormat` - Un objeto TextFormat.

Ejemplo

En el ejemplo siguiente, se crea un nuevo campo de texto (denominado `my_txt`) en tiempo de ejecución y se definen varias propiedades. Se aplica el formato del texto que se ha insertado.

```
var my_fmt:TextFormat = new TextFormat();
my_fmt.bold = true;
my_fmt.font = "Arial";
my_fmt.color = 0xFF9900;

this.createTextField("my_txt", 999, 0, 0, 400, 300);
my_txt.wordWrap = true;
my_txt.multiline = true;
my_txt.border = true;
my_txt.type = "input";
my_txt.setNewTextFormat(my_fmt);
my_txt.text = "Oranges are a good source of vitamin C";
```

Véase también

`getNewTextFormat (método TextField.getNewTextFormat)`, `getTextFormat (método TextField.getTextFormat)`, `setTextFormat (método TextField.setTextFormat)`

setTextFormat (método TextField.setTextFormat)

```
public setTextFormat([beginIndex:Number], [endIndex:Number],
  textFormat:TextFormat) : Void
```

Aplica el formato de texto especificado por el parámetro `textFormat` a una parte o a todo el texto de un campo de texto. `textFormat` debe ser un objeto TextFormat que especifique los cambios de formato de texto deseados. Sólo se aplican al campo de texto las propiedades de textFormat que no tengan valor null. No se aplicará ninguna propiedad de `textFormat` configurada con el valor null. De manera predeterminada, todas las propiedades del objeto TextFormat de nueva creación están configuradas con el valor `null`.

Existen dos tipos de información de formato en un objeto TextFormat: formato de nivel de carácter y de nivel de párrafo. Cada carácter de un campo de texto puede tener su propia configuración de formato de carácter, como nombre de fuente, tamaño de fuente, negrita y cursiva.

En el caso de los párrafos, se examina el primer carácter del párrafo en busca de la configuración de formato de párrafo para el párrafo completo. Margen izquierdo, margen derecho y sangría son ejemplos de configuración de formato de párrafo.

El método `setTextFormat()` cambia el formato de texto aplicado a un carácter individual, a un rango de caracteres o a todo el cuerpo del texto de un campo de texto. Estas sintaxis figuran en la tabla siguiente:

Sintaxis	Descripción
`my_textField.setTextFormat(textFormat:TextFormat)`	Aplica las propiedades de `textFormat` a todo el texto del campo de texto.
`my_textField.setTextFormat(beginIndex:Number, textFormat:TextFormat)`	Aplica las propiedades de `textFormat` al carácter situado en la posición `beginIndex`.
`my_textField.setTextFormat(beginIndex:Number, endIndex:Number, textFormat:TextFormat)`	Aplica al fragmento de texto las propiedades del parámetro `textFormat` desde la posición `beginIndex` hasta la posición `endIndex`.

Tenga en cuenta que el texto insertado manualmente por el usuario o sustituido mediante `TextField.replaceSel()`, recibe el formato predeterminado de campo de texto para texto nuevo, no el formato especificado para el punto de inserción del texto. Para definir el formato de texto predeterminado de un campo de texto, utilice `TextField.setNewTextFormat()`.

Disponibilidad: ActionScript 1.0; Flash Player 6

Parámetros

beginIndex:Number [opcional] - Un entero que especifica el primer carácter de un fragmento de texto. Si no especifica beginIndex y endIndex, el objeto TextFormat devuelto se aplica al TextField completo.

endIndex:Number [opcional] - Un entero que especifica el primer carácter después de un fragmento de texto. Si especifica beginIndex pero no especifica endIndex, TextFormat se aplica al único carácter que especifica beginIndex.

textFormat:TextFormat - Un objeto TextFormat que contiene información de formato de carácter y de párrafo.

Ejemplo

El ejemplo siguiente define el formato de texto de dos cadenas de texto diferentes. Se llama al método setTextFormat y se aplica al campo de texto my_txt.

```
var format1_fmt:TextFormat = new TextFormat();
format1_fmt.font = "Arial";
var format2_fmt:TextFormat = new TextFormat();
format2_fmt.font = "Courier";

var string1:String = "Sample string number one."+newline;
var string2:String = "Sample string number two."+newline;

this.createTextField("my_txt", this.getNextHighestDepth(), 0, 0, 300, 200);
my_txt.multiline = true;
my_txt.wordWrap = true;
my_txt.text = string1;
var firstIndex:Number = my_txt.length;
my_txt.text += string2;
var secondIndex:Number = my_txt.length;

my_txt.setTextFormat(0, firstIndex, format1_fmt);
my_txt.setTextFormat(firstIndex, secondIndex, format2_fmt);
```

El método MovieClip.getNextHighestDepth() utilizado en este ejemplo necesita Flash Player 7 o una versión posterior. Si el archivo SWF incluye un componente de la versión 2, utilice la clase DepthManager de los componentes de la versión 2 en lugar del método MovieClip.getNextHighestDepth().

Véase también

TextFormat, setNewTextFormat (método TextField.setNewTextFormat)

sharpness (propiedad TextField.sharpness)

```
public sharpness : Number
```

La nitidez de los bordes de glifo en esta instancia de TextField. Esta propiedad sólo se aplica si la propiedad `antiAliasType` del campo de texto se define como `"advanced"`. El rango para `sharpness` es un número del -400 al 400. Si intenta definir `sharpness` con un valor fuera del rango, Flash selecciona el valor más cercano en el rango (-400 o 400) para la propiedad.

Disponibilidad: ActionScript 1.0; Flash Player 8

Ejemplo

Este ejemplo crea tres campos de texto con `sharpness` como `400`, `0` y `-400`. Se asume que tiene una fuente incorporada en la Biblioteca con el identificador de vinculación definido como `"Times-12"`. Para incorporar la fuente, siga estos pasos:

- Abra la biblioteca.
- Haga clic en el menú de opciones Biblioteca en el ángulo superior derecho de la Biblioteca.
- Seleccione "Nueva fuente" en la lista desplegable.
- Asigne a la fuente el nombre "Times-12".
- Seleccione "Times New Roman" en la lista desplegable de fuentes.
- Presione el botón "Aceptar".
- Haga clic con el botón derecho en la fuente que ha creado y seleccione "Vinculación..."
- Active la casilla de verificación "Exportar para ActionScript".
- Acepte el identificador predeterminado "Times-12" haciendo clic en el botón "Aceptar".

```
var my_format:TextFormat = new TextFormat();
my_format.font = "Times-12";

var my_text1:TextField = this.createTextField("my_text1",
  this.getNextHighestDepth(), 10, 10, 400, 100);
my_text1.text = "This text has sharpness set to 400."
my_text1.embedFonts = true;
my_text1.antiAliasType = "advanced";
my_text1.gridFitType = "pixel";
my_text1.sharpness = 400;
my_text1.setTextFormat(my_format);

var my_text2:TextField = this.createTextField("my_text2",
  this.getNextHighestDepth(), 10, 40, 400, 100);
my_text2.text = "This text has sharpness set to 0."
my_text2.embedFonts = true;
my_text2.antiAliasType = "advanced";
my_text2.gridFitType = "pixel";
```

```
my_text2.sharpness = 0;
my_text2.setTextFormat(my_format);

var my_text3:TextField = this.createTextField("my_text3",
  this.getNextHighestDepth(), 10, 70, 400, 100);
my_text3.text = "This text has sharpness set to -400."
my_text3.embedFonts = true;
my_text3.antiAliasType = "advanced";
my_text3.gridFitType = "pixel";
my_text3.sharpness = -400;
my_text3.setTextFormat(my_format);
```

Si el archivo SWF incluye un componente de la versión 2, utilice la clase DepthManager de los componentes de la versión 2 en lugar del método `MovieClip.getNextHighestDepth()` utilizado en este ejemplo.

Véase también

`gridFitType (propiedad TextField.gridFitType)`, `antiAliasType (propiedad TextField.antiAliasType)`

_soundbuftime (propiedad TextField._soundbuftime)

```
public _soundbuftime : Number
```

El número de segundos que un sonido debe acumularse previamente en el búfer antes de que comience a reproducirse sin interrupción.

Nota: Aunque puede especificar esta propiedad para un objeto TextField, se trata en realidad de una propiedad global que se aplica a todos los sonidos cargados, por lo que puede especificar su valor simplemente como `_soundbuftime`. Al establecer esta propiedad para un objeto TextField, en realidad se define la propiedad global.

Para más información y ver un ejemplo, consulte la propiedad `_soundbuftime` global.

Disponibilidad: ActionScript 1.0; Flash Player 6

Véase también

`Propiedad _soundbuftime`

styleSheet (propiedad TextField.styleSheet)

```
public styleSheet : StyleSheet
```

Asocia una hoja de estilos al campo de texto. Para más información sobre la creación de hojas de estilos, consulte la entrada de la clase TextField.StyleSheet.

La hoja de estilos asociada a un campo de texto puede cambiarse en cualquier momento. Si se cambia la hoja de estilos en uso, el campo de texto vuelve a dibujarse utilizando la nueva hoja de estilos. Para eliminar la hoja de estilos, puede definirse en `null` o `undefined`. Si se elimina la hoja de estilos en uso, el campo de texto vuelve a dibujarse sin ninguna hoja de estilos. El formato aplicado por una hoja de estilos no se conserva si se elimina la hoja de estilos.

Disponibilidad: ActionScript 1.0; Flash Player 7

Ejemplo

El ejemplo siguiente crea un nuevo campo de texto en tiempo de ejecución denominado `news_txt`. En el escenario se utilizan tres botones, `css1_btn`, `css2_btn` y `clearCss_btn`, para cambiar la hoja de estilos que se aplica a `news_txt` o eliminar la hoja de estilos del campo de texto. Añada el siguiente código ActionScript al archivo FLA o AS:

```
this.createTextField("news_txt", this.getNextHighestDepth(), 0, 0, 300,
  200);
news_txt.wordWrap = true;
news_txt.multiline = true;
news_txt.html = true;
var newsText:String = "<p class='headline'>Description</p> Method; "
  + "starts loading the CSS file into styleSheet. The load operation is
  asynchronous; "
  + "use the <span class='bold'>TextField.StyleSheet.onLoad</span> "
  + "callback handler to determine when the file has finished loading. "
  + "<span class='important'>The CSS file must reside in exactly the same "
  + "domain as the SWF file that is loading it.</span> For more information
  about "
  + "restrictions on loading data across domains, see Flash Player security
  features.";

news_txt.htmlText = newsText;

css1_btn.onRelease = function() {
  var styleObj:TextField.StyleSheet = new TextField.StyleSheet();
  styleObj.onLoad = function(success:Boolean) {
  if (success) {
    news_txt.styleSheet = styleObj;
    news_txt.htmlText = newsText;
  }
  };
  styleObj.load("styles.css");
};

css2_btn.onRelease = function() {
  var styleObj:TextField.StyleSheet = new TextField.StyleSheet();
  styleObj.onLoad = function(success:Boolean) {
  if (success) {
    news_txt.styleSheet = styleObj;
```

```
    news_txt.htmlText = newsText;
  }
  };
  styleObj.load("styles2.css");
};

clearCss_btn.onRelease = function() {
  news_txt.styleSheet = undefined;
  news_txt.htmlText = newsText;
};
```

Se aplican los siguientes estilos al campo de texto. Guarde los dos archivos CSS siguientes en el mismo directorio que el archivo FLA o AS creado anteriormente:

```
// in styles.css
.important {
  color: #FF0000;
}
.bold {
  font-weight: bold;
}
.headline {
  color: #000000;
  font-family: Arial,Helvetica,sans-serif;
  font-size: 18px;
  font-weight: bold;
  display: block;
}

// in styles2.css
.important {
  color: #FF00FF;
}
.bold {
  font-weight: bold;
}
.headline {
  color: #00FF00;
  font-family: Arial,Helvetica,sans-serif;
  font-size: 18px;
  font-weight: bold;
  display: block;
}
```

Si el archivo SWF incluye un componente de la versión 2, utilice la clase DepthManager de los componentes de la versión 2 en lugar del método `MovieClip.getNextHighestDepth()` utilizado en este ejemplo.

Véase también

`StyleSheet (TextField.StyleSheet)`

tabEnabled (propiedad TextField.tabEnabled)

```
public tabEnabled : Boolean
```

Especifica si el campo de texto se incluye en el orden de tabulación automático. De manera predeterminada, tiene el valor `undefined`.

Si la propiedad `tabEnabled` es `undefined` o `true`, el objeto se incluirá en el orden de tabulación automático. Si la propiedad `tabIndex` también está configurada con un valor, el objeto se incluirá también en el orden de tabulación personalizado. Si `tabEnabled` tiene el valor `false`, el objeto no se incluirá en el orden de tabulación automático ni en el personalizado aunque se establezca la propiedad `tabIndex`.

Disponibilidad: ActionScript 1.0; Flash Player 6

Ejemplo

El ejemplo siguiente crea varios campos de texto, denominados `one_txt` y `two_txt`, `three_txt` y `four_txt`. El campo `three_txt` tiene la propiedad `tabEnabled` definida como `false`, por lo que se excluye del orden de tabulación automático.

```
this.createTextField("one_txt", this.getNextHighestDepth(), 10, 10, 100,
  22);
one_txt.border = true;
one_txt.type = "input";
this.createTextField("two_txt", this.getNextHighestDepth(), 10, 40, 100,
  22);
two_txt.border = true;
two_txt.type = "input";
this.createTextField("three_txt", this.getNextHighestDepth(), 10, 70, 100,
  22);
three_txt.border = true;
three_txt.type = "input";
this.createTextField("four_txt", this.getNextHighestDepth(), 10, 100, 100,
  22);
four_txt.border = true;
four_txt.type = "input";

three_txt.tabEnabled = false;
three_txt.text = "tabEnabled = false;";
```

El método `MovieClip.getNextHighestDepth()` utilizado en este ejemplo necesita Flash Player 7 o una versión posterior. Si el archivo SWF incluye un componente de la versión 2, utilice la clase DepthManager de los componentes de la versión 2 en lugar del método `MovieClip.getNextHighestDepth()`.

Véase también

tabEnabled (propiedad Button.tabEnabled), tabEnabled (propiedad MovieClip.tabEnabled)

tabIndex (propiedad TextField.tabIndex)

```
public tabIndex : Number
```

Permite personalizar el orden de tabulación de los objetos de un archivo SWF. Puede establecer la propiedad `tabIndex` de un botón, un clip de película o una instancia de campo de texto; la configuración predeterminada es `undefined` (no definido).

Si algún objeto mostrado actualmente en el archivo SWF contiene una propiedad `tabIndex`, se desactivará el orden de tabulación automático y el orden de tabulación se calculará a partir de las propiedades `tabIndex` de los objetos del archivo SWF. El orden de tabulación personalizado sólo incluye objetos que tienen propiedades `tabIndex`.

La propiedad `tabIndex` debe ser un entero positivo. Los objetos se ordenan conforme a lo que indiquen las propiedades `tabIndex` y en orden ascendente. Un objeto que tenga el valor `1` para `tabIndex` precederá a otro objeto que tenga el valor 2 para `tabIndex`. Si dos objetos tienen el mismo valor de `tabIndex`, su clasificación en el orden de tabulación no estará definida (`undefined`).

El orden de tabulación personalizado definido por la propiedad `tabIndex` es *flat* (plano o no jerárquico). Esto significa que no se tienen en cuenta las relaciones jerárquicas entre los objetos del archivo SWF. Todos los objetos del archivo SWF con propiedades `tabIndex` tendrán su lugar en el orden de tabulación, que se determinará por el orden de los valores de `tabIndex`. Si dos objetos tienen el mismo valor de `tabIndex`, el que va en primer lugar no estará definido (`undefined`). No debe utilizar el mismo valor de `tabIndex` para varios objetos.

Disponibilidad: ActionScript 1.0; Flash Player 6

Ejemplo

El siguiente código ActionScript crea de forma dinámica cuatro campos de texto y les asigna un orden de tabulación personalizado. Añada el siguiente código ActionScript al archivo FLA o AS:

```
this.createTextField("one_txt", this.getNextHighestDepth(), 10, 10, 100,
  22);
one_txt.border = true;
one_txt.type = "input";
this.createTextField("two_txt", this.getNextHighestDepth(), 10, 40, 100,
  22);
two_txt.border = true;
```

```
two_txt.type = "input";
this.createTextField("three_txt", this.getNextHighestDepth(), 10, 70, 100,
  22);
three_txt.border = true;
three_txt.type = "input";
this.createTextField("four_txt", this.getNextHighestDepth(), 10, 100, 100,
  22);
four_txt.border = true;
four_txt.type = "input";

one_txt.tabIndex = 3;
two_txt.tabIndex = 1;
three_txt.tabIndex = 2;
four_txt.tabIndex = 4;
```

El método `MovieClip.getNextHighestDepth()` utilizado en este ejemplo necesita Flash Player 7 o una versión posterior. Si el archivo SWF incluye un componente de la versión 2, utilice la clase DepthManager de los componentes de la versión 2 en lugar del método `MovieClip.getNextHighestDepth()`.

Véase también

`tabIndex (propiedad Button.tabIndex)`, `tabIndex (propiedad MovieClip.tabIndex)`

_target (propiedad TextField._target)

```
public _target : String [read-only]
```

Ruta de destino de la instancia del campo de texto. El destino `_self` especifica el marco actual de la ventana actual, `_blank` especifica una ventana nueva, `_parent` especifica el nivel superior del marco actual y `_top` especifica el marco del nivel más alto de la ventana actual.

Disponibilidad: ActionScript 1.0; Flash Player 6

Ejemplo

El siguiente código ActionScript crea un campo de texto denominado `my_txt` y devuelve la ruta de destino del nuevo campo, en notación con barras y con punto.

```
this.createTextField("my_txt", this.getNextHighestDepth(), 10, 10, 100,
  22);
trace(my_txt._target); // output: /my_txt
trace(eval(my_txt._target)); // output: _level0.my_txt
```

El método `MovieClip.getNextHighestDepth()` utilizado en este ejemplo necesita Flash Player 7 o una versión posterior. Si el archivo SWF incluye un componente de la versión 2, utilice la clase DepthManager de los componentes de la versión 2 en lugar del método `MovieClip.getNextHighestDepth()`.

text (propiedad TextField.text)

```
public text : String
```

Indica el texto actual del campo de texto. Las líneas se separan mediante el carácter de retorno de carro ("\r", ASCII 13). Esta propiedad contiene el texto normal sin formato del campo de texto, sin etiquetas HTML, aunque el campo de texto sea HTML.

Disponibilidad: ActionScript 1.0; Flash Player 6

Ejemplo

El ejemplo siguiente crea un campo de texto HTML llamado `my_txt` y le asigna una cadena de texto con formato HTML. Cuando controla la propiedad `htmlText`, el panel Salida muestra la cadena con formato HTML. Cuando controla el valor de la propiedad `text`, aparece la cadena no formateada con etiquetas HTML en el panel Salida.

```
this.createTextField("my_txt", this.getNextHighestDepth(), 10, 10, 400,
  22);
my_txt.html = true;
my_txt.htmlText = "<B>Lorem ipsum dolor sit amet.</B>";

trace("htmlText: "+my_txt.htmlText);
trace("text: "+my_txt.text);
```

Se genera el siguiente resultado:

```
htmlText: <P ALIGN="LEFT"><FONT FACE="Times New Roman" SIZE="12"
 COLOR="#000000" KERNING="0">
<B>Lorem ipsum dolor sit amet.</B></FONT></P>
text: Lorem ipsum dolor sit amet.
```

El método `MovieClip.getNextHighestDepth()` utilizado en este ejemplo necesita Flash Player 7 o una versión posterior. Si el archivo SWF incluye un componente de la versión 2, utilice la clase DepthManager de los componentes de la versión 2 en lugar del método `MovieClip.getNextHighestDepth()`.

Véase también

`htmlText (propiedad TextField.htmlText)`

textColor (propiedad TextField.textColor)

```
public textColor : Number
```

Indica el color del texto de un campo de texto. El sistema de colores hexadecimal utiliza seis dígitos para representar los valores de color. Cada dígito tiene dieciséis valores o caracteres posibles. El rango de caracteres de 0 a 9 y, seguidamente, de A a F. Negro se representa mediante (`#000000`) y blanco, situado en el extremo opuesto del sistema de colores, mediante (`#FFFFFF`).

Disponibilidad: ActionScript 1.0; Flash Player 6

Ejemplo

El siguiente código ActionScript crea un campo de texto y cambia su propiedad color a red.

```
this.createTextField("my_txt", 99, 10, 10, 100, 300);
my_txt.text = "this will be red text";
my_txt.textColor = 0xFF0000;
```

El método `MovieClip.getNextHighestDepth()` utilizado en este ejemplo necesita Flash Player 7 o una versión posterior. Si el archivo SWF incluye un componente de la versión 2, utilice la clase DepthManager de los componentes de la versión 2 en lugar del método `MovieClip.getNextHighestDepth()`.

textHeight (propiedad TextField.textHeight)

```
public textHeight : Number
```

Indica la altura del texto.

Disponibilidad: ActionScript 1.0; Flash Player 6

Ejemplo

El ejemplo siguiente crea un campo de texto y le asigna una cadena de texto. Se utiliza una sentencia trace para mostrar la altura y la anchura del texto en el panel Salida. A continuación se utiliza la propiedad `autoSize` para redimensionar el campo de texto, y las nuevas altura y anchura también aparecerán en el panel Salida.

```
this.createTextField("my_txt", 99, 10, 10, 100, 300);
my_txt.text = "Sample text";
trace("textHeight: "+my_txt.textHeight+", textWidth: "+my_txt.textWidth);
trace("_height: "+my_txt._height+", _width: "+my_txt._width+"\n");
my_txt.autoSize = true;
trace("after my_txt.autoSize = true;");
trace("_height: "+my_txt._height+", _width: "+my_txt._width);
```

Que devuelve la siguiente información:

```
textHeight: 15, textWidth: 56
_height: 300, _width: 100

after my_txt.autoSize = true;
_height: 19, _width: 60
```

Véase también

`textWidth (propiedad TextField.textWidth)`

textWidth (propiedad TextField.textWidth)

```
public textWidth : Number
```

Indica la anchura del texto.

Disponibilidad: ActionScript 1.0; Flash Player 6

Ejemplo

Consulte el ejemplo de `TextField.textHeight`.

Véase también

`textHeight (propiedad TextField.textHeight)`

thickness (propiedad TextField.thickness)

```
public thickness : Number
```

El grosor de los bordes de glifo en esta instancia de TextField. Esta propiedad sólo se aplica cuando `antiAliasType()` es `"advanced"`.

El rango para `thickness` es un número del -200 al 200. Si intenta definir `thickness` con un valor fuera del rango, Flash selecciona el valor más cercano en el rango (-200 o 200) para la propiedad.

Disponibilidad: ActionScript 1.0; Flash Player 8

Ejemplo

Este ejemplo crea dos campos de texto y aplica un grosor (`thickness`) de -200 a uno y 200 al otro. Se asume que tiene una fuente incorporada en la Biblioteca con el identificador de vinculación definido como `"Times-12"`. Para incorporar la fuente, siga estos pasos:

- Abra la biblioteca.
- Haga clic en el menú de opciones Biblioteca en el ángulo superior derecho de la Biblioteca.
- Seleccione "Nueva fuente" en la lista desplegable.
- Asigne a la fuente el nombre "Times-12".
- Seleccione "Times New Roman" en la lista desplegable de fuentes.
- Presione el botón "Aceptar".
- Haga clic con el botón derecho en la fuente que ha creado y seleccione "Vinculación..."
- Active la casilla de verificación "Exportar para ActionScript".
- Acepte el identificador predeterminado "Times-12" haciendo clic en el botón "Aceptar".

```
var my_format:TextFormat = new TextFormat();
my_format.font = "Times-12";

var my_text1:TextField = this.createTextField("my_text1",
  this.getNextHighestDepth(), 10, 10, 300, 30);
my_text1.text = "thickness = 200";
my_text1.antiAliasType = "advanced";
my_text1.border = true;
my_text1.thickness = 200;
my_text1.embedFonts = true;
my_text1.setTextFormat(my_format);

var my_text2:TextField = this.createTextField("my_text2",
  this.getNextHighestDepth(), 10, 50, 300, 30);
my_text2.text = "thickness = -200."
my_text2.antiAliasType = "advanced";
my_text2.thickness = -200;
my_text2.border = true;
my_text2.embedFonts = true;
my_text2.setTextFormat(my_format);
```

Si el archivo SWF incluye un componente de la versión 2, utilice la clase DepthManager de los componentes de la versión 2 en lugar del método `MovieClip.getNextHighestDepth()` utilizado en este ejemplo.

Véase también

`antiAliasType (propiedad TextField.antiAliasType)`

type (propiedad TextField.type)

```
public type : String
```

Especifica el tipo de campo de texto. Existen dos valores: `"dynamic"`, que especifica un campo de texto dinámico que el usuario no puede modificar, e `"input"`, que especifica un campo de introducción de texto.

Disponibilidad: ActionScript 1.0; ActionScript 1.0; Flash Player 6

Ejemplo

El ejemplo siguiente crea dos campos de texto: `username_txt` y `password_txt`. Se introduce texto en los dos campos de texto; sin embargo, `password_txt` tiene la propiedad `password` definida como `true`. Por lo tanto, los caracteres se muestran como asteriscos en el campo `password_txt`.

```
this.createTextField("username_txt", this.getNextHighestDepth(), 10, 10,
  100, 22);
username_txt.border = true;
username_txt.type = "input";
username_txt.maxChars = 16;
username_txt.text = "hello";

this.createTextField("password_txt", this.getNextHighestDepth(), 10, 40,
  100, 22);
password_txt.border = true;
password_txt.type = "input";
password_txt.maxChars = 16;
password_txt.password = true;
password_txt.text = "world";
```

El método `MovieClip.getNextHighestDepth()` utilizado en este ejemplo necesita Flash Player 7 o una versión posterior. Si el archivo SWF incluye un componente de la versión 2, utilice la clase DepthManager de los componentes de la versión 2 en lugar del método `MovieClip.getNextHighestDepth()`.

_url (propiedad TextField._url)

```
public _url : String [read-only]
```

Recupera el URL del archivo SWF que creó el campo de texto.

Disponibilidad: ActionScript 1.0; Flash Player 6

Ejemplo

El ejemplo siguiente recupera la URL del archivo SWF que creó el campo de texto y un archivo SWF que se carga en él.

```
this.createTextField("my_txt", 1, 10, 10, 100, 22);
trace(my_txt._url);

var mclListener:Object = new Object();
mclListener.onLoadInit = function(target_mc:MovieClip) {
  trace(target_mc._url);
};
var holder_mcl:MovieClipLoader = new MovieClipLoader();
holder_mcl.addListener(mclListener);
holder_mcl.loadClip("best_flash_ever.swf",
  this.createEmptyMovieClip("holder_mc", 2));
```

Cuando pruebe este ejemplo, la URL del archivo SWF probado y el archivo best_flash_ever.swf aparecen en el panel Salida.

La clase MovieClipLoader utilizada en este ejemplo necesita Flash Player 7 o una versión posterior.

variable (propiedad TextField.variable)

```
public variable : String
```

El nombre de la variable a la que está asociado el campo de texto. El tipo de esta propiedad es String.

Disponibilidad: ActionScript 1.0; Flash Player 6

Ejemplo

El ejemplo siguiente crea un campo de texto con el nombre `my_txt` y le asocia la variable `today_date`. Al cambiar la variable `today_date`, se actualiza el texto que aparece en `my_txt`.

```
this.createTextField("my_txt", 1, 10, 10, 200, 22);
my_txt.variable = "today_date";
var today_date:Date = new Date();

var date_interval:Number = setInterval(updateDate, 500);
function updateDate():Void {
  today_date = new Date();
}
```

_visible (propiedad TextField._visible)

```
public _visible : Boolean
```

Valor booleano que indica si el campo de texto *my_txt* puede seleccionarse. Los campos de texto no visibles (que tienen la propiedad `_visible` configurada como `false`) se desactivan.

Disponibilidad: ActionScript 1.0; Flash Player 6

Ejemplo

El ejemplo siguiente crea un campo de texto llamado `my_txt`. Un botón llamado `visible_btn` muestra y oculta `my_txt`.

```
this.createTextField("my_txt", 1, 10, 10, 200, 22);
my_txt.background = true;
my_txt.backgroundColor = 0xDFDFDF;
my_txt.border = true;
my_txt.type = "input";

visible_btn.onRelease = function() {
  my_txt._visible = !my_txt._visible;
};
```

Véase también

`_visible (propiedad Button._visible)`, `_visible (propiedad MovieClip._visible)`

_width (propiedad TextField._width)

```
public _width : Number
```

Anchura del campo de texto, expresada en píxeles.

Disponibilidad: ActionScript 1.0; Flash Player 6

Ejemplo

El ejemplo siguiente crea dos campos de texto que pueden utilizarse para cambiar la anchura y la altura de un tercer campo de texto en el escenario. Añada el siguiente código ActionScript a un archivo AS o FLA.

```
this.createTextField("my_txt", this.getNextHighestDepth(), 10, 40, 160,
  120);
my_txt.background = true;
my_txt.backgroundColor = 0xFF0000;
my_txt.border = true;
my_txt.multiline = true;
my_txt.type = "input";
my_txt.wordWrap = true;
```

```
this.createTextField("width_txt", this.getNextHighestDepth(), 10, 10, 30,
  20);
width_txt.border = true;
width_txt.maxChars = 3;
width_txt.restrict = "0-9";
width_txt.type = "input";
width_txt.text = my_txt._width;
width_txt.onChanged = function() {
  my_txt._width = this.text;
}

this.createTextField("height_txt", this.getNextHighestDepth(), 70, 10, 30,
  20);
height_txt.border = true;
height_txt.maxChars = 3;
height_txt.restrict = "0-9";
height_txt.type = "input";
height_txt.text = my_txt._height;
height_txt.onChanged = function() {
  my_txt._height = this.text;
}
```

Al probar el ejemplo, intente especificar nuevos valores en `width_txt` y `height_txt` para cambiar las dimensiones de `my_txt`.

El método `MovieClip.getNextHighestDepth()` utilizado en este ejemplo necesita Flash Player 7 o una versión posterior. Si el archivo SWF incluye un componente de la versión 2, utilice la clase DepthManager de los componentes de la versión 2 en lugar del método `MovieClip.getNextHighestDepth()`.

Véase también

`_height (propiedad TextField._height)`

wordWrap (propiedad TextField.wordWrap)

`public wordWrap : Boolean`

Valor booleano que indica si el campo de texto tiene ajuste de texto. Si el valor de `wordWrap` es `true`, el campo de texto tiene ajuste de texto; si el valor es `false`, el campo de texto no tiene ajuste de texto.

Disponibilidad: ActionScript 1.0; Flash Player 6

Ejemplo

El ejemplo siguiente muestra cómo afecta `wordWrap` al texto largo en un campo de texto creado en tiempo de ejecución.

```
this.createTextField("my_txt", 99, 10, 10, 100, 200);
my_txt.text = "This is very long text that will certainly extend beyond the
  width of this text field";
my_txt.border = true;
```

Para probar el archivo SWF en Flash Player, seleccione Control > Probar película. A continuación, vuelva a ActionScript y añada la línea siguiente al código, y pruebe de nuevo el archivo SWF:

```
my_txt.wordWrap = true;
```

_x (propiedad TextField._x)

```
public _x : Number
```

Entero que establece la coordenada x de un campo de texto como relativa a las coordenadas locales del clip de película principal. Si un campo de texto se encuentra en la línea de tiempo principal, su sistema de coordenadas hará referencia a la esquina superior izquierda del escenario como (0, 0). Si el campo de texto está dentro de un clip de película que incluye transformaciones, el campo de texto estará en el sistema de coordenadas local del clip de película en el que está contenido. Por consiguiente, en el caso de un clip de película con un giro de 90 grados en sentido contrario al de las agujas del reloj, el campo de texto contenido hereda un sistema de coordenadas con un giro de 90 grados en sentido contrario al de las agujas del reloj. Las coordenadas del campo de texto hacen referencia a la posición del punto de registro.

Disponibilidad: ActionScript 1.0; Flash Player 6

Ejemplo

El ejemplo siguiente crea un campo de texto donde se haga clic con el ratón. Cuando crea un campo de texto, ese campo muestra las coordenadas *x* e *y* actuales del campo de texto.

```
this.createTextField("coords_txt", this.getNextHighestDepth(), 0, 0, 60,
  22);
coords_txt.autoSize = true;
coords_txt.selectable = false;
coords_txt.border = true;

var mouseListener:Object = new Object();
mouseListener.onMouseDown = function() {
  coords_txt.text = "X:"+Math.round(_xmouse)+", Y:"+Math.round(_ymouse);
  coords_txt._x = _xmouse;
  coords_txt._y = _ymouse;
```

```
};
Mouse.addListener(mouseListener);
```

El método `MovieClip.getNextHighestDepth()` utilizado en este ejemplo necesita Flash Player 7 o una versión posterior. Si el archivo SWF incluye un componente de la versión 2, utilice la clase DepthManager de los componentes de la versión 2 en lugar del método `MovieClip.getNextHighestDepth()`.

Véase también

`_xscale (propiedad TextField._xscale)`, `_y (propiedad TextField._y)`, `_yscale (propiedad TextField._yscale)`

_xmouse (propiedad TextField._xmouse)

```
public _xmouse : Number [read-only]
```

Devuelve la coordenada x de la posición del ratón relativa al campo de texto.

Disponibilidad: ActionScript 1.0; Flash Player 6

Ejemplo

El ejemplo siguiente crea tres campos de texto en el escenario. La instancia `mouse_txt` muestra la posición actual del ratón en relación al escenario. La instancia `textfield_txt` muestra la posición actual del puntero del ratón en relación a la instancia `my_txt`. Añada el siguiente código ActionScript a un archivo AS o FLA:

```
this.createTextField("mouse_txt", this.getNextHighestDepth(), 10, 10, 200,
  22);
mouse_txt.border = true;
this.createTextField("textfield_txt", this.getNextHighestDepth(), 220, 10,
  200, 22);
textfield_txt.border = true;
this.createTextField("my_txt", this.getNextHighestDepth(), 100, 100, 160,
  120);
my_txt.border = true;

var mouseListener:Object = new Object();
mouseListener.onMouseMove = function() {
  mouse_txt.text = "MOUSE ... X:" + Math.round(_xmouse) + ",\tY:" +
  Math.round(_ymouse);
  textfield_txt.text = "TEXTFIELD ... X:" + Math.round(my_txt._xmouse) +
  ",\tY:" +
  Math.round(my_txt._ymouse);
}

Mouse.addListener(mouseListener);
```

El método `MovieClip.getNextHighestDepth()` utilizado en este ejemplo necesita Flash Player 7 o una versión posterior. Si el archivo SWF incluye un componente de la versión 2, utilice la clase DepthManager de los componentes de la versión 2 en lugar del método `MovieClip.getNextHighestDepth()`.

Véase también

`_ymouse (propiedad TextField._ymouse)`

_xscale (propiedad TextField._xscale)

```
public _xscale : Number
```

Determina la escala horizontal del campo de texto aplicada desde el punto de registro del campo de texto, expresada como porcentaje. El punto de registro predeterminado es (0,0).

Disponibilidad: ActionScript 1.0; Flash Player 6

Ejemplo

El ejemplo siguiente cambia la escala de la instancia `my_txt` cuando hace clic en las instancias `scaleUp_btn` y `scaleDown_btn`.

```
this.createTextField("my_txt", 99, 10, 40, 100, 22);
my_txt.autoSize = true;
my_txt.border = true;
my_txt.selectable = false;
my_txt.text = "Sample text goes here.";

scaleUp_btn.onRelease = function() {
  my_txt._xscale = 2;
  my_txt._yscale = 2;
}
scaleDown_btn.onRelease = function() {
  my_txt._xscale /= 2;
  my_txt._yscale /= 2;
}
```

Véase también

`_x (propiedad TextField._x)`, `_y (propiedad TextField._y)`, `_yscale (propiedad TextField._yscale)`

_y (propiedad TextField._y)

```
public _y : Number
```

Establece la coordenada y de un campo de texto como relativa a las coordenadas locales del clip de película principal. Si un campo de texto se encuentra en la línea de tiempo principal, su sistema de coordenadas hará referencia a la esquina superior izquierda del escenario como (0, 0). Si el campo de texto está dentro de otro clip de película que incluye transformaciones, el campo de texto estará en el sistema de coordenadas local del clip de película en el que está contenido. Por consiguiente, en el caso de un clip de película con un giro de 90 grados en sentido contrario al de las agujas del reloj, el campo de texto contenido hereda un sistema de coordenadas con un giro de 90 grados en sentido contrario al de las agujas del reloj. Las coordenadas del campo de texto hacen referencia a la posición del punto de registro.

Disponibilidad: ActionScript 1.0; Flash Player 6

Ejemplo

Consulte el ejemplo de `TextField._x`.

Véase también

`_x (propiedad TextField._x)`, `_xscale (propiedad TextField._xscale)`, `_yscale (propiedad TextField._yscale)`

_ymouse (propiedad TextField._ymouse)

```
public _ymouse : Number [read-only]
```

Indica la coordenada y de la posición del ratón relativa al campo de texto.

Disponibilidad: ActionScript 1.0; Flash Player 6

Ejemplo

Consulte el ejemplo de `TextField._xmouse`.

Véase también

`_xmouse (propiedad TextField._xmouse)`

_yscale (propiedad TextField._yscale)

```
public _yscale : Number
```

La escala vertical del campo de texto aplicada desde el punto de registro del campo de texto, expresada como porcentaje. El punto de registro predeterminado es (0,0).

Disponibilidad: ActionScript 1.0; Flash Player 6

Ejemplo

Consulte el ejemplo de `TextField._xscale`.

Véase también

`_x (propiedad TextField._x)`, `_xscale (propiedad TextField._xscale)`, `_y (propiedad TextField._y)`

TextFormat

```
Object
  |
  +-TextFormat
```

```
public class TextFormat
extends Object
```

La clase TextFormat representa la información de formato de caracteres. Utilice la clase TextFormat para crear formato de texto específico para campos de texto. Puede aplicar formato de texto tanto a campos de texto estáticos como dinámicos. Algunas propiedades de la clase TextFormat no están disponibles para fuentes incorporadas y de dispositivo.

Disponibilidad: ActionScript 1.0; Flash Player 6

Véase también

`setTextFormat (método TextField.setTextFormat)`, `getTextFormat (método TextField.getTextFormat)`

Resumen de propiedades

Modificadores	Propiedad	Descripción
	`align:String`	Cadena que indica la alineación del párrafo.
	`blockIndent:Number`	Un número que indica la sangría de bloque en puntos.
	`bold:Boolean`	Un valor booleano que especifica si el texto está en negrita.

Modificadores	Propiedad	Descripción
	`bullet:Boolean`	Un valor booleano que indica que el texto forma parte de una lista con viñetas.
	`color:Number`	Indica el color del texto.
	`font:String`	El nombre de la fuente del texto con este formato, expresado como una cadena.
	`indent:Number`	Un entero que indica la sangría desde el margen izquierdo hasta el primer carácter del párrafo.
	`italic:Boolean`	Un valor booleano que indica si el texto con este formato de texto está en cursiva.
	`kerning:Boolean`	Un valor booleano que indica si el ajuste de caracteres está activado o desactivado.
	`leading:Number`	Un entero que representa la cantidad de espacio vertical entre líneas (denominado *interlineado*) expresado en píxeles.
	`leftMargin:Number`	El margen izquierdo del párrafo expresado en puntos.
	`letterSpacing:Number`	La cantidad de espacio distribuido uniformemente entre caracteres.
	`rightMargin:Number`	El margen derecho del párrafo expresado en puntos.
	`size:Number`	El tamaño en puntos del texto con este formato de texto.
	`tabStops:Array`	Especifica las tabulaciones personalizadas como una matriz de enteros no negativos.
	`target:String`	Indica la ventana de destino en la que se muestra el hipervínculo.
	`underline:Boolean`	Un valor booleano que indica si el texto que utiliza este formato de texto está subrayado (`true`) o no (`false`).
	`url:String`	Indica la URL con la que está vinculado el texto con este formato de texto.

Propiedades heredadas de la clase Object

`constructor (propiedad Object.constructor), __proto__ (propiedad Object.__proto__), prototype (propiedad Object.prototype), __resolve (propiedad Object.__resolve)`

Resumen de constructores

Firma	Descripción
`TextFormat([font:String], [size:Number], [color:Number], [bold:Boolean], [italic:Boolean], [underline:Boolean], [url:String], [target:String], [align:String], [leftMargin:Number], [rightMargin:Number], [indent:Number], [leading:Number])`	Crea un objeto TextFormat con las propiedades especificadas.

Resumen de métodos

Modificadores	Firma	Descripción
	`getTextExtent(text:String, [width:Number]) : Object`	*Desfasado* desde Flash Player 8. No hay opción sustitutiva. Devuelve información sobre las medidas del texto para la cadena `text` en el formato especificado por `my_fmt`.

Métodos heredados de la clase Object

addProperty (método Object.addProperty), hasOwnProperty (método Object.hasOwnProperty), isPropertyEnumerable (método Object.isPropertyEnumerable), isPrototypeOf (método Object.isPrototypeOf), registerClass (método Object.registerClass), toString (método Object.toString), unwatch (método Object.unwatch), valueOf (método Object.valueOf), watch (método Object.watch)

align (propiedad TextFormat.align)

```
public align : String
```

Cadena que indica la alineación del párrafo. Puede aplicar esta propiedad a texto estático y dinámico. En la lista siguiente se muestran los valores que puede tener esta propiedad:

- `"left"`-El párrafo está alineado a la izquierda.
- `"center"`-El párrafo está centrado.
- `"right"`-El párrafo está alineado a la derecha.
- `"justify"`-El párrafo está justificado. (Este valor se añadió Flash Player 8.)

El valor predeterminado es `null`, que indica que la propiedad no se ha definido.

Disponibilidad: ActionScript 1.0; Flash Player 6

Ejemplo

El ejemplo siguiente muestra la propiedad `align` que se va a definir como justificada; los caracteres de cada línea aparecerán extendidos de forma que el espaciado del texto es más uniforme a la derecha.

```
var format:TextFormat = new TextFormat();
format.align = "justify";

var txtField:TextField = this.createTextField("txtField",
  this.getNextHighestDepth(), 100, 100, 300, 100);
txtField.multiline = true;
txtField.wordWrap = true;
txtField.border = true;
txtField.text = "When this text is justified, it will be "
  + "spread out to more cleanly fill the horizontal "
  + "space for each line. This can be considered an "
  + "improvement over regular left-aligned text that "
  + "will simply wrap and do no more.";
txtField.setTextFormat(format);
```

El método `MovieClip.getNextHighestDepth()` utilizado en este ejemplo necesita Flash Player 7 o una versión posterior. Si el archivo SWF incluye un componente de la versión 2, utilice la clase DepthManager de los componentes de la versión 2 en lugar del método `MovieClip.getNextHighestDepth()`.

blockIndent (propiedad TextFormat.blockIndent)

```
public blockIndent : Number
```

Un número que indica la sangría de bloque en puntos. La sangría de bloque se aplica a un bloque de texto completo; es decir, a todas las líneas del texto. Por contra, la sangría normal (`TextFormat.indent`) sólo afecta a la primera línea de cada párrafo. Si esta propiedad tiene el valor `null`, el objeto TextFormat no especifica la sangría de bloque.

Disponibilidad: ActionScript 1.0; Flash Player 6

Ejemplo

En este ejemplo se crea un campo de texto con borde y blockIndent se establece en 20.

```
this.createTextField("mytext",1,100,100,100,100);
mytext.multiline = true;
mytext.wordWrap = true;
mytext.border = true;

var myformat:TextFormat = new TextFormat();
myformat.blockIndent = 20;

mytext.text = "This is my first test field object text";
mytext.setTextFormat(myformat);
```

bold (propiedad TextFormat.bold)

```
public bold : Boolean
```

Un valor booleano que especifica si el texto está en negrita. El valor predeterminado es `null`, que indica que la propiedad no se ha definido. Si el valor es `true`, el texto está en negrita.

Disponibilidad: ActionScript 1.0; Flash Player 6

Ejemplo

En el ejemplo siguiente se crea un campo de texto que incluye caracteres en negrita.

```
var my_fmt:TextFormat = new TextFormat();
my_fmt.bold = true;

this.createTextField("my_txt", 1, 100, 100, 300, 100);
my_txt.multiline = true;
my_txt.wordWrap = true;
my_txt.border = true;
my_txt.text = "This is my test field object text";
my_txt.setTextFormat(my_fmt);
```

bullet (propiedad TextFormat.bullet)

```
public bullet : Boolean
```

Un valor booleano que indica que el texto forma parte de una lista con viñetas. En una lista con viñetas, todos los párrafos de texto están sangrados. A la izquierda de la primera línea de cada párrafo se muestra un símbolo de viñeta. El valor predeterminado es `null`.

Disponibilidad: ActionScript 1.0; Flash Player 6

Ejemplo

En el ejemplo siguiente se crea un nuevo campo de texto durante la ejecución y se introduce una cadena con un salto de línea en el campo. La clase TextFormat se utiliza para asignar formato a los caracteres añadiendo viñetas a cada línea del campo de texto. Esto se ilustra en el siguiente código ActionScript:

```
var my_fmt:TextFormat = new TextFormat();
my_fmt.bullet = true;

this.createTextField("my_txt", 1, 100, 100, 300, 100);
my_txt.multiline = true;
my_txt.wordWrap = true;
my_txt.border = true;
my_txt.text = "this is my text"+newline;
my_txt.text += "this is more text"+newline;
my_txt.setTextFormat(my_fmt);
```

color (propiedad TextFormat.color)

```
public color : Number
```

Indica el color del texto. Número que contiene tres componentes RGB de 8 bits; por ejemplo, 0xFF0000 es rojo y 0x00FF00, verde.

Disponibilidad: ActionScript 1.0; Flash Player 6

Ejemplo

En el ejemplo siguiente se crea un campo de texto y el color del texto se establece en rojo.

```
var my_fmt:TextFormat = new TextFormat();
my_fmt.blockIndent = 20;
my_fmt.color = 0xFF0000; // hex value for red
```

```
this.createTextField("my_txt", 1, 100, 100, 300, 100);
my_txt.multiline = true;
my_txt.wordWrap = true;
my_txt.border = true;
my_txt.text = "this is my first test field object text";
my_txt.setTextFormat(my_fmt);
```

font (propiedad TextFormat.font)

```
public font : String
```

El nombre de la fuente del texto con este formato, expresado como una cadena. El valor predeterminado es `null`, que indica que la propiedad no se ha definido.

Disponibilidad: ActionScript 1.0; Flash Player 6

Ejemplo

En el ejemplo siguiente se crea un campo de texto y la fuente se define como Courier.

```
this.createTextField("mytext",1,100,100,100,100);
mytext.multiline = true;
mytext.wordWrap = true;
mytext.border = true;

var myformat:TextFormat = new TextFormat();
myformat.font = "Courier";

mytext.text = "this is my first test field object text";
mytext.setTextFormat(myformat);
```

getTextExtent (método TextFormat.getTextExtent)

```
public getTextExtent(text:String, [width:Number]) : Object
```

Desfasado desde Flash Player 8. No hay opción sustitutiva.

Devuelve información sobre las medidas del texto para la cadena `text` en el formato especificado por `my_fmt`. La cadena de texto se considera como texto normal (que no es HTML).

El método devuelve un objeto con seis propiedades: `ascent`, `descent`, `width`, `height`, `textFieldHeight` y `textFieldWidth`. Todas las medidas se expresan en píxeles.

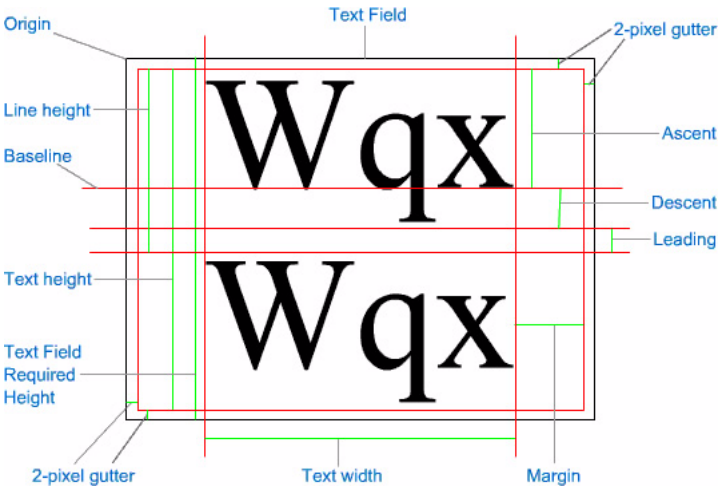
Si se especifica un parámetro `width`, se aplicará ajuste de texto al texto especificado. Esto le permite determinar la altura a la que un cuadro de texto muestra todo el texto especificado.

Las medidas `ascent` (ascendente) y `descent` (descendente) proporcionan, respectivamente, la distancia por encima y por debajo de la línea de base de una línea de texto. La línea de base de la primera línea de texto se sitúa en el origen del campo de texto más su medida de `ascent`.

Las medidas `width` y `height` proporcionan la anchura y la altura de la cadena de texto. Las medidas `textFieldHeight` y `textFieldWidth` proporcionan la altura y la anchura necesarias para que un objeto de campo de texto muestre la cadena de texto completa. Los campos de texto tienen alrededor un espacio de 2 píxeles de ancho, por lo que el valor de `textFieldHeight` es igual al valor de `height` + 4; de igual forma, el valor de `textFieldWidth` siempre es igual al valor de `width` + 4.

Si está creando un campo de texto basado en las medidas del texto, utilice `textFieldHeight` en lugar de `height` y `textFieldWidth` en lugar de `width`.

En la siguiente figura se ilustran estas medidas.



Al configurar el objeto TextFormat, establezca todos los atributos exactamente de la forma que se establecerán para la creación del campo de texto, incluido el nombre de la fuente, el tamaño de fuente y el espacio interlineal. El valor predeterminado de espacio interlineal (leading) es 2.

Disponibilidad: ActionScript 1.0; Flash Player 6

Parámetros

`text``:String` - Una cadena.

`width``:Number` [opcional] - Un número que representa la anchura, en píxeles, a la que debe ajustarse el texto especificado.

Valor devuelto

`Object` - Un objeto con las propiedades `width`, `height`, `ascent`, `descent`, `textFieldHeight`, `textFieldWidth`.

Ejemplo

Este ejemplo crea un campo de texto de una línea lo suficientemente grande para mostrar una cadena de texto con el formato especificado.

```
var my_str:String = "Small string";

// Create a TextFormat object,
// and apply its properties.
var my_fmt:TextFormat = new TextFormat();
with (my_fmt) {
  font = "Arial";
  bold = true;
}

// Obtain metrics information for the text string
// with the specified formatting.
var metrics:Object = my_fmt.getTextExtent(my_str);

// Create a text field just large enough to display the text.
this.createTextField("my_txt", this.getNextHighestDepth(), 100, 100,
  metrics.textFieldWidth,
metrics.textFieldHeight);
my_txt.border = true;
my_txt.wordWrap = true;
// Assign the same text string and TextFormat object to the my_txt object.
my_txt.text = my_str;
my_txt.setTextFormat(my_fmt);
```

El ejemplo siguiente crea un campo de texto de varias líneas, de 100 píxeles de ancho lo suficientemente alto para mostrar una cadena con el formato especificado.

```
// Create a TextFormat object.
var my_fmt:TextFormat = new TextFormat();
// Specify formatting properties for the TextFormat object:
my_fmt.font = "Arial";
my_fmt.bold = true;
my_fmt.leading = 4;

// The string of text to be displayed
var textToDisplay:String = "Macromedia Flash Player 7, now with improved
  text metrics.";

// Obtain text measurement information for the string,
// wrapped at 100 pixels.
var metrics:Object = my_fmt.getTextExtent(textToDisplay, 100);

// Create a new TextField object using the metric
// information just obtained.
this.createTextField("my_txt", this.getNextHighestDepth(), 50, 50-
  metrics.ascent, 100,
metrics.textFieldHeight);
```

```
my_txt.wordWrap = true;
my_txt.border = true;
// Assign the text and the TextFormat object to the TextObject:
my_txt.text = textToDisplay;
my_txt.setTextFormat(my_fmt);
```

indent (propiedad TextFormat.indent)

```
public indent : Number
```

Un entero que indica la sangría desde el margen izquierdo hasta el primer carácter del párrafo. Un valor positivo indica sangrado normal. Puede utilizar un valor negativo, pero el sangrado negativo sólo se aplica si el margen izquierdo es mayor que 0. Para definir el margen mayor que 0, utilice las propiedades `indent` o `blockIndent` del objeto TextFormat. El valor predeterminado es `null`, que indica que la propiedad no se ha definido.

Disponibilidad: ActionScript 1.0; Flash Player 6

Ejemplo

En el ejemplo siguiente se crea un campo de texto y se establece la sangría en 10:

```
this.createTextField("mytext",1,100,100,100,100);
mytext.multiline = true;
mytext.wordWrap = true;
mytext.border = true;

var myformat:TextFormat = new TextFormat();
myformat.indent = 10;

mytext.text = "this is my first test field object text";
mytext.setTextFormat(myformat);
```

Véase también

`blockIndent (propiedad TextFormat.blockIndent)`

italic (propiedad TextFormat.italic)

```
public italic : Boolean
```

Un valor booleano que indica si el texto con este formato de texto está en cursiva. El valor predeterminado es `null`, que indica que la propiedad no se ha definido.

Disponibilidad: ActionScript 1.0; Flash Player 6

Ejemplo

En el ejemplo siguiente se crea un campo de texto y el estilo del texto se define en cursiva.

```
this.createTextField("mytext",1,100,100,100,100);
mytext.multiline = true;
mytext.wordWrap = true;
mytext.border = true;

var myformat:TextFormat = new TextFormat();
myformat.italic = true;

mytext.text = "This is my first text field object text";
mytext.setTextFormat(myformat);
```

kerning (propiedad TextFormat.kerning)

```
public kerning : Boolean
```

Un valor booleano que indica si el ajuste de caracteres está activado o desactivado. Mediante el ajuste entre caracteres se introduce una cantidad de espacio predeterminada entre pares concretos de caracteres para permitir una legibilidad óptima. El valor predeterminado es `false`, que indica que el ajuste entre caracteres está desactivado.

El ajuste entre caracteres sólo se admite en las fuentes incorporadas. Algunas fuentes, como Courier New, no admiten el ajuste entre caracteres.

La propiedad `kerning` sólo puede utilizarse en archivos SWF creados en Windows; no en archivos SWF creados en Macintosh. Sin embargo, estos archivos SWF *pueden* reproducirse en versiones no Windows de Flash Player y se aplica el ajuste entre caracteres.

Utilice este ajuste exclusivamente cuando sea necesario, como en encabezados en tamaños de fuente grandes.

Disponibilidad: ActionScript 1.0; Flash Player 8

Ejemplo

El ejemplo siguiente muestra dos campos de texto: El formato del primero utiliza texto en rojo con `kerning` como `false` y el formato del segundo emplea texto en azul con `kerning` como `true`. Para usar este ejemplo, se agrega un símbolo de fuente a la Biblioteca y, a continuación, se selecciona Arial como la fuente. El cuadro de diálogo Propiedades de vinculación para la fuente, defina el nombre de identificador como `"Font 1"`, seleccione Exportar para ActionScript y elija Exportar en primer fotograma.

```
var fmt1:TextFormat = new TextFormat();
fmt1.font = "Font 1";
fmt1.size = 50;
```

```
fmt1.color = 0xFF0000;
fmt1.kerning = false;

var fmt2:TextFormat = new TextFormat();
fmt2.font = "Font 1";
fmt2.size = 50;
fmt2.color = 0x0000FF;
fmt2.kerning = true;

this.createTextField("tf1", this.getNextHighestDepth(), 10, 10, 400, 100);
tf1.embedFonts = true;
tf1.text = "Text 7AVA-7AVA";
tf1.setTextFormat(fmt1);

this.createTextField("tf2", this.getNextHighestDepth(), 10, 40, 400, 100);
tf2.embedFonts = true;
tf2.text = tf1.text;
tf2.setTextFormat(fmt2);
```

Si el archivo SWF incluye un componente de la versión 2, utilice la clase DepthManager de los componentes de la versión 2 en lugar del método `MovieClip.getNextHighestDepth()` utilizado en este ejemplo.

leading (propiedad TextFormat.leading)

```
public leading : Number
```

Un entero que representa la cantidad de espacio vertical entre líneas (denominado *interlineado*) expresado en píxeles. El valor predeterminado es `null`, que indica que la propiedad no se ha definido.

Flash Player 8 admite *interlineado negativo*, lo que significa que la cantidad de espacio entre líneas es menor que la altura del texto. El interlineado negativo puede resultar útil para aproximar mucho las líneas de texto, como en los encabezados. Para evitar que el texto se solape, aplique el interlineado negativo a líneas de texto que no contengan trazos descendentes, como el texto que aparece todo en mayúsculas.

Disponibilidad: ActionScript 1.0; Flash Player 6

Ejemplo

En el ejemplo siguiente se crea un campo de texto y se establece el interlineado en 10.

```
var my_fmt:TextFormat = new TextFormat();
my_fmt.leading = 10;

this.createTextField("my_txt", 1, 100, 100, 100, 100);
my_txt.multiline = true;
my_txt.wordWrap = true;
```

```
my_txt.border = true;
my_txt.text = "This is my first text field object text";
my_txt.setTextFormat(my_fmt);
```

leftMargin (propiedad TextFormat.leftMargin)

```
public leftMargin : Number
```

El margen izquierdo del párrafo expresado en puntos. El valor predeterminado es `null`, que indica que la propiedad no se ha definido.

Disponibilidad: ActionScript 1.0; Flash Player 6

Ejemplo

En el ejemplo siguiente se crea un campo de texto y el margen izquierdo se establece en 20 puntos.

```
this.createTextField("mytext",1,100,100,100,100);
mytext.multiline = true;
mytext.wordWrap = true;
mytext.border = true;

var myformat:TextFormat = new TextFormat();
myformat.leftMargin = 20;

mytext.text = "this is my first test field object text";
mytext.setTextFormat(myformat);
```

letterSpacing (propiedad TextFormat.letterSpacing)

```
public letterSpacing : Number
```

La cantidad de espacio distribuido uniformemente entre caracteres. El valor de Number especifica el número de píxeles que se añaden al espacio después de cada carácter. Un valor negativo condensa el espacio entre caracteres.

Las fuentes del sistema admiten sólo valores enteros, sin embargo, para fuentes incorporadas, puede especificar valores de coma flotante (no entero), como 2,6.

Disponibilidad: ActionScript 1.0; Flash Player 8

Ejemplo

El código siguiente utiliza dos objetos TextFormat para aplicar valores positivos y negativos de `letterSpacing` a distintos rangos de texto en un campo de texto.

```
this.createTextField("mytext", this.getNextHighestDepth(), 10, 10, 200,
  100);
mytext.multiline = true;
```

```
mytext.wordWrap = true;
mytext.border = true;

var format1:TextFormat = new TextFormat();
format1.letterSpacing = -1;

var format2:TextFormat = new TextFormat();
format2.letterSpacing = 10;

mytext.text = "Eat at \nJOE'S.";
mytext.setTextFormat(0, 7, format1);
mytext.setTextFormat(8, 12, format2);
```

Si el archivo SWF incluye un componente de la versión 2, utilice la clase DepthManager de los componentes de la versión 2 en lugar del método `MovieClip.getNextHighestDepth()` utilizado en este ejemplo.

rightMargin (propiedad TextFormat.rightMargin)

```
public rightMargin : Number
```

El margen derecho del párrafo expresado en puntos. El valor predeterminado es `null`, que indica que la propiedad no se ha definido.

Disponibilidad: ActionScript 1.0; Flash Player 6

Ejemplo

En el ejemplo siguiente se crea un campo de texto y el margen derecho se establece en 20 puntos.

```
this.createTextField("mytext",1,100,100,100,100);
mytext.multiline = true;
mytext.wordWrap = true;
mytext.border = true;

var myformat:TextFormat = new TextFormat();
myformat.rightMargin = 20;

mytext.text = "this is my first test field object text";
mytext.setTextFormat(myformat);
```

size (propiedad TextFormat.size)

```
public size : Number
```

El tamaño en puntos del texto con este formato de texto. El valor predeterminado es `null`, que indica que la propiedad no se ha definido.

Disponibilidad: ActionScript 1.0; Flash Player 6

Ejemplo

En el ejemplo siguiente se crea un campo de texto y el tamaño del texto se establece en 20 puntos.

```
this.createTextField("mytext",1,100,100,100,100);
mytext.multiline = true;
mytext.wordWrap = true;
mytext.border = true;

var myformat:TextFormat = new TextFormat();
myformat.size = 20;

mytext.text = "This is my first text field object text";
mytext.setTextFormat(myformat);
```

tabStops (propiedad TextFormat.tabStops)

```
public tabStops : Array
```

Especifica las tabulaciones personalizadas como una matriz de enteros no negativos. Cada tabulación se especifica en píxeles. Si no se han especificado tabulaciones personalizadas (`null`), la tabulación predeterminada es 4 (la anchura media de los caracteres).

Disponibilidad: ActionScript 1.0; Flash Player 6

Ejemplo

En el ejemplo siguiente se crean dos campos de texto, uno con tabulaciones cada 40 píxeles y otro con tabulaciones cada 75 píxeles.

```
this.createTextField("mytext",1,100,100,400,100);
mytext.border = true;
var myformat:TextFormat = new TextFormat();
myformat.tabStops = [40,80,120,160];
mytext.text = "A\tB\tC\tD"; // \t is the tab stop character
mytext.setTextFormat(myformat);

this.createTextField("mytext2",2,100,220,400,100);
mytext2.border = true;
var myformat2:TextFormat = new TextFormat();
myformat2.tabStops = [75,150,225,300];
mytext2.text ="A\tB\tC\tD";
mytext2.setTextFormat(myformat2);
```

target (propiedad TextFormat.target)

```
public target : String
```

Indica la ventana de destino en la que se muestra el hipervínculo. Si la ventana de destino es una cadena vacía, el texto se muestra en la ventana de destino predeterminada, `_self`. Puede elegir un nombre personalizado o uno de los cuatro nombres siguientes: `_self` especifica el marco actual de la ventana actual, `_blank` especifica una ventana nueva, `_parent` especifica el nivel superior del marco actual y `_top` especifica el marco del nivel más alto de la ventana actual. Si la propiedad `TextFormat.url` es una cadena vacía o tiene el valor `null`, puede obtener o establecer esta propiedad, pero ésta no tendrá ningún efecto.

Disponibilidad: ActionScript 1.0; Flash Player 6

Ejemplo

En el ejemplo siguiente se crea un campo de texto vinculado con el sitio Web de Macromedia. En el ejemplo se utiliza `TextFormat.target` para mostrar el sitio Web de Macromedia en una ventana del navegador nueva.

```
var myformat:TextFormat = new TextFormat();
myformat.url = "http://www.macromedia.com";
myformat.target = "_blank";

this.createTextField("mytext",1,100,100,200,100);
mytext.multiline = true;
mytext.wordWrap = true;
mytext.border = true;
mytext.html = true;
mytext.text = "Go to Macromedia.com";
mytext.setTextFormat(myformat);
```

Véase también

`url (propiedad TextFormat.url)`

Constructor TextFormat

```
public TextFormat([font:String], [size:Number], [color:Number],
  [bold:Boolean], [italic:Boolean], [underline:Boolean], [url:String],
  [target:String], [align:String], [leftMargin:Number],
  [rightMargin:Number], [indent:Number], [leading:Number])
```

Crea un objeto TextFormat con las propiedades especificadas. Puede cambiar las propiedades del objeto TextFormat para cambiar el formato de los campos de texto.

Cualquier parámetro puede establecerse en `null` para indicar que no está definido. Todos los parámetros son opcionales; se considerará que los parámetros que se omitan tienen el valor `null`.

Disponibilidad: ActionScript 1.0; Flash Player 6

Parámetros

`font``:String` - [opcional] El nombre de una fuente de texto como cadena.

`size``:Number` [opcional] - Un entero que indica el tamaño en puntos.

`color``:Number` [opcional] - El color del texto que utiliza este formato de texto. Número que contiene tres componentes RGB de 8 bits; por ejemplo, 0xFF0000 es rojo y 0x00FF00, verde.

`bold``:Boolean` [opcional] - Un valor booleano que indica si el texto está en negrita.

`italic``:Boolean` [opcional] - Un valor booleano que indica si el texto está en cursiva.

`underline``:Boolean` [opcional] - Un valor booleano que indica si el texto está subrayado.

`url``:String` [opcional] - La URL con la que está vinculado el texto con este formato de texto. Si `url` es una cadena vacía, el texto no tiene hipervínculo.

`target``:String` [opcional] - La ventana de destino en la que se muestra el hipervínculo. Si la ventana de destino es una cadena vacía, el texto se muestra en la ventana de destino predeterminada, `_self`. Si el parámetro `url` se define como una cadena vacía o en el valor `null`, puede obtener o establecer esta propiedad, pero ésta no tendrá ningún efecto.

`align``:String` [opcional] - La alineación del párrafo representado como una cadena. Si es `"left"`, párrafo está alineado a la izquierda. Si es `"center"`, el párrafo está centrado. Si es `"right"`, párrafo está alineado a la derecha.

`leftMargin``:Number` [opcional] - Indica el margen izquierdo del párrafo, expresado en puntos.

`rightMargin``:Number` [opcional] - Indica el margen derecho del párrafo, expresado en puntos.

`indent``:Number` [opcional] - Un entero que indica la sangría desde el margen izquierdo hasta el primer carácter del párrafo.

`leading``:Number` [opcional] - Un número que indica la cantidad de espacio vertical entre las líneas (interlineado).

Ejemplo

En el ejemplo siguiente se crea un objeto TextFormat, se asigna formato al campo de texto `stats_txt` y se crea un nuevo campo de texto para mostrar el texto en:

```
// Define a TextFormat which is used to format the stats_txt text field.
var my_fmt:TextFormat = new TextFormat();
my_fmt.bold = true;
my_fmt.font = "Arial";
my_fmt.size = 12;
my_fmt.color = 0xFF0000;
// Create a text field to display the player's statistics.
this.createTextField("stats_txt", 5000, 10, 0, 530, 22);
// Apply the TextFormat to the text field.
stats_txt.setNewTextFormat(my_fmt);
stats_txt.selectable = false;
stats_txt.text = "Lorem ipsum dolor sit amet...";
```

Para ver otro ejemplo, consulte el archivo animations.fla de la carpeta de ejemplos de ActionScript. La lista siguiente muestra rutas habituales a la carpeta de ejemplos de ActionScript:

- Windows: *unidad de inicio*\Archivos de programa\Macromedia\Flash 8\Samples and Tutorials\Samples\ActionScript
- Macintosh: *Disco duro de Macintosh*/Applications/Macromedia Flash 8/Samples and Tutorials/Samples/ActionScript

underline (propiedad TextFormat.underline)

```
public underline : Boolean
```

Un valor booleano que indica si el texto que utiliza este formato de texto está subrayado (`true`) o no (`false`). Este subrayado es similar al que se obtiene mediante la etiqueta `<U>`, aunque ésta última no es un subrayado real, ya que no omite los trazos descendentes correctamente. El valor predeterminado es `null`, que indica que la propiedad no se ha definido.

Disponibilidad: ActionScript 1.0; Flash Player 6

Ejemplo

En el ejemplo siguiente se crea un campo de texto y el estilo del texto se define en subrayado.

```
this.createTextField("mytext",1,100,100,200,100);
mytext.multiline = true;
mytext.wordWrap = true;
mytext.border = true;

var myformat:TextFormat = new TextFormat();
```

```
myformat.underline = true;
mytext.text = "This is my first text field object text";
mytext.setTextFormat(myformat);
```

url (propiedad TextFormat.url)

```
public url : String
```

Indica la URL con la que está vinculado el texto con este formato de texto. Si la propiedad `url` es una cadena vacía, el texto no tiene hipervínculo. El valor predeterminado es `null`, que indica que la propiedad no se ha definido.

Disponibilidad: ActionScript 1.0; Flash Player 6

Ejemplo

En el ejemplo siguiente se crea un campo de texto que está vinculado con el sitio Web de Macromedia.

```
var myformat:TextFormat = new TextFormat();
myformat.url = "http://www.macromedia.com";

this.createTextField("mytext",1,100,100,200,100);
mytext.multiline = true;
mytext.wordWrap = true;
mytext.border = true;
mytext.html = true;
mytext.text = "Go to Macromedia.com";
mytext.setTextFormat(myformat);
```

TextRenderer (flash.text.TextRenderer)

```
Object
  |
  +-flash.text.TextRenderer
```

```
public class TextRenderer
extends Object
```

La clase TextRenderer proporciona funciones el suavizado avanzado de fuentes incorporadas. El suavizado avanzado permite representar los tipos de fuentes en tamaño pequeño con una calidad muy elevada. Utilice suavizado avanzado con aplicaciones que tengan gran cantidad de texto pequeño. Macromedia no recomienda el uso de suavizado avanzado para fuentes grandes (más de 48 puntos). El suavizado avanzado sólo se encuentra disponible en Flash Player 8.

Para definir el suavizado avanzado en un campo de texto, establezca la propiedad `antiAliasType` de la instancia de TextField: El ejemplo siguiente requiere una fuente compartida en la biblioteca con un identificador de vinculación denominado `"CustomFont"`.

```
var txtFormat:TextFormat = new TextFormat();
txtFormat.font = "CustomFont";

var label:TextField = this.createTextField("label",
  this.getNextHighestDepth(), 10, 10, 200, 20);
label.setNewTextFormat(txtFormat);
label.text = "Hello World";
label.embedFonts = true;
label.antiAliasType = "advanced";
```

El suavizado avanzado proporciona modulación de trazo continua, que es modulación continua del grosor del trazo y la nitidez del borde. Como es una función avanzada, puede utilizar el método `setAdvancedAntialiasingTable()` para definir la configuración de tipos de letra y tamaños de fuente específicos.

Disponibilidad: ActionScript 1.0; Flash Player 8

Véase también

`antiAliasType (propiedad TextField.antiAliasType)`

Resumen de propiedades

Modificadores	Propiedad	Descripción
`static`	`maxLevel:Number`	Nivel de calidad de campos de distancia con muestreo adaptable (ADF) de suavizado avanzado.

Propiedades heredadas de la clase Object

`constructor (propiedad Object.constructor), __proto__ (propiedad Object.__proto__), prototype (propiedad Object.prototype), __resolve (propiedad Object.__resolve)`

Resumen de métodos

Modificadores	Firma	Descripción
static	setAdvancedAntialiasingTable(fontName:String, fontStyle:String, colorType:String, advancedAntialiasingTable:Array) : Void	Establece una tabla de búsqueda de modulación de trazo continua (CSM) personalizada para una fuente.

Métodos heredados de la clase Object

addProperty (método Object.addProperty), hasOwnProperty (método Object.hasOwnProperty), isPropertyEnumerable (método Object.isPropertyEnumerable), isPrototypeOf (método Object.isPrototypeOf), registerClass (método Object.registerClass), toString (método Object.toString), unwatch (método Object.unwatch), valueOf (método Object.valueOf), watch (método Object.watch)

maxLevel (propiedad TextRenderer.maxLevel)

```
public static maxLevel : Number
```

Nivel de calidad de campos de distancia con muestreo adaptable (ADF) de suavizado avanzado. Los únicos valores aceptables con 3, 4 y 7.

El suavizado avanzado utiliza campos ADF para representar los contornos que delimitan un glifo. Cuanto mayor sea la calidad, más espacio en caché se necesitará para las estructuras ADF. Con un valor de `3` se consume una cantidad mínima de memoria, pero la calidad obtenida es la más baja. Las fuentes más grandes requieren más espacio en caché; con un tamaño de fuente de 64 píxeles, el nivel de calidad aumenta de `3` a `4` o de `4` a `7` a menos que el nivel ya sea `7`.

Disponibilidad: ActionScript 1.0; Flash Player 8

Ejemplo

El ejemplo siguiente especifica el valor `maxLevel` para el archivo SWF completo y, a continuación, muestra un campo de texto con el valor definido. Para que el texto de este ejemplo se muestre correctamente, debe haber un símbolo de fuente disponible con un identificador de vinculación de `"CustomFont"`.

```
import flash.text.TextRenderer;
TextRenderer.maxLevel = 3;
```

```
var txtFormat:TextFormat = new TextFormat();
txtFormat.font = "CustomFont";
txtFormat.size = 64;

var label:TextField = this.createTextField("label",
  this.getNextHighestDepth(), 10, 10, 500, 100);
label.setNewTextFormat(txtFormat);
label.text = "Hello World";
label.embedFonts = true;
trace("TextRenderer.maxLevel: " + TextRenderer.maxLevel);
```

setAdvancedAntialiasingTable (método TextRenderer.setAdvancedAntialiasingTable)

```
public static setAdvancedAntialiasingTable(fontName:String,
  fontStyle:String, colorType:String, advancedAntialiasingTable:Array) :
  Void
```

Establece una tabla de búsqueda de modulación de trazo continua (CSM) personalizada para una fuente. Se trata de un método avanzado.

Flash Player sólo incluye valores de suavizado avanzado para 10 fuentes básicas, y solamente con tamaño de fuente entre 6 y 20. En el caso de estas fuentes, se utiliza el valor 6 para todos los tamaños por debajo de 6, y 20 para todos los tamaño por encima de 20. A los datos de fuentes suministrados se asignan otras fuentes. El método `setAdvancedAntialiasingTable()` permite definir datos de suavizado personalizados para otras fuentes y tamaños de fuente, o bien sustituir los valores predeterminados de las fuentes proporcionadas.

Disponibilidad: ActionScript 1.0; Flash Player 8

Parámetros

`fontName``:String` - El nombre de la fuente que se está configurando.

`fontStyle``:String` - El estilo de fuente puede ser `"bold"`, `"bolditalic"`, `"italic"` y `"none"`.

`colorType``:String` - Este valor puede ser `"dark"` o `"light"`.

`advancedAntialiasingTable``:Array` - Una matriz de valores CSM para la fuente especificada. Cada valor es un objeto con las siguientes propiedades:

- `fontSize`
- `insideCutOff`
- `outsideCutOff`

La matriz `advancedAntialiasingTable` puede contener varias entradas que especifican configuraciones CSM para distintos tamaños de fuentes. (Ver ejemplo.)

`fontSize` es el tamaño, en píxeles, para el que se considera válido la configuración.

El suavizado avanzado utiliza campos de distancia con muestreo adaptable (ADF) para representar los contornos que delimitan un glifo. Macromedia Flash emplea un valor de corte externo `outsideCutOff`), por debajo del cual las densidades se establecen en cero, y un valor de corte interno (`insideCutOff`), por encima del cual las densidades se definen en un valor de densidad máximo (como 255). Entre estos dos valores de corte, la función de asignación es una curva lineal que va desde cero en el corte externo hasta la densidad máxima en el corte interno.

El ajuste de los valores de corte externo e interno afecta al grosor del trazo y a la nitidez del borde. El espaciado entre estos dos parámetros es comparable al doble del radio del filtro de los métodos de suavizado clásicos; un espaciado menor resulta en un borde más nítido, mientras que un espaciado mayor ofrece bordes más suaves y filtrados. Cuando el espaciado es cero, la imagen de densidad resultante es un mapa de bits en dos niveles. Si el espaciado es muy reducido, la imagen de densidad resultante tiene bordes diluidos tipo acuarela.

Por lo general, el usuario prefiere los bordes nítidos muy contrastados en tamaños de punto pequeños, y bordes más suaves para texto animado y tamaños de punto grandes.

El corte externo suele tener un valor negativo, el corte interno un valor positivo y su punto intermedio un valor próximo a cero. El ajuste de estos parámetros para desplazar el punto intermedio hacia un valor infinito negativo aumentará el grosor del trazo; el desplazamiento del punto intermedio hacia un valor infinito positivo reducirá el grosor del trazo. Compruebe que el corte externo tenga siempre un valor menor o igual que el corte interno.

En algunos casos se considera adecuado un exponente gamma equivalente a 1. Sin embargo, en la representación de subpíxeles (modo LCD), el exponente gamma se utiliza para reducir las distorsiones relacionadas con la aberración cromática que pueden aparecer al representar tipos de letra con trazos finos (por ejemplo, Times Roman) en tamaños de punto pequeños. También puede utilizar el exponente gamma para aumentar el contraste en modos Cathode Ray Tube (CRT) y LCD.

Ejemplo

El ejemplo siguiente crea dos entradas de suavizado y dos campos de texto para mostrar su uso. Para que este ejemplo funcione, el archivo SWF debe tener una fuente compartida incorporada con un identificador de vinculación `"myArial"`. Para incorporar la fuente, siga estos pasos:

- Abra la biblioteca.
- Haga clic en el menú de opciones Biblioteca en el ángulo superior derecho de la Biblioteca.
- Seleccione Nueva fuente en el menú emergente.
- Asigne a la fuente el nombre *myArial*.
- Seleccione Arial en el menú emergente de fuentes.
- Haga clic en Aceptar.
- Haga clic con el botón derecho en la fuente que ha creado y seleccione Vinculación.
- Active la casilla de verificación Exportar para ActionScript.
- Haga clic en Aceptar para aceptar el identificador predeterminado, myArial.

```
import flash.text.TextRenderer;

var antiAliasEntry_1 = {fontSize:24, insideCutoff:1.61, outsideCutoff:-
  3.43};
var antiAliasEntry_2 = {fontSize:48, insideCutoff:0.8, outsideCutoff:-0.8};
var arialTable:Array = new Array(antiAliasEntry_1, antiAliasEntry_2);

var lbl_1:TextField = createLabel(0, 0, 300, 100, 24);
var lbl_2:TextField = createLabel(0, 100, 300, 100, 48);

TextRenderer.setAdvancedAntialiasingTable("Arial", "none", "dark",
  arialTable);

function createLabel(x:Number, y:Number, width:Number, height:Number,
  fontSize:Number):TextField {
  var depth:Number = this.getNextHighestDepth();

  var tmpTxt = this.createTextField("txt_" + depth, depth, x, y, width,
  height);
  tmpTxt.antiAliasType = "advanced";
  tmpTxt.gridFitType = "pixel";
  tmpTxt.border = true;
  tmpTxt.text = "Hello World";
  tmpTxt.embedFonts = true;
  tmpTxt.setTextFormat(getTextFormat(fontSize));
  return tmpTxt;
}
```

```
function getTextFormat(fontSize:Number):TextFormat {
  var tf:TextFormat = new TextFormat();
  tf.align = "center";
  tf.size = fontSize;
  tf.font = "myArial";
  return tf;
}
```

TextSnapshot

```
Object
  |
  +-TextSnapshot
```

public class *TextSnapshot*
extends Object

Los objetos TextSnapshot le permiten trabajar con texto estático en un clip de película. Puede utilizar los objetos TextSnapshot, por ejemplo, para disponer el texto con mayor precisión que la que permite el texto dinámico, pero continuar accediendo al texto en modo de sólo lectura.

No es preciso utilizar un constructor para crear un objeto TextSnapshot, ya que lo devuelve `MovieClip.getTextSnapshot()`.

Disponibilidad: ActionScript 1.0; Flash Player 7

Véase también

getTextSnapshot (método MovieClip.getTextSnapshot)

Resumen de propiedades

Propiedades heredadas de la clase Object

constructor (propiedad Object.constructor), __proto__ (propiedad Object.__proto__), prototype (propiedad Object.prototype), __resolve (propiedad Object.__resolve)

Resumen de métodos

Modificadores	Firma	Descripción
	`findText(startIndex:Number, textToFind:String, caseSensitive:Boolean) : Number`	Realiza una búsqueda en el objeto TextSnapshot especificado y devuelve la posición en la que aparece por primera vez `textToFind` en la posición de `startIndex` o después de dicha posición.
	`getCount() : Number`	Devuelve el número de caracteres del objeto TextSnapshot.
	`getSelected(start:Number, [end:Number]) : Boolean`	Devuelve un valor booleano que especifica si un objeto TextSnapshot contiene texto seleccionado en el rango especificado.
	`getSelectedText([includeLineEndings:Boolean]) : String`	Devuelve una cadena que contiene todos los caracteres especificados por el método `TextSnapshot.setSelected()` correspondiente.
	`getText(start:Number, end:Number, [includeLineEndings:Boolean]) : String`	Devuelve una cadena que contiene todos los caracteres especificados por los parámetros `start` y `end`.
	`getTextRunInfo(beginIndex:Number, endIndex:Number) : Array`	Devuelve una matriz de objetos que contiene información acerca de una extensión de texto.
	`hitTestTextNearPos(x:Number, y:Number, [closeDist:Number]) : Number`	Le permite determinar qué carácter de un objeto TextSnapshot está en las coordenadas *x*, *y* especificadas (o cerca de ellas) del clip de película que contiene el texto del objeto TextSnapshot.
	`setSelectColor(color:Number) : Void`	Especifica el color que se va a utilizar al resaltar caracteres seleccionados con el método `TextSnapshot.setSelected()`.
	`setSelected(start:Number, end:Number, select:Boolean) : Void`	Especifica un rango de caracteres de un objeto TextSnapshot que debe seleccionarse o no seleccionarse.

addProperty (método Object.addProperty), hasOwnProperty (método Object.hasOwnProperty), isPropertyEnumerable (método Object.isPropertyEnumerable), isPrototypeOf (método Object.isPrototypeOf), registerClass (método Object.registerClass), toString (método Object.toString), unwatch (método Object.unwatch), valueOf (método Object.valueOf), watch (método Object.watch)

findText (método TextSnapshot.findText)

```
public findText(startIndex:Number, textToFind:String,
  caseSensitive:Boolean) : Number
```

Realiza una búsqueda en el objeto TextSnapshot especificado y devuelve la posición en la que aparece por primera vez `textToFind` en la posición de `startIndex` o después de dicha posición. Si no se encuentra `textToFind`, el método devuelve `-1`.

Disponibilidad: ActionScript 1.0; Flash Player 7

Parámetros

`startIndex``:Number` - Especifica el índice del punto inicial en el texto de TextSnapshot en el que se debe buscar el texto especificado.

`textToFind``:String` - El texto que se va a buscar. Especifique un literal de cadena (entre comillas) o una variable variable.

`caseSensitive``:Boolean` - Valor booleano que especifica si las mayúsculas/minúsculas del texto encontrado deben coincidir con la cadena de `textToFind` (`true`); en caso contrario, `false`.

Valor devuelto

`Number` - La posición de índice basada en cero de la primera vez que aparece el texto especificado, o bien `-1` si no se encuentra ningún texto.

Ejemplo

El ejemplo siguiente ilustra varios modos de utilizar este método. Para utilizar este código, coloque en el escenario un campo de texto estático que contenga el texto "TextSnapshot Example".

```
var my_mc:MovieClip = this;
var my_snap:TextSnapshot = my_mc.getTextSnapshot();
var index1:Number = my_snap.findText(0, "Snap", true);
var index2:Number = my_snap.findText(0, "snap", true);
var index3:Number = my_snap.findText(0, "snap", false);
```

```
trace(index1); // 4
trace(index2); // -1
trace(index3); // 4
```

Véase también

getText (método TextSnapshot.getText)

getCount (método TextSnapshot.getCount)

```
public getCount() : Number
```

Devuelve el número de caracteres del objeto TextSnapshot.

Disponibilidad: ActionScript 1.0; Flash Player 7

Valor devuelto

`Number` - El número de caracteres del objeto TextSnapshot.

Ejemplo

En el ejemplo siguiente se ilustra la forma de obtener el número de caracteres de un objeto TextSnapshot. Para utilizar este código, coloque en el escenario un campo de texto estático que contenga el texto "TextSnapshot Example" (sólo ese texto).

```
var my_mc:MovieClip = this;
var my_snap:TextSnapshot = my_mc.getTextSnapshot();
var count:Number = my_snap.getCount();
var theText:String = my_snap.getText(0, count, false);
trace(count); // 20
trace(theText); // TextSnapshot Example
```

Véase también

getText (método TextSnapshot.getText)

getSelected (método TextSnapshot.getSelected)

```
public getSelected(start:Number, [end:Number]) : Boolean
```

Devuelve un valor booleano que especifica si un objeto TextSnapshot contiene texto seleccionado en el rango especificado.

Para buscar todos los caracteres, pase un valor de `0` para `start` y `TextSnapshot.getCount()` (o un valor muy grande) para `end`. Para buscar un solo carácter, pase al parámetro `end` un valor superior en uno al parámetro `start`.

Disponibilidad: ActionScript 1.0; Flash Player 7

Parámetros

start:Number - La posición de índice del primer carácter que se va a examinar. Los valores válidos para start son desde 0 hasta TextSnapshot.getCount() - 1. Si start es un valor negativo, se utiliza 0.

end:Number [opcional] - La posición de índice que es un valor superior al último valor que se va a examinar. Los valores válidos para end son desde 0 hasta TextSnapshot.getCount(). El carácter indexado por el parámetro end no se incluye en la cadena extraída. Si omite este parámetro, se utiliza TextSnapshot.getCount(). Cuando el valor de end es menor o igual que el valor de start, se utiliza start + 1.

Valor devuelto

Boolean - El valor booleano que indica si al menos un carácter del rango especificado se ha seleccionado mediante el correspondiente método TextSnapshot.setSelected() (true); false en caso contrario.

Ejemplo

El ejemplo siguiente ilustra varios modos de utilizar este método. Para utilizar este código, coloque en el escenario un campo de texto estático que contenga el texto "TextSnapshot Example". En la biblioteca, incluya la fuente utilizada por el campo de texto estático y en las opciones de Vinculación, seleccione Exportar para ActionScript. Añada el siguiente código ActionScript al fotograma 1 de la línea de tiempo:

```
var my_snap:TextSnapshot = this.getTextSnapshot();
var count:Number = my_snap.getCount();
my_snap.setSelected(0, 4, true);
my_snap.setSelected(1, 2, false);

var firstCharIsSelected:Boolean = my_snap.getSelected(0, 1);
var secondCharIsSelected:Boolean = my_snap.getSelected(1, 2);
trace(firstCharIsSelected); // true
trace(secondCharIsSelected); // false
```

Véase también

getCount (método TextSnapshot.getCount), getText (método TextSnapshot.getText), getSelectedText (método TextSnapshot.getSelectedText), setSelected (método TextSnapshot.setSelected)

getSelectedText (método TextSnapshot.getSelectedText)

```
public getSelectedText([includeLineEndings:Boolean]) : String
```

Devuelve una cadena que contiene todos los caracteres especificados por el método `TextSnapshot.setSelected()` correspondiente. Si no hay ningún carácter seleccionado (por el método `TextSnapshot.setSelected()`), se devuelve una cadena vacía.

Si pasa `true` para `includeLineEndings`, se insertan caracteres de nueva línea en la cadena devuelta, y es posible que ésta sea más larga que el rango de entrada. Si `includeLineEndings` es `false` o se omite, el método devuelve el texto seleccionado sin añadir ningún carácter.

Disponibilidad: ActionScript 1.0; Flash Player 7

Parámetros

`includeLineEndings``:Boolean` [opcional] - Un valor booleano que especifica si los caracteres de nueva línea se han insertado (`true`) o no se han insertado (`false`) en la cadena devuelta. El valor predeterminado es `false`.

Valor devuelto

`String` - Devuelve una cadena que contiene todos los caracteres especificados por el método `TextSnapshot.setSelected()` correspondiente.

Ejemplo

El ejemplo siguiente ilustra varios modos de utilizar este método. Para utilizar este código, coloque en el escenario un campo de texto estático que contenga el texto "TextSnapshot Example". A continuación, en la biblioteca, incluya la fuente utilizada por el campo de texto estático y en las opciones de Vinculación, seleccione Exportar para ActionScript. Añada el siguiente código ActionScript al fotograma 1 de la línea de tiempo:

```
var my_snap:TextSnapshot = this.getTextSnapshot();
var count:Number = my_snap.getCount();
my_snap.setSelected(0, 4, true);
my_snap.setSelected(1, 2, false);

var theText:String = my_snap.getSelectedText(false);
trace(theText); // Text
```

Cuando pruebe el archivo SWF, verá un rectángulo de color alrededor de los caracteres especificados.

Véase también

getSelected (método TextSnapshot.getSelected), setSelected (método TextSnapshot.setSelected)

getText (método TextSnapshot.getText)

```
public getText(start:Number, end:Number, [includeLineEndings:Boolean]) :
  String
```

Devuelve una cadena que contiene todos los caracteres especificados por los parámetros `start` y `end`. Si no se especifican caracteres, el método devuelve una cadena vacía.

Para obtener todos los caracteres, pase un valor de `0` para `start` y `TextSnapshot.getCount()` (o un valor muy grande) para `end`. Para devolver un solo carácter, pase el valor `start +1` para `end`.

Si pasa `true` para `includeLineEndings`, se insertan caracteres de nueva línea en la cadena devuelta cuando se considere apropiado, y es posible que ésta sea más larga que el rango de entrada. Si `includeLineEndings` es `false` o se omite, el método devuelve el texto seleccionado sin añadir ningún carácter.

Disponibilidad: ActionScript 1.0; Flash Player 7

Parámetros

`start``:Number` - Un entero que indica la posición del primer carácter que se va a incluir en la cadena devuelta. Los valores válidos para `start` son desde `0` hasta `TextSnapshot.getCount() - 1`. Si `start` es un valor negativo, se utiliza `0`.

`end``:Number` - Un entero equivalente a 1+ el índice del último carácter que se va a examinar en el objeto TextSnapshot. Los valores válidos para `end` son desde `0` hasta `TextSnapshot.getCount()`. El carácter indexado por el parámetro `end` no se incluye en la cadena extraída. Si omite este parámetro, se utiliza `TextSnapshot.getCount()`. Cuando el valor de `end` es menor o igual que el valor de `start`, se utiliza `start + 1`.

`includeLineEndings``:Boolean` [opcional] - Un valor booleano que especifica si los caracteres de nueva línea se han insertado (`true`) o no se han insertado (`false`) en la cadena devuelta. El valor predeterminado es `false`.

Valor devuelto

`String` - Una cadena que contiene los caracteres incluidos en el rango especificado, o una cadena vacía si no se encuentran caracteres en el rango especificado.

Ejemplo

En el ejemplo siguiente se ilustra la forma de obtener el número de caracteres de un objeto TextSnapshot. Para utilizar este código, coloque en el escenario un campo de texto estático que contenga el texto "TextSnapshot Example".

```
var my_mc:MovieClip = this;
var my_snap:TextSnapshot = my_mc.getTextSnapshot();
var count:Number = my_snap.getCount();
var theText:String = my_snap.getText(0, count, false);
trace(count); // 20
trace(theText); // TextSnapshot Example
```

Véase también

`getCount (método TextSnapshot.getCount)`, `getSelectedText (método TextSnapshot.getSelectedText)`

getTextRunInfo (método TextSnapshot.getTextRunInfo)

```
public getTextRunInfo(beginIndex:Number, endIndex:Number) : Array
```

Devuelve una matriz de objetos que contiene información acerca de una extensión de texto. Cada objeto se corresponde con un carácter del rango de caracteres especificado por los dos parámetros del método.

Nota: Si se usa el método `getTextRunInfo()` para un gran rango de texto se puede obtener un objeto grande. Macromedia recomienda limitar el rango de texto definido por los parámetros `beginIndex` y `endIndex`.

Disponibilidad: ActionScript 1.0; Flash Player 7

Parámetros

`beginIndex``:Number` - El valor de índice del primer carácter en el rango de caracteres.

`endIndex``:Number` - El valor de índice del último carácter en el rango de caracteres.

Valor devuelto

`Array` - Una matriz de objetos en la que cada objeto contiene información sobre un carácter determinado del rango especificado. Cada objeto contiene las propiedades siguientes:

- `indexInRun` - Un índice de número entero basado en cero del carácter (en relación a la cadena completa, en lugar de a la extensión de texto seleccionada).
- `selected` - Valor booleano que indica si el carácter está seleccionado `true`; `false`, en caso contrario.

- `font` El nombre de la fuente del carácter.
- `color` El valor de color y alfa combinado del carácter. Los dos primeros dígitos hexadecimales representan el valor alfa y los demás dígitos representan el valor de color. (El ejemplo incluye un método para convertir valores decimales en valores hexadecimales.)
- `height` - Altura del carácter, expresada en píxeles.
- `matrix_a`, `matrix_b`, `matrix_c`, `matrix_d`, `matrix_tx` y `matrix_ty` Los valores de una matriz que define la transformación geométrica en el carácter. El texto normal vertical tiene siempre una matriz `[1 0 0 1 x y]`, donde `x` e `y` equivalen a la posición del carácter en el clip de película principal, independientemente de la altura del texto. La matriz está en el sistema de coordenadas del clip de película principal y no incluye ninguna transformación que pueda estar en el propio clip de película (o en su elemento principal).
- `corner0x`, `corner0y`, `corner1x`, `corner1y`, `corner2x`, `corner2y`, `corner3x` y `corner3y` Las esquinas del cuadro de delimitación del carácter, basadas en el sistema de coordenadas del clip de película principal. Estos valores sólo están disponibles si la fuente que utiliza el carácter está incorporada en el archivo SWF.

Ejemplo

El ejemplo siguiente ilustra varios modos de utilizar este método. Para utilizar este código, cree en el escenario un campo de texto estático que contenga el texto "AB". Gire el campo de texto 45 grados y seleccione que el segundo carácter sea superíndice con un color 0xFFFFFF, alfa 50%, como muestra la figura siguiente:

El siguiente script muestra las propiedades `getTextRunInfo()` de cada carácter del campo de texto:

```
var myTS:TextSnapshot = this.getTextSnapshot();
var myArray:Array = myTS["getTextRunInfo"](0, myTS.getCount());
for (var i = 0; i < myTS.getCount(); i++) {
  trace("indexInRun: " + myArray[i].indexInRun);
  trace("selected: " + myArray[i].selected);
  trace("font: " + myArray[i].font);
  trace("color: " + decToHex(myArray[i].color));
  trace("height: " + myArray[i].height);
  trace("matrix_a: " + myArray[i].matrix_a);
  trace("matrix_b: " + myArray[i].matrix_b);
  trace("matrix_c: " + myArray[i].matrix_c);
  trace("matrix_d: " + myArray[i].matrix_d);
  trace("matrix_ty: " + myArray[i].matrix_tx);
  trace("matrix_tx: " + myArray[i].matrix_ty);
  trace(" ");
}
```

```
function decToHex(dec:Number) {
  var hexString:String = "";
  if (dec > 15) {
    hexString = decToHex(Math.floor(dec / 16));
  }
  var hexDigit = dec - 16 * (Math.floor(dec / 16));
    if (hexDigit > 9) {
      hexDigit = String.fromCharCode(hexDigit + 55);
    }
    hexString = hexString + hexDigit;
    return hexString;
}
```

Se genera el siguiente resultado:

```
indexInRun: 0
selected: false
font: Times New Roman
color: FF000000
height: 28.6
matrix_a: 0.0316612236983293
matrix_b: 0.0385940558426864
matrix_c: -0.0385940558426864
matrix_d: 0.0316612236983293
matrix_ty: 22.75
matrix_tx: 40.35

indexInRun: 0
selected: false
font: Times New Roman
color: 80000000
height: 28.6
matrix_a: 0.0316612236983293
matrix_b: 0.0385940558426864
matrix_c: -0.0385940558426864
matrix_d: 0.0316612236983293
matrix_ty: 49
matrix_tx: 45.5
```

Este ejemplo utiliza un método `decToHex()` para convertir el valor decimal de la propiedad `color` en un valor hexadecimal.

Véase también

Matrix (flash.geom.Matrix)

hitTestTextNearPos (método TextSnapshot.hitTestTextNearPos)

```
public hitTestTextNearPos(x:Number, y:Number, [closeDist:Number]) : Number
```

Le permite determinar qué carácter de un objeto TextSnapshot está en las coordenadas *x*, *y* especificadas (o cerca de ellas) del clip de película que contiene el texto del objeto TextSnapshot.

Si omite o pasa el valor `0` para `closeDist`, la ubicación especificada por las coordenadas *x*, *y* deberá encontrarse dentro del recuadro de delimitación del objeto TextSnapshot.

Este método sólo funciona correctamente con fuentes que incluyen información de medidas de caracteres; sin embargo, de manera predeterminada, Macromedia Flash no incluye esta información para campos de texto estáticos. Por lo tanto, el método puede devolver `-1` en lugar de un valor de índice. Para asegurarse de que se devuelve un valor de índice, puede forzar a la herramienta de edición de Flash a que incluya información de medidas de caracteres para una fuente. Para ello, añada un campo de texto dinámico que utilice dicha fuente, seleccione Opciones de caracteres para dicho campo de texto dinámico y luego especifique que los contornos de fuentes deben incorporarse al menos para un carácter. (No importa qué caracteres especifique ni tampoco si son los caracteres utilizados en los campos de texto estáticos.)

Disponibilidad: ActionScript 1.0; Flash Player 7

Parámetros

x`:Number` - La coordenada *x* del clip de película que contiene el texto en el objeto TextSnapshot.

y`:Number` - La coordenada *y* del clip de película que contiene el texto en el objeto TextSnapshot.

closeDist`:Number` [opcional] - La distancia máxima desde *x*, *y* en la que se puede buscar texto. La distancia se mide desde el punto central de cada carácter. El valor predeterminado es `0`.

Valor devuelto

`Number` - El valor de índice del carácter del objeto TextSnapshot situado más cerca de la coordenada *x*, *y* especificada. El método devuelve `-1` si no se encuentra ningún carácter o si la fuente no contiene información de medidas de caracteres.

Ejemplo

El ejemplo siguiente ilustra varios modos de utilizar este método. Para utilizar este código, coloque en el escenario un campo de texto estático que contenga el texto "TextSnapshot Example". En la biblioteca, incluya la fuente utilizada por el campo de texto estático y en las opciones de Vinculación, seleccione Exportar para ActionScript. Para probar el código, ejecute el archivo SWF y sitúe el puntero del ratón en el texto en pantalla.

```
var my_ts:TextSnapshot = getTextSnapshot();
this.onMouseMove = function() {
  var hitIndex:Number = my_ts.hitTestTextNearPos(_xmouse, _ymouse, 0);
  my_ts.setSelected(0, my_ts.getCount(), false);
  if (hitIndex >= 0) {
    my_ts.setSelected(hitIndex, hitIndex + 1, true);
  }
};
```

Véase también

getTextSnapshot (método MovieClip.getTextSnapshot), _x (propiedad MovieClip._x), _y (propiedad MovieClip._y)

setSelectColor (método TextSnapshot.setSelectColor)

```
public setSelectColor(color:Number) : Void
```

Especifica el color que se va a utilizar al resaltar caracteres seleccionados con el método `TextSnapshot.setSelected()`. El color es siempre opaco; no puede especificar un valor de transparencia.

Este método sólo funciona correctamente con fuentes que incluyen información de medidas de caracteres; sin embargo, de manera predeterminada, Macromedia Flash no incluye esta información para campos de texto estáticos. Por lo tanto, el método puede devolver `-1` en lugar de un valor de índice. Para asegurarse de que se devuelve un valor de índice, puede forzar a la herramienta de edición de Flash a que incluya información de medidas de caracteres para una fuente. Para ello, añada un campo de texto dinámico que utilice dicha fuente, seleccione Opciones de caracteres para dicho campo de texto dinámico y luego especifique que los contornos de fuentes deben incorporarse al menos para un carácter. (No importa qué caracteres especifique ni tampoco si son los caracteres utilizados en los campos de texto estáticos.)

Disponibilidad: ActionScript 1.0; Flash Player 7

Parámetros

color:Number - El color utilizado en el borde que rodea los caracteres que se han seleccionado mediante el método TextSnapshot.setSelected() correspondiente, expresado en formato 0x*RRGGBB*.

Ejemplo

El ejemplo siguiente ilustra varios modos de utilizar este método. Para utilizar este código, coloque en el escenario un campo de texto estático que contenga el texto "TextSnapshot Example". En la biblioteca, incluya la fuente utilizada por el campo de texto estático y en las opciones de Vinculación, seleccione Exportar para ActionScript. Añada el siguiente código ActionScript al fotograma 1 de la línea de tiempo:

```
var my_snap:TextSnapshot = this.getTextSnapshot();
var count:Number = my_snap.getCount();
my_snap.setSelectColor(0xFF0000);
my_snap.setSelected(0, 4, true);
my_snap.setSelected(1, 2, false);

var theText:String = my_snap.getSelectedText(false); // get the selected
  text
trace(theText); // Text
```

Cuando pruebe el archivo SWF, verá un rectángulo de color alrededor de los caracteres especificados.

Véase también

setSelected (método TextSnapshot.setSelected)

setSelected (método TextSnapshot.setSelected)

```
public setSelected(start:Number, end:Number, select:Boolean) : Void
```

Especifica un rango de caracteres de un objeto TextSnapshot que debe seleccionarse o no seleccionarse. Los caracteres seleccionados se dibujan con un rectángulo de color detrás de ellos que coincide con el recuadro de delimitación del carácter. El color del recuadro de delimitación se define mediante TextSnapshot.setSelectColor().

Para seleccionar todos los caracteres, pase un valor de 0 para start y TextSnapshot.getCount() (o un valor muy grande) para end. Para especificar un solo carácter, pase el valor start + 1 para end.

Dado que los caracteres se marcan individualmente como seleccionados, puede llamar a este método varias veces para seleccionar varios caracteres; es decir, el uso de este método no anula la selección de otros caracteres establecidos por este método.

Este método sólo funciona correctamente con fuentes que incluyen información de medidas de caracteres; de manera predeterminada, Flash no incluye esta información para campos de texto estáticos. Por lo tanto, es posible que el texto seleccionado no parezca seleccionado en la pantalla. Para asegurarse de que todo el texto seleccionado aparezca como seleccionado, puede forzar a la herramienta de edición de Flash a que incluya información de medidas de caracteres para una fuente. Para ello, en la biblioteca, incluya la fuente utilizada por el campo de texto estático y en las opciones de Vinculación, seleccione Exportar para ActionScript.

Disponibilidad: ActionScript 1.0; Flash Player 7

Parámetros

start:Number - El número del primer carácter que se va a seleccionar. Los valores válidos para start son desde 0 hasta TextSnapshot.getCount() - 1. Si start es un valor negativo, se utiliza 0.

end:Number - Un entero equivalente a 1+ el índice del último carácter que se va a examinar. Los valores válidos para end son desde 0 hasta TextSnapshot.getCount(). El carácter indexado por el parámetro end no se incluye en la cadena extraída. Si omite este parámetro, se utiliza TextSnapshot.getCount(). Cuando el valor de end es menor o igual que el valor de start, se utiliza start + 1.

select:Boolean - Un valor booleano que especifica si el texto debe seleccionarse (true) o no (false).

Ejemplo

El ejemplo siguiente ilustra varios modos de utilizar este método. Para utilizar este código, coloque en el escenario un campo de texto estático que contenga el texto "TextSnapshot Example". En la biblioteca, incluya la fuente utilizada por el campo de texto estático y en las opciones de Vinculación, seleccione Exportar para ActionScript. Añada el siguiente código ActionScript al fotograma 1 de la línea de tiempo:

```
var my_snap:TextSnapshot = this.getTextSnapshot();
var count:Number = my_snap.getCount();
my_snap.setSelected(0, 4, true);
my_snap.setSelected(1, 2, false);

var theText:String = my_snap.getSelectedText(false);
trace(theText); // Text
```

Véase también

getCount (método TextSnapshot.getCount)

Transform (flash.geom.Transform)

```
Object
  |
  +-flash.geom.Transform
```

```
public class Transform
extends Object
```

La clase Transform recopila datos acerca de las transformaciones de color y manipulaciones de coordenadas que se aplican a un objeto MovieClip.

Normalmente los objetos Transform se obtienen a través del valor de la propiedad `transform` de un objeto MovieClip.

Disponibilidad: ActionScript 1.0; Flash Player 8

Véase también

`transform (propiedad MovieClip.transform)`, `ColorTransform (flash.geom.ColorTransform)`, `Matrix (flash.geom.Matrix)`

Resumen de propiedades

Modificadores	Propiedad	Descripción
	`colorTransform:ColorTransform`	Objeto ColorTransform que contiene valores que ajustan globalmente los colores en el clip de película.
	`concatenatedColorTransform:ColorTransform` [read-only]	Objeto ColorTransform que representa las transformaciones de color aplicadas a este objeto y todas las de sus objetos principales hasta el nivel de la raíz, combinadas.
	`concatenatedMatrix:Matrix` [read-only]	Objeto Matrix que representa las matrices de transformación aplicadas a este objeto y todas las de sus objetos principales hasta el nivel de la raíz, combinadas.
	`matrix:Matrix`	Objeto Matrix de transformación que contiene valores que afectan al ajuste de tamaño, rotación y conversión del clip de película.
	`pixelBounds:Rectangle`	Un objeto Rectangle que define el rectángulo de delimitación del objeto MovieClip en el escenario.

Propiedades heredadas de la clase Object

constructor (propiedad Object.constructor), __proto__ (propiedad Object.__proto__), prototype (propiedad Object.prototype), __resolve (propiedad Object.__resolve)

Resumen de constructores

Firma	Descripción
Transform(mc:MovieClip)	Crea un nuevo objeto Transform asociado con el objeto MovieClip especificado.

Resumen de métodos

Métodos heredados de la clase Object

addProperty (método Object.addProperty), hasOwnProperty (método Object.hasOwnProperty), isPropertyEnumerable (método Object.isPropertyEnumerable), isPrototypeOf (método Object.isPrototypeOf), registerClass (método Object.registerClass), toString (método Object.toString), unwatch (método Object.unwatch), valueOf (método Object.valueOf), watch (método Object.watch)

colorTransform (propiedad Transform.colorTransform)

```
public colorTransform : ColorTransform
```

Objeto ColorTransform que contiene valores que ajustan globalmente los colores en el clip de película.

Disponibilidad: ActionScript 1.0; Flash Player 8

Ejemplo

El ejemplo siguiente aplica el objeto ColorTransform `blueColorTransform` al objeto Transform `trans`. Este ColorTransform convierte el color del clip de película `rect` de rojo a azul.

```
import flash.geom.Transform;
import flash.geom.ColorTransform;

var rect:MovieClip = createRectangle(20, 80, 0xFF0000);

var trans:Transform = new Transform(rect);
trace(trans.colorTransform);
```

```
// (redMultiplier=1, greenMultiplier=1, blueMultiplier=1,
  alphaMultiplier=1, redOffset=0, greenOffset=0, blueOffset=0,
  alphaOffset=0)

var blueColorTransform:ColorTransform = new ColorTransform(0, 1, 1, 1, 0,
  0, 255, 0);

rect.onPress = function() {
  trans.colorTransform = blueColorTransform;
  trace(trans.colorTransform);
  // (redMultiplier=0, greenMultiplier=1, blueMultiplier=1,
  alphaMultiplier=1, redOffset=0, greenOffset=0, blueOffset=255,
  alphaOffset=0)
}

function createRectangle(width:Number, height:Number, color:Number,
  scope:MovieClip):MovieClip {
  scope = (scope == undefined) ? this : scope;
  var depth:Number = scope.getNextHighestDepth();
  var mc:MovieClip = scope.createEmptyMovieClip("mc_" + depth, depth);
    mc.beginFill(color);
  mc.lineTo(0, height);
  mc.lineTo(width, height);
  mc.lineTo(width, 0);
  mc.lineTo(0, 0);
  return mc;
}
```

Véase también

ColorTransform (flash.geom.ColorTransform)

concatenatedColorTransform (propiedad Transform.concatenatedColorTransform)

```
public concatenatedColorTransform : ColorTransform [read-only]
```

Objeto ColorTransform que representa las transformaciones de color aplicadas a este objeto y todas las de sus objetos principales hasta el nivel de la raíz, combinadas. Si se han aplicado transformaciones de color diferentes en niveles distintos, para esta propiedad se concatenarán todas en un objeto ColorTransform.

Disponibilidad: ActionScript 1.0; Flash Player 8

Ejemplo

El siguiente ejemplo se aplica dos objetos Transform a un objeto MovieClip principal y a uno secundario. A continuación, se aplica una variable `blueColorTransform` al objeto Transform `parentTrans`, que ajusta el color de los objetos MovieClip principal y secundario al azul. Puede ver cómo `child.concatenatedColorTransform` es la combinación de `parentTrans` y `childTrans`.

```
import flash.geom.Transform;
import flash.geom.ColorTransform;

var parentRect:MovieClip = createRectangle(20, 80, 0xFF0000);
var childRect:MovieClip = createRectangle(10, 40, 0x00FF00, parentRect);

var parentTrans:Transform = new Transform(parentRect);
var childTrans:Transform = new Transform(childRect);

var blueColorTransform:ColorTransform = new ColorTransform(0, 1, 1, 1, 0,
  0, 255, 0);

parentTrans.colorTransform = blueColorTransform;

trace(childTrans.concatenatedColorTransform);
// (redMultiplier=0, greenMultiplier=1, blueMultiplier=1,
  alphaMultiplier=1, redOffset=0, greenOffset=0, blueOffset=255,
  alphaOffset=0)
trace(childTrans.colorTransform);
// (redMultiplier=1, greenMultiplier=1, blueMultiplier=1,
  alphaMultiplier=1, redOffset=0, greenOffset=0, blueOffset=0,
  alphaOffset=0)
trace(parentTrans.concatenatedColorTransform);
// (redMultiplier=0, greenMultiplier=1, blueMultiplier=1,
  alphaMultiplier=1, redOffset=0, greenOffset=0, blueOffset=255,
  alphaOffset=0)

function createRectangle(width:Number, height:Number, color:Number,
  scope:MovieClip):MovieClip {
  scope = (scope == undefined) ? this : scope;
  var depth:Number = scope.getNextHighestDepth();
  var mc:MovieClip = scope.createEmptyMovieClip("mc_" + depth, depth);
  mc.beginFill(color);
  mc.lineTo(0, height);
  mc.lineTo(width, height);
  mc.lineTo(width, 0);
  mc.lineTo(0, 0);
  return mc;
}
```

Véase también

ColorTransform (flash.geom.ColorTransform)

concatenatedMatrix (propiedad Transform.concatenatedMatrix)

```
public concatenatedMatrix : Matrix [read-only]
```

Objeto Matrix que representa las matrices de transformación aplicadas a este objeto y todas las de sus objetos principales hasta el nivel de la raíz, combinadas. Si se han aplicado matrices de transformación diferentes en niveles distintos, para esta propiedad se concatenarán todas en un objeto Matrix.

Disponibilidad: ActionScript 1.0; Flash Player 8

Ejemplo

El siguiente ejemplo se aplica dos objetos Transform a un objeto MovieClip principal y a uno secundario. A continuación se aplica `scaleMatrix` al objeto Transform `parentTrans`, que ajusta la escala de los objetos MovieClip principal y secundario. Puede ver cómo `child.concatenatedMatrix` es la combinación de `parentTrans` y `childTrans`.

```
import flash.geom.Transform;
import flash.geom.Matrix;

var parentRect:MovieClip = createRectangle(20, 80, 0xFF0000);
var childRect:MovieClip = createRectangle(10, 40, 0x00FF00, parentRect);

var parentTrans:Transform = new Transform(parentRect);
var childTrans:Transform = new Transform(childRect);

var scaleMatrix:Matrix = new Matrix();
scaleMatrix.scale(2, 2);

parentTrans.matrix = scaleMatrix;

trace(childTrans.concatenatedMatrix); // (a=2, b=0, c=0, d=2, tx=0, ty=0)
trace(childTrans.matrix); // (a=1, b=0, c=0, d=1, tx=0, ty=0)
trace(parentTrans.concatenatedMatrix); // (a=2, b=0, c=0, d=2, tx=0, ty=0)

function createRectangle(width:Number, height:Number, color:Number,
  scope:MovieClip):MovieClip {
  scope = (scope == undefined) ? this : scope;
  var depth:Number = scope.getNextHighestDepth();
  var mc:MovieClip = scope.createEmptyMovieClip("mc_" + depth, depth);
  mc.beginFill(color);
  mc.lineTo(0, height);
  mc.lineTo(width, height);
```

```
  mc.lineTo(width, 0);
  mc.lineTo(0, 0);
  return mc;
}
```

matrix (propiedad Transform.matrix)

```
public matrix : Matrix
```

Objeto Matrix de transformación que contiene valores que afectan al ajuste de tamaño, rotación y conversión del clip de película.

Disponibilidad: ActionScript 1.0; Flash Player 8

Ejemplo

El ejemplo siguiente aplica el objeto Matrix `scaleMatrix` al objeto Transform `trans`. Este objeto Matrix asigna al objeto MovieClip `rect` una escala de factor dos.

```
import flash.geom.Transform;
import flash.geom.Matrix;

var rect:MovieClip = createRectangle(20, 80, 0xFF0000);

var trans:Transform = new Transform(rect);
trace(trans.matrix); // (a=1, b=0, c=0, d=1, tx=0, ty=0)

var scaleMatrix:Matrix = new Matrix();
scaleMatrix.scale(2, 2);

rect.onPress() = function() {
  trans.matrix = scaleMatrix;
  trace(trans.matrix); // (a=2, b=0, c=0, d=2, tx=0, ty=0)
}

function createRectangle(width:Number, height:Number, color:Number,
  scope:MovieClip):MovieClip {
  scope = (scope == undefined) ? this : scope;
  var depth:Number = scope.getNextHighestDepth();
  var mc:MovieClip = scope.createEmptyMovieClip("mc_" + depth, depth);
  mc.beginFill(color);
  mc.lineTo(0, height);
  mc.lineTo(width, height);
  mc.lineTo(width, 0);
  mc.lineTo(0, 0);
  return mc;
}
```

Véase también

Matrix (flash.geom.Matrix)

pixelBounds (propiedad Transform.pixelBounds)

```
public pixelBounds : Rectangle
```

Un objeto Rectangle que define el rectángulo de delimitación del objeto MovieClip en el escenario.

Disponibilidad: ActionScript 1.0; Flash Player 8

Ejemplo

El ejemplo siguiente crea un objeto Transform `trans` y controla su propiedad `pixelBounds`. Observe que `pixelBounds` devuelve un cuadro de delimitación con valores iguales a los métodos `getBounds()` y `getRect()` del objeto MovieClip.

```
import flash.geom.Transform;

var rect:MovieClip = createRectangle(20, 80, 0xFF0000);
var trans:Transform = new Transform(rect);
trace(trans.pixelBounds); // (x=0, y=0, w=20, h=80)

var boundsObj:Object = rect.getBounds();
trace(boundsObj.xMin); // 0
trace(boundsObj.yMin); // 0
trace(boundsObj.xMax); // 20
trace(boundsObj.yMax); // 80

var rectObj:Object = rect.getRect();
trace(rectObj.xMin); // 0
trace(rectObj.yMin); // 0
trace(rectObj.xMax); // 20
trace(rectObj.yMax); // 80

function createRectangle(width:Number, height:Number, color:Number,
  scope:MovieClip):MovieClip {
  scope = (scope == undefined) ? this : scope;
  var depth:Number = scope.getNextHighestDepth();
  var mc:MovieClip = scope.createEmptyMovieClip("mc_" + depth, depth);
  mc.beginFill(color);
  mc.lineTo(0, height);
  mc.lineTo(width, height);
  mc.lineTo(width, 0);
  mc.lineTo(0, 0);
  return mc;
}
```

Constructor Transform

```
public Transform(mc:MovieClip)
```

Crea un nuevo objeto Transform asociado con el objeto MovieClip especificado.

Una vez creado, el nuevo objeto Transform puede recuperarse a través de la propiedad `transform` del objeto MovieClip en cuestión.

Disponibilidad: ActionScript 1.0; Flash Player 8

Parámetros

mc`:MovieClip` - El objeto MovieClip al que se aplica el nuevo objeto Transform.

Ejemplo

El ejemplo siguiente crea el objeto Transform `trans` y lo aplica al objeto MovieClip rect. Observará que `trans` y `rect.transform` del objeto Transform no dan el mismo resultado aunque contienen los mismos valores.

```
import flash.geom.Transform;

var rect:MovieClip = createRectangle(20, 80, 0xFF0000);

var trans:Transform = new Transform(rect);

trace(rect.transform == trans); // false

for(var i in trans) {
  trace(">> " + i + ": " + trans[i]);
  // >> pixelBounds: (x=0, y=0, w=20, h=80)
  // >> concatenatedColorTransform: (redMultiplier=1, greenMultiplier=1,
  blueMultiplier=1, alphaMultiplier=1, redOffset=0, greenOffset=0,
  blueOffset=0, alphaOffset=0)
  // >> colorTransform: (redMultiplier=1, greenMultiplier=1,
  blueMultiplier=1, alphaMultiplier=1, redOffset=0, greenOffset=0,
  blueOffset=0, alphaOffset=0)
  // >> concatenatedMatrix: (a=1, b=0, c=0, d=1, tx=0, ty=0)
  // >> matrix: (a=1, b=0, c=0, d=1, tx=0, ty=0)
}

for(var i in rect.transform) {
  trace(">> " + i + ": " + rect.transform[i]);
  // >> pixelBounds: (x=0, y=0, w=20, h=80)
  // >> concatenatedColorTransform: (redMultiplier=1, greenMultiplier=1,
  blueMultiplier=1, alphaMultiplier=1, redOffset=0, greenOffset=0,
  blueOffset=0, alphaOffset=0)
  // >> colorTransform: (redMultiplier=1, greenMultiplier=1,
  blueMultiplier=1, alphaMultiplier=1, redOffset=0, greenOffset=0,
  blueOffset=0, alphaOffset=0)
```

```
  // >> concatenatedMatrix: (a=1, b=0, c=0, d=1, tx=0, ty=0)
  // >> matrix: (a=1, b=0, c=0, d=1, tx=0, ty=0)
}

function createRectangle(width:Number, height:Number, color:Number,
  scope:MovieClip):MovieClip {
  scope = (scope == undefined) ? this : scope;
  var depth:Number = scope.getNextHighestDepth();
  var mc:MovieClip = scope.createEmptyMovieClip("mc_" + depth, depth);
  mc.beginFill(color);
  mc.lineTo(0, height);
  mc.lineTo(width, height);
  mc.lineTo(width, 0);
  mc.lineTo(0, 0);
  return mc;
}
```

Video

```
Object
  |
  +-Video
```

public class *Video*
extends Object

La clase Video permite mostrar un flujo de vídeo en vivo en el escenario sin incorporarlo al archivo SWF. Puede capturar el vídeo utilizando `Camera.get()`. En archivos publicados para Flash Player 7 y versiones posteriores, también puede utilizar la clase Video para reproducir archivos de Flash Video (FLV) a través de HTTP o desde el sistema de archivos local. Para más información, consulte las entradas de las clases NetConnection y NetStream.

Flash Player 7 admite Flash video (FLV) codificado con códec de vídeo Sorenson Spark. Flash Player 8 admite vídeo Flash (FLV) codificado con códec Sorenson o On2 VP6 y también admite un canal alfa. El códec de vídeo On2 VP6 utiliza menos ancho de banda que tecnologías más antiguas y además ofrece filtros de desbloqueo y reducción de estelas.

Si su contenido de Flash carga dinámicamente vídeo de Flash (mediante la descarga progresiva o Flash Communication Server), puede utilizar vídeo On2 VP6 sin necesidad de volver a publicar su SWF para Flash Player 8, siempre que sus usuarios vean el contenido con Flash Player 8. Implementado mediante flujo o descarga el vídeo On2 VP6 en las versiones 6 y 7 de Flash SWF y reproduciendo el contenido con Flash Player 8, evitará la necesidad de recrear los archivos SWF.

Códec	Contenido (SWF)Versión(versión publicada)	Versión de Flash Player(versión necesaria para reproducción)
Sorenson Spark	6	6, 7, 8
	7	7, 8
On2 VP6	6	8*
	7	8
	8	8

* Si su contenido de Flash carga vídeo Flash (FLV) de forma dinámica, puede utilizar vídeo On2 VP6 sin necesidad de volver a publicar su SWF para Flash Player 8, siempre que los usuarios utilicen Flash Player 8 para ver el contenido. Sólo Flash Player 8 permite publicar y reproducir vídeo On2 VP6.

Un objeto Video puede utilizarse como un clip de película. Al igual que otros objetos que usted sitúa en el escenario, puede controlar diversas propiedades de los objetos Video. Por ejemplo, puede desplazar el objeto Video por el escenario utilizando sus propiedades `_x` e `_y`, puede cambiar su tamaño utilizando sus propiedades `_height` y `_width`, etc.

Para mostrar el flujo de vídeo, sitúe primero un objeto Video en el escenario. A continuación utilice `Video.attachVideo()` para asociar el flujo de vídeo al objeto de vídeo.

- Si el panel Biblioteca no está visible, seleccione Ventana > Biblioteca para mostrarlo.
- Añada un objeto Video incorporado a la biblioteca haciendo clic en el menú Opciones de la parte derecha de la barra de título del panel Biblioteca y seleccionando Nuevo vídeo.
- Arrastre el objeto Video hasta el escenario y utilice el inspector de propiedades para asignarle un nombre de instancia exclusivo, como, por ejemplo,`my_video`. (No le asigne el nombre Video.)

Disponibilidad: ActionScript 1.0; Flash Player 6

Véase también

`NetConnection`, `NetStream`

Resumen de propiedades

Modificadores	Propiedad	Descripción
	`_alpha:Number`	Indica el valor de transparencia alfa del objeto Video especificado.
	`deblocking:Number`	Indica el tipo de filtro de desbloqueo aplicado al vídeo descodificado como parte del proceso posterior.
	`_height:Number`	Indica la altura del objeto Video, expresada en píxeles.
	`height:Number` [read-only]	Un entero que especifica la altura del flujo de vídeo en píxeles.
	`_name:String`	Indica el nombre de instancia del objeto Video especificado.
	`_parent:MovieClip`	Indica el clip de película o el objeto que contiene el objeto Video actual.
	`_rotation:Number`	Indica el giro del objeto Video, expresado en grados, con respecto a su orientación original.
	`smoothing:Boolean`	Especifica si el vídeo debe suavizarse (interpolarse) al aplicarle una escala.
	`_visible:Boolean`	Indica si el objeto Video especificado por *`my_video`* es visible.
	`_width:Number`	Indica la anchura del objeto Video, expresada en píxeles.
	`width:Number` [read-only]	Un entero que especifica la anchura del flujo de vídeo en píxeles.
	`_x:Number`	Indica la coordenada x de un objeto Video en relación a las coordenadas locales del clip de película principal.
	`_xmouse:Number` [read-only]	Indica la coordenada *x* de la posición del ratón.
	`_xscale:Number`	Indica la escala horizontal (*`percentage`*) del objeto Video aplicada desde el punto de registro del objeto Video.
	`_y:Number`	Indica la coordenada *y* de un objeto Video en relación a las coordenadas locales del clip de película principal.
	`_ymouse:Number` [read-only]	Indica la coordenada *y* de la posición del ratón.
	`_yscale:Number`	Indica la escala vertical (*`percentage`*) del objeto Video aplicada desde el punto de registro del objeto Video.

Propiedades heredadas de la clase Object

```
constructor (propiedad Object.constructor), __proto__ (propiedad
Object.__proto__), prototype (propiedad Object.prototype), __resolve
(propiedad Object.__resolve)
```

Resumen de métodos

Modificadores	Firma	Descripción
	`attachVideo(source:Object) : Void`	Especifica un flujo de vídeo (*source*) para que se muestre dentro de los límites del objeto Video del escenario.
	`clear() : Void`	Borra la imagen actualmente mostrada en el objeto Video.

Métodos heredados de la clase Object

```
addProperty (método Object.addProperty), hasOwnProperty (método
Object.hasOwnProperty), isPropertyEnumerable (método
Object.isPropertyEnumerable), isPrototypeOf (método Object.isPrototypeOf),
registerClass (método Object.registerClass), toString (método
Object.toString), unwatch (método Object.unwatch), valueOf (método
Object.valueOf), watch (método Object.watch)
```

_alpha (propiedad Video._alpha)

```
public _alpha : Number
```

Indica el valor de transparencia alfa del objeto Video especificado. Los valores válidos son los comprendidos entre 0 (totalmente transparente) y 100 (totalmente opaco). El valor predeterminado es 100. Los objetos existentes en un clip de película que tenga configurado `_alpha` con el valor 0 continuarán activos aunque no sean visibles.

Disponibilidad: ActionScript 1.0; Flash Player 8

Véase también

`_visible (propiedad Video._visible)`

attachVideo (método Video.attachVideo)

```
public attachVideo(source:Object) : Void
```

Especifica un flujo de vídeo (*source*) para que se muestre dentro de los límites del objeto Video del escenario. El flujo de vídeo es un archivo FLV que se está mostrando mediante el comando `NetStream.play()`, un objeto Camera o `null`. Si *source* tiene el valor `null`, dejará de reproducirse vídeo en el objeto Video.

No es preciso que utilice este método si el archivo FLV sólo contiene audio; la parte de audio de los archivos FLV se reproduce automáticamente cuando se envía el comando `NetStream.play()`.

Si desea controlar el audio asociado a un archivo FLV, puede utilizar `MovieClip.attachAudio()` para dirigir el audio a un clip de película; posteriormente, podrá crear un objeto Sound para controlar algunos aspectos del audio. Para más información, consulte `MovieClip.attachAudio()`.

Disponibilidad: ActionScript 1.0; Flash Player 6

Parámetros

source:`Object` - Un objeto Camera que captura datos de vídeo o un objeto NetStream. Para perder la conexión al objeto Video, pase `null` para *source* .

Ejemplo

El ejemplo siguiente reproduce vídeo en vivo de forma local:

```
var my_video:Video; //my_video is a Video object on the Stage
var active_cam:Camera = Camera.get();
my_video.attachVideo(active_cam);
```

El siguiente ejemplo reproduce un archivo grabado anteriormente con el nombre myVideo.flv almacenado en el mismo directorio que el archivo SWF.

```
var my_video:Video; // my_video is a Video object on the Stage
var my_nc:NetConnection = new NetConnection();
my_nc.connect(null);
var my_ns:NetStream = new NetStream(my_nc);
my_video.attachVideo(my_ns);
my_ns.play("video1.flv");
```

Véase también

`Camera`, `NetStream`

clear (método Video.clear)

```
public clear() : Void
```

Borra la imagen actualmente mostrada en el objeto Video. Esto resulta útil, por ejemplo, cuando se desea mostrar información de espera sin tener que ocultar el objeto Video.

Disponibilidad: ActionScript 1.0; Flash Player 6

Ejemplo

El ejemplo siguiente detiene y borra video1.flv que se está reproduciendo en un objeto Video (denominado `my_video`) cuando el usuario hace clic en la instancia `pause_btn`.

```
var pause_btn:Button;
var my_video:Video; // my_video is a Video object on the Stage
var my_nc:NetConnection = new NetConnection();
my_nc.connect(null);
var my_ns:NetStream = new NetStream(my_nc);
my_video.attachVideo(my_ns);
my_ns.play("video1.flv");
pause_btn.onRelease = function() {
  my_ns.pause();
  my_video.clear();
};
```

Véase también

`attachVideo (método Video.attachVideo)`

deblocking (propiedad Video.deblocking)

```
public deblocking : Number
```

Indica el tipo de filtro de desbloqueo aplicado al vídeo descodificado como parte del proceso posterior. Hay dos filtros de desbloqueo disponibles: uno en códec Sorenson y uno en códec On2 VP6. Los valores siguientes son aceptables:

- 0 (predeterminado)-Permite que el compresor de vídeo aplique el filtro de desbloqueo si es preciso.
- 1-No utilizar ningún filtro de desbloqueo.
- 2-Utilizar el filtro de desbloqueo Sorenson.
- 3-Utilizar el filtro de desbloqueo On2 y no utilizar el filtro de reducción de estelas.
- 4-Utilizar el filtro de desbloqueo de On2 y de reducción de estelas de On2 rápido.
- 5-Utilizar el filtro de desbloqueo de On2 y de reducción de estelas de On2 mejor.
- 6-Igual que la opción 5.
- 7-Igual que la opción 5.

Si se selecciona un modo superior a 2 para vídeo con el códec Sorenson, el descodificador Sorenson usa de forma predeterminada el modo 0 internamente.

El uso del filtro de desbloqueo tiene un efecto global en el rendimiento de la reproducción y normalmente no es necesario para vídeo de banda ancha. Si su sistema no es lo suficientemente potente, es posible que tenga dificultades para reproducir vídeo con este filtro activado.

Disponibilidad: ActionScript 1.0; Flash Player 6

Ejemplo

El ejemplo siguiente reproduce video1.flv en el objeto `my_video` y permite al usuario cambiar el comportamiento del filtro de desbloqueo en video1.flv. Añada un objeto de vídeo denominado `my_video` y una instancia de ComboBox con el nombre `deblocking_cb` a su archivo, y, a continuación, añada el siguiente código ActionScript al archivo FLA o AS.

```
var deblocking_cb:mx.controls.ComboBox;
var my_video:Video; // my_video is a Video object on the Stage
var my_nc:NetConnection = new NetConnection();
my_nc.connect(null);
var my_ns:NetStream = new NetStream(my_nc);
my_video.attachVideo(my_ns);
my_ns.play("video1.flv");

deblocking_cb.addItem({data:0, label:'Auto'});
deblocking_cb.addItem({data:1, label:'No'});
deblocking_cb.addItem({data:2, label:'Yes'});

var cbListener:Object = new Object();
cbListener.change = function(evt:Object) {
  my_video.deblocking = evt.target.selectedItem.data;
};
deblocking_cb.addEventListener("change", cbListener);
```

Utilice la instancia de ComboBox para cambiar el comportamiento del filtro de desbloqueo en video1.flv.

_height (propiedad Video._height)

```
public _height : Number
```

Indica la altura del objeto Video, expresada en píxeles.

Disponibilidad: ActionScript 1.0; Flash Player 8

Véase también

`_width (propiedad Video._width)`

height (propiedad Video.height)

```
public height : Number [read-only]
```

Un entero que especifica la altura del flujo de vídeo en píxeles. En el caso de flujos en vivo, este valor es el mismo que el de la propiedad `Camera.height` del objeto Camera que está capturando el flujo de vídeo. En el caso de archivos FLV, este valor es la altura del archivo exportado como FLV.

Puede utilizar esta propiedad, por ejemplo, para asegurarse de que el usuario está viendo el vídeo con el mismo tamaño con el que se capturó, con independencia del tamaño real del objeto Video existente en el escenario.

Disponibilidad: ActionScript 1.0; Flash Player 6

Ejemplo

El ejemplo siguiente define las propiedades `_height` y `_width` de la instancia Video Symbol igual a `height` y `width` del archivo FLV.

Para utilizar este ejemplo, cree primero un nuevo símbolo Video con el nombre de instancia "myVideo" y colóquelo en el mismo contexto que este script. Las propiedades `height` y `width` serán cero cuando se active el código de estado de `NetStream.Play.Start`. Si cambia el tamaño del vídeo cuando reciba el estado de `NetStream.Buffer.Full`, asegurará que su tamaño sea correcto antes de que se muestre el primer fotograma.

```
var netConn:NetConnection = new NetConnection();
netConn.connect(null);
var netStrm:NetStream = new NetStream(netConn);

myVideo.attachVideo(netStrm);
netStrm.play("Video.flv");

netStrm.onStatus = function(infoObject:Object) {
  switch (infoObject.code) {
    case 'NetStream.Play.Start' :
    case 'NetStream.Buffer.Full' :
      myVideo._width = myVideo.width;
      myVideo._height = myVideo.height;
      break;
  }
}
```

Véase también

`_height (propiedad MovieClip._height)`, `width (propiedad Video.width)`

_name (propiedad Video._name)

```
public _name : String
```

Indica el nombre de instancia del objeto Video especificado.

Disponibilidad: ActionScript 1.0; Flash Player 8

_parent (propiedad Video._parent)

```
public _parent : MovieClip
```

Indica el clip de película o el objeto que contiene el objeto Video actual. El objeto actual es el que contiene el código ActionScript que hace referencia a `_parent`. Utilice la propiedad `_parent` para especificar una ruta de acceso relativa a los clips de película u objetos que se encuentran por encima del objeto actual.

Puede utilizar `_parent` para subir múltiples niveles en la lista de visualización, como se muestra a continuación:

```
this._parent._parent._alpha = 20;
```

Disponibilidad: ActionScript 1.0; Flash Player 8

Véase también

`Propiedad _root`, `_target (propiedad MovieClip._target)`

_rotation (propiedad Video._rotation)

```
public _rotation : Number
```

Indica el giro del objeto Video, expresado en grados, con respecto a su orientación original. Los valores comprendidos entre 0 y 180 representan un giro en el sentido de las agujas del reloj, mientras que los comprendidos entre 0 y -180 representan un giro en sentido contrario al de las agujas del reloj. Los valores situados fuera de este rango se suman o restan de 360 para obtener un valor que sí esté comprendido en el rango. Por ejemplo, la sentencia `my_video._rotation = 450` es igual que `my_video._rotation = 90`.

Disponibilidad: ActionScript 1.0; Flash Player 8

smoothing (propiedad Video.smoothing)

```
public smoothing : Boolean
```

Especifica si el vídeo debe suavizarse (interpolarse) al aplicarle una escala. Para que funcione el suavizado, el reproductor debe estar en modo de alta calidad. El valor predeterminado es `false` (sin suavizado).

Disponibilidad: ActionScript 1.0; Flash Player 6

Ejemplo

El ejemplo siguiente utiliza un botón (con el nombre `smoothing_btn`) para activar/desactivar la propiedad smoothing aplicada al vídeo `my_video` cuando se reproduce en un archivo SWF. Cree un botón denominado `smoothing_btn` y añada el siguiente código ActionScript al archivo FLA o AS:

```
this.createTextField("smoothing_txt", this.getNextHighestDepth(), 0, 0,
  100, 22);
smoothing_txt.autoSize = true;

var my_nc:NetConnection = new NetConnection();
my_nc.connect(null);
var my_ns:NetStream = new NetStream(my_nc);
my_video.attachVideo(my_ns);
my_ns.play("video1.flv");
my_ns.onStatus = function(infoObject:Object) {
  updateSmoothing();
};
smoothing_btn.onRelease = function() {
  my_video.smoothing = !my_video.smoothing;
  updateSmoothing();
};
function updateSmoothing():Void {
  smoothing_txt.text = "smoothing = "+my_video.smoothing;
}
```

El método `MovieClip.getNextHighestDepth()` utilizado en este ejemplo necesita Flash Player 7 o una versión posterior. Si el archivo SWF incluye un componente de la versión 2, utilice la clase DepthManager de los componentes de la versión 2 en lugar del método `MovieClip.getNextHighestDepth()`.

_visible (propiedad Video._visible)

```
public _visible : Boolean
```

Indica si el objeto Video especificado por *`my_video`* es visible.

Disponibilidad: ActionScript 1.0; Flash Player 8

_width (propiedad Video._width)

```
public _width : Number
```

Indica la anchura del objeto Video, expresada en píxeles.

Disponibilidad: ActionScript 1.0; Flash Player 8

Véase también

`_height (propiedad Video._height)`

width (propiedad Video.width)

```
public width : Number [read-only]
```

Un entero que especifica la anchura del flujo de vídeo en píxeles. En el caso de flujos en vivo, este valor es el mismo que el de la propiedad `Camera.width` del objeto Camera que está capturando el flujo de vídeo. En el caso de archivos FLV, este valor es la anchura del archivo exportado como archivo FLV.

Puede utilizar esta propiedad, por ejemplo, para asegurarse de que el usuario está viendo el vídeo con el mismo tamaño con el que se capturó, con independencia del tamaño real del objeto Video existente en el escenario.

Disponibilidad: ActionScript 1.0; Flash Player 6

Ejemplo

Consulte el ejemplo de `Video.height`.

_x (propiedad Video._x)

```
public _x : Number
```

Indica la coordenada x de un objeto Video en relación a las coordenadas locales del clip de película principal. Si un objeto Video se encuentra en la línea de tiempo principal, su sistema de coordenadas hará referencia a la esquina superior izquierda del escenario como (0, 0). Si el objeto Video está dentro de un clip de película que incluye transformaciones, el objeto Video estará en el sistema de coordenadas local del clip de película en el que está contenido. Por consiguiente, en el caso de un clip de película con un giro de 90° en sentido contrario al de las agujas del reloj, los clips de película secundarios de este clip de película heredarán un sistema de coordenadas con un giro de 90° en sentido contrario al de las agujas del reloj. Las coordenadas del objeto Video hacen referencia a la posición del punto de registro.

Disponibilidad: ActionScript 1.0; Flash Player 8

Véase también

_xscale (propiedad Video._xscale), _y (propiedad Video._y), _yscale (propiedad Video._yscale)

_xmouse (propiedad Video._xmouse)

```
public _xmouse : Number [read-only]
```

Indica la coordenada *x* de la posición del ratón.

Disponibilidad: ActionScript 1.0; Flash Player 8

Véase también

Mouse, _ymouse (propiedad Video._ymouse)

_xscale (propiedad Video._xscale)

```
public _xscale : Number
```

Indica la escala horizontal (*percentage*) del objeto Video aplicada desde el punto de registro del objeto Video. El punto de registro predeterminado es (0,0).

La escala del sistema de coordenadas local afecta a la configuración de las propiedades _x e _y, que se definen en píxeles completos.

Disponibilidad: ActionScript 1.0; Flash Player 8

Véase también

_x (propiedad Video._x), _y (propiedad Video._y), _yscale (propiedad Video._yscale), _width (propiedad Video._width)

_y (propiedad Video._y)

```
public _y : Number
```

Indica la coordenada *y* de un objeto Video en relación a las coordenadas locales del clip de película principal. Si un objeto Video se encuentra en la línea de tiempo principal, su sistema de coordenadas hará referencia a la esquina superior izquierda del escenario como (0, 0). Si el objeto Video está dentro de un clip de película que incluye transformaciones, el objeto Video estará en el sistema de coordenadas local del clip de película en el que está contenido. Por consiguiente, en el caso de un clip de película con un giro de 90° en sentido contrario al de las agujas del reloj, los clips de película secundarios de este clip de película heredarán un sistema de coordenadas con un giro de 90° en sentido contrario al de las agujas del reloj. Las coordenadas del objeto Video hacen referencia a la posición del punto de registro.

Disponibilidad: ActionScript 1.0; Flash Player 8

Véase también

`_x (propiedad Video._x)`, `_xscale (propiedad Video._xscale)`, `_yscale (propiedad Video._yscale)`

_ymouse (propiedad Video._ymouse)

```
public _ymouse : Number [read-only]
```

Indica la coordenada *y* de la posición del ratón.

Disponibilidad: ActionScript 1.0; Flash Player 8

Véase también

`Mouse`, `_xmouse (propiedad Video._xmouse)`

_yscale (propiedad Video._yscale)

```
public _yscale : Number
```

Indica la escala vertical (*percentage*) del objeto Video aplicada desde el punto de registro del objeto Video. El punto de registro predeterminado es (0,0).

La escala del sistema de coordenadas local afecta a la configuración de las propiedades `_x` e `_y`, que se definen en píxeles completos.

Disponibilidad: ActionScript 1.0; Flash Player 8

Véase también

`_x (propiedad Video._x)`, `_xscale (propiedad Video._xscale)`, `_y (propiedad Video._y)`, `_height (propiedad Video._height)`

XML

```
Object
  |
  +-XMLNode
    |
    +-XML
```

```
public class XML
extends XMLNode
```

Utilice los métodos y propiedades de la clase XML para cargar, analizar, enviar, crear y manipular árboles de documentos XML.

Deberá utilizar el constructor `new XML()` para crear un objeto XML antes de llamar a los métodos de la clase XML.

Un documento XML se representa en Flash mediante la clase XML. Cada elemento del documento jerárquico se representa mediante un objeto XMLNode.

Para obtener información sobre los métodos y propiedades siguientes, puede ver la clase XMLNode, más concretamente `appendChild()`, `attributes`, `childNodes`, `cloneNode()`, `firstChild`, `hasChildNodes()`, `insertBefore()`, `lastChild`, `nextSibling`, `nodeName`, `nodeType`, `nodeValue`, `parentNode`, `previousSibling`, `removeNode()` y `toString()`.

En versiones anteriores de Referencia del lenguaje ActionScript de Flash, los métodos y propiedades indicados se incluían en la clase XML. Ahora se describen en la clase XMLNode.

Nota: El modelo de los objetos XML y XMLNode sigue la recomendación W3C DOM Nivel 1, que puede encontrar en: http://www.w3.org/tr/1998/REC-DOM-Level-1-19981001/level-one-core.html. Esta recomendación especifica una interfaz de nodo y una interfaz de documento. La interfaz de documento hereda de la interfaz de nodo y añade métodos tales como `createElement()` y `createTextNode()`. En ActionScript, los objetos XML y XMLNode están diseñados para dividir la funcionalidad en líneas similares.

Disponibilidad: ActionScript 1.0; Flash Player 5

Véase también

`appendChild (método XMLNode.appendChild)`, `attributes (propiedad XMLNode.attributes)`, `childNodes (propiedad XMLNode.childNodes)`, `cloneNode (método XMLNode.cloneNode)`, `firstChild (propiedad XMLNode.firstChild)`, `hasChildNodes (método XMLNode.hasChildNodes)`, `insertBefore (método XMLNode.insertBefore)`, `lastChild (propiedad XMLNode.lastChild)`, `nextSibling (propiedad XMLNode.nextSibling)`, `nodeName (propiedad XMLNode.nodeName)`, `nodeType (propiedad XMLNode.nodeType)`, `nodeValue (propiedad XMLNode.nodeValue)`, `parentNode (propiedad XMLNode.parentNode)`, `previousSibling (propiedad XMLNode.previousSibling)`, `removeNode (método XMLNode.removeNode)`, `toString (método XMLNode.toString)`

Resumen de propiedades

Modificadores	Propiedad	Descripción
	`contentType:String`	Contenido MIME que se envía al servidor cuando se llama al método `XML.send()` o `XML.sendAndLoad()`.
	`docTypeDecl:String`	Especifica información sobre la declaración `DOCTYPE` del documento XML.
	`idMap:Object`	Un objeto con los nodos del archivo XML que tienen un atributo `id` asignado.
	`ignoreWhite:Boolean`	El valor predeterminado es `false`.
	`loaded:Boolean`	La propiedad que indica si el documento XML se ha cargado correctamente.
	`status:Number`	Establece automáticamente y devuelve un valor numérico que indica si un documento XML se ha analizado correctamente en un objeto XML.
	`xmlDecl:String`	Cadena que especifica información sobre una declaración XML de un documento.

Propiedades heredadas de la clase XMLNode

```
attributes (propiedad XMLNode.attributes), childNodes (propiedad
XMLNode.childNodes), firstChild (propiedad XMLNode.firstChild), lastChild
(propiedad XMLNode.lastChild), localName (propiedad XMLNode.localName),
namespaceURI (propiedad XMLNode.namespaceURI), nextSibling (propiedad
XMLNode.nextSibling), nodeName (propiedad XMLNode.nodeName), nodeType
(propiedad XMLNode.nodeType), nodeValue (propiedad XMLNode.nodeValue),
parentNode (propiedad XMLNode.parentNode), prefix (propiedad
XMLNode.prefix), previousSibling (propiedad XMLNode.previousSibling)
```

Propiedades heredadas de la clase Object

```
constructor (propiedad Object.constructor), __proto__ (propiedad
Object.__proto__), prototype (propiedad Object.prototype), __resolve
(propiedad Object.__resolve)
```

Resumen de eventos

Evento	Descripción
`onData = function(src:String) {}`	Invocado cuando se ha descargado completamente el texto XML del servidor o cuando se produce un error mientras se está descargando el texto XML de un servidor.
`onHTTPStatus = function(httpStatus:Number) {}`	Se invoca cuando Flash Player recibe un código de estado HTTP del servidor.
`onLoad = function(success:Boolean) {}`	Lo invoca Flash Player cuando se recibe un documento XML del servidor.

Resumen de constructores

Firma	Descripción
`XML(text:String)`	Crea un nuevo objeto XML.

Resumen de métodos

Modificadores	Firma	Descripción
	`addRequestHeader(header:Object, headerValue:String) : Void`	Añade o cambia los encabezados de peticiones HTTP (como `Content-Type` o `SOAPAction`) enviados con las acciones `POST`.
	`createElement(name:String) : XMLNode`	Crea un nuevo elemento XML con el nombre especificado en el parámetro.
	`createTextNode(value:String) : XMLNode`	Crea un nuevo nodo de texto XML con el texto especificado.
	`getBytesLoaded() : Number`	Devuelve el número de bytes cargados (sin interrupción) para el documento XML.
	`getBytesTotal() : Number`	Devuelve el tamaño en bytes del documento XML.
	`load(url:String) : Boolean`	Carga un documento XML de la URL especificada y sustituye el contenido del objeto XML especificado por los datos XML descargados.
	`parseXML(value:String) : Void`	Analiza el texto XML especificado en el parámetro `value` y llena el objeto XML especificado con el árbol XML resultante.

Modificadores	Firma	Descripción
	`send(url:String, [target:String], [method:String]) : Boolean`	Codifica el objeto XML especificado en un documento XML y lo envía a la URL `target` especificada.
	`sendAndLoad(url:String, resultXML:XML) : Void`	Codifica el objeto XML especificado en un documento XML, lo envía a la URL especificada empleando el método `POST`, descarga la respuesta del servidor y la carga en el `resultXMLobject` especificado en los parámetros.

Métodos heredados de la clase XMLNode

`appendChild (método XMLNode.appendChild)`, `cloneNode (método XMLNode.cloneNode)`, `getNamespaceForPrefix (método XMLNode.getNamespaceForPrefix)`, `getPrefixForNamespace (método XMLNode.getPrefixForNamespace)`, `hasChildNodes (método XMLNode.hasChildNodes)`, `insertBefore (método XMLNode.insertBefore)`, `removeNode (método XMLNode.removeNode)`, `toString (método XMLNode.toString)`

Métodos heredados de la clase Object

`addProperty (método Object.addProperty)`, `hasOwnProperty (método Object.hasOwnProperty)`, `isPropertyEnumerable (método Object.isPropertyEnumerable)`, `isPrototypeOf (método Object.isPrototypeOf)`, `registerClass (método Object.registerClass)`, `toString (método Object.toString)`, `unwatch (método Object.unwatch)`, `valueOf (método Object.valueOf)`, `watch (método Object.watch)`

addRequestHeader (método XML.addRequestHeader)

`public addRequestHeader(header:Object, headerValue:String) : Void`

Añade o cambia los encabezados de peticiones HTTP (como `Content-Type` o `SOAPAction`) enviados con las acciones `POST`. En la primera sintaxis, se pasan dos cadenas al método: `header` y `headerValue`. En la segunda sintaxis, se pasa una matriz de cadenas, alternando los nombres de los encabezados y los valores de éstos.

Si se realizan múltiples llamadas para establecer el mismo nombre de encabezado, cada valor sucesivo reemplaza al valor establecido en la llamada anterior.

No puede añadir ni cambiar los siguientes encabezados HTTP estándar con este método: `Accept-Ranges`, `Age`, `Allow`, `Allowed`, `Connection`, `Content-Length`, `Content-Location`, `Content-Range`, `ETag`, `Host`, `Last-Modified`, `Locations`, `Max-Forwards`, `Proxy-Authenticate`, `Proxy-Authorization`, `Public`, `Range`, `Retry-After`, `Server`, `TE`, `Trailer`, `Transfer-Encoding`, `Upgrade`, `URI`, `Vary`, `Via`, `Warning` y `WWW-Authenticate`.

Disponibilidad: ActionScript 1.0; Flash Player 6

Parámetros

`header``:Object` - Una cadena que representa un nombre de encabezado de solicitud HTTP.

`headerValue``:String` - Una cadena que representa el valor asociado con `header`.

Ejemplo

El ejemplo siguiente añade un encabezado HTTP personalizado denominado `SOAPAction` con un valor de `Foo` a un objeto XML con el nombre `my_xml`:

```
my_xml.addRequestHeader("SOAPAction", "'Foo'");
```

El ejemplo siguiente crea una matriz `headers` que contiene dos encabezados HTTP alternativos y sus valores asociados. La matriz se pasa como un parámetro al método `addRequestHeader()`.

```
var headers:Array = new Array("Content-Type", "text/plain", "X-
  ClientAppVersion", "2.0");
my_xml.addRequestHeader(headers);
```

Véase también

`addRequestHeader (método LoadVars.addRequestHeader)`

contentType (propiedad XML.contentType)

```
public contentType : String
```

Contenido MIME que se envía al servidor cuando se llama al método `XML.send()` o `XML.sendAndLoad()`. La configuración predeterminada es `application/x-www-form-urlencoded`, que es el tipo de contenido MIME estándar empleado para la mayoría de los formularios HTML.

Disponibilidad: ActionScript 1.0; Flash Player 6

Ejemplo

El siguiente ejemplo crea un nuevo documento XML y comprueba su tipo de contenido predeterminado:

```
// create a new XML document
var doc:XML = new XML();

// trace the default content type
trace(doc.contentType); // output: application/x-www-form-urlencoded
```

El ejemplo siguiente define un paquete XML y establece el tipo de contenido para el objeto XML. A continuación se envían los datos a un servidor y se muestra un resultado en una ventana de navegador.

```
var my_xml:XML = new XML("<highscore><name>Ernie</name><score>13045</
  score></highscore>");
my_xml.contentType = "text/xml";
my_xml.send("http://www.flash-mx.com/mm/highscore.cfm", "_blank");
```

Presione F12 para probar este ejemplo en un navegador.

Véase también

`send (método XML.send)`, `sendAndLoad (método XML.sendAndLoad)`

createElement (método XML.createElement)

```
public createElement(name:String) : XMLNode
```

Crea un nuevo elemento XML con el nombre especificado en el parámetro. El nuevo elemento carece inicialmente de elemento principal, de elementos secundarios o de elementos del mismo nivel. El método devuelve una referencia al objeto XML de nueva creación que representa al elemento. Este método y `XML.createTextNode()` son los métodos constructor para crear nodos para un objeto XML.

Disponibilidad: ActionScript 1.0; Flash Player 5

Parámetros

`name``:String` - El nombre de etiqueta del elemento XML que se va a crear.

Valor devuelto

`XMLNode` - Un objeto XMLNode; un elemento XML.

Ejemplo

El ejemplo siguiente crea tres nodos XML utilizando el método `createElement()`:

```
// create an XML document
var doc:XML = new XML();

// create three XML nodes using createElement()
var element1:XMLNode = doc.createElement("element1");
var element2:XMLNode = doc.createElement("element2");
var element3:XMLNode = doc.createElement("element3");

// place the new nodes into the XML tree
doc.appendChild(element1);
element1.appendChild(element2);
element1.appendChild(element3);

trace(doc);
// output: <element1><element2 /><element3 /></element1>
```

Véase también

`createTextNode (método XML.createTextNode)`

createTextNode (método XML.createTextNode)

```
public createTextNode(value:String) : XMLNode
```

Crea un nuevo nodo de texto XML con el texto especificado. El nuevo nodo carece inicialmente de elemento principal, de elementos secundarios o de elementos del mismo nivel. Este método devuelve una referencia al objeto XML que representa al nuevo nodo de texto. Este método y `XML.createElement()` son los métodos constructor para crear nodos para un objeto XML.

Disponibilidad: ActionScript 1.0; Flash Player 5

Parámetros

`value``:String` - Una cadena; el texto utilizado para crear el nuevo nodo de texto.

Valor devuelto

`XMLNode` - Un objeto XMLNode.

Ejemplo

El ejemplo siguiente crea dos nodos de texto XML con el método `createTextNode()` y los coloca en los nodos XML existentes:

```
// create an XML document
var doc:XML = new XML();

// create three XML nodes using createElement()
var element1:XMLNode = doc.createElement("element1");
var element2:XMLNode = doc.createElement("element2");
var element3:XMLNode = doc.createElement("element3");

// place the new nodes into the XML tree
doc.appendChild(element1);
element1.appendChild(element2);
element1.appendChild(element3);

// create two XML text nodes using createTextNode()
var textNode1:XMLNode = doc.createTextNode("textNode1 String value");
var textNode2:XMLNode = doc.createTextNode("textNode2 String value");

// place the new nodes into the XML tree
element2.appendChild(textNode1);
element3.appendChild(textNode2);

trace(doc);
// output (with line breaks added between tags):
// <element1>
// <element2>textNode1 String value</element2>
// <element3>textNode2 String value</element3>
// </element1>
```

Véase también

`createElement (método XML.createElement)`

docTypeDecl (propiedad XML.docTypeDecl)

```
public docTypeDecl : String
```

Especifica información sobre la declaración `DOCTYPE` del documento XML. Una vez que el texto XML se ha analizado en un objeto XML, la propiedad `XML.docTypeDecl` del objeto XML se define como el texto de la declaración `DOCTYPE` del documento XML (por ejemplo, `<!DOCTYPE greeting SYSTEM "hello.dtd">`). Esta propiedad se establece utilizando una representación de cadena de la declaración `DOCTYPE`, no un objeto de nodo XML.

El analizador XML de ActionScript no es un analizador validador. El analizador lee la declaración `DOCTYPE`, que se almacena en la propiedad `XML.docTypeDecl`, pero no se lleva a cabo ninguna validación Dtd.

Si no se ha encontrado ninguna declaración `DOCTYPE` durante la operación de análisis, la propiedad `XML.docTypeDecl` se establece en `undefined` (no definida). El método `XML.toString()` genera el contenido de `XML.docTypeDecl` inmediatamente después de la declaración XML almacenada en `XML.xmlDecl` y antes que cualquier otro texto del objeto XML. Si `XML.docTypeDecl` tiene el valor undefined, no se genera ninguna declaración `DOCTYPE`.

Disponibilidad: ActionScript 1.0; Flash Player 5

Ejemplo

El ejemplo siguiente utiliza la propiedad `XML.docTypeDecl` para establecer la declaración `DOCTYPE` para un objeto XML:

```
my_xml.docTypeDecl = "<!DOCTYPE greeting SYSTEM \"hello.dtd\">";
```

Véase también

`xmlDecl (propiedad XML.xmlDecl)`

getBytesLoaded (método XML.getBytesLoaded)

```
public getBytesLoaded() : Number
```

Devuelve el número de bytes cargados (sin interrupción) para el documento XML. Puede comparar el valor de `getBytesLoaded()` con el valor de `getBytesTotal()` para determinar el porcentaje de un documento XML que se ha cargado.

Disponibilidad: ActionScript 1.0; Flash Player 6

Valor devuelto

`Number` - Un entero que indica el número de bytes cargados.

Ejemplo

El ejemplo siguiente muestra cómo utilizar el método `XML.getBytesLoaded()` con el método `XML.getBytesTotal()` para seguir el progreso de un comando `XML.load()`. Debe reemplazar el parámetro URL del comando `XML.load()` de modo que el parámetro se refiera a un archivo XML válido utilizando HTTP. Si intenta utilizar este ejemplo para cargar un archivo local que resida en el disco duro, el ejemplo no funcionará correctamente porque, en el modo Probar película, Flash Player carga los archivos locales íntegramente.

```
// create a new XML document
```

```
var doc:XML = new XML();

var checkProgress = function(xmlObj:XML) {
  var bytesLoaded:Number = xmlObj.getBytesLoaded();
  var bytesTotal:Number = xmlObj.getBytesTotal();
  var percentLoaded:Number = Math.floor((bytesLoaded / bytesTotal ) 100);
  trace ("milliseconds elapsed: " + getTimer());
  trace ("bytesLoaded: " + bytesLoaded);
  trace ("bytesTotal: " + bytesTotal);
  trace ("percent loaded: " + percentLoaded);
  trace ("---------------------------------");
}

doc.onLoad = function(success:Boolean) {
  clearInterval(intervalID);
  trace("intervalID: " + intervalID);
}
doc.load("[place a valid URL pointing to an XML file here]");
var intervalID:Number = setInterval(checkProgress, 100, doc);
```

Véase también

getBytesTotal (método XML.getBytesTotal)

getBytesTotal (método XML.getBytesTotal)

```
public getBytesTotal() : Number
```

Devuelve el tamaño en bytes del documento XML.

Disponibilidad: ActionScript 1.0; Flash Player 6

Valor devuelto

`Number` - Un entero.

Ejemplo

Véase el ejemplo de `XML.getBytesLoaded()`.

Véase también

getBytesLoaded (método XML.getBytesLoaded)

idMap (propiedad XML.idMap)

```
public idMap : Object
```

Un objeto con los nodos del archivo XML que tienen un atributo `id` asignado. Los nombres de las propiedades del objeto (cada una contiene un nodo) corresponden a los valores de los atributos `id`.

Considere el siguiente objeto XML:

```
<employee id='41'>
<name>
John Doe
</name>
<address>
601 Townsend St.
</address>
</employee>

<employee id='42'>
<name>
Jane Q. Public
</name>
</employee>
<department id="IT">
Information Technology
</department>
```

En este ejemplo, la propiedad `idMap` para este XML es un objeto con tres propiedades: `41`, `42` y `IT`. Cada una de estas propiedades es un nodo XMLNode que tiene el correspondiente valor `id`. Por ejemplo, la propiedad `IT` del objeto `idMap` es este nodo:

```
<department id="IT">
Information Technology
</department>
```

Debe utilizar el método `parse()` en el objeto XML para que se cree una instancia de la propiedad `idMap`.

Si hay más de un XMLNode con el mismo valor de `id`, la propiedad correspondiente del objeto `idNode` es la del último nodo analizado, de la forma siguiente:

```
var x1:XML = new XML("<a id='1'><b id='2' /><c id='1' /></a>");
x2 = new XML();
x2.parseXML(x1);
trace (x2.idMap['1']);
```

El código siguiente devolverá el nodo `<c>`:

```
<c id='1' />
```

Disponibilidad: ActionScript 1.0; Flash Player 8

Ejemplo

Puede crear un archivo de texto con el nombre `idMapTest.xml` que contenga el texto siguiente:

```
<?xml version="1.0"?>
<doc xml:base="http://example.org/today/" xmlns:xlink="http://www.w3.org/
 1999/xlink">
<head>
<title>Virtual Library</title>
</head>
<body>
<paragraph id="linkP1">See <link xlink:type="simple"
 xlink:href="new.xml">what's
new</link>!</paragraph>
<paragraph>Check out the hot picks of the day!</paragraph>
<olist xml:base="/hotpicks/">
<item>
<link id="foo" xlink:type="simple" xlink:href="pick1.xml">Hot Pick #1</
 link>
</item>
<item>
<link id="bar" xlink:type="simple" xlink:href="pick2.xml">Hot Pick #2</
 link>
</item>
<item>
<link xlink:type="simple" xlink:href="pick3.xml">Hot Pick #3</link>
</item>
</olist>
</body>
</doc>
```

A continuación, puede crear un archivo SWF en el mismo directorio que el archivo XML. Puede incluir el siguiente script en el SWF.

```
var readXML = new XML();
readXML.load("idMapTest.xml");
readXML.onLoad = function(success) {
  myXML = new XML();
  myXML.parseXML(readXML);
  for (var x in myXML.idMap){
    trace('idMap.' + x + " = " + newline + myXML.idMap[x]);
    trace('_____________' + newline);
  }
}
```

Cuando pruebe el archivo SWF, se genera el siguiente resultado:

```
idMap.bar =
<link id="bar" xlink:type="simple" xlink:href="pick2.xml">Hot Pick #2</
 link>
_____________

idMap.foo =
<link id="foo" xlink:type="simple" xlink:href="pick1.xml">Hot Pick #1</
 link>
_____________

idMap.linkP1 =
<paragraph id="linkP1">See <link xlink:type="simple"
 xlink:href="new.xml">what's

new</link>!</paragraph>
_____________
```

ignoreWhite (propiedad XML.ignoreWhite)

```
public ignoreWhite : Boolean
```

El valor predeterminado es `false`. Cuando se establece con el valor `true`, los nodos de texto que contienen sólo espacio en blanco se descartan durante el proceso de análisis. Los nodos de texto con espacio en blanco al principio o al final no se ven afectados.

Sintaxis 1: Puede establecer la propiedad `ignoreWhite` para objetos XML individuales, como se muestra en el siguiente código:

```
my_xml.ignoreWhite = true;
```

Sintaxis 2: Puede establecer la propiedad `ignoreWhite` predeterminada para objetos XML, como se muestra en el siguiente código:

```
XML.prototype.ignoreWhite = true;
```

Disponibilidad: ActionScript 1.0; Flash Player 5

Ejemplo

El ejemplo siguiente carga un archivo XML con un nodo de texto que contiene solamente espacio blanco; la etiqueta `foyer` incluye catorce espacios. Para ejecutar este ejemplo, cree un archivo de texto llamado *flooring.xml* y copie en él las siguientes etiquetas:

```
<house>
<kitchen> ceramic tile </kitchen>
<bathroom>linoleum</bathroom>
<foyer> </foyer>
</house>
```

Cree un nuevo documento Flash con el nombre *flooring.fla* y guárdelo en el mismo directorio que el archivo XML. Sitúe el siguiente código en la línea de tiempo principal:

```
// Create a new XML object.
var flooring:XML = new XML();

// Set the ignoreWhite property to true (default value is false)
flooring.ignoreWhite = true;

// After loading is complete, trace the XML object.
flooring.onLoad = function(success:Boolean) {
  trace(flooring);
}

// Load the XML into the flooring object.
flooring.load("flooring.xml");

// Output (line breaks added for clarity):
<house>
  <kitchen> ceramic tile </kitchen>
  <bathroom>linoleum</bathroom>
  <foyer />
</house>
```

Si cambia la definición de `flooring.ignoreWhite` a `false`, o simplemente elimina esa línea de texto, se conservan los catorce espacios de la etiqueta `foyer`:

```
...
// Set the ignoreWhite property to false (default value).
flooring.ignoreWhite = false;
...
// Output (line breaks added for clarity):
<house>
  <kitchen> ceramic tile </kitchen>
  <bathroom>linoleum</bathroom>
  <foyer> </foyer>
</house>
```

Los archivos XML_blogTracker.fla y XML_languagePicker.fla de la carpeta de ejemplos de ActionScript también contienen un ejemplo de código. Estas son rutas de acceso habituales a esta carpeta:

- Windows: *unidad de inicio*\Archivos de programa\Macromedia\Flash 8\Samples and Tutorials\Samples\ActionScript
- Macintosh: *Disco duro de Macintosh*/Applications/Macromedia Flash 8/Samples and Tutorials/Samples/ActionScript

load (método XML.load)

```
public load(url:String) : Booleano
```

Carga un documento XML de la URL especificada y sustituye el contenido del objeto XML especificado por los datos XML descargados. La URL es relativa y se llama a ella mediante HTTP. El proceso de carga es asíncrono; no finaliza inmediatamente después de que se ejecute el método `load()`.

Cuando se ejecuta el método `load()`, la propiedad `loaded` del objeto XML se establece con el valor `false`. Cuando terminan de descargarse los datos XML, la propiedad `loaded` se establece con el valor `true` y se invoca el controlador de eventos `onLoad`. Los datos XML no se analizan hasta que terminan de descargarse. Si el objeto XML contenía anteriormente árboles XML, estos se descartarán.

Puede definir una función personalizada que se ejecute cuando se invoque el controlador de eventos `onLoad` del objeto XML.

Nota: Si un archivo que se está cargando contiene caracteres no ASCII (como ocurre en muchos idiomas, no así en inglés), se recomienda que guarde el archivo con codificación UTF-8 o UTF-16, en lugar de utilizar un formato no Unicode, como ASCII.

Cuando utilice este método, puede ser conveniente usar el modelo de seguridad de Flash Player:

Para Flash Player 8:

- No se permite la carga de datos si el archivo SWF que realiza la llamada se encuentra en el entorno limitado local con sistema de archivos y el recurso se encuentra en un entorno limitado de red.
- Tampoco se permite la carga de datos si el archivo SWF que realiza la llamada se encuentra en un entorno limitado de red y el recurso de destino es local.

Para más información, consulte lo siguiente:

- Capítulo 17, "Aspectos básicos de la seguridad," en *Aprendizaje de ActionScript 2.0 en Flash*
- El documento técnico sobre la seguridad de Flash Player 8 en http://www.macromedia.com/go/fp8_security
- El documento técnico sobre interfaces API relativas a seguridad de Flash Player 8 en http://www.macromedia.com/go/fp8_security_apis

Para Flash Player 7 y versiones posteriores, los sitios Web pueden permitir el acceso de distintos dominios a un recurso mediante un archivo de política de varios dominios. En archivos SWF de cualquier versión que se ejecuten en Flash Player 7 y posterior, `url` debe estar exactamente en el mismo dominio. Por ejemplo, un archivo SWF de www.someDomain.com puede cargar datos únicamente de orígenes que estén también en www.someDomain.com.

En archivos SWF que se ejecuten en una versión del reproductor anterior a Flash Player 7, el parámetro `url` debe estar en el mismo superdominio que el archivo SWF que emite esta llamada. El *superdominio* puede obtenerse eliminando el componente situado más a la izquierda de la URL de un archivo. Por ejemplo, un archivo SWF situado en www.someDomain.com puede cargar datos de fuentes situadas en store.someDomain.com porque ambos archivos se encuentran en el mismo superdominio, denominado someDomain.com.

Disponibilidad: ActionScript 1.0; Flash Player 5

Parámetros

`url``:String` - Una cadena que representa la URL donde se encuentra el documento XML que se va a cargar. Si el archivo SWF que realiza esta llamada se ejecuta en un navegador Web, el valor `url` debe pertenecer al mismo dominio que el archivo SWF.

Valor devuelto

`Boolean` - Un valor booleano `false` si no se pasa ningún parámetro (null); `true`, en caso contrario. Utilice el controlador de eventos `onLoad()` para comprobar el estado de un documento XML cargado.

Ejemplo

En el ejemplo siguiente se utiliza el método `XML.load()`:

```
// Create a new XML object.
var flooring:XML = new XML();

// Set the ignoreWhite property to true (default value is false).
flooring.ignoreWhite = true;

// After loading is complete, trace the XML object.
flooring.onLoad = function(success) {
  trace(flooring);
};

// Load the XML into the flooring object.
flooring.load("flooring.xml");
```

Para ver el contenido del archivo flooring.xml y la salida que produce, véase el ejemplo de `XML.ignoreWhite`.

Véase también

`ignoreWhite (propiedad XML.ignoreWhite)`, `loaded (propiedad XML.loaded)`, `onLoad (controlador XML.onLoad)`, `useCodepage (propiedad System.useCodepage)`

loaded (propiedad XML.loaded)

```
public loaded : Boolean
```

La propiedad que indica si el documento XML se ha cargado correctamente. Si no hay ningún controlador de eventos `onLoad()` personalizado definido para el objeto XML, esta propiedad se establece con el valor `true` cuando el proceso de carga de documento iniciado por la llamada a `XML.load()` ha finalizado completamente; en caso contrario, el valor es `false`. No obstante, si define un comportamiento personalizado para el controlador de eventos `onLoad()` para el objeto XML, asegúrese de que establece `onload` en dicha función.

Disponibilidad: ActionScript 1.0; Flash Player 5

Ejemplo

El ejemplo siguiente utiliza la propiedad `XML.loaded` en un script simple.

```
var my_xml:XML = new XML();
my_xml.ignoreWhite = true;
my_xml.onLoad = function(success:Boolean) {
  trace("success: "+success);
  trace("loaded: "+my_xml.loaded);
  trace("status: "+my_xml.status);
};
my_xml.load("http://www.flash-mx.com/mm/problems/products.xml");
```

Cuando Flash invoca el controlador `onLoad()` aparece información en el panel Salida. Si la llamada se realiza correctamente, aparece `true` como estado `loaded` en el panel Salida.

```
success: true
loaded: true
status: 0
```

Véase también

`load (método XML.load)`, `onLoad (controlador XML.onLoad)`

onData (controlador XML.onData)

```
onData = function(src:String) {}
```

Invocado cuando se ha descargado completamente el texto XML del servidor o cuando se produce un error mientras se está descargando el texto XML de un servidor. Este controlador se invoca antes de que se analice el XML y puede utilizarlo para llamar a una rutina de análisis personalizada en lugar de utilizar el analizador XML de Flash. El parámetro src es una cadena que contiene texto XML descargado del servidor, a no ser que se produzca un error durante la descarga, en cuyo caso el parámetro `src` no estará definido (`undefined`).

De forma predeterminada, el controlador de eventos `XML.onData` llama a `XML.onLoad`. Puede sustituir el controlador de eventos `XML.onData` con comportamiento personalizado, pero no se llama a `XML.onLoad` a menos que lo haga en su implementación de `XML.onData`.

Disponibilidad: ActionScript 1.0; Flash Player 5

Parámetros

`src``:String` - Una cadena o `undefined`; los datos, normalmente en formato XML, que envía el servidor.

Ejemplo

El ejemplo siguiente muestra cómo aparece de forma predeterminada el controlador de eventos XML.onData:

```
XML.prototype.onData = function (src:String) {
  if (src == undefined) {
    this.onLoad(false);
  } else {
    this.parseXML(src);
    this.loaded = true;
    this.onLoad(true);
  }
}
```

Puede sustituir el controlador de eventos XML.onData para interceptar el texto XML sin analizarlo.

Véase también

`onLoad (controlador XML.onLoad)`

onHTTPStatus (controlador XML.onHTTPStatus)

```
onHTTPStatus = function(httpStatus:Number) {}
```

Se invoca cuando Flash Player recibe un código de estado HTTP del servidor. Este controlador permite capturar y trabajar con códigos de estado HTTP.

El controlador `onHTTPStatus` se llama antes que `onData`, que activa llamadas a `onLoad` con un valor de `undefined` si falla la carga. Es importante destacar que cuando se activa `onHTTPStatus`, se activa también `onData`, haya o no sustituido `onHTTPStatus`. Para utilizar mejor el controlador `onHTTPStatus`, escriba una función apropiada para capturar el resultado de la llamada a `onHTTPStatus`; a continuación puede utilizar el resultado de sus funciones de controladores `onData` o `onLoad`. Si no se llama a `onHTTPStatus`, el reproductor no ha intentado realizar la petición URL. Esto puede ocurrir debido a que la petición quebranta las reglas de seguridad del entorno limitado para el archivo SWF.

Si Flash Player no puede obtener el código de estado del servidor o si Flash Player no puede comunicarse con el servidor, se pasa el valor predeterminado 0 al código ActionScript. Es posible generar un valor 0 en cualquier reproductor; por ejemplo, si se solicita una dirección URL mal formada, y el plug-in de Flash Player siempre lo genera cuando se ejecuta en los navegadores siguientes, ya que no pasan códigos de estado HTTP al reproductor: Netscape, Mozilla, Safari, Opera o Internet Explorer para Macintosh.

Disponibilidad: ActionScript 1.0; Flash Player 8

Parámetros

`httpStatus``:Number` - El código de estado HTTP que devuelve el servidor. Por ejemplo, un valor de 404 indica que el servidor no encontró nada que coincida con el URI solicitado. Los códigos de estado HTTP se encuentran en las secciones 10.4 y 10.5 de la especificación HTTP en ftp://ftp.isi.edu/in-notes/rfc2616.txt.

Ejemplo

El siguiente ejemplo muestra cómo utilizar el método `onHTTPStatus` para facilitar la depuración. El ejemplo recoge códigos de estado HTTP y asigna su valor y tipo a una instancia del objeto `XML` (este ejemplo crea los miembros de instancia `this.httpStatus` y `this.httpStatusType` en tiempo de ejecución). El método los `onData` utiliza para controlar información sobre la respuesta HTTP que puede resultar útil en la depuración.

```
var myXml:XML = new XML();

myXml.onHTTPStatus = function(httpStatus:Number) {
  this.httpStatus = httpStatus;
  if(httpStatus < 100) {
    this.httpStatusType = "flashError";
```

```
  }
  else if(httpStatus < 200) {
    this.httpStatusType = "informational";
  }
  else if(httpStatus < 300) {
    this.httpStatusType = "successful";
  }
  else if(httpStatus < 400) {
    this.httpStatusType = "redirection";
  }
  else if(httpStatus < 500) {
    this.httpStatusType = "clientError";
  }
  else if(httpStatus < 600) {
    this.httpStatusType = "serverError";
  }
}

myXml.onData = function(src:String) {
  trace(">> " + this.httpStatusType + ": " + this.httpStatus);
  if(src != undefined) {
    this.parseXML(src);
    this.loaded = true;
    this.onLoad(true);
  }
  else {
    this.onLoad(false);
  }
}

myXml.onLoad = function(success:Boolean) {
}

myXml.load("http://weblogs.macromedia.com/mxna/xml/
  rss.cfm?query=byMostRecent&languages=1");
```

Véase también

onData (controlador LoadVars.onHTTPStatus), load (método XML.load), sendAndLoad (método XML.sendAndLoad)

onLoad (controlador XML.onLoad)

```
onLoad = function(success:Boolean) {}
```

Lo invoca Flash Player cuando se recibe un documento XML del servidor. Si se recibe correctamente el documento XML, el parámetro `success` tiene el valor `true`. Si no se ha recibido el documento o si se ha producido un error al recibir la respuesta del servidor, el parámetro `success` tiene el valor `false`. En la implementación predeterminada, este método está desactivado. Para sustituir la implementación predeterminada, debe asignar una función que contenga acciones personalizadas.

Disponibilidad: ActionScript 1.0; Flash Player 5

Parámetros

`success``:Boolean` - Un valor booleano que es `true` si el objeto XML se ha cargado correctamente con una operación `XML.load()` o `XML.sendAndLoad()`; en caso contrario, es `false`.

Ejemplo

El ejemplo siguiente incluye código ActionScript para una aplicación de comercio electrónico sencilla. El método `sendAndLoad()` transmite un elemento XML que contiene el nombre de usuario y la contraseña, y utiliza un controlador `XML.onLoad` para procesar la respuesta del servidor.

```
var login_str:String = "<login username=\""+username_txt.text+"\"
  password=\""+password_txt.text+"\" />";
var my_xml:XML = new XML(login_str);
var myLoginReply_xml:XML = new XML();

myLoginReply_xml.ignoreWhite = true;

myLoginReply_xml.onLoad = function(success:Boolean){

  if (success) {

    if ((myLoginReply_xml.firstChild.nodeName == "packet") &&
      (myLoginReply_xml.firstChild.attributes.success == "true")) {
      gotoAndStop("loggedIn");
    } else {
      gotoAndStop("loginFailed");
    }

  } else {
    gotoAndStop("connectionFailed");
  }
```

```
};

my_xml.sendAndLoad("http://www.flash-mx.com/mm/login_xml.cfm",
  myLoginReply_xml);
```

Véase también

load (método XML.load), sendAndLoad (método XML.sendAndLoad),

parseXML (método XML.parseXML)

```
public parseXML(value:String) : Void
```

Analiza el texto XML especificado en el parámetro `value` y llena el objeto XML especificado con el árbol XML resultante. Se descartarán los árboles existentes en el objeto XML.

Disponibilidad: ActionScript 1.0; Flash Player 5

Parámetros

`value``:String` - Una cadena que representa el texto XML que se va a analizar y pasar al objeto XML especificado.

Ejemplo

El ejemplo siguiente crea y analiza un objeto XML:

```
var xml_str:String = "<state name=\"California\">
<city>San Francisco</city></state>"

// defining the XML source within the XML constructor:
var my1_xml:XML = new XML(xml_str);
trace(my1_xml.firstChild.attributes.name); // output: California

// defining the XML source using the XML.parseXML method:
var my2_xml:XML = new XML();
my2_xml.parseXML(xml_str);
trace(my2_xml.firstChild.attributes.name); // output: California
```

send (método XML.send)

`public send(url:String, [target:String], [method:String]) : Booleano`

Codifica el objeto XML especificado en un documento XML y lo envía a la URL `target` especificada.

Cuando utilice este método, puede ser conveniente usar el modelo de seguridad de Flash Player:

- En el caso de Flash Player 8, el método no se admite si el archivo SWF que realiza la llamada se encuentra en un entorno limitado local que no es de confianza.
- En el caso de Flash Player 7, el método no se admite si el archivo SWF que realiza la llamada es un archivo local.

Para más información, consulte lo siguiente:

- Capítulo 17, "Aspectos básicos de la seguridad," en *Aprendizaje de ActionScript 2.0 en Flash*
- El documento técnico sobre la seguridad de Flash Player 8 en http://www.macromedia.com/go/fp8_security
- El documento técnico sobre interfaces API relativas a seguridad de Flash Player 8 en http://www.macromedia.com/go/fp8_security_apis

Disponibilidad: ActionScript 1.0; Flash Player 5

Parámetros

url:`String` - La URL de destino del objeto XML especificado.

target:`String` [opcional] - La ventana del navegador para mostrar los datos que devuelve el servidor:

- `_self` especifica el fotograma actual en la ventana actual.
- `_blank` especifica una nueva ventana.
- `_parent` especifica el elemento principal del fotograma actual.
- `_top` especifica el fotograma de nivel superior de la ventana actual.

Si no especifica un parámetro `target`, es igual que si especifica `_self`.

method:`String` [opcional] - El método del protocolo HTTP utilizado: `"GET"` o `"POST"`. En un navegador, el valor predeterminado es `"POST"`. En el entorno de prueba de Flash, el valor predeterminado es `"GET"`.

Valor devuelto

`Boolean` - `false` si no se especifican parámetros; `true`, en caso contrario.

Ejemplo

El ejemplo siguiente define un paquete XML y establece el tipo de contenido para el objeto XML. A continuación se envían los datos a un servidor y se muestra un resultado en una ventana de navegador.

```
var my_xml:XML = new XML("<highscore><name>Ernie</name><score>13045</
  score></highscore>");
my_xml.contentType = "text/xml";
my_xml.send("http://www.flash-mx.com/mm/highscore.cfm", "_blank");
```

Presione F12 para probar este ejemplo en un navegador.

Véase también

`sendAndLoad (método XML.sendAndLoad)`

sendAndLoad (método XML.sendAndLoad)

```
public sendAndLoad(url:String, resultXML:XML) : Void
```

Codifica el objeto XML especificado en un documento XML, lo envía a la URL especificada empleando el método `POST`, descarga la respuesta del servidor y la carga en el `resultXMLobject` especificado en los parámetros. La respuesta del servidor se carga de la misma forma empleada por el método `XML.load()`.

Cuando se ejecuta el método `sendAndLoad()`, la propiedad `loaded` del objeto XML se establece con el valor `false`. Cuando terminan de descargarse los datos XML, la propiedad `loaded` se establece con el valor `true`, si se han cargado correctamente los datos, y se invoca el controlador de eventos `onLoad`. Los datos XML no se analizan hasta que terminan de descargarse. Si el objeto XML contenía anteriormente árboles XML, estos se descartarán.

Cuando utilice este método, puede ser conveniente usar el modelo de seguridad de Flash Player:

Para Flash Player 8:

- No se permite la carga de datos si el archivo SWF que realiza la llamada se encuentra en el entorno limitado local con sistema de archivos y el recurso se encuentra en un entorno limitado de red.
- Tampoco se permite la carga de datos si el archivo SWF que realiza la llamada se encuentra en un entorno limitado de red y el recurso de destino es local.

Para más información, consulte lo siguiente:

- Capítulo 17, "Aspectos básicos de la seguridad," en *Aprendizaje de ActionScript 2.0 en Flash*
- El documento técnico sobre la seguridad de Flash Player 8 en http://www.macromedia.com/go/fp8_security
- El documento técnico sobre interfaces API relativas a seguridad de Flash Player 8 en http://www.macromedia.com/go/fp8_security_apis

Para Flash Player 7 y versiones posteriores, los sitios Web pueden permitir el acceso de distintos dominios a un recurso mediante un archivo de política de varios dominios. En archivos SWF de cualquier versión que se ejecuten en Flash Player 7 y posterior, `url` debe estar exactamente en el mismo dominio. Por ejemplo, un archivo SWF de www.someDomain.com puede cargar datos únicamente de orígenes que estén también en www.someDomain.com.

En archivos SWF que se ejecuten en una versión del reproductor anterior a Flash Player 7, el parámetro `url` debe estar en el mismo superdominio que el archivo SWF que emite esta llamada. El *superdominio* puede obtenerse eliminando el componente situado más a la izquierda de la URL de un archivo. Por ejemplo, un archivo SWF situado en www.someDomain.com puede cargar datos de fuentes situadas en store.someDomain.com porque ambos archivos se encuentran en el mismo superdominio, denominado someDomain.com.

Disponibilidad: ActionScript 1.0; Flash Player 5

Parámetros

`url``:String` - Una cadena; la URL de destino del objeto XML especificado. Si el archivo SWF que realiza esta llamada se ejecuta en un navegador Web, el valor `url` debe pertenecer al mismo dominio que el archivo SWF; para ver más detalles, consulte la sección Descripción.

`resultXML``:XML` - Un objeto XML de destino creado con el método constructor XML que recibirá la información devuelta del servidor.

Ejemplo

El ejemplo siguiente incluye código ActionScript para una aplicación de comercio electrónico sencilla. El método `XML.sendAndLoad()` transmite un elemento XML que contiene el nombre de usuario y la contraseña, y utiliza un controlador `onLoad` para procesar la respuesta del servidor.

```
var login_str:String = "<login username=\""+username_txt.text+"\"
  password=\""+password_txt.text+"\" />";
var my_xml:XML = new XML(login_str);
var myLoginReply_xml:XML = new XML();
```

```
myLoginReply_xml.ignoreWhite = true;
myLoginReply_xml.onLoad = myOnLoad;
my_xml.sendAndLoad("http://www.flash-mx.com/mm/login_xml.cfm",
  myLoginReply_xml);
function myOnLoad(success:Boolean) {
  if (success) {
    if ((myLoginReply_xml.firstChild.nodeName == "packet") &&
      (myLoginReply_xml.firstChild.attributes.success == "true")) {
      gotoAndStop("loggedIn");
    } else {
      gotoAndStop("loginFailed");
    }
  } else {
    gotoAndStop("connectionFailed");
  }
}
```

Véase también

`send (método XML.send)`, `load (método XML.load)`, `loaded (propiedad XML.loaded)`, `onLoad (controlador XML.onLoad)`

status (propiedad XML.status)

```
public status : Number
```

Establece automáticamente y devuelve un valor numérico que indica si un documento XML se ha analizado correctamente en un objeto XML. A continuación se indican los códigos numéricos de status (estado) con las correspondientes descripciones:

- 0 No se ha producido ningún error; el análisis ha finalizado correctamente.
- -2 Una sección CDATA no se ha terminado correctamente.
- -3 La declaración `XML` no se ha terminado correctamente.
- -4 La declaración `DOCTYPE` no se ha terminado correctamente.
- -5 Un comentario no se ha terminado correctamente.
- -6 Un elemento XML no tiene la forma correcta.
- -7 Memoria insuficiente.
- -8 El valor de un atributo no se ha terminado correctamente.
- -9 Una etiqueta start carece de la correspondiente etiqueta end.
- -10 Se ha encontrado una etiqueta end sin que exista una etiqueta start.

Disponibilidad: ActionScript 1.0; Flash Player 5

Ejemplo

El ejemplo siguiente carga un paquete XML en un archivo SWF. Aparece un mensaje de estado, que indica si el XML se carga y analiza correctamente. Añada el siguiente código ActionScript al archivo FLA o AS:

```
var my_xml:XML = new XML();
my_xml.onLoad = function(success:Boolean) {
  if (success) {
  if (my_xml.status == 0) {
    trace("XML was loaded and parsed successfully");
  } else {
    trace("XML was loaded successfully, but was unable to be parsed.");
  }
  var errorMessage:String;
  switch (my_xml.status) {
  case 0 :
    errorMessage = "No error; parse was completed successfully.";
    break;
  case -2 :
    errorMessage = "A CDATA section was not properly terminated.";
    break;
  case -3 :
    errorMessage = "The XML declaration was not properly terminated.";
    break;
  case -4 :
    errorMessage = "The DOCTYPE declaration was not properly terminated.";
    break;
  case -5 :
    errorMessage = "A comment was not properly terminated.";
    break;
  case -6 :
    errorMessage = "An XML element was malformed.";
    break;
  case -7 :
    errorMessage = "Out of memory.";
    break;
  case -8 :
    errorMessage = "An attribute value was not properly terminated.";
    break;
  case -9 :
    errorMessage = "A start-tag was not matched with an end-tag.";
    break;
  case -10 :
    errorMessage = "An end-tag was encountered without a matching
    start-tag.";
    break;
  default :
    errorMessage = "An unknown error has occurred.";
    break;
```

```
  }
  trace("status: "+my_xml.status+" ("+errorMessage+")");
  } else {
  trace("Unable to load/parse XML. (status: "+my_xml.status+")");
  }
};
my_xml.load("http://www.helpexamples.com/flash/badxml.xml");
```

Constructor XML

```
public XML(text:String)
```

Crea un nuevo objeto XML. Deberá utilizar el constructor para crear un objeto XML antes de llamar a cualquiera de los métodos de la clase XML.

Nota: Utilice los métodos `createElement()` y `createTextNode()` para añadir elementos y nodos de texto a un árbol de documentos XML.

Disponibilidad: ActionScript 1.0; Flash Player 5

Parámetros

`text``:String` - Una cadena; el texto XML analizado para crear el nuevo objeto XML.

Ejemplo

El ejemplo siguiente crea un nuevo objeto XML vacío:

```
var my_xml:XML = new XML();
```

El ejemplo siguiente crea un objeto XML analizando el texto XML especificado en el parámetro `source` y llena el objeto XML creado con el árbol de documentos XML resultante:

```
var other_xml:XML = new XML("<state name=\"California\"><city>San
  Francisco</city></state>");
```

Véase también

`createElement (método XML.createElement)`, `createTextNode (método XML.createTextNode)`

xmlDecl (propiedad XML.xmlDecl)

```
public xmlDecl : String
```

Cadena que especifica información sobre una declaración XML de un documento. Una vez que el documento XML se ha analizado e incorporado a un objeto XML, esta propiedad se establece con el texto de la declaración XML del documento. Esta propiedad se establece utilizando una representación de cadena de la declaración XML, no un objeto de nodo XML. Si no se ha encontrado ninguna declaración XML durante la operación de análisis, la propiedad se establece con el valor `undefined`.XML. El método `XML.toString()` devuelve el contenido de la propiedad `XML.xmlDecl` antes que ningún otro texto del objeto XML. Si la propiedad `XML.xmlDecl` contiene el tipo `undefined`, no se proporciona ninguna declaración XML.

Disponibilidad: ActionScript 1.0; Flash Player 5

Ejemplo

El ejemplo siguiente crea un campo de texto con el nombre `my_txt` con las mismas dimensiones que el escenario. El campo de texto muestra las propiedades del paquete XML que carga en el archivo SWF. Aparece la declaración docType en `my_txt`. Añada el siguiente código ActionScript al archivo FLA o AS:

```
var my_fmt:TextFormat = new TextFormat();
my_fmt.font = "_typewriter";
my_fmt.size = 12;
my_fmt.leftMargin = 10;

this.createTextField("my_txt", this.getNextHighestDepth(), 0, 0,
  Stage.width, Stage.height);
my_txt.border = true;
my_txt.multiline = true;
my_txt.wordWrap = true;
my_txt.setNewTextFormat(my_fmt);

var my_xml:XML = new XML();
my_xml.ignoreWhite = true;
my_xml.onLoad = function(success:Boolean) {
  var endTime:Number = getTimer();
  var elapsedTime:Number = endTime-startTime;
  if (success) {
    my_txt.text = "xmlDecl:"+newline+my_xml.xmlDecl+newline+newline;
    my_txt.text +=
  "contentType:"+newline+my_xml.contentType+newline+newline;
    my_txt.text +=
  "docTypeDecl:"+newline+my_xml.docTypeDecl+newline+newline;
    my_txt.text += "packet:"+newline+my_xml.toString()+newline+newline;
  } else {
```

```
    my_txt.text = "Unable to load remote XML."+newline+newline;
  }
  my_txt.text += "loaded in: "+elapsedTime+" ms.";
};
my_xml.load("http://www.helpexamples.com/crossdomain.xml");
var startTime:Number = getTimer();
```

El método `MovieClip.getNextHighestDepth()` utilizado en este ejemplo necesita Flash Player 7 o una versión posterior. Si el archivo SWF incluye un componente de la versión 2, utilice la clase DepthManager de los componentes de la versión 2 en lugar del método `MovieClip.getNextHighestDepth()`.

Véase también

`docTypeDecl (propiedad XML.docTypeDecl)`

XMLNode

```
Object
  |
  +-XMLNode
```

`public class` *XMLNode*
`extends Object`

Un documento XML se representa en Flash mediante la clase XML. Cada elemento del documento jerárquico se representa mediante un objeto XMLNode.

Disponibilidad: ActionScript 1.0; Flash Player 5

Véase también

`XML`

Resumen de propiedades

Modificadores	Propiedad	Descripción
	`attributes:Object`	Objeto que contiene todos los atributos de la instancia XML especificada.
	`childNodes:Array` [read-only]	Matriz de los elementos secundarios del objeto XML especificado.
	`firstChild:XMLNode` [read-only]	Evalúa el objeto XML especificado y hace referencia al primer elemento secundario de la lista de elementos secundarios del nodo principal.

Modificadores	Propiedad	Descripción
	`lastChild:XMLNode` [read-only]	Valor XMLNode que hace referencia al último elemento secundario de la lista de elementos secundarios del nodo.
	`localName:String` [read-only]	La parte de nombre local del nombre del nodo XML.
	`namespaceURI:String` [read-only]	Si el nodo XML tiene un prefijo, `namespaceURI` es el valor de la declaración `xmlns` para ese prefijo (el URI), que normalmente se denomina URI de espacio de nombres.
	`nextSibling:XMLNode` [read-only]	Valor XMLNode que hace referencia al siguiente elemento secundario de la lista de elementos secundarios del nodo principal.
	`nodeName:String`	Cadena que representa el nombre del nodo del objeto XML.
	`nodeType:Number` [read-only]	Valor de `nodeType`, que puede ser 1 para un elemento XML o 3 para un nodo de texto.
	`nodeValue:String`	Valor de nodo del objeto XML.
	`parentNode:XMLNode` [read-only]	Valor de XMLNode que hace referencia al nodo principal del objeto XML especificado o devuelve `null` si el nodo no tiene elemento principal.
	`prefix:String` [read-only]	La parte de prefijo del nombre del nodo XML.
	`previousSibling:XMLNode` [read-only]	Valor XMLNode que hace referencia al elemento secundario anterior de la lista de elementos secundarios del nodo principal.

Propiedades heredadas de la clase Object

```
constructor (propiedad Object.constructor), __proto__ (propiedad
Object.__proto__), prototype (propiedad Object.prototype), __resolve
(propiedad Object.__resolve)
```

Resumen de constructores

Firma	Descripción
`XMLNode(type:Number, value:String)`	El constructor XMLNode permite crear una instancia de un nodo XML basada en una cadena que especifique su contenido y en un número que represente el tipo de nodo.

Resumen de métodos

Modificadores	Firma	Descripción
	`appendChild(newChild:XMLNode) : Void`	Añade el nodo especificado a la lista secundaria del objeto XML.
	`cloneNode(deep:Boolean) : XMLNode`	Construye y devuelve un nuevo nodo XML del mismo tipo, nombre, valor y atributos que el objeto XML especificado.
	`getNamespaceForPrefix(prefix:String) : String`	Devuelve el URI de espacio de nombres asociado con el prefijo especificado para el nodo.
	`getPrefixForNamespace(nsURI:String) : String`	Devuelve el prefijo de nombre asociado con el URI de espacio de nombres especificado para el nodo.
	`hasChildNodes() : Boolean`	Especifica si el objeto XML tiene nodos secundarios.
	`insertBefore(newChild:XMLNode, insertPoint:XMLNode) : Void`	Inserta un nodo `newChild` en la lista de secundarios del objeto XML, delante del nodo `insertPoint`.
	`removeNode() : Void`	Quita el objeto XML especificado de su elemento principal.
	`toString() : String`	Evalúa el objeto XML especificado, construye una representación textual de la estructura XML, incluidos el nodo, los elementos secundarios y los atributos, y devuelve el resultado como una cadena.

Métodos heredados de la clase Object

addProperty (método Object.addProperty), hasOwnProperty (método Object.hasOwnProperty), isPropertyEnumerable (método Object.isPropertyEnumerable), isPrototypeOf (método Object.isPrototypeOf), registerClass (método Object.registerClass), toString (método Object.toString), unwatch (método Object.unwatch), valueOf (método Object.valueOf), watch (método Object.watch)

appendChild (método XMLNode.appendChild)

```
public appendChild(newChild:XMLNode) : Void
```

Añade el nodo especificado a la lista secundaria del objeto XML. Este método opera directamente en el nodo al que hace referencia el parámetro `childNode`; no añade una copia del nodo. Si el nodo que se va a añadir ya existe en otra estructura de árbol, la adición del nodo a la nueva ubicación lo eliminará de su ubicación actual. Si el parámetro `childNode` hace referencia a un nodo que ya existe en otra estructura de árbol XML, el nodo secundario añadido se sitúa en la nueva estructura de árbol tras eliminarlo de su nodo principal actual.

Disponibilidad: ActionScript 1.0; Flash Player 5

Parámetros

`newChild``:XMLNode` - Un XMLNode que representa el nodo que se va a mover de su ubicación actual a la lista de elementos secundarios del objeto *`my_xml`* .

Ejemplo

Este ejemplo hace lo que se detalla a continuación, en el orden indicado:

- Crea dos documentos XML vacíos: `doc1` y `doc2`.
- Crea un nuevo nodo mediante el método `createElement()` y lo anexa, con el método `appendChild()`, al documento XML `doc1`.
- Muestra cómo mover un nodo mediante el método `appendChild()`, moviendo el nodo raíz desde `doc1` a `doc2`.
- Copia el nodo raíz de `doc2` y lo anexa a `doc1`.
- Crea un nuevo nodo y lo anexa al nodo raíz del documento XML `doc1`.

```
var doc1:XML = new XML();
var doc2:XML = new XML();

// create a root node and add it to doc1
var rootnode:XMLNode = doc1.createElement("root");
doc1.appendChild(rootnode);
trace ("doc1: " + doc1); // output: doc1: <root />
trace ("doc2: " + doc2); // output: doc2:

// move the root node to doc2
doc2.appendChild(rootnode);
trace ("doc1: " + doc1); // output: doc1:
trace ("doc2: " + doc2); // output: doc2: <root />

// clone the root node and append it to doc1
var clone:XMLNode = doc2.firstChild.cloneNode(true);
doc1.appendChild(clone);
```

```
trace ("doc1: " + doc1); // output: doc1: <root />
trace ("doc2: " + doc2); // output: doc2: <root />

// create a new node to append to root node (named clone) of doc1
var newNode:XMLNode = doc1.createElement("newbie");
clone.appendChild(newNode);
trace ("doc1: " + doc1); // output: doc1: <root><newbie /></root>
```

attributes (propiedad XMLNode.attributes)

```
public attributes : Object
```

Objeto que contiene todos los atributos de la instancia XML especificada. El objeto XML.attributes contiene una variable para cada atributo de la instancia XML. Dado que estas variables se definen como parte del objeto, normalmente se hace referencia a ellas como propiedades del objeto. El valor de cada atributo se almacena en la correspondiente propiedad como una cadena. Por ejemplo, si tiene un atributo denominado color, podrá recuperar el valor del atributo especificando color como nombre de propiedad, como se muestra en el siguiente código:

```
var myColor:String = doc.firstChild.attributes.color;
```

Disponibilidad: ActionScript 1.0; Flash Player 5

Ejemplo

El ejemplo siguiente muestra cómo leer y escribir los atributos de un nodo XML:

```
var doc:XML = new XML("<mytag name='Val'> item </mytag>");
trace(doc.firstChild.attributes.name); // Val

doc.firstChild.attributes.order = "first";
trace (doc.firstChild); // <mytag order="first" name="Val"> item </mytag>

for (attr in doc.firstChild.attributes) {
  trace (attr + " = " + doc.firstChild.attributes[attr]);
}

// order = first
// name = Val
```

childNodes (propiedad XMLNode.childNodes)

```
public childNodes : Array [read-only]
```

Matriz de los elementos secundarios del objeto XML especificado. Cada elemento de la matriz es una referencia a un objeto XML que representa un nodo secundario. Esta es una propiedad de sólo lectura que no puede utilizarse para manipular nodos secundarios. Utilice los métodos `appendChild()`, `insertBefore()` y `removeNode()` para manipular nodos secundarios.

Esta propiedad no está definida para los nodos de texto (`nodeType == 3`).

Disponibilidad: ActionScript 1.0; Flash Player 5

Ejemplo

El ejemplo siguiente muestra cómo utilizar la propiedad XML.childNodes para devolver una matriz de nodos secundarios:

```
// create a new XML document
var doc:XML = new XML();

// create a root node
var rootNode:XMLNode = doc.createElement("rootNode");

// create three child nodes
var oldest:XMLNode = doc.createElement("oldest");
var middle:XMLNode = doc.createElement("middle");
var youngest:XMLNode = doc.createElement("youngest");

// add the rootNode as the root of the XML document tree
doc.appendChild(rootNode);

// add each of the child nodes as children of rootNode
rootNode.appendChild(oldest);
rootNode.appendChild(middle);
rootNode.appendChild(youngest);

// create an array and use rootNode to populate it
var firstArray:Array = doc.childNodes;
trace (firstArray);
// output: <rootNode><oldest /><middle /><youngest /></rootNode>

// create another array and use the child nodes to populate it
var secondArray:Array = rootNode.childNodes;
trace(secondArray);
// output: <oldest />,<middle />,<youngest />
```

Véase también

nodeType (propiedad XMLNode.nodeType), appendChild (método XMLNode.appendChild), insertBefore (método XMLNode.insertBefore), removeNode (método XMLNode.removeNode)

cloneNode (método XMLNode.cloneNode)

```
public cloneNode(deep:Boolean) : XMLNode
```

Construye y devuelve un nuevo nodo XML del mismo tipo, nombre, valor y atributos que el objeto XML especificado. Si `deep` se establece con el valor `true`, todos los elementos secundarios se clonan sucesivamente, lo que da como resultado una copia exacta del árbol de documentos del objeto original.

El clon del nodo que se devuelve ya no está asociado al árbol del elemento clonado. En consecuencia, `nextSibling`, `parentNode` y `previousSibling` tendrán el valor `null`. Si el parámetro `deep` se define como `false`, o el nodo *`my_xml`* no tiene nodos secundarios, `firstChild` y `lastChild` también son null.

Disponibilidad: ActionScript 1.0; Flash Player 5

Parámetros

`deep``:Boolean` - Un valor booleano; si es `true`, los elementos secundarios del objeto XML especificado se clonarán sucesivamente.

Valor devuelto

`XMLNode` - Un objeto XMLNode.

Ejemplo

El ejemplo siguiente muestra cómo utilizar el método `XML.cloneNode()` para crear una copia de un nodo:

```
// create a new XML document
var doc:XML = new XML();

// create a root node
var rootNode:XMLNode = doc.createElement("rootNode");

// create three child nodes
var oldest:XMLNode = doc.createElement("oldest");
var middle:XMLNode = doc.createElement("middle");
var youngest:XMLNode = doc.createElement("youngest");

// add the rootNode as the root of the XML document tree
doc.appendChild(rootNode);

// add each of the child nodes as children of rootNode
rootNode.appendChild(oldest);
rootNode.appendChild(middle);
rootNode.appendChild(youngest);
```

```
// create a copy of the middle node using cloneNode()
var middle2:XMLNode = middle.cloneNode(false);

// insert the clone node into rootNode between the middle and youngest nodes
rootNode.insertBefore(middle2, youngest);
trace(rootNode);
// output (with line breaks added):
// <rootNode>
// <oldest />
// <middle />
// <middle />
// <youngest />
// </rootNode>

// create a copy of rootNode using cloneNode() to demonstrate a deep copy
var rootClone:XMLNode = rootNode.cloneNode(true);

// insert the clone, which contains all child nodes, to rootNode
rootNode.appendChild(rootClone);
trace(rootNode);
// output (with line breaks added):
// <rootNode>
// <oldest />
// <middle />
// <middle />
// <youngest />
// <rootNode>
// <oldest />
// <middle />
// <middle />
// <youngest />
// </rootNode>
// </rootNode>
```

firstChild (propiedad XMLNode.firstChild)

```
public firstChild : XMLNode [read-only]
```

Evalúa el objeto XML especificado y hace referencia al primer elemento secundario de la lista de elementos secundarios del nodo principal. Esta propiedad tiene el valor `null` si el nodo no tiene elementos secundarios. Esta propiedad tiene el valor `undefined` si el nodo es un nodo de texto. Esta es una propiedad de sólo lectura que no puede utilizarse para manipular nodos secundarios; utilice los métodos `appendChild()`, `insertBefore()` y `removeNode()` para manipular nodos secundarios.

Disponibilidad: ActionScript 1.0; Flash Player 5

Ejemplo

El ejemplo siguiente muestra cómo utilizar `XML.firstChild` para recorrer los nodos secundarios de un nodo:

```
// create a new XML document
var doc:XML = new XML();

// create a root node
var rootNode:XMLNode = doc.createElement("rootNode");

// create three child nodes
var oldest:XMLNode = doc.createElement("oldest");
var middle:XMLNode = doc.createElement("middle");
var youngest:XMLNode = doc.createElement("youngest");

// add the rootNode as the root of the XML document tree
doc.appendChild(rootNode);

// add each of the child nodes as children of rootNode
rootNode.appendChild(oldest);
rootNode.appendChild(middle);
rootNode.appendChild(youngest);

// use firstChild to iterate through the child nodes of rootNode
for (var aNode:XMLNode = rootNode.firstChild; aNode != null; aNode =
  aNode.nextSibling) {
  trace(aNode);
}

// output:
// <oldest />
// <middle />
// <youngest />
```

El ejemplo siguiente procede del archivo FLA XML_languagePicker en el directorio Examples y se puede encontrar en la definición de función de controlador de eventos languageXML.onLoad:

```
// loop through the strings in each language node
// adding each string as a new element in the language array
for (var stringNode:XMLNode = childNode.firstChild; stringNode != null;
  stringNode = stringNode.nextSibling, j++) {
  masterArray[i][j] = stringNode.firstChild.nodeValue;
}
```

Para ver el script completo, consulte el archivo XML_languagePicker.fla de la carpeta de ejemplos de ActionScript:

- Windows: *unidad de inicio*\Archivos de programa\Macromedia\Flash 8\Samples and Tutorials\Samples\ActionScript
- Macintosh: *Disco duro de Macintosh*/Applications/Macromedia Flash 8/Samples and Tutorials/Samples/ActionScript

Véase también

`appendChild (método XMLNode.appendChild)`, `insertBefore (método XMLNode.insertBefore)`, `removeNode (método XMLNode.removeNode)`

getNamespaceForPrefix (método XMLNode.getNamespaceForPrefix)

```
public getNamespaceForPrefix(prefix:String) : Cadena
```

Devuelve el URI de espacio de nombres asociado con el prefijo especificado para el nodo. Para determinar el URI, `getPrefixForNamespace()` busca en la jerarquía de XML desde el nodo y devuelve el URI de espacio de nombres de la primera declaración `xmlns` para el `prefix` concreto.

Si no hay espacio de nombres definido para el prefijo especificado, el método devuelve `null`.

Si especifica una cadena vacía (`""`) como `prefix` y hay un espacio de nombres predeterminado definido para el nodo (como `xmlns="http://www.example.com/"`), el método devuelve el URI de espacio de nombres predeterminado.

Disponibilidad: ActionScript 1.0; Flash Player 8

Parámetros

`prefix``:String` - El prefijo para el que el método devuelve el espacio de nombres asociado.

Valor devuelto

`String` - El espacio de nombres asociado con el prefijo especificado.

Ejemplo

El siguiente ejemplo crea un objeto XML muy simple y devuelve el resultado de una llamada a `getNamespaceForPrefix()`

```
function createXML():XMLNode {
  var str:String = "<Outer xmlns:exu=\"http://www.example.com/util\">"
    + "<exu:Child id='1' />"
    + "<exu:Child id='2' />"
    + "<exu:Child id='3' />"
    + "</Outer>";
  return new XML(str).firstChild;
}

var xml:XMLNode = createXML();
trace(xml.getNamespaceForPrefix("exu")); // output: http://www.example.com/
  util
trace(xml.getNamespaceForPrefix("")); // output: null
```

Véase también

`getPrefixForNamespace (método XMLNode.getPrefixForNamespace)`, `namespaceURI (propiedad XMLNode.namespaceURI)`

getPrefixForNamespace (método XMLNode.getPrefixForNamespace)

```
public getPrefixForNamespace(nsURI:String) : Cadena
```

Devuelve el prefijo de nombre asociado con el URI de espacio de nombres especificado para el nodo. Para determinar el prefijo, `getPrefixForNamespace()` busca en la jerarquía de XML desde el nodo y devuelve el prefijo de la primera declaración `xmlns` para el URI de espacio de nombres correspondiente a `nsURI`.

Si no hay asignación de `xmlns` para el URI, el método devuelve `null`. Si hay una asignación de `xmlns` para el URI, pero no hay ningún prefijo asociado a la asignación, el método devuelve una cadena vacía (`""`).

Disponibilidad: ActionScript 1.0; Flash Player 8

Parámetros

`nsURI``:String` - El URI de espacio de nombres para el que el método devuelve el prefijo asociado.

Valor devuelto

`String` - El prefijo asociado con el espacio de nombres especificado.

Ejemplo

El siguiente ejemplo crea un objeto XML muy simple y devuelve el resultado de una llamada al método `getPrefixForNamespace()` El nodo XML `Outer`, que se representa mediante la variable `xmlDoc` define un URI de espacio de nombres y lo asigna al prefijo `exu`. Al llamar al método `getPrefixForNamespace()` con el URI de espacio de nombres ("http://www.example.com/util") se obtiene el prefijo `exu`, pero si se llama a este método con un URI no definido ("http://www.example.com/other") devuelve `null`. El primer nodo `exu:Child`, que se representa mediante la variable `child1`, define también un URI de espacio de nombres ("http://www.example.com/child"), pero no lo asigna a un prefijo. Si se llama a este método en el URI de espacio de nombres definido, pero no asignado,

```
function createXML():XMLNode {
  var str:String = "<Outer xmlns:exu=\"http://www.example.com/util\">"
    + "<exu:Child id='1' xmlns=\"http://www.example.com/child\"/>"
    + "<exu:Child id='2' />"
    + "<exu:Child id='3' />"
    + "</Outer>";
  return new XML(str).firstChild;
}

var xmlDoc:XMLNode = createXML();
trace(xmlDoc.getPrefixForNamespace("http://www.example.com/util")); //
  output: exu
trace(xmlDoc.getPrefixForNamespace("http://www.example.com/other")); //
  output: null

var child1:XMLNode = xmlDoc.firstChild;
trace(child1.getPrefixForNamespace("http://www.example.com/child")); //
  output: [empty string]
trace(child1.getPrefixForNamespace("http://www.example.com/other")); //
  output: null
```

Véase también

`getNamespaceForPrefix (método XMLNode.getNamespaceForPrefix)`, `namespaceURI (propiedad XMLNode.namespaceURI)`

hasChildNodes (método XMLNode.hasChildNodes)

`public hasChildNodes() : Booleano`

Especifica si el objeto XML tiene nodos secundarios.

Disponibilidad: ActionScript 1.0; Flash Player 5

Valor devuelto

`Boolean` - `true` si el nodo XMLNode especificado tiene uno o varios nodos secundarios; en caso contrario devuelve `false`.

Ejemplo

El ejemplo siguiente crea un nuevo paquete XML. Si el nodo raíz tiene nodos secundarios, el código se repite por todos los nodos secundarios para mostrar el nombre y el valor de cada nodo. Añada el siguiente código ActionScript al archivo FLA o AS:

```
var my_xml:XML = new XML("hankrudolph");
if (my_xml.firstChild.hasChildNodes()) {
// use firstChild to iterate through the child nodes of rootNode
  for (var aNode:XMLNode = my_xml.firstChild.firstChild; aNode != null;
  aNode=aNode.nextSibling) {
    if (aNode.nodeType == 1) {
      trace(aNode.nodeName+":\t"+aNode.firstChild.nodeValue);
    }
  }
}
```

El panel Salida muestra esta información:

```
output:
username: hank
password: rudolph
```

insertBefore (método XMLNode.insertBefore)

```
public insertBefore(newChild:XMLNode, insertPoint:XMLNode) : Void
```

Inserta un nodo `newChild` en la lista de secundarios del objeto XML, delante del nodo `insertPoint`. Si `insertPoint` *no* es un elemento secundario del objeto XMLNode, se produce un error al insertar.

Disponibilidad: ActionScript 1.0; Flash Player 5

Parámetros

`newChild``:XMLNode` - El objeto XMLNode que se va a insertar.

`insertPoint``:XMLNode` - El objeto XMLNode que seguirá al nodo `newChild` cuando se invoque el método.

Ejemplo

El ejemplo siguiente inserta un nuevo nodo XML entre los dos nodos existentes:

```
var my_xml:XML = new XML("<a>1</a>\n<c>3</c>");
var insertPoint:XMLNode = my_xml.lastChild;
var newNode:XML = new XML("<b>2</b>\n");
my_xml.insertBefore(newNode, insertPoint);
trace(my_xml);
```

Véase también

XML, cloneNode (método XMLNode.cloneNode)

lastChild (propiedad XMLNode.lastChild)

```
public lastChild : XMLNode [read-only]
```

Valor XMLNode que hace referencia al último elemento secundario de la lista de elementos secundarios del nodo. La propiedad `XML.lastChild` tiene el valor `null` si el nodo no tiene elementos secundarios. Esta propiedad no puede utilizarse para manipular nodos secundarios; utilice los métodos `appendChild()`, `insertBefore()` y `removeNode()` para manipular nodos secundarios.

Disponibilidad: ActionScript 1.0; Flash Player 5

Ejemplo

El ejemplo siguiente utiliza la propiedad `XML.lastChild` para repetir por los nodos secundarios de un nodo XML, empezando por el último elemento de la lista de elementos secundarios del nodo y terminando por el primer elemento secundario de la lista:

```
// create a new XML document
var doc:XML = new XML();

// create a root node
var rootNode:XMLNode = doc.createElement("rootNode");

// create three child nodes
var oldest:XMLNode = doc.createElement("oldest");
var middle:XMLNode = doc.createElement("middle");
var youngest:XMLNode = doc.createElement("youngest");

// add the rootNode as the root of the XML document tree
doc.appendChild(rootNode);

// add each of the child nodes as children of rootNode
rootNode.appendChild(oldest);
rootNode.appendChild(middle);
rootNode.appendChild(youngest);
```

```
// use lastChild to iterate through the child nodes of rootNode
for (var aNode:XMLNode = rootNode.lastChild; aNode != null; aNode =
  aNode.previousSibling) {
  trace(aNode);
}

// output:
// <youngest />
// <middle />
// <oldest />
```

El ejemplo siguiente crea un nuevo paquete XML y utiliza la propiedad `XML.lastChild` para repetir por los nodos secundarios del nodo raíz:

```
// create a new XML document
var doc:XML = new XML("");

var rootNode:XMLNode = doc.firstChild;

// use lastChild to iterate through the child nodes of rootNode
for (var aNode:XMLNode = rootNode.lastChild; aNode != null;
  aNode=aNode.previousSibling) {
  trace(aNode);
}

// output:
// <youngest />
// <middle />
// <oldest />
```

Véase también

appendChild (método XMLNode.appendChild), insertBefore (método XMLNode.insertBefore), removeNode (método XMLNode.removeNode), XML

localName (propiedad XMLNode.localName)

```
public localName : String [read-only]
```

La parte de nombre local del nombre del nodo XML. Es el nombre de elemento sin el prefijo de espacio de nombres. Por ejemplo, el nodo `<contact:mailbox/>bob@example.com</contact:mailbox>` tiene el nombre local "mailbox" y el prefijo "contact", que forman el nombre completo del elemento "contact.mailbox".

Puede acceder al prefijo a través de la propiedad `prefix` del objeto de nodo XML. La propiedad `nodeName` devuelve el nombre completo (incluidos el prefijo y el nombre local).

Disponibilidad: ActionScript 1.0; Flash Player 8

Ejemplo

Este ejemplo utiliza un archivo SWF y un archivo XML situados en el mismo directorio. El archivo XML, denominado "SoapSample.xml" contiene el código siguiente:

```
<?xml version="1.0"?>
<soap:Envelope xmlns:soap="http://www.w3.org/2001/12/soap-envelope">
<soap:Body xmlns:w="http://www.example.com/weather">
<w:GetTemperature>
<w:City>San Francisco</w:City>
</w:GetTemperature>
</soap:Body>
</soap:Envelope>
```

El origen del archivo SWF contiene el script siguiente (observe los comentarios de las cadenas de Salida):

```
var xmlDoc:XML = new XML()
xmlDoc.ignoreWhite = true;
xmlDoc.load("SoapSample.xml")
xmlDoc.onLoad = function(success:Boolean)
{
  var tempNode:XMLNode = xmlDoc.childNodes[0].childNodes[0].childNodes[0];
  trace("w:GetTemperature localname: " + tempNode.localName); // Output:
  ... GetTemperature
  var soapEnvNode:XMLNode = xmlDoc.childNodes[0];
  trace("soap:Envelope localname: " + soapEnvNode.localName); // Output:
  ... Envelope
}
```

namespaceURI (propiedad XMLNode.namespaceURI)

```
public namespaceURI : String [read-only]
```

Si el nodo XML tiene un prefijo, `namespaceURI` es el valor de la declaración `xmlns` para ese prefijo (el URI), que normalmente se denomina URI de espacio de nombres. La declaración `xmlns` está en el nodo actual o en un nodo de un nivel superior en la jerarquía XML.

Si el nodo XML no tiene un prefijo, el valor de la propiedad `namespaceURI` depende de si hay un espacio de nombres predeterminado definido (como en `xmlns="http://www.example.com/"`). Si hay un espacio de nombres predeterminado, el valor de la propiedad `namespaceURI` es el valor del espacio de nombres. Si no hay un espacio de nombres predeterminado, la propiedad `namespaceURI` para ese nodo es una cadena vacía (`""`).

Puede utilizar el método `getNamespaceForPrefix()` para identificar el espacio de nombres asociado con un prefijo concreto. La propiedad `namespaceURI` devuelve el prefijo asociado con el nombre de nodo.

Disponibilidad: ActionScript 1.0; Flash Player 8

Ejemplo

El ejemplo siguiente muestra cómo afecta a la propiedad `namespaceURI` el uso de prefijos. Un directorio contiene un archivo SWF y un archivo XML. El archivo XML, denominado `SoapSample.xml` contiene las siguientes etiquetas.

```
<?xml version="1.0"?>
<soap:Envelope xmlns:soap="http://www.w3.org/2001/12/soap-envelope">
<soap:Body xmlns:w="http://www.example.com/weather">
<w:GetTemperature>
<w:City>San Francisco</w:City>
</w:GetTemperature>
</soap:Body>
</soap:Envelope>
```

El origen del archivo SWF contiene el script siguiente (observe los comentarios de las cadenas de Salida). Para `tempNode`, que representa el nodo `w:GetTemperature`, el valor de `namespaceURI` está definido en la etiqueta `soap:Body`. Para `soapBodyNode`, que representa el nodo `soap:Body`, el valor de `namespaceURI` viene determinado por el prefijo `soap` del nodo situado encima, en lugar de la definición del prefijo `w` que contiene el nodo `soap:Body`.

```
var xmlDoc:XML = new XML();
xmlDoc.load("SoapSample.xml");
xmlDoc.ignoreWhite = true;
xmlDoc.onLoad = function(success:Boolean)
{
  var tempNode:XMLNode = xmlDoc.childNodes[0].childNodes[0].childNodes[0];
  trace("w:GetTemperature namespaceURI: " + tempNode.namespaceURI);
    // Output: ... http://www.example.com/weather

  trace("w:GetTemperature soap namespace: " +
  tempNode.getNamespaceForPrefix("soap"));
    // Output: ... http://www.w3.org/2001/12/soap-envelope

  var soapBodyNode:XMLNode = xmlDoc.childNodes[0].childNodes[0];
  trace("soap:Envelope namespaceURI: " + soapBodyNode.namespaceURI);
    // Output: ... http://www.w3.org/2001/12/soap-envelope
}
```

El ejemplo siguiente utiliza etiquetas XML sin prefijos. Usa un archivo SWF y un archivo XML situados en el mismo directorio. El archivo XML, denominado `NoPrefix.xml` contiene las siguientes etiquetas.

```
<?xml version="1.0"?>
<rootnode>
<simplenode xmlns="http://www.w3.org/2001/12/soap-envelope">
<innernode />
</simplenode>
</rootnode>
```

El origen del archivo SWF contiene el script siguiente (observe los comentarios de las cadenas de Salida). `rootNode` no tiene un espacio de nombres predeterminado, por lo que su valor `namespaceURI` devuelve una cadena vacía. `simpleNode` define un espacio de nombres predeterminado, por lo que su valor `namespaceURI` es dicho espacio de nombres. `innerNode` no define un espacio de nombres predeterminado, pero utiliza el espacio de nombres predeterminado definido por `simpleNode`; su valor `namespaceURI` es el mismo que el de `simpleNode`.

```
var xmlDoc:XML = new XML()
xmlDoc.load("NoPrefix.xml");
xmlDoc.ignoreWhite = true;
xmlDoc.onLoad = function(success:Boolean)
{
  var rootNode:XMLNode = xmlDoc.childNodes[0];
  trace("rootNode Node namespaceURI: " + rootNode.namespaceURI);
    // Output: [empty string]

  var simpleNode:XMLNode = xmlDoc.childNodes[0].childNodes[0];
  trace("simpleNode Node namespaceURI: " + simpleNode.namespaceURI);
    // Output: ... http://www.w3.org/2001/12/soap-envelope

  var innerNode:XMLNode = xmlDoc.childNodes[0].childNodes[0].childNodes[0];
  trace("innerNode Node namespaceURI: " + innerNode.namespaceURI);
    // Output: ... http://www.w3.org/2001/12/soap-envelope
}
```

Véase también

`getNamespaceForPrefix (método XMLNode.getNamespaceForPrefix)`,
`getPrefixForNamespace (método XMLNode.getPrefixForNamespace)`

nextSibling (propiedad XMLNode.nextSibling)

```
public nextSibling : XMLNode [read-only]
```

Valor XMLNode que hace referencia al siguiente elemento secundario de la lista de elementos secundarios del nodo principal. Esta propiedad tiene el valor `null` si el nodo no tiene otro nodo del mismo nivel. Esta propiedad no puede utilizarse para manipular nodos secundarios; utilice los métodos `appendChild()`, `insertBefore()` y `removeNode()` para manipular nodos secundarios.

Disponibilidad: ActionScript 1.0; Flash Player 5

Ejemplo

El ejemplo siguiente es un extracto del ejemplo de la propiedad `XML.firstChild` y muestra cómo se puede utilizar la propiedad `XML.nextSibling` para repetir por los nodos secundarios de un nodo XML:

```
for (var aNode:XMLNode = rootNode.firstChild; aNode != null; aNode =
  aNode.nextSibling) {
  trace(aNode);
}
```

Véase también

`firstChild (propiedad XMLNode.firstChild)`, `appendChild (método XMLNode.appendChild)`, `insertBefore (método XMLNode.insertBefore)`, `removeNode (método XMLNode.removeNode)`, `XML`

nodeName (propiedad XMLNode.nodeName)

```
public nodeName : String
```

Cadena que representa el nombre del nodo del objeto XML. Si el objeto XML es un elemento XML (`nodeType == 1`), `nodeName` es el nombre de la etiqueta que representa al nodo del archivo XML. Por ejemplo, `TITLE` es el nombre del nodo (`nodeName`) de una etiqueta HTML `TITLE`. Si el objeto XML es un nodo de texto (`nodeType == 3`), nodeName tiene el valor `null`.

Disponibilidad: ActionScript 1.0; Flash Player 5

Ejemplo

El ejemplo siguiente crea un nodo de elemento y un nodo de texto, y comprueba el nombre de cada uno:

```
// create an XML document
var doc:XML = new XML();

// create an XML node using createElement()
var myNode:XMLNode = doc.createElement("rootNode");

// place the new node into the XML tree
doc.appendChild(myNode);

// create an XML text node using createTextNode()
var myTextNode:XMLNode = doc.createTextNode("textNode");

// place the new node into the XML tree
myNode.appendChild(myTextNode);

trace(myNode.nodeName);
trace(myTextNode.nodeName);

// output:
// rootNode
// null
```

El ejemplo siguiente crea un nuevo paquete XML. Si el nodo raíz tiene nodos secundarios, el código se repite por todos los nodos secundarios para mostrar el nombre y el valor de cada nodo. Añada el siguiente código ActionScript al archivo FLA o AS:

```
var my_xml:XML = new XML("hankrudolph");
if (my_xml.firstChild.hasChildNodes()) {
  // use firstChild to iterate through the child nodes of rootNode
  for (var aNode:XMLNode = my_xml.firstChild.firstChild; aNode != null;
  aNode=aNode.nextSibling) {
    if (aNode.nodeType == 1) {
      trace(aNode.nodeName+":\t"+aNode.firstChild.nodeValue);
    }
  }
}
```

El panel Salida muestra los siguientes nombres de nodos:

```
output:
username: hank
password: rudolph
```

Véase también

nodeType (propiedad XMLNode.nodeType)

nodeType (propiedad XMLNode.nodeType)

```
public nodeType : Number [read-only]
```

Valor de `nodeType`, que puede ser 1 para un elemento XML o 3 para un nodo de texto.

`nodeType` es un valor numérico de la enumeración NodeType de la recomendación W3C DOM Nivel 1: www.w3.org/tr/1998/REC-DOM-Level-1-19981001/level-one-core.html. Los valores se muestran en la tabla siguiente:

Valor entero	Constante definida
1	ELEMENT_NODE
2	ATTRIBUTE_NODE
3	TEXT_NODE
4	CDATA_SECTION_NODE
5	ENTITY_REFERENCE_NODE
6	ENTITY_NODE
7	PROCESSING_INSTRUCTION_NODE
8	COMMENT_NODE
9	DOCUMENT_NODE
10	DOCUMENT_TYPE_NODE
11	DOCUMENT_FRAGMENT_NODE
12	NOTATION_NODE

En Flash Player, la clase XML incorporada sólo admite 1 (`ELEMENT_NODE`) y 3 (`TEXT_NODE`).

Disponibilidad: ActionScript 1.0; Flash Player 5

Ejemplo

El ejemplo siguiente crea un nodo de elemento y un nodo de texto, y comprueba los tipos de nodos:

```
// create an XML document
var doc:XML = new XML();

// create an XML node using createElement()
var myNode:XMLNode = doc.createElement("rootNode");
```

```
// place the new node into the XML tree
doc.appendChild(myNode);

// create an XML text node using createTextNode()
var myTextNode:XMLNode = doc.createTextNode("textNode");

// place the new node into the XML tree
myNode.appendChild(myTextNode);

trace(myNode.nodeType);
trace(myTextNode.nodeType);

// output:
// 1
// 3
```

Véase también

`nodeValue (propiedad XMLNode.nodeValue)`

nodeValue (propiedad XMLNode.nodeValue)

```
public nodeValue : String
```

Valor de nodo del objeto XML. Si el objeto XML es un nodo de texto, `nodeType` es 3, y `nodeValue` es el texto del nodo. Si el objeto XML es un elemento XML (`nodeType` es 1), `nodeValue` es `null` y de sólo lectura.

Disponibilidad: ActionScript 1.0; Flash Player 5

Ejemplo

El ejemplo siguiente crea un nodo de elemento y un nodo de texto, y comprueba el valor de cada uno:

```
// create an XML document
var doc:XML = new XML();

// create an XML node using createElement()
var myNode:XMLNode = doc.createElement("rootNode");

// place the new node into the XML tree
doc.appendChild(myNode);

// create an XML text node using createTextNode()
var myTextNode:XMLNode = doc.createTextNode("textNode");

// place the new node into the XML tree
myNode.appendChild(myTextNode);
```

```
trace(myNode.nodeValue);
trace(myTextNode.nodeValue);

// output:
// null
// myTextNode
```

El ejemplo siguiente crea y analiza un paquete XML: El código se reproduce indefinidamente por cada nodo secundario y muestra el valor del nodo utilizando la propiedad `firstChild` y `firstChild.nodeValue`. Cuando utiliza `firstChild` para mostrar el contenido del nodo, mantiene la entidad `&`. Sin embargo, si utiliza de forma explícita `nodeValue`, se convierte al carácter ampersand (&).

```
var my_xml:XML = new XML("mortongood&evil");
trace("using firstChild:");
for (var i = 0; i<my_xml.firstChild.childNodes.length; i++) {
  trace("\t"+my_xml.firstChild.childNodes[i].firstChild);
}
trace("");
trace("using firstChild.nodeValue:");
for (var i = 0; i<my_xml.firstChild.childNodes.length; i++) {
  trace("\t"+my_xml.firstChild.childNodes[i].firstChild.nodeValue);
}
```

El panel Salida muestra la siguiente información:

```
 using firstChild:
 morton
 good&evil

 using firstChild.nodeValue:
 morton
 good&evil
```

Véase también

`nodeType (propiedad XMLNode.nodeType)`

parentNode (propiedad XMLNode.parentNode)

```
public parentNode : XMLNode [read-only]
```

Valor de XMLNode que hace referencia al nodo principal del objeto XML especificado o devuelve `null` si el nodo no tiene elemento principal. Esta es una propiedad de sólo lectura que no puede utilizarse para manipular nodos secundarios; utilice los métodos `appendChild()`, `insertBefore()` y `removeNode()` para manipular nodos secundarios.

Disponibilidad: ActionScript 1.0; Flash Player 5

Ejemplo

El ejemplo siguiente crea un paquete XML y muestra el nodo principal del nodo username en el panel Salida:

```
var my_xml:XML = new XML("mortongood&evil");

// first child is the <login /> node
var rootNode:XMLNode = my_xml.firstChild;

// first child of the root is the <username /> node
var targetNode:XMLNode = rootNode.firstChild;
trace("the parent node of '"+targetNode.nodeName+"' is:
  "+targetNode.parentNode.nodeName);
trace("contents of the parent node are:\n"+targetNode.parentNode);
```

Salida (con saltos de línea para que se vea más claramente):

```
the parent node of 'username' is: login
contents of the parent node are:
morton
good&evil
```

Véase también

appendChild (método XMLNode.appendChild), insertBefore (método XMLNode.insertBefore), removeNode (método XMLNode.removeNode), XML

prefix (propiedad XMLNode.prefix)

```
public prefix : String [read-only]
```

La parte de prefijo del nombre del nodo XML. Por ejemplo, el nodo `<contact:mailbox/>bob@example.com</contact:mailbox>` tiene el prefijo "contact" y el nombre local "mailbox", que forman el nombre completo del elemento "contact.mailbox".

La propiedad `nodeName` de un objeto de nodo XML devuelve el nombre completo (incluidos el prefijo y el nombre local). Puede acceder a la parte del nombre local del nombre del elemento utilizando la propiedad `localName`.

Disponibilidad: ActionScript 1.0; Flash Player 8

Ejemplo

Un directorio contiene un archivo SWF y un archivo XML. El archivo XML, denominado "SoapSample.xml" contiene el código siguiente:

```
<?xml version="1.0"?>
<soap:Envelope xmlns:soap="http://www.w3.org/2001/12/soap-envelope">
```

```
<soap:Body xmlns:w="http://www.example.com/weather">
<w:GetTemperature>
<w:City>San Francisco</w:City>
</w:GetTemperature>
</soap:Body>
</soap:Envelope>
```

El origen del archivo SWF contiene el script siguiente (observe los comentarios de las cadenas de Salida):

```
var xmlDoc:XML = new XML();
xmlDoc.ignoreWhite = true;
xmlDoc.load("SoapSample.xml");
xmlDoc.onLoad = function(success:Boolean)
{
  var tempNode:XMLNode = xmlDoc.childNodes[0].childNodes[0].childNodes[0];
  trace("w:GetTemperature prefix: " + tempNode.prefix); // Output: ... w
  var soapEnvNode:XMLNode = xmlDoc.childNodes[0];
  trace("soap:Envelope prefix: " + soapEnvNode.prefix); // Output: ... soap
}
```

previousSibling (propiedad XMLNode.previousSibling)

```
public previousSibling : XMLNode [read-only]
```

Valor XMLNode que hace referencia al elemento secundario anterior de la lista de elementos secundarios del nodo principal. La propiedad tiene el valor null si el nodo no tiene un nodo del mismo nivel que le preceda. Esta propiedad no puede utilizarse para manipular nodos secundarios; utilice los métodos `appendChild()`, `insertBefore()` y `removeNode()` para manipular nodos secundarios.

Disponibilidad: ActionScript 1.0; Flash Player 5

Ejemplo

El ejemplo siguiente es un extracto del ejemplo de la propiedad `XML.lastChild` y muestra cómo se puede utilizar la propiedad `XML.previousSibling` para reproducirse indefinidamente por los nodos secundarios de un nodo XML:

```
for (var aNode:XMLNode = rootNode.lastChild; aNode != null; aNode =
  aNode.previousSibling) {
  trace(aNode);
}
```

Véase también

lastChild (propiedad XMLNode.lastChild), appendChild (método XMLNode.appendChild), insertBefore (método XMLNode.insertBefore), removeNode (método XMLNode.removeNode), XML

removeNode (método XMLNode.removeNode)

```
public removeNode() : Void
```

Quita el objeto XML especificado de su elemento principal. También elimina todos los descendientes del nodo.

Disponibilidad: ActionScript 1.0; Flash Player 5

Ejemplo

El ejemplo siguiente crea un paquete XML y, a continuación, elimina el objeto XML especificado y sus nodos descendientes:

```
var xml_str:String = "<state name=\"California\"><city>San Francisco</
  city></state>";

var my_xml:XML = new XML(xml_str);
var cityNode:XMLNode = my_xml.firstChild.firstChild;
trace("before XML.removeNode():\n"+my_xml);
cityNode.removeNode();
trace("");
trace("after XML.removeNode():\n"+my_xml);

// output (line breaks added for clarity):
//
// before XML.removeNode():
// <state name="California">
// <city>San Francisco</city>
// </state>
//
// after XML.removeNode():
// <state name="California" />
```

toString (método XMLNode.toString)

```
public toString() : Cadena
```

Evalúa el objeto XML especificado, construye una representación textual de la estructura XML, incluidos el nodo, los elementos secundarios y los atributos, y devuelve el resultado como una cadena.

En el caso de objetos XML de nivel superior (los creados con el constructor), el método `XML.toString()` genera la declaración XML del documento (almacenada en la propiedad `XML.xmlDecl`), seguida de la declaración `DOCTYPE` del documento (almacenada en la propiedad `XML.docTypeDecl`), seguida de la representación textual de todos los nodos XML del objeto. La declaración XML no se genera si la propiedad `XML.xmlDecl` no está definida. La declaración `DOCTYPE` no se genera si la propiedad `XML.docTypeDecl` no está definida (`undefined`).

Disponibilidad: ActionScript 1.0; Flash Player 5

Valor devuelto

`String` - Cadena.

Ejemplo

El código siguiente utiliza el método `toString()` para convertir un objeto XMLNode en una cadena (String) y, a continuación, utiliza el método `toUpperCase()` de la clase String:

```
var xString = "<first>Mary</first>"
    + "<last>Ng</last>"
var my_xml:XML = new XML(xString);
var my_node:XMLNode = my_xml.childNodes[1];
trace(my_node.toString().toUpperCase());
    // output: <LAST>NG</LAST>
```

Véase también

`docTypeDecl (propiedad XML.docTypeDecl)`, `xmlDecl (propiedad XML.xmlDecl)`

Constructor XMLNode

```
public XMLNode(type:Number, value:String)
```

El constructor XMLNode permite crear una instancia de un nodo XML basada en una cadena que especifique su contenido y en un número que represente el tipo de nodo.

Disponibilidad: ActionScript 1.0; Flash Player 8

Parámetros

`type``:Number` - Un entero que representa el tipo de nodo:

Valor entero	Constante definida
1	ELEMENT_NODE
2	ATTRIBUTE_NODE

Valor entero	Constante definida
3	TEXT_NODE
4	CDATA_SECTION_NODE
5	NTITY_REFERENCE_NODE
6	ENTITY_NODE
7	PROCESSING_INSTRUCTION_NODE
8	COMMENT_NODE
9	DOCUMENT_NODE
10	DOCUMENT_TYPE_NODE
11	DOCUMENT_FRAGMENT_NODE
12	NOTATION_NODE

En Flash Player, la clase XML sólo admite tipos de nodo 1 (ELEMENT_NODE) y 3 (TEXT_NODE).

value:String - Para un nodo de texto, éste es el texto del nodo; para un nodo de elemento, es el contenido de la etiqueta.

Ejemplo

```
var ELEMENT_NODE:Number = 1;
var node1:XMLNode = new XMLNode(ELEMENT_NODE, "fullName");

var TEXT_NODE:Number = 3;
var node2:XMLNode = new XMLNode(TEXT_NODE, "Justin Case");

// Create a new XML document
var doc:XML = new XML();

// Create a root node
var rootNode:XMLNode = doc.createElement("root");

// Add the rootNode as the root of the XML document tree
doc.appendChild(rootNode);

// Build the rest of the document:
rootNode.appendChild(node1);
node1.appendChild(node2);

trace(doc);

// Output: Justin Case
```

XMLSocket

```
Object
  |
  +-XMLSocket
```

```
public class XMLSocket
extends Object
```

La clase XMLSocket implementa sockets de cliente que permiten al equipo que ejecuta Flash Player comunicarse con un equipo servidor identificado mediante una dirección IP o un nombre de dominio. La clase XMLSocket es útil para aplicaciones de servidor que requieran una baja latencia, como los sistemas de chat en tiempo real. Una solución de chat tradicional basada en HTTP suele sondear el servidor y descargar los mensajes nuevos mediante una solicitud HTTP. Por contra, una solución de chat con XMLSocket mantiene una conexión abierta con el servidor, lo que permite que el servidor envíe de inmediato los mensajes entrantes sin que se produzca una solicitud del cliente. Para utilizar la clase XMLSocket, el equipo servidor debe ejecutar un demonio que entienda el protocolo utilizado por la clase XMLSocket. El protocolo se describe en la siguiente lista:

- Los mensajes XML se envían a través de una conexión de socket ininterrumpida TCP/IP dúplex.
- Cada mensajes XML es un documento XML completo terminado en un byte cero (0).
- Pueden enviarse y recibirse un número ilimitado de mensajes XML a través de una misma conexión XMLSocket.

Las siguientes restricciones afectan a la forma y el lugar en el que el objeto XMLSocket puede conectar con el servidor:

- El método XMLSocket.connect() puede conectar sólo números de puertos TCP superiores o iguales a 1024. Como consecuencia de esta limitación, el servidor daemons que se comunica con el objeto XMLSocket también debe asignarse a números de puerto superiores o iguales a 1024. Los números de puerto inferiores a 1024 suelen utilizarse para servicios del sistema como FTP, Telnet y HTTP, por lo que los objetos XMLSocket no pueden utilizar estos puertos por razones de seguridad. La restricción de número de puerto limita la posibilidad de que se pueda acceder y hacer un uso indebido de estos recursos.
- El método XMLSocket.connect() sólo puede conectar con equipos del mismo dominio en el que reside el archivo SWF. Esta restricción no afecta a los archivos SWF que se ejecutan desde un disco local. (Esta restricción es idéntica a las reglas de seguridad para loadVariables(), XML.sendAndLoad() y XML.load().) Para conectar con un dominio de servidor que se ejecuta en un dominio diferente a aquél en el que reside el archivo SWF, puede crear un archivo de política de seguridad en el servidor que permita el acceso desde determinados dominios.

La configuración de un servidor para que se comunique con el objeto XMLSocket puede resultar compleja. Si su aplicación no requiere interactividad en tiempo real, utilice la función loadVariables() o conectividad de servidor XML basada en HTTP de Flash (XML.load(), XML.sendAndLoad(), XML.send()) en lugar de la clase XMLSocket. Para utilizar los métodos de la clase XMLSocket, primero debe utilizar el constructor, new XMLSocket, para crear un objeto XMLSocket.

Disponibilidad: ActionScript 1.0; Flash Player 5

Resumen de propiedades

Propiedades heredadas de la clase Object

```
constructor (propiedad Object.constructor), __proto__ (propiedad
Object.__proto__), prototype (propiedad Object.prototype), __resolve
(propiedad Object.__resolve)
```

Resumen de eventos

Evento	Descripción
`onClose = function() {}`	Se invoca sólo cuando el servidor cierra una conexión abierta.
`onConnect = function(success :Boolean) {}`	Lo invoca Flash Player cuando una solicitud de conexión iniciada a través de `XMLSocket.connect()` ha sido correcta o incorrecta.
`onData = function(src:Str ing) {}`	Se invoca cuando se ha descargado del servidor un mensaje terminado en byte cero (0).
`onXML = function(src:XML ) {}`	Lo invoca Flash Player cuando el objeto XML que contiene un documento XML llega a través de una conexión XMLSocket abierta.

Resumen de constructores

Firma	Descripción
`XMLSocket()`	Crea un nuevo objeto XMLSocket

Resumen de métodos

Modificadores	Firma	Descripción
	`close() : Void`	Cierra la conexión especificada por el objeto XMLSocket.
	`connect(url:String, port:Number) : Boolean`	Establece una conexión con el servidor de Internet especificado utilizando el puerto TCP indicado (que debe ser mayor o igual a 1024) y devuelve `true` o `false`,, dependiendo de si se ha establecido correctamente una conexión.
	`send(data:Object) : Void`	Convierte el objeto XML o los datos especificados en el parámetro `object` en una cadena y la transmite al servidor seguida de un byte cero (0).

Métodos heredados de la clase Object

`addProperty (método Object.addProperty)`, `hasOwnProperty (método Object.hasOwnProperty)`, `isPropertyEnumerable (método Object.isPropertyEnumerable)`, `isPrototypeOf (método Object.isPrototypeOf)`, `registerClass (método Object.registerClass)`, `toString (método Object.toString)`, `unwatch (método Object.unwatch)`, `valueOf (método Object.valueOf)`, `watch (método Object.watch)`

close (método XMLSocket.close)

`public close() : Void`

Cierra la conexión especificada por el objeto XMLSocket.

Disponibilidad: ActionScript 1.0; Flash Player 5

Ejemplo

El ejemplo siguiente crea un objeto XMLSocket, intenta conectarse al servidor y, a continuación, cierra la conexión.

```
var socket:XMLSocket = new XMLSocket();
socket.connect(null, 2000);
socket.close();
```

Véase también

`connect (método XMLSocket.connect)`

connect (método XMLSocket.connect)

```
public connect(url:String, port:Number) : Booleano
```

Establece una conexión con el servidor de Internet especificado utilizando el puerto TCP indicado (que debe ser mayor o igual a 1024) y devuelve `true` o `false`,, dependiendo de si se ha establecido correctamente una conexión. Si desconoce el número de puerto de su equipo servidor de Internet, póngase en contacto con el administrador de la red.

Si especifica el valor `null` para el parámetro `host`, el servidor con el que se entra en contacto es en el que reside el archivo SWF que llama a `XMLSocket.connect()`. Por ejemplo, si el archivo SWF se ha descargado de www.yoursite.com, especificar `null` para el parámetro host equivaldrá a introducir la dirección IP de www.yoursite.com.

En archivos SWF que se ejecuten en una versión del reproductor anterior a Flash Player 7, `host` debe estar en el mismo superdominio que el archivo SWF que emite esta llamada. Por ejemplo, un archivo SWF situado en www.someDomain.com puede cargar variables de un archivo SWF situado en store.someDomain.com porque ambos archivos se encuentran en el mismo superdominio, denominado someDomain.com.

En archivos SWF de cualquier versión que se ejecuten en Flash Player 7 o posterior, `host` debe estar exactamente en el mismo dominio. Por ejemplo, un archivo SWF situado en www.someDomain.com publicado para Flash Player 5, pero abierto en Flash Player 7 o posterior sólo puede cargar variables de archivos SWF que también se encuentren en www.someDomain.com. Si desea cargar variables de un dominio diferente, puede colocar un *archivo de política para distintos dominios* en el servidor en el que se alberga el archivo SWF al que se está accediendo.

Cuando se ejecuta el método `load()`, la propiedad `loaded` del objeto XML se establece con el valor `false`. Cuando terminan de descargarse los datos XML, la propiedad `loaded` se establece con el valor `true` y se invoca el controlador de eventos `onLoad`. Los datos XML no se analizan hasta que terminan de descargarse. Si el objeto XML contenía anteriormente árboles XML, estos se descartarán.

Si `XMLSocket.connect()` devuelve el valor `true`, la etapa inicial del proceso de conexión habrá sido correcta; posteriormente, se invoca al método `XMLSocket.onConnect` para determinar si la conexión final ha sido correcta o incorrecta. Si `XMLSocket.connect()` devuelve `false`, ello indicará que no se ha podido establecer una conexión.

Cuando utilice este método puede usar el modelo de seguridad de Flash Player.

- En el caso de Flash Player 8, no se admite el método XMLSocket.connect() si el archivo SWF que realiza la llamada se encuentra en el entorno limitado local con sistema de archivos.

- Para Flash Player 7 y versiones posteriores, los sitios Web pueden permitir el acceso de distintos dominios a un recurso mediante el despliegue de un archivo de política de varios dominios.

Para más información, consulte lo siguiente:

- Capítulo 17, "Aspectos básicos de la seguridad," en *Aprendizaje de ActionScript 2.0 en Flash*
- El documento técnico sobre la seguridad de Flash Player 8 en http://www.macromedia.com/go/fp8_security
- El documento técnico sobre interfaces API relativas a seguridad de Flash Player 8 en http://www.macromedia.com/go/fp8_security_apis

Disponibilidad: ActionScript 1.0; Flash Player 5

Parámetros

url:String - Una cadena; un nombre de dominio DNS completo o una dirección IP con el formato *aaa.bbb.ccc.ddd*. También puede especificar `null` para conectarse al servidor host en el que se encuentra el archivo SWF. Si el archivo SWF que realiza esta llamada se ejecuta en un navegador Web, el valor `host` debe pertenecer al mismo dominio que el archivo SWF; para ver más detalles, consulte la información acerca de las limitaciones de dominios para archivos SWF en la descripción principal de este método.

port:Number - Un número; el número de puerto TCP en el host utilizado para establecer una conexión. El número de puerto debe ser 1024 o superior.

Valor devuelto

`Boolean` - `true` si la conexión se realiza correctamente; `false` en caso contrario.

Ejemplo

El ejemplo siguiente utiliza `XMLSocket.connect()` para conectarse al host donde reside el archivo SWF y `trace` para obtener el valor que indica si se ha realizado correctamente la conexión:

```
var socket:XMLSocket = new XMLSocket()
socket.onConnect = function (success:Boolean) {
  if (success) {
  trace ("Connection succeeded!")
  } else {
  trace ("Connection failed!")
  }
}
```

```
if (!socket.connect(null, 2000)) {
  trace ("Connection failed!")
}
```

Véase también

onConnect (controlador XMLSocket.onConnect), Sentencia function

onClose (controlador XMLSocket.onClose)

```
onClose = function() {}
```

Se invoca sólo cuando el servidor cierra una conexión abierta. En la implementación predeterminada, este método no realiza ninguna acción. Para sustituir la implementación predeterminada, debe asignar una función que contenga acciones personalizadas.

Disponibilidad: ActionScript 1.0; Flash Player 5

Ejemplo

El ejemplo siguiente ejecuta una sentencia trace si el servidor cierra una conexión abierta:

```
var socket:XMLSocket = new XMLSocket();
socket.connect(null, 2000);
socket.onClose = function () {
  trace("Connection to server lost.");
}
```

Véase también

onConnect (controlador XMLSocket.onConnect), Sentencia function

onConnect (controlador XMLSocket.onConnect)

```
onConnect = function(success:Boolean) {}
```

Lo invoca Flash Player cuando una solicitud de conexión iniciada a través de `XMLSocket.connect()` ha sido correcta o incorrecta. Si la conexión se realiza correctamente, el parámetro `success` es `true`; en caso contrario, el parámetro `success` es `false`.

En la implementación predeterminada, este método no realiza ninguna acción. Para sustituir la implementación predeterminada, debe asignar una función que contenga acciones personalizadas.

Disponibilidad: ActionScript 1.0; Flash Player 5

Parámetros

success:Boolean - Un valor booleano que indica si la conexión de socket se ha realizado correctamente (true o false).

Ejemplo

El ejemplo siguiente muestra el proceso para especificar una función de sustitución para el método onConnect en una aplicación de chat sencilla.

Tras crear el objeto XMLSocket mediante el método constructor, el script define la función personalizada que se va a ejecutar cuando se invoque el controlador de eventos onConnect. La función controla la pantalla que ven los usuarios, dependiendo de si se ha realizado la conexión correctamente. Si la conexión se ha realizado correctamente, los usuarios verán la pantalla de chat principal en el fotograma startChat. Si la conexión no se ha realizado, los usuarios verán una pantalla con información para resolver problemas en el fotograma connectionFailed.

```
var socket:XMLSocket = new XMLSocket();
socket.onConnect = function (success) {
  if (success) {
    gotoAndPlay("startChat");
  } else {
    gotoAndStop("connectionFailed");
  }
}
```

Por último, se inicia la conexión. Si connect() devuelve false, el archivo SWF se envía directamente al fotograma connectionFailed, y nunca se llama a onConnect. Si connect() devuelve true, el archivo SWF salta a un fotograma waitForConnection, que es la pantalla "Please wait" (Espere por favor). El archivo SWF sigue en el fotograma waitForConnection hasta que se llama al controlador onConnect, lo que ocurre en algún momento posterior según la latencia de la red.

```
if (!socket.connect(null, 2000)) {
  gotoAndStop("connectionFailed");
} else {
  gotoAndStop("waitForConnection");
}
```

Véase también

connect (método XMLSocket.connect), Sentencia function

onData (controlador XMLSocket.onData)

```
onData = function(src:String) {}
```

Se invoca cuando se ha descargado del servidor un mensaje terminado en byte cero (0). Puede sustituir `XMLSocket.onData` para interceptar los datos enviados por el servidor sin analizarlos como XML. Esto resulta útil si transmite paquetes de datos con formato arbitrario y prefiere manipularlos directamente cuando lleguen en lugar de que Flash Player los analice como XML.

De forma predeterminada, el método `XMLSocket.onData` llama al método `XMLSocket.onXML`. Si sustituye `XMLSocket.onData` por un comportamiento personalizado, no se llamará a `XMLSocket.onXML` a no ser que efectúe la llamada en su implementación de `XMLSocket.onData`.

Disponibilidad: ActionScript 1.0; Flash Player 5

Parámetros

`src``:String` - Una cadena que contiene los datos enviados por el servidor.

Ejemplo

En este ejemplo, el parámetro `src` es una cadena que contiene texto XML descargado del servidor. El terminador byte cero (0) no se incluye en la cadena.

```
XMLSocket.prototype.onData = function (src) {
  this.onXML(new XML(src));
}
```

onXML (controlador XMLSocket.onXML)

```
onXML = function(src:XML) {}
```

Lo invoca Flash Player cuando el objeto XML que contiene un documento XML llega a través de una conexión XMLSocket abierta. Una conexión XMLSocket puede utilizarse para transferir un número ilimitado de documentos XML entre el cliente y el servidor. Cada documento termina en un byte cero (0). Cuando Flash Player recibe el byte cero, analiza todos los datos XML recibidos desde el anterior byte cero o desde que se estableció la conexión si éste es el primer mensaje recibido. Cada lote de datos XML analizados se considera como un único documento XML y se analiza con el método `onXML`.

En la implementación predeterminada, este método no realiza ninguna acción. Para sustituir la implementación predeterminada, debe asignar una función que contenga acciones definidas por usted.

Disponibilidad: ActionScript 1.0; Flash Player 5

Parámetros

src:XML - Un objeto XML que contiene un documento XML analizado recibido de un servidor.

Ejemplo

La siguiente función sustituye la implementación predeterminada del método onXML en una aplicación de chat sencilla. La función myOnXML indica a la aplicación de chat que reconozca un elemento XML único, MESSAGE, con el formato siguiente.

```
<MESSAGE USER="John" TEXT="Hello, my name is John!" />.
```

Se supone que la función siguiente displayMessage() es una función definida por el usuario que muestra el mensaje que recibe el usuario:

```
var socket:XMLSocket = new XMLSocket();
socket.onXML = function (doc) {
  var e = doc.firstChild;
  if (e != null && e.nodeName == "MESSAGE") {
    displayMessage(e.attributes.user, e.attributes.text);
  }
}
```

Véase también

Sentencia function

send (método XMLSocket.send)

```
public send(data:Object) : Void
```

Convierte el objeto XML o los datos especificados en el parámetro object en una cadena y la transmite al servidor seguida de un byte cero (0). Si object es un objeto XML, la cadena es una representación textual XML del objeto XML. La operación de envío (send) es asíncrona; se devuelve inmediatamente, pero puede que los datos se transmitan posteriormente. El método XMLSocket.send() no devuelve ningún valor que indique si los datos se han transmitido correctamente.

Si el objeto *myXMLSocket* no está conectado al servidor (mediante XMLSocket.connect()), se producirá un error en la operación *myXMLSocket* .

Disponibilidad: ActionScript 1.0; Flash Player 5

Parámetros

data:Object - Un objeto XML u otros datos que se van a transmitir al servidor.

Ejemplo

El ejemplo siguiente muestra cómo especificar un nombre de usuario y una contraseña para enviar el objeto XML `my_xml` al servidor:

```
var myXMLSocket:XMLSocket = new XMLSocket();
var my_xml:XML = new XML();
var myLogin:XMLNode = my_xml.createElement("login");
myLogin.attributes.username = usernameTextField;
myLogin.attributes.password = passwordTextField;
my_xml.appendChild(myLogin);
myXMLSocket.send(my_xml);
```

Véase también

`connect (método XMLSocket.connect)`

Constructor XMLSocket

```
public XMLSocket()
```

Crea un nuevo objeto XMLSocket El objeto XMLSocket no está conectado inicialmente a ningún servidor. Debe llamar a XMLSocket.connect() para conectar el objeto con un servidor.

Disponibilidad: ActionScript 1.0; Flash Player 5

Ejemplo

En el ejemplo siguiente se crea un objeto XMLSocket:

```
var socket:XMLSocket = new XMLSocket();
```

XMLUI

```
Object
  |
  +-XMLUI
```

```
public class XMLUI
extends Object
```

El objeto XMLUI permite la comunicación con archivos SWF que se utilizan como interfaces de usuario personalizadas para las funciones de extensibilidad de la herramienta de edición de Flash (como, por ejemplo, Comportamientos, Comandos, Efectos y Herramientas).

Macromedia Flash MX 2004 y Macromedia Flash MX Professional 2004 se suministran con varias funciones de extensibilidad que incluyen Comportamientos, Comandos (API JavaScript), Efectos y Herramientas. Mediante estas funciones, los usuarios avanzados pueden ampliar o automatizar la funcionalidad de la herramienta de edición. El motor de XML a interfaz de usuario funciona con todas estas funciones de extensibilidad para crear cuadros de diálogo que el usuario ve si la extensión requiere o acepta parámetros. Pueden definirse cuadros de diálogo mediante etiquetas XML o mediante la creación de un SWF para que se muestre. El objeto XMLUI proporciona un mecanismo mediante el cual un usuario avanzado puede comunicarse con un archivo SWF de esta forma.

Disponibilidad: ActionScript 1.0; Flash Player 7

Resumen de propiedades

Propiedades heredadas de la clase Object

```
constructor (propiedad Object.constructor), __proto__ (propiedad
Object.__proto__), prototype (propiedad Object.prototype), __resolve
(propiedad Object.__resolve)
```

Resumen de métodos

Modificadores	Firma	Descripción
`static`	`accept() : Void`	Provoca la salida del cuadro de diálogo XMLUI actual con el estado "accept".
`static`	`cancel() : Void`	Provoca la salida del cuadro de diálogo XMLUI actual con el estado "cancel".
`static`	`get(name:String) : String`	Recupera el valor de la propiedad especificada del cuadro de diálogo XMLUI actual.
`static`	`set(name:String, value:String) : Void`	Modifica el valor de la propiedad especificada del cuadro de diálogo XMLUI actual.

Métodos heredados de la clase Object

```
addProperty (método Object.addProperty), hasOwnProperty (método
Object.hasOwnProperty), isPropertyEnumerable (método
Object.isPropertyEnumerable), isPrototypeOf (método Object.isPrototypeOf),
registerClass (método Object.registerClass), toString (método
Object.toString), unwatch (método Object.unwatch), valueOf (método
Object.valueOf), watch (método Object.watch)
```

accept (método XMLUI.accept)

```
public static accept() : Void
```

Provoca la salida del cuadro de diálogo XMLUI actual con el estado "accept". Equivale a que el usuario haga clic en el botón Aceptar.

Disponibilidad: ActionScript 1.0; Flash Player 7

cancel (método XMLUI.cancel)

```
public static cancel() : Void
```

Provoca la salida del cuadro de diálogo XMLUI actual con el estado "cancel". Equivale a que el usuario haga clic en el botón Cancelar.

Disponibilidad: ActionScript 1.0; Flash Player 7

get (método XMLUI.get)

```
public static get(name:String) : Cadena
```

Recupera el valor de la propiedad especificada del cuadro de diálogo XMLUI actual.

Disponibilidad: ActionScript 1.0; Flash Player 7

Parámetros

name:String - El nombre de la propiedad XMLUI que se va a recuperar.

Valor devuelto

String - Devuelve el valor de la propiedad como una cadena.

set (método XMLUI.set)

```
public static set(name:String, value:String) : Void
```

Modifica el valor de la propiedad especificada del cuadro de diálogo XMLUI actual.

Disponibilidad: ActionScript 1.0; Flash Player 7

Parámetros

name:String - El nombre de la propiedad XMLUI que se va a modificar.

value:String - El valor que se va a definir para la propiedad especificada.

CAPÍTULO 3

Código ActionScript desfasado

3

Debido a la evolución de ActionScript muchos de los elementos del lenguaje han quedado desfasados. En esta sección se incluyen los elementos desfasados y, si existen, se sugieren alternativas. Aunque los elementos desfasados aún funcionan en Flash Player 8, Macromedia recomienda que no se sigan usando en el código. No se garantiza el servicio de soporte para elementos desfasados en el futuro.

Resumen de clases desfasadas

Modificadores	Nombre de clase	Descripción
	Color	*Desfasada* a partir de Flash Player 8. La clase Color está desfasada y en su lugar debe utilizarse la clase flash.geom.ColorTransform.

Resumen de funciones desfasadas

Modificadores	Nombre de función	Descripción
	call(frame:Object)	*Desfasada* desde Flash Player 5. Esta acción está desfasada y en su lugar debe utilizarse la sentencia `function`.
	chr(number:Number)	*Desfasada* desde Flash Player 5. Esta función está desfasada y en su lugar debe utilizarse `String.fromCharCode()`.
	TextFormat.getTextExtent(text:String, [width:Number])	*Desfasada* desde Flash Player 8. No hay opción sustitutiva.

Modificadores	Nombre de función	Descripción
	ifFrameLoaded([scene:String], frame:Object)	*Desfasada* desde Flash Player 5. Esta función está desfasada. Macromedia recomienda que utilice la propiedad `MovieClip._framesloaded`.
	int(value:Number)	*Desfasada* desde Flash Player 5. Esta función está desfasada y en su lugar debe utilizarse `Math.round()`.
	length(expression:String, variable:Object)	*Desfasada* desde Flash Player 5. Esta función, junto con todas las funciones string, está desfasada. Macromedia recomienda que utilice los métodos de la clase String y la propiedad `String.length` para realizar las mismas operaciones.
	mbchr(number:Number)	*Desfasada* desde Flash Player 5. Esta función está desfasada y en su lugar debe utilizarse el método `String.fromCharCode()`.
	mblength(string:String)	*Desfasada* desde Flash Player 5. Esta función está desfasada y en su lugar deben utilizarse los métodos y propiedades de la clase String.
	mbord(character:String)	*Desfasada* desde Flash Player 5. Esta función está desfasada y en su lugar debe utilizarse `String.charCodeAt()`.
	mbsubstring(value:String, index:Number, count:Number)	*Desfasada* desde Flash Player 5. Esta función está desfasada y en su lugar debe utilizarse `String.substr()`.
	ord(character:String)	*Desfasada* desde Flash Player 5. Esta función está desfasada y en su lugar deben utilizarse los métodos y propiedades de la clase String.
	random(value:Number)	*Desfasada* desde Flash Player 5. Esta función está desfasada y en su lugar debe utilizarse `Math.random()`.
	substring(string:String, index:Number, count:Number)	*Desfasada* desde Flash Player 5. Esta función está desfasada y en su lugar debe utilizarse `String.substr()`.
	tellTarget(target:String, statement(s))	*Desfasada* desde Flash Player 5. Macromedia recomienda el uso de notación con puntos (.) y la sentencia `with`.
	toggleHighQuality()	*Desfasada* desde Flash Player 5. Esta función está desfasada y en su lugar debe utilizarse `_quality`.

Resumen de propiedades desfasadas

Modificadores	Nombre de la propiedad	Descripción
	Button._highquality	*Desfasada* desde Flash Player 7. Esta propiedad está desfasada y en su lugar debe utilizarse `Button._quality`.
	MovieClip._highquality	*Desfasada* desde Flash Player 7. Esta propiedad está desfasada y en su lugar debe utilizarse `MovieClip._quality`.
	TextField._highquality	*Desfasada* desde Flash Player 7. Esta propiedad está desfasada y en su lugar debe utilizarse `TextField._quality`.
	_highquality	*Desfasada* desde Flash Player 5. Esta propiedad está desfasada y en su lugar debe utilizarse `_quality`.
	maxscroll	*Desfasada* desde Flash Player 5. Esta propiedad está desfasada y en su lugar debe utilizarse `TextField.maxscroll`.
	scroll	*Desfasada* desde Flash Player 5. Esta propiedad está desfasada y en su lugar debe utilizarse `TextField.scroll`.

Resumen de operadores desfasados

Operador	Descripción
<> (desigualdad)	*Desfasado* desde Flash Player 5. Este operador está desfasado. Macromedia recomienda que utilice el operador `!= (inequality)`.
suma (concatenación (cadenas))	*Desfasado* desde Flash Player 5. Macromedia recomienda que utilice el operador de suma (+) cuando cree contenido para Flash Player 5 o versiones posteriores. Este operador no se admite en Flash Player 8 ni en versiones posteriores.
and (AND lógico)	*Desfasado* desde Flash Player 5. Macromedia recomienda el uso del operador lógico AND (`&&`).
eq (igualdad (cadenas))	*Desfasado* desde Flash Player 5. Este operador está desfasado y en su lugar debe utilizarse `== (equality)`.

Operador	Descripción
ge (mayor o igual que (cadenas))	*Desfasado* desde Flash Player 5. Este operador está desfasado y en su lugar debe utilizarse >= (mayor o igual que).
gt (mayor que (cadenas))	*Desfasado* desde Flash Player 5. Este operador está desfasado y en su lugar debe utilizarse > (mayor que).
le (menor o igual que (cadenas))	*Desfasado* desde Flash Player 5. Este operador está desfasado y en su lugar debe utilizarse <= (menor o igual que).
lt (menor que (cadenas))	*Desfasado* desde Flash Player 5. Este operador está desfasado y en su lugar debe utilizarse < (menor que).
ne (distinto (cadenas))	*Desfasado* desde Flash Player 5. Este operador está desfasado y en su lugar debe utilizarse `!= (inequality)`.
not (NOT lógico)	*Desfasado* desde Flash Player 5. Este operador está desfasado y en su lugar debe utilizarse `! (logical NOT)`.
or (OR lógico)	*Desfasado* desde Flash Player 5. Este operador está desfasado y en su lugar debe utilizarse `|| (logical OR)`.

Índice alfabético

Símbolos

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Y