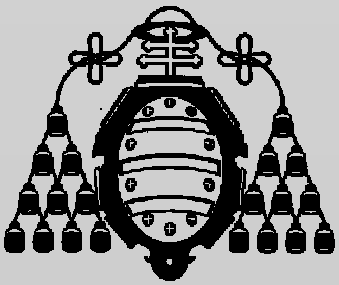


Diseño gráfico de páginas web

HTML 4.0, hojas de estilo y uso de JavaScript en HTML dinámico

EJERCICIOS

Universidad de Oviedo
Área de Expresión Gráfica en la Ingeniería

Abril-Mayo de 2000
E.T.S.I.I.I.G. Campus de Gijón

Daniel Gayo Avello

Índice Módulo 0 (Presentación de la documentación)

0. Presentación de la documentación

Bienvenido al curso “Diseño Gráfico en el web”. La presente documentación está dividida en varios módulos, donde cada módulo se compone de varios capítulos relacionados entre sí, excepto este módulo que sólo contiene un capítulo.

0.1 Contenidos de la documentación

La documentación se organiza en cuatro módulos que se describen a continuación. Los capítulos que integran los tres últimos módulos se pueden consultar en la página 0-2, en la sección titulada “Contenidos de los capítulos”.

0.1.1 Descripción de los módulos

Módulo 0: Presentación de la documentación

La parte que está leyendo en estos momentos. Contiene información relativa a la organización de la documentación para que tenga una visión de conjunto del curso y pueda acceder lo más cómodamente a la información que requiera. También incluye los ejercicios del curso, una sección de referencias y un glosario.

Módulo 1: HTML 4.0

El segundo módulo del documento introduce las bases mínimas para poder comenzar a desarrollar una página *web*; esto es, el lenguaje de descripción de documentos y algunos conceptos relacionados con páginas web “estáticas”. Consta de nueve pequeños capítulos. El capítulo 1 sirve de introducción al lector a los conceptos de *web*, *URI* o *HTML*; presentándolos de una forma lógica y procurando dar una perspectiva histórica a los mismos. Al acabar los capítulos 2, 3 y 4 el lector tendrá clara la estructura que debe seguir un documento *HTML*, sabrá cómo establecer la jerarquía de información en el mismo, así como crear listas y tablas. El capítulo 5 introduce el concepto de hipertexto mediante el uso de enlaces. En el capítulo 6 el lector aprenderá a crear y manejar *frames*, esto es, páginas *web* en las que se muestran simultáneamente varios documentos *HTML*. Con el capítulo 7 aprenderá a manejar el color y las fuentes de texto; y el capítulo 8 a utilizar imágenes. El módulo 1 se cierra con el capítulo 9, en el que se explica la forma de incrustar objetos en las páginas *web*, así como conceptos “avanzados” tales como el uso de metainformación.

Módulo 2: Hojas de estilo

El siguiente módulo presenta el concepto de hojas de estilo (*CSS*) que se emplean con dos fines básicos: por un lado, separar las cuestiones estéticas del documento (color, fuentes elegidas, alineación de texto) de la información del mismo; por otro, como base del *HTML* dinámico. Este módulo se compone de cuatro capítulos. El capítulo 10, presenta al lector el concepto de hoja de estilo, indicando las ventajas de su uso, así como las posibilidades que ofrecen. En el capítulo 11 se muestra la forma en que se deben manejar el color y las características del texto mediante hojas de estilo y qué propiedades del documento deberían dejar de indicarse mediante *HTML*. En el capítulo 12 se explica el empleo de capas y se termina el tercer módulo con el capítulo 13 en el que se presentan una serie de conceptos avanzados (hojas de estilo externas, funcionamiento de la herencia, concepto de cascada, pseudoelementos, etc.).

Módulo 3. JavaScript y DHTML

Con el último módulo del documento el lector será capaz de crear páginas de gran riqueza visual así como totalmente dinámicas e interactivas. Consta de dos capítulos. El capítulo 14 introduce al lector en el lenguaje *JavaScript*/*Jscript*, desde los conceptos más básicos hasta el uso de objetos; así mismo, explica la forma de utilizar este lenguaje en las páginas *web* mediante el uso del modelo de objetos del navegador y el empleo de eventos. Por último el capítulo 15 entra en el mundo del *HTML* dinámico, explicando cómo determinar la plataforma de visualización del documento, cómo crear y manejar capas, o la forma en que se manipulan de forma dinámica las características de la página *web*. Una vez haya finalizado este capítulo el

lector será capaz de crear páginas *web* no solamente correctas, sino visualmente ricas, dotadas de una gran interactividad y totalmente independientes del navegador utilizado.

0.1.2 Contenidos de los capítulos

A continuación se presenta la relación completa de los contenidos de cada capítulo del presente documento.

Módulo 1: HTML 4.0

0.1.3 ¿Qué es y qué no es este documento?

El presente documento podría considerarse un tutorial sobre diseño *web*, es decir, un texto mediante el cual el lector puede aprender a desarrollar sus propias páginas de forma correcta. Se comentan las características más habituales de los temas que se tocan (*HTML*, hojas de estilo y *JavaScript*), aunque evidentemente NO todas las posibilidades que ofrece el *web*. El objetivo fundamental del texto es servir de acompañamiento a las clases impartidas durante el curso, no sustituirlas; sin embargo, una vez finalizado el curso siempre será una referencia útil para localizar conceptos que el lector puede necesitar en su práctica diaria.

El texto no es de ninguna manera una especificación de *HTML* o de *CSS*, como tampoco es un manual de programación en *JavaScript*/*Jscript*; para acceder a ese tipo de documentos el lector debe consultar la referencia donde se indica abundante bibliografía sobre tales temas.

0.1.4 ¿Cómo puedo utilizar este documento?

Como ya se indicó en el punto anterior, la finalidad primera de este texto es servir de apoyo a las clases impartidas durante el curso; por ello se ha estructurado de tal forma que personas con escaso o ningún conocimiento en diseño *web* puedan seguir el curso con aprovechamiento, en tal caso, lo más aconsejable es seguir el texto en el mismo orden que se presenta procurando leer con antelación los capítulos que se vayan a tratar en cada clase. Sin embargo, como también se dijo con anterioridad, una vez finalizado el curso el presente texto puede ser un material de consulta muy valioso para el lector que puede acceder según sus necesidades a los puntos que necesite o emplear cada módulo o capítulo como un "tutorial" para consolidar aspectos del curso que todavía no domine al completo.

0.1.5 ¿Qué *software* se necesita?

Para seguir el curso se necesita un navegador que permita visualizar *HTML 4.0*, maneje hojas de estilo (especificación *CSS 2*) e interprete *JavaScript*/*Jscript*. Los navegadores más extendidos en sus versiones 4.0 o superiores cumplen, aproximadamente, estos requisitos con lo cual el principal requisito es disponer de *Navigator* 4.0 o superior o *Internet Explorer* 4.0 o superior.

Por otra parte, el objetivo de este curso no es enseñar a emplear ninguna herramienta sino el uso de una serie de lenguajes que posibilitan el diseño *web* con lo cual el otro *software* requerido es un editor de textos. A la hora de tratar las imágenes que se vayan a incluir en las páginas *web* sí se necesitará alguna herramienta; en opinión del autor la mejor aplicación disponible para este tipo de trabajo es el programa *Paint Shop Pro* 5.0 (que, por cierto, es *shareware*).

0.2 (Apéndice A) Ejercicios prácticos

0.2.1 Introducción al *HTML*

En este ejercicio se trata de introducir al alumno en el uso de características básicas del lenguaje HTML, tales como la jerarquía, el empleo de encabezamientos, párrafos, saltos de línea, estilos físicos y lógicos, así como el empleo de listas.

0.2.2 Tablas y enlaces

El alumno deberá desarrollar en esta práctica tablas complejas así como crear enlaces internos, externos y enlaces a distintos tipos de recursos en *Internet*.

0.2.3 Empleo de *frames*

El objetivo fundamental de este ejercicio es el desarrollo de documentos complejos mediante el empleo de *frames*.

0.2.4 Uso del color

Con esta práctica se pretende que el alumno utilice las características avanzadas de formateo de texto (color, alineación y fuentes) además de emplear el color en sus tablas

0.2.5 Imágenes

En este ejercicio se aplicará lo aprendido sobre las imágenes, su inclusión, la forma de emplearlas como enlaces o fondos de páginas y tablas.

0.2.6 *HTML*. Conceptos avanzados

Esta práctica permitirá al alumno comprender mejor el empleo de metainformación así como la forma de incrustar objetos en sus documentos *HTML*.

0.2.7 Introducción a las hojas de estilo

El alumno se aproximará a las hojas de estilo mediante un ejemplo sencillo, con el que podrá experimentar algunas de las características de las mismas, así como las diferentes formas de asociar estilos a un documento *HTML*.

0.2.8 Colores y texto con *CSS*

En este ejercicio se aprenderá a emplear el color mediante hojas de estilo, así como la forma de manipular las características del texto mediante *CSS*.

0.2.9 Hojas de estilo y *HTML 4.0* estricto

En esta práctica el alumno tendrá que tomar un documento *HTML 4.0 transitorio* y convertirlo en un documento *HTML 4.0 estricto* con hojas de estilo; para orientarse empleará el validador de documentos del "Consorcio W3".

0.2.10 Manejo de capas

Este ejercicio tiene como finalidad practicar todos los conceptos relacionados con capas *CSS*: márgenes, posicionamiento, visualización...

0.2.11 *CSS*. Conceptos avanzados

En esta práctica se hará un uso intensivo de las clases y la herencia para formatear un texto de forma adecuada; además de emplear las pseudoclases de enlaces y dinámicas.

0.2.12 *JavaScript* (y I)

Este ejercicio tiene como finalidad aproximar al alumno al lenguaje *JavaScript*, para ello se practicarán conceptos tales como variables, operadores y expresiones.

0.2.13 *JavaScript* (y II)

En esta práctica se practicarán conceptos tales como las funciones o las estructuras de control.

0.2.14 *JavaScript* en el *web*

Con este ejercicio el alumno conocerá algunos de los objetos predefinidos en los navegadores *web*, la forma de emplearlos y como aprovechar una serie de eventos habituales para disparar acciones implementadas en *JavaScript*.

0.2.15 *HTML* dinámico (y I)

Este práctica constituye la primera de un ejercicio de dos partes; en esta el usuario desarrollará la estructura de una página *web* dinámica mediante *HTML*, *CSS* y *JavaScript* para, en la segunda parte, programar la parte dinámica propiamente dicha.

0.2.16 *HTML* dinámico (y II)

Segunda parte del ejercicio anterior; en esta práctica se procederá a implementar el código *JavaScript* necesario para realizar la precarga de las imágenes, una animación de capas y una animación de imágenes.

0.2.17 *GIF's* transparentes y animados

Práctica introductoria al manejo de herramientas de tratamiento de imágenes para el desarrollo de *GIF's* transparentes y animados; se emplearán para ello los programas *Paint Shop Pro* y *Animation Shop*.

0.3 (Apéndice B) Caracteres ISO Latín-1 y códigos numéricos

&#00; - &#08;	sin uso	
	tabulador horizontal	
nueva línea	
&#11; - &#31;	sin uso	
	espacio	
!	exclamación de cierre	!
"	comillas dobles	"
#	parrilla	#
$	dólar	$
%	tanto por ciento	%
&	ampersand	&
'	comilla simple	'
(	paréntesis izquierdo	)
)	paréntesis derecho	(
*	asterisco	*
+	signo de suma	+
,	coma	,
-	guión	-
.	punto	.
/	barra inclinada derecha	/
0 - 9	dígitos 0-9	0-9
:	dos puntos	:
;	punto y coma	;
<	menor que	<
=	igual	=
?	interrogación de cierre	?
@	arroba	@
A - Z	letras A-Z	A-Z
[	corchete izquierdo	[
\	barra inclinada izquierda	\
]	corchete derecho	]
^	acento circunflejo	^
_	subrayado	_
`	acento agudo	´
a - &#122:	letras a-z	a-z
{	llave izquierda	{
		barra vertical	|
}	llave derecha	}
~	tilde	~
&#127; -	sin uso	

&#161;	exclamación de apertura	¡
&#162;	símbolo de centavo	¢
&#163;	libra esterlina	£
&#164;	???	¤
&#165;	símbolo de yen	¥
&#166;	barra vertical quebrada	¦
&#167;	símbolo de sección	§
&#168;	diéresis	¨
&#169;	copyright	©
&#170;	ordinal femenino	ª
&#171;	comillas angulosas apertura	«
&#172;	símbolo de negación	¬
&#173;	guión	-
&#174;	marca registrada	®
&#175;	???	¯
&#176;	símbolo de grados	º
&#177;	más-menos	±
&#178;	dos en superíndice	²
&#179;	tres en superíndice	³
&#180;	acento agudo	´
&#181;	símbolo de micro	µ
&#182;	símbolo de párrafo	¶
&#183;	punto medio	·
&#184;	cedilla	ç
&#185;	uno en superíndice	¹
&#186;	ordinal masculino	º
&#187;	comillas angulosas, cierre	»
&#188;	fracción, un cuarto	¼
&#189;	fracción, un medio	½
&#190;	fracción, tres cuartos	¾
&#191;	interrogación de apertura	¿
&#192;	A mayúscula, acento agudo	Á
&#193;	A mayúscula, acento grave	À
&#194;	A mayúscula, acento circunflejo	Â
&#195;	A mayúscula, tilde	Ã
&#196;	A mayúscula, aro	Å
&#197;	A mayúscula, diéresis	Ä
&#198;	AE mayúscula, diptongo	Æ
&#199;	C mayúscula, cedilla	Ç
&#200;	E mayúscula, acento agudo	É
&#201;	E mayúscula, acento grave	È
&#202;	E mayúscula, acento circunflejo	Ê

&#203;	E mayúscula, diéresis	Ë
&#204;	I mayúscula, acento agudo	Í
&#205;	I mayúscula, acento grave	Ì
&#206;	I mayúscula, acento circunflejo	Î
&#207;	I mayúscula, diéresis	Ï
&#208;	ETH mayúscula (Islandés)	Ð
&#209;	N mayúscula, tilde	Ñ
&#210;	O mayúscula, acento agudo	Ó
&#211;	O mayúscula, acento grave	Ò
&#212;	O mayúscula, acento circunflejo	Ô
&#213;	O mayúscula, tilde	Õ
&#214;	O mayúscula, diéresis	Ö
&#215;	símbolo de multiplicación	×
&#216;	O mayúscula, slash	Ø
&#217;	U mayúscula, acento agudo	Ú
&#218;	U mayúscula, acento grave	Ù
&#219;	U mayúscula, acento circunflejo	Û
&#220;	U mayúscula, diéresis	Ü
&#221;	Y mayúscula, acento agudo	Ý
&#222;	THORN mayúscula (Islandés)	Þ
&#223;	unión sz minúscula (Alemán)	ß
&#224;	a minúscula, acento agudo	á
&#225;	a minúscula, acento grave	à
&#226;	a minúscula, acento circunflejo	â
&#227;	a minúscula, tilde	ã
&#228;	a minúscula, diéresis	ä
&#229;	a minúscula, aro	å
&#230;	ae minúscula, diptongo	æ
&#231;	c minúscula, cedilla	ç
&#232;	e minúscula, acento agudo	é
&#233;	e minúscula, acento grave	è
&#234;	e minúscula, acento circunflejo	ê
&#235;	e minúscula, diéresis	ë
&#236;	i minúscula, acento agudo	í
&#237;	i minúscula, acento grave	ì
&#238;	i minúscula, acento circunflejo	î
&#239;	i minúscula, diéresis	ï
&#240;	eth minúscula (Islandés)	ð
&#241;	n minúscula, tilde	ñ
&#242;	o minúscula, acento agudo	ó
&#243;	o minúscula, acento grave	ò
&#244;	o minúscula, acento circunflejo	ô

&#245;	o minúscula, tilde	õ
&#246;	o minúscula, diéresis	ö
&#247;	símbolo de división	÷
&#248;	o minúscula, slash	ø
&#249;	u minúscula, acento agudo	ú
&#250;	u minúscula, acento grave	ù
&#251;	u minúscula, acento circunflejo	û
&#252;	u minúscula, diéresis	ü
&#253;	y minúscula, acento agudo	ý
&#254;	thorn minúscula (Islandés)	þ
&#255;	y minúscula, diéresis	ÿ

0.4 (Apéndice C) Material de referencia

0.4.1 *HTML*

- **http://www.w3.org/TR/REC-html40**
  Especificación del lenguaje HTML 4.0 ☆☆☆
- **http://www.programacion.net/html/**
  Tutorial sobre *HTML* 4.0
- **http://www.webreference.com/**
  Multitud de artículos ☆☆☆

0.4.2 Hojas de estilo

- **http://www.w3.org/TR/REC-CSS1**
  Recomendación CSS 1 ☆☆☆
- **http://www.w3.org/TR/REC-CSS2**
  Especificación *CSS* 2 ☆☆☆
- **http://www.webreference.com/**
  Multitud de artículos ☆☆☆
- **http://www.w3.org/pub/WWW/Style**
  Todo sobre hojas de estilo. Información y recursos.

0.4.3 *JavaScript/JScript*

- **http://www.ecma.ch/stand/ecma-262.htm**
  Especificación de *ECMAScript* (*JavaScript* estándar)
- **http://developer.netscape.com/tech/javascript/index.html**
  Información general sobre *JavaScript*
- **http://developer.netscape.com/docs/manuals/communicator/jsref/index.htm**
  Referencia técnica de *JavaScript* ☆☆☆
- **http://developer.netscape.com/docs/manuals/communicator/jsguide4/index.htm**
  Guía de *JavaScript* ☆☆☆
- **http://wsabstract.com/**
  Muchísimos tutoriales ☆☆☆
- **http://www.webreference.com/**
  Multitud de artículos ☆☆☆

0.4.4 *DHTML*

- **http://developer.netscape.com/tech/dynhtml/index.html**
  Información sobre *DHTML*
- **http://developer.netscape.com/docs/manuals/communicator/dynhtml/index.htm**
  HTML dinámico en *Netscape Communicator* ☆☆☆
- **http://www.programacion.net/html/**
  Tutorial sobre *DHTML*
- **http://wsabstract.com/**
  Muchísimos tutoriales ☆☆☆
- **http://www.webreference.com/**
  Multitud de artículos ☆☆☆
- **http://www.webreference.com/dhtml/**
  Gran cantidad de información y trucos ☆☆☆

0.5 (Apéndice D) Glosario

_

_BLANK Valor del atributo **TARGET**. El recurso apuntado se carga en una nueva ventana.

_PARENT Valor del atributo **TARGET**. El recurso apuntado se carga en el *frameset* padre del *frame* en que se encuentra el enlace. Si el *frame* no tiene padre equivale a **_TOP**.

_SELF Valor del atributo **TARGET**. El recurso apuntado se carga en el *frame* en que se encuentra el enlace.

_TOP Valor del atributo **TARGET**. El recurso apuntado se carga en la ventana original eliminando todos los *frames*.

<A

<A> Etiqueta de enlace, permite definir enlaces de partida (atributo **HREF**) o de llegada (atributo **NAME**). Requiere etiqueta de cierre.

<ADDRESS> Estilo de párrafo. Puede ser utilizado por los autores para proporcionar información de contacto. Debería situarse al principio o al final del documento. Requiere etiqueta de cierre.

<AREA> Etiqueta que define una zona activable de un mapa en el cliente. Debe ir incluida en un elemento **<MAP>**. No requiere etiqueta de cierre.

<B

<B> Negrita (estilo físico de texto). Requiere etiqueta de cierre.

<BLOCKQUOTE> Estilo de párrafo. Utilizado para citar texto de otras fuentes. Requiere etiqueta de cierre.

<BODY> El contenido de un documento *HTML* va en el cuerpo del mismo; enmarcado por esta etiqueta y su correspondiente etiqueta de cierre.

**
** Fuerza un salto de línea. No requiere etiqueta de cierre.

<C

<CAPTION> Se utiliza para indicar el título de una tabla.

<CITE> Cita de otras fuentes (estilo lógico). Requiere etiqueta de cierre.

<CODE> Muestra de un fragmento de código (estilo lógico). Requiere etiqueta de cierre.

<D

<DD> Precede a la descripción de un elemento de una lista descriptiva. No requiere etiqueta de cierre.

<DIV> Etiqueta que permite especificar divisiones en un documento *HTML*. Requiere etiqueta de cierre.

<DL> Indica la apertura de una lista descriptiva. Requiere etiqueta de cierre.

<DT> Precede a un elemento de una lista descriptiva. No requiere etiqueta de cierre.

<E

<EM> Se utiliza para mostrar énfasis (estilo lógico). Requiere etiqueta de cierre.

<F

<FONT> Etiqueta que permite modificar el tamaño, color y fuente del texto que delimita junto con su etiqueta de cierre.

<FRAME> Etiqueta que describe las características de un *frame*. No requiere etiqueta de cierre.

<FRAMESET> Etiqueta utilizada en un documento de *frames* para describir la forma en que se dividirá la ventana del navegador. Requiere etiqueta de cierre.

<H

<HEAD> Esta etiqueta junto con su correspondiente etiqueta de cierre delimita la cabecera del documento. La cabecera contiene información sobre el documento como título, palabras clave, etc. Los navegadores no visualizan, en general, la información de la cabecera aunque pueden hacerla accesible al usuario mediante otros mecanismos.

<H*n*> Un título o encabezamiento describe brevemente la sección que precede. En *HTML* hay 6 niveles de títulos, numerados desde 1 hasta 6, donde el 1 es el mayor importancia y el 6 el de menor. Requiere etiqueta de cierre.

<HTML> Esta etiqueta junto con su correspondiente etiqueta de cierre delimita todo el documento *HTML*.

<I

<I> Cursiva (estilo físico de texto). Requiere etiqueta de cierre.

<IFRAME> Etiqueta que describe las características de un *inline frame*. Es aconsejable utilizar la etiqueta de cierre para delimitar texto explicativo para los navegadores que no soporten esta etiqueta.

<IMG> Etiqueta que permite la inclusión de imágenes en un documento *HTML*. No requiere etiqueta de cierre.

<K

<KBD> Simula una entrada de teclado (estilo lógico). Requiere etiqueta de cierre.

<L

<LI> Precede a un elemento de una lista ordenada o no ordenada. No requiere etiqueta de cierre.

<LINK> Etiqueta que permite definir relaciones entre o incluir documentos en un documento *HTML*.

<M

<MAP> Etiqueta que permite definir mapas en el cliente. Requiere etiqueta de cierre.

<META> Etiqueta que permite incluir metainformación en un documento *HTML*. No requiere etiqueta de cierre.

<N

<NOFRAMES> Etiqueta que delimita el texto a mostrar en el caso de que el navegador no soporte *frames*. Requiere etiqueta de cierre.

<O

<OL> Indica la apertura de una lista ordenada. Requiere etiqueta de cierre.

<P

<P> Esta etiqueta junto con su correspondiente etiqueta de cierre delimitan un párrafo nuevo.

<PRE> Estilo de párrafo. Permite visualizar el texto tal y como aparece en el documento *HTML*. Requiere etiqueta de cierre.

<S

<SAMP> Muestra un texto de ejemplo (estilo lógico). Requiere etiqueta de cierre.

<SCRIPT> Etiqueta que permite incrustar código en un lenguaje de *script* en un documento *HTML*. Requiere etiqueta de cierre.

<STRONG> Se utiliza para mostrar énfasis (estilo lógico). Requiere etiqueta de cierre.

<T

<TABLE> Indica la apertura de una tabla. Requiere etiqueta de cierre.

<TD> Indica la apertura de una celda en una fila de una tabla. Requiere etiqueta de cierre.

<TH> Etiqueta análoga a **<TD>** con la diferencia que el texto de estas celdas se considera texto de cabecera con lo cual se centra automáticamente y se visualiza en negrita.

<TITLE> Los autores de documentos *HTML* deberían utilizar el título para identificar el contenido de un documento; los títulos deben ser suficientemente explícitos y permitir al usuario hacerse una idea de la información que proporciona un documento en cuestión. Requiere etiqueta de cierre.

<TR> Indica la apertura de una fila en una tabla. Requiere etiqueta de cierre.

<TT> Teletipo (estilo físico de texto). Requiere etiqueta de cierre.

<U

<U> Subrayado (estilo físico de texto). Requiere etiqueta de cierre.

<UL> Indica la apertura de una lista no ordenada. Requiere etiqueta de cierre.

<V

<VAR> Indica una variable de un programa informático (estilo lógico). Requiere etiqueta de cierre.

A

a Valor del atributo **TYPE** de la etiqueta **<OL>**. Permite listas ordenadas con índices alfabéticos (minúsculas).

A Valor del atributo **TYPE** de la etiqueta **<OL>**. Permite listas ordenadas con índices alfabéticos (mayúsculas).

ALIGN Atributo de las etiquetas **<TABLE>**, **<TR>** y **<TD>** que permite la alineación horizontal del contenido de las celdas de una tabla.

Atributo de la etiqueta **<P>** que permite seleccionar la alineación del texto de un párrafo: justificado (**JUSTIFY**), centrado (**CENTER**), alineado a la derecha (**RIGHT**) o alineado a la izquierda (**LEFT**).

Atributo de la etiqueta **<IMG>** que permite indicar la alineación de una imagen respecto al texto que la rodea: parte superior (**TOP**), centrada (**MIDDLE**) y parte inferior (**BOTTOM**).

ALINK Atributo de la etiqueta **<BODY>**. Permite indicar el color de los enlaces del documento *HTML* cuando son activados.

ALT Atributo de la etiqueta **<IMG>**. Permite especificar un texto alternativo para la imagen; **debe utilizarse siempre**.

ámbito Se denomina ámbito de una variable a la parte de un programa en que se puede utilizar.

ASCII *American Standard Code for Information Interchange* (Código Estándar Americano para Intercambio de Información). Estándar mundial que recoge 128 caracteres, letras, números y símbolos utilizados para escribir ficheros de texto plano (sin formato).

atributo Característica modificable de una etiqueta *HTML*.

B

BACKGROUND Atributo de las etiquetas **<BODY>**, **<TABLE>**, **<TR>**, **<TD>** y **<TH>**. Permite indicar el *URI* de una imagen que se utilizará como fondo del documento *HTML*, de una tabla, una fila o una celda, respectivamente. La imagen que se visualiza para un elemento si tiene más de una definida es la más restrictiva.

background Propiedad que permite emplear una imagen como fondo de un elemento de un documento *HTML*.

background-color Propiedad que indica el color de fondo de un elemento de un documento *HTML*.

BGCOLOR Atributo de las etiqueta **<BODY>**, **<TABLE>**, **<TR>**, **<TD>** y **<TH>**. Permite indicar el color de fondo del documento *HTML*, de una tabla, una fila o una celda, respectivamente. El color que se visualiza para un elemento si tiene más de uno definido es el más restrictivo.

BORDER Atributo de la etiqueta **<TABLE>**, indica el ancho del borde de una tabla.

Atributo de la etiqueta **<IMG>**, indica el ancho del borde de una imagen.

BOTTOM Valor del atributo **VALIGN** de las etiquetas **<TABLE>**, **<TR>** y **<TD>** que permite colocar el contenido de las celdas de una tabla en la parte inferior.

Valor del atributo **ALIGN** de la etiqueta **<IMG>** que permite alinear una imagen respecto a la parte inferior del texto que la rodea.

C

capa Denominación que se da a una división en un documento *HTML* que contiene uno o más elementos *HTML* y que puede colocarse dentro del documento en 3 dimensiones: las dos de la pantalla y un índice de profundidad respecto a otras capas.

CELLPADDING Atributo de la etiqueta **<TABLE>**, indica la distancia desde el contenido de una celda hasta el borde de la misma.

CELLSPACING Atributo de la etiqueta **<TABLE>**, indica la distancia entre celdas de una misma tabla.

CENTER Valor del atributo **ALIGN** de las etiquetas **<TABLE>**, **<TR>** y **<TD>** que permite centrar horizontalmente el contenido de las celdas de una tabla.

Valor del atributo **ALIGN** de la etiqueta **<P>** que permite centrar horizontalmente el texto del párrafo.

CERN *Centro Europeo de Investigaciones Nucleares*; lugar donde se desarrolló el *HTML* y nació el *web*.

CIRCLE Valor del atributo **TYPE** de la etiqueta **<UL>**. Permite listas no ordenadas utilizando como ítem un círculo vacío.

Valor del atributo **SHAPE** de la etiqueta **<AREA>** que indica que la zona activable de un mapa es un círculo.

CLASS Atributo que permite indicar la clase a que pertenece un elemento de un documento *HTML*; muy utilizado junto con hojas de estilo.

cliente Cualquier elemento de un sistema de información que requiere un servicio mediante el envío de solicitudes a otro elemento encargado de atenderlas (*servidor*).

COLOR Atributo de la etiqueta **<FONT>** que permite indicar el color de un texto. Funciona de forma análoga a **TEXTCOLOR**.

color Propiedad que indica el color del texto de un elemento de un documento *HTML*.

COLS Atributo de la etiqueta **<FRAMESET>** que permite describir la división en *frames* verticales de la ventana del navegador.

COLSPAN Atributo de la etiqueta **<TD>** que permite definir celdas cuya anchura es un múltiplo entero de la columna base.

comentario Texto que permite documentar un código para mejorar su legibilidad y mantenimiento; los comentarios no son visibles para el usuario. En *HTML* se encierran entre los símbolos **<!--** y **-->**.

COORDS Atributo de la etiqueta **<AREA>**. Indica las coordenadas de la zona activable de un mapa.

correo electrónico Conjunto de tecnologías que permiten la interconexión de ordenadores para el intercambio de mensajes, documentos, etc.

CSS *Cascade Style Sheets. Véase hojas de estilo.*

D

descarga *Véase download.*

DHTML *Véase HTML dinámico*

DISC Valor del atributo **TYPE** de la etiqueta **<UL>**. Permite listas no ordenadas utilizando como ítem un círculo lleno.

download Transferencia de ficheros mediante *FTP* que se hace en el sentido servidor → cliente; es decir, el usuario recibe uno o más ficheros desde el servidor.

E

e-mail *Véase correo electrónico.*

enlace Palabra, frase o imagen que aparece en un documento *HTML* y que, al ser activada, invoca a un recurso accesible vía *web*, generalmente otro documento *HTML*.

enlace externo Aquel que al ser activado carga un recurso distinto al documento *HTML* actual.

enlace interno Aquel que al ser activado desplaza al usuario hasta una sección distinta del documento *HTML* actual.

etiqueta Palabra clave encerrada entre los signos “<” y “>” y que modifica el texto al que precede.

F

FACE Atributo de la etiqueta **<FONT>** que permite indicar la fuente de un texto; debiendo elegir el navegador la fuente disponible más parecida a la seleccionada.

`float`	Propiedad que permite indicar si un elemento de un documento *HTML* flotará a la derecha (**`right`**), a la izquierda (**`left`**) o no flotará (**`none`**).
`font-family`	Propiedad que permite seleccionar el tipo de fuente a utilizar para representar un texto.
`font-size`	Propiedad que permite indicar el tamaño de un texto.
`font-style`	Propiedad que permite seleccionar un estilo para representar un texto; admite los valores **`normal`**, **`italic`** y **`oblique`** (las dos últimas son cursivas).
`font-variant`	Propiedad que permite seleccionar una variante para representar un texto; admite los valores **`normal`** o **`small-caps`** (todo en mayúsculas, aunque las correspondientes a las letras minúsculas se muestren de menor tamaño).
`font-weight`	Propiedad que permite especificar el nivel de negrita para un texto; admite valores, expresados en centenas, desde 100 hasta 900 además de cuatro valores predefinidos **`normal`** (400), **`bold`** (700), **`bolder`** (+100 ó 900, el menor) y **`lighter`** (-100 ó 100, el mayor).
frame	Subdivisión de la ventana de un navegador que puede mostrar su propio *URI*, ser nombrada, apuntada y manipulada independientemente del resto.
`FRAMEBORDER`	Atributo de las etiquetas **`<FRAME>`**, **`<IFRAME>`** que permite visualizar (valor **`1`**) o no (valor **`0`**) el borde en *frames* e *inline frames*.
FTP	*File Transfer Protocol* (Protocolo de Transferencia de Ficheros). Protocolo que permite a dos ordenadores (un servidor y un cliente) comunicarse, recibiendo el cliente uno o más ficheros (de cualquier tipo) que le proporciona el servidor

H

`height`	Propiedad que permite especificar la altura de un elemento en un documento *HTML*.
`HEIGHT`	Atributo de las etiquetas **`<TABLE>`**, **`<TR>`** y **`<TD>`**. Permite especificar la altura del elemento al que modifica, bien en pixels o bien como un porcentaje del elemento jerárquicamente superior (ventana, tabla y fila, respectivamente). Atributo de la etiqueta **`<IMG>`**. Permite especificar la altura de la imagen.
hiperenlace	*Ver enlace.*
hipertexto	Forma no lineal de presentar información textual de manera que el lector puede navegar a través de la misma mediante el uso de *hiperenlaces*.
hoja de estilo	Conjunto de reglas que describen las propiedades de uno o más elementos de un documento *HTML*.

HREF Atributo de la etiqueta **<A>**; define un enlace de partida. Toma como valor un *URI*.

Atributo de la etiqueta **<AREA>**. Indica el *URI* del recurso apuntado por la zona activable de un mapa.

HTML *HiperText Mark-Up Language* (Lenguaje HiperTextual de Etiquetas). Lenguaje que permite desarrollar documentos hipertextuales en *Internet*.

HTML dinámico Forma de desarrollar páginas *web* mediante el empleo conjunto de *HTML* 4.0, hojas de estilo y *JavaScript*.

HTTP *HiperText Transfer Protocol* (Protocolo de Transferencia de HiperTexto). Protocolo que permite a dos ordenadores (un servidor y un cliente) comunicarse, recibiendo el cliente los documentos *HTML* que le proporciona el servidor.

I

I Valor del atributo **TYPE** de la etiqueta **<OL>**. Permite listas ordenadas con índices numéricos romanos (minúsculas).

I Valor del atributo **TYPE** de la etiqueta **<OL>**. Permite listas ordenadas con índices numéricos romanos (mayúsculas).

ID Atributo que permite indicar un identificador para un elemento de un documento *HTML*; muy utilizado junto con hojas de estilo.

imagen mapa Imagen que contiene varias zonas activables distintas, apuntando cada una a un recurso distinto.

inline frame Propuesta de *HTML 4.0* que permite “incrustar” una página *web* dentro de un documento *HTML*.

Internet Conjunto de redes interconectadas, basadas en los protocolos TCP/IP y derivada de la red ARPANET que conectaba universidades y centros de investigación estadounidenses.

J

JUSTIFY Valor del atributo **ALIGN** de la etiqueta **<P>** que permite justificar el texto del párrafo.

L

LANGUAGE Atributo de la etiqueta **<SCRIPT>** que permite indicar el lenguaje en que está escrito el *script* incrustado. Un valor típico es **JavaScript**.

LEFT Valor del atributo **ALIGN** de las etiquetas **<TABLE>**, **<TR>** y **<TD>** que permite colocar el contenido de las celdas de una tabla en la parte izquierda.

Valor del atributo **ALIGN** de la etiqueta **<P>** que permite alinear a la izquierda el texto del párrafo.

LINK Atributo de la etiqueta **<BODY>**. Permite indicar el color de los enlaces no visitados del documento *HTML*.

lista Enumeración de elementos.

lista descriptiva Enumeración de elementos compuestos de un ítem y la descripción del mismo.

lista no ordenada Enumeración de elementos precedidos por un símbolo gráfico.

lista ordenada Enumeración de elementos precedidos, generalmente, de un valor numérico.

lista regular Una lista ordenada o no ordenada.

M

mail *Véase correo electrónico.*

margen Espacio que queda entre el borde de una caja y los márgenes de otras cajas.

margin
margin-bottom
margin-left
margin-right
margin-top
Propiedades que permiten especificar las dimensiones de los márgenes de una caja en *CSS*.

MARGINHEIGHT Atributo de la etiqueta **<FRAME>**, indica la altura en pixels del *frame*.

MARGINWIDTH Atributo de la etiqueta **<FRAME>**, indica la anchura en pixels del *frame*.

MIDDLE Valor del atributo **VALIGN** de las etiquetas **<TABLE>**, **<TR>** y **<TD>** que permite la centrar verticalmente el contenido de las celdas de una tabla.

Valor del atributo **ALIGN** de la etiqueta **<IMG>** que permite centrar una imagen respecto al texto que la rodea.

Mosaic Primer navegador gráfico.

motor de búsqueda Servidor que permite localizar recursos en el *web* a partir de un conjunto de palabras clave.

N

NAME — Atributo de la etiqueta **<FRAME>**, permite identificar al *frame* para apuntarlo desde enlaces en otros documentos.

Atributo de la etiqueta **<MAP>**, permite identificar al mapa para indicarlo en el atributo **USEMAP** de la imagen que lo utilice.

NAME — Atributo de la etiqueta **<A>**; define un enlace de llegada. Toma como valor una cadena.

NCSA — *National Center for Supercomputing Applications* (Centro Nacional para Aplicaciones de Supercomputación). Organismo de la Universidad de Illinois donde se desarrolló *Mosaic*.

Netscape — *Communications Corporation*: Compañía desarrolladora del navegador *Netscape Navigator*. Pionera en algunas mejoras importantes de los navegadores así como del *HTML*.

Navigator: Navegador comercial.

news — (También *newsgroups*). Espacios en *Internet* en los que personas con intereses similares envían y contestas mensajes sobre los mismos.

NOHREF — Atributo de la etiqueta **<AREA>**. No tiene valor. Indica que la zona activable de un mapa no tiene enlace asociado.

NORESIZE — Atributo sin valor de la etiqueta **<FRAME>**, impide el cambio de tamaño del *frame*.

O

overflow — Propiedad que permite indicar el comportamiento del contenido de una caja que sobresale de los límites de la misma; admite los valores **visible** (se muestra), **hidden** (no se muestra), **scroll** (puede desplazarse) y **auto** (equivale a **visible** si no hay desbordamiento y a **scroll** si lo hay).

P

padding
padding-bottom
padding-left
padding-right
padding-top — Propiedades que permiten especificar las dimensiones del *padding* de una caja en *CSS*.

padding — Espacio que queda entre el contenido de una caja en *CSS* y el borde de la misma.

POLY Valor del atributo **SHAPE** de la etiqueta **<AREA>** que indica que la zona activable de un mapa es un polígono.

protocolo Conjunto de normas y/o procedimientos para la transmisión de datos que han de ser observadas por los dos extremos de un proceso comunicativo.

R

RECT Valor del atributo **SHAPE** de la etiqueta **<AREA>** que indica que la zona activable de un mapa es un rectángulo.

regla Cada una de las construcciones de una hoja de estilo que describe una propiedad específica de un elemento de un documento *HTML*.

REL Atributo de la etiqueta **<LINK>**, indica una relación (hacia delante) entre el documento actual y el documento indicado por la etiqueta.

REV Atributo de la etiqueta **<LINK>**, indica una relación (hacia atrás) entre el documento actual y el documento indicado por la etiqueta.

RIGHT Valor del atributo **ALIGN** de las etiquetas **<TABLE>**, **<TR>** y **<TD>** que permite colocar el contenido de las celdas de una tabla en la parte derecha.

Valor del atributo **ALIGN** de la etiqueta **<P>** que permite alinear a la derecha el texto del párrafo.

ROWS Atributo de la etiqueta **<FRAMESET>** que permite describir la división en *frames* horizontales de la ventana del navegador.

ROWSPAN Atributo de la etiqueta **<TD>** que permite definir celdas cuya altura es un múltiplo entero de la fila base.

S

script Se suelen denominar de esta forma programas cortos escritos con lenguajes relativamente sencillos para realizar tareas poco complejas; los *scripts* son habitualmente interpretados y están experimentando un gran auge gracias a *Internet*. *JavaScript*/*JScript*, *VBScript* son ejemplos de lenguajes destinados al desarrollo de *scripts*.

SCROLLING Atributo de la etiqueta **<FRAME>**, admite los valores **YES** (fuerza la existencia de barras de *scroll*), **NO** (impide la existencia de barras de *scroll*) y **AUTO** (valor por defecto, deja la elección al navegador).

servidor Elemento de un sistema que resuelve las peticiones de otros elementos denominados clientes.

SHAPE Atributo de la etiqueta **<AREA>**. Indica la forma de la zona activable de un mapa.

SIZE Atributo de la etiqueta **<FONT>** que permite indicar el tamaño absoluto de un texto mediante un entero en la escala 1..7, o un tamaño relativo indicando un incremento o decremento que se ajustará a la misma escala.

SQUARE Valor del atributo **TYPE** de la etiqueta **<UL>**. Permite listas no ordenadas utilizando como ítem un cuadrado.

SRC Atributo de la etiqueta **<FRAME>**, indica el *URI* a cargar en dicho *frame*.

Atributo de la etiqueta **<SCRIPT>**, indica el *URI* del fichero que contiene el código del *script*.

STYLESHEET Valor que toma el atributo **REL** de la etiqueta **<LINK>** para indicar que se debe cargar una hoja de estilo externa.

T

TARGET Atributo de la etiqueta **<A>**, indica en qué ventana o *frame* se cargara el recurso apuntado por el enlace.

TEXT Atributo de la etiqueta **<BODY>**. Permite indicar el color del texto del documento *HTML*.

text-decoration Propiedad que permite indicar una decoración para el texto; admite los valores **underline** (subrayado), **overline** (“superrayado”), **line-through** (tachado) y **blink** (parpadeo).

text-indent Propiedad que permite especificar un sangrado en la primera línea de un texto; recibe como valor una distancia desde el margen.

text-transform Propiedad que permite manejar el uso de mayúsculas y minúsculas independientemente del texto escrito; admite los valores **capitalize** (pasa a mayúscula la primera letra de cada palabra), **uppercase** (escribe todo en mayúsculas), **lowercase** (escribe todo en minúsculas), **none** (no hace nada).

TOP Valor del atributo **VALIGN** de las etiquetas **<TABLE>**, **<TR>** y **<TD>** que permite colocar el contenido de las celdas de una tabla en la parte superior.

Valor del atributo **ALIGN** de la etiqueta **<IMG>** que permite alinear una imagen respecto a la parte superior del texto que la rodea.

TYPE Atributo de las etiquetas **<OL>** y **<UL>**. Permite especificar la naturaleza del ítem que precederá a los elementos de la lista.

U

upload Transferencia de ficheros mediante *FTP* que se hace en el sentido cliente → servidor; es decir, el usuario envía uno o más ficheros hacia el servidor.

URI — *Universal Resource Identifier* (Identificador Universal de Recursos). Dirección que permite localizar un recurso en *Internet* de forma inequívoca puesto que cada recurso tiene un *URI* único e irrepetible.

USEMAP — Atributo de la etiqueta **<IMG>** que toma como valor el *URI* donde se encuentra definido el mapa para la imagen.

V

VALIGN — Atributo de las etiquetas **<TABLE>**, **<TR>** y **<TD>** que permite la alineación vertical del contenido de las celdas de una tabla.

variable — Un identificador que puede cambiar de valor a lo largo de la ejecución de un programa.

visibility — Propiedad que indica si un elemento de un documento *HTML* es visible (**visible**) o no (**hidden**).

VLINK — Atributo de la etiqueta **<BODY>**. Permite indicar el color de los enlaces visitados del documento *HTML*.

W

web — También *World Wide Web*. Red mundial de documentos hipertextuales y recursos accesibles mediante la red *Internet*.

width — Propiedad que permite especificar la anchura de un elemento en un documento *HTML*.

WIDTH — Atributo de las etiquetas **<TABLE>**, **<TR>** y **<TD>**. Permite especificar la anchura del elemento al que modifica, bien en pixels o bien como un porcentaje del elemento jerárquicamente superior (ventana, tabla y fila, respectivamente).

Atributo de la etiqueta **<IMG>**. Permite especificar la anchura de la imagen.

Z

z-index — Propiedad que permite indicar la posición en que se encuentra una capa en la pila de capas de un documento *HTML*.

Índice Módulo 1 (*HTML* 4.0)

1. Presentación del *web* y del lenguaje *HTML*

1.1 El web

1.1.1 ¿Qué es el *web*?

El *World Wide Web*, o simplemente el *web*, es una red mundial de recursos[1] accesibles mediante la red *Internet*. El *web* funciona gracias a tres mecanismos distintos aunque relacionados entre sí:

- Un esquema que permite nombrar un recurso de forma única en todo el mundo (el *URI*). Por ejemplo, **`http://giworks.uniovi.es/cursos/cursos.shtml`** se refiere a un único recurso (en este caso un documento con información sobre los cursos que imparte el grupo ***GIworks***) y ningún otro recurso del mundo puede llamarse de la misma forma.
- *Protocolos* que permiten acceder a los recursos; en el caso del *web* el protocolo *HTTP* (Protocolo de Transferencia de HiperTexto). Por ejemplo, el *URI* anterior indica al navegador que el recurso es accesible mediante e protocolo *HTTP*.
- El lenguaje *HTML* que permite desarrollar documentos *hipertextuales* gracias a los cuales se pueden enlazar unos documentos con otros, "tejiendo" esa red virtual que es el *web*.

La historia del *web* es pareja a la historia del lenguaje *HTML* que se describe en el punto "1.2.2 Breve historia del *HTML*".

1.1.2 Introducción a los *URI's*

Como ya se mencionó en el punto anterior, uno de los mecanismos en los que se basa el *web* es aquel que permite que cada recurso tenga una dirección única mediante la cual es accesible. Esta dirección se codifica mediante un *Identificador Universal de Recursos* (*URI*).

Los *URI's* se componen, generalmente[2], de tres partes:

- El esquema de acceso al recurso.
- El nombre de la máquina donde se encuentra físicamente el recurso.
- El nombre del recurso propiamente dicho.

Analicemos el *URI* que se utilizó como ejemplo en el punto anterior:

`http://` **`giworks.uniovi.es`** **`/cursos/cursos.shtml`**

Podría traducirse de la forma siguiente: en la máquina **`giworks.uniovi.es`**, existe un documento accesible mediante el protocolo *HTTP*, en el camino **`/cursos/cursos.shtml`**.

Como se verá más adelante, en los documentos *HTML* pueden emplearse muchos otros tipos de *URI's*, para transferencia de ficheros, envío de correo, acceso a grupos de noticias... (Sección "5.3 Distintos tipos de recursos accesibles".)

1.2 El HTML

1.2.1 ¿Qué es el *HTML*?

El *HTML* (Lenguaje HiperTextual de Etiquetas) es un lenguaje estandarizado (esto es conocido y empleado por todos de la misma forma) para describir documentos de texto que contienen referencias a otros documentos; estas referencias permiten acceder a dichos recursos y se conocen con el nombre de enlaces.

[1] Un documento puede ser una página escrita en *HTML*, una imagen, un vídeo, un mundo virtual, este documento... En definitiva, cualquier tipo de fichero es susceptible de convertirse en un recurso accesible vía *web* y, por tanto, visible para usuarios de todo el mundo.

[2] Consultar el punto "5.3.1 Enlaces externos" para conocer una cuarta sección de los *URI's*.

La última versión del lenguaje permite al autor de documentos *HTML*:

- Crear documentos con un formato análogo al de una publicación impresa (títulos jerárquicos, estilos de texto, tablas, listas, imágenes, etc.).
- Establecer enlaces que permiten al lector de sus documentos acceder a otros recursos (documentos *HTML*, vídeo, sonido...) mediante un *click* de ratón.
- Diseñar formularios para recibir información del usuario, realizar transacciones, hacer búsquedas, etc.

1.2.2 Breve historia del *HTML*

El lenguaje *HTML*, así como el protocolo que lo soporta, el *HTTP*, fueron desarrollados por Tim Berners-Lee en el *CERN* (Centro Europeo de Investigaciones Nucleares) a finales de 1989, principios de 1990. La motivación que llevó a desarrollar este sistema fue la necesidad de compartir información entre los físicos de alta energía que trabajaban por todo el mundo. El lenguaje fue popularizado por el navegador *Mosaic* del *NCSA* (Centro Nacional para Aplicaciones de Supercomputación) y supuso la eclosión del *web*.

Desde entonces las distintas versiones del lenguaje se han sucedido para incorporar nuevas características y posibilidades al mismo. A la especificación del lenguaje hecha en el *CERN* siguieron *HTML 2.0* (1995), *HTML 3.0* (también en 1995), *HTML 3.2* (1997), hasta llegar a la versión 4.0 (última revisión en abril de 1998).

Las diferentes versiones del lenguaje tienen los siguientes inconvenientes:

- Los cambios introducidos no siempre fueron del todo adecuados pues en muchas ocasiones se hicieron para soportar “extensiones” introducidas de forma unilateral por los principales vendedores de navegadores (*Netscape* y *Microsoft*).
- Ningún navegador soporta por completo la última versión estandarizada y/o lo hace de forma incorrecta.
- Sigue habiendo extensiones propietarias que la marca rival no soporta.

Sin embargo, esto no debe desanimar de ningún modo al lector puesto que, a pesar de tales inconvenientes, se pueden desarrollar páginas *web* magníficas sin necesidad de recurrir a las opciones avanzadas (y no implementadas) de *HTML 4.0* ni emplear extensiones propietarias.

1.2.3 ¿Por qué usar *HTML 4.0 transitorio*?

Como se ha dicho, este módulo va a tratar *HTML 4.0*, pero ¿qué significa el término *transitorio* que se le añade? Con la última versión del lenguaje se pretende que los documentos *HTML* tan sólo se empleen para indicar características inherentes al documento tales como encabezamientos, divisiones, tablas, etc.; es decir, *HTML 4.0* sólo describe el contenido del documento, la forma (colores, fuentes, alineación de párrafos...) debería indicarse mediante otro mecanismo (tal mecanismo no es otro que las hojas de estilo, que se explican en el Módulo 2).

Sin embargo, para facilitar las cosas tanto a los autores de páginas *web* como a los desarrolladores de navegadores, existe una versión *transitoria* que admite tanto etiquetas de versiones anteriores para indicar características de forma como el uso de hojas de estilo. Por ello, aunque la filosofía de este curso es que el lector desarrolle sus páginas siguiendo de la forma más fiel posible las especificaciones, también se explican tales etiquetas *desaprobadas* puesto que, en cierta manera, esta versión transitoria de *HTML 4.0* puede considerarse un superconjunto de las versiones anteriores del lenguaje y le permitirá entender el código de páginas desarrolladas de una forma no acorde con la “ortodoxia” del estándar.

2. Estructura y jerarquía de un documento. Utilización de estilos

2.1 Estructura de un documento

2.1.1 Las etiquetas *HTML*

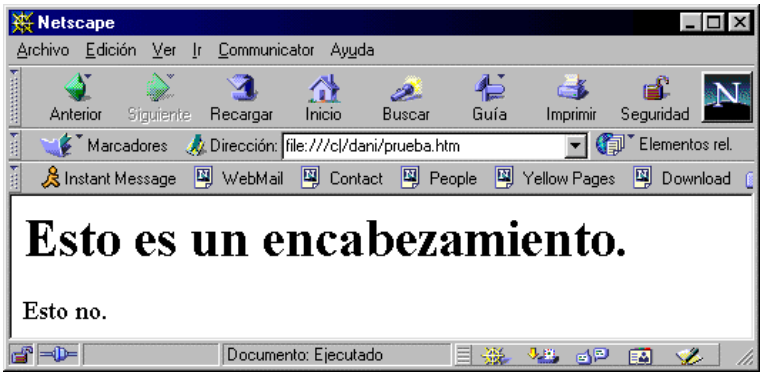
El lenguaje *HTML* no es un lenguaje de descripción de páginas; se limita a describir la estructura y el contenido del documento y no el formato de la página y su apariencia. Esto se logra encerrando cada porción del texto entre un par de etiquetas, una de apertura y otra de cierre, que no son visibles en el momento de visualizar el documento aunque afectan a su visualización.

En *HTML* las etiquetas se delimitan con los signos "<" (menor que) y ">" (mayor que); todo esto quedará más claro con un ejemplo:

```
<H1>Esto es un encabezamiento.</H1>
Esto no.
```

La visualización correspondiente a ese código *HTML* es la siguiente (Figura 2-1):

Figura 2-1



Al utilizar los signos mayor que y menor que para delimitar las etiquetas si se quieren utilizar como caracteres normales hay que indicárselo al navegador mediante un código; así el carácter "<" es sustituido por **`<`** (*less than*) y el carácter ">" por **`>`** (*greater than*).

Si se escribe la siguiente línea de código:

```
Los signos &gt; y &lt; se representan mediante &#38;gt; y &#38;lt;.
```

Se obtendría lo siguiente (Figura 2-2):

Figura 2-2

2.1.2 Caracteres *ISO Latín-1* y códigos numéricos

El ejemplo anterior permitir pasar a tratar otro tema, que es el de la acentuación y la utilización de símbolos. Para asegurar que cualquier navegador va a mostrar el texto de la forma deseada no se debe acentuar directamente el texto ni introducir símbolos como ©, ® o ™; un documento *HTML* sólo debe contener caracteres del código *ASCII* de 7 bits, indicándose los otros caracteres mediante una serie de códigos.

Tales códigos (disponibles en el "Apéndice A") pueden indicarse de forma numérica (por ejemplo **`&#38;`** representa el carácter "&") o bien de una forma un poco más explícita (el caso de **`<`** y **`>`**).

Sin embargo, a efectos prácticos y como hispanohablante tan sólo hay 12 de esos códigos que deban preocupar al lector: 10 para las vocales acentuadas (minúsculas y mayúsculas) y 2 para la "ñ" (minúscula y mayúscula).

Los códigos correspondientes aparecen en la tabla 2-1.

Tabla 2-1 (Códigos para vocales acentuadas y la "ñ")

	á	é	Í	ó	ú	ñ
Minúscula	`&aacute;`	`&eacute;`	`&iacute;`	`&oacute;`	`&uacute;`	`&ntilde;`
Mayúscula	`&Aacute;`	`&Eacute;`	`&Iacute;`	`&Oacute;`	`&Uacute;`	`&Ntilde;`

El lector debe recordar también que los caracteres de apertura para la interrogación y la exclamación también deben indicarse mediante un código puesto que no pertenecen al mencionado código *ASCII* de 7 bits.

2.1.3 Estructura de un documento *HTML*

Todos los documentos *HTML* comienzan con la etiqueta **`<HTML>`** y terminan con la etiqueta **`</HTML>`**. Además, se dividen en dos zonas: la cabecera y el cuerpo; estas zonas se delimitan con las etiquetas **`<HEAD>`**...**`</HEAD>`** para la cabecera y **`<BODY>`**...**`</BODY>`** para el cuerpo del documento.

En la cabecera se incluye diversa información sobre el documento que no afecta a la apariencia del mismo en el navegador; por el momento sólo se verá una etiqueta relacionada con la cabecera y que permite dar un título al documento (para saber más sobre la información incluida en la cabecera consúltese la sección "9.3 Uso de metadatos").

El título es un texto "plano" delimitado por las marcas **`<TITLE>`**...**`</TITLE>`** y que debe identificar al documento. Aunque no se limita la longitud del título ésta no debe ser excesiva pues podría ser truncado por el navegador perdiendo el sentido; tampoco se debe ir al otro extremo, pues títulos excesivamente cortos no facilitan al usuario la tarea de identificar la naturaleza del documento.

El contenido del documento reside en el cuerpo que, como ya se vio, está delimitado por las marcas **`<BODY>`**...**`</BODY>`**; por el momento no es necesario saber más sobre esta zona puesto que en los siguientes puntos se tratará en profundidad.

Antes de avanzar más en el conocimiento del lenguaje se hace necesario presentar dos conceptos muy importantes. El primero es el de *comentario*, aquel texto cuya misión es facilitar la comprensión del código a la persona que vaya a leerlo pero sin ser visible para el usuario; en *HTML* los comentarios se encierran entre los símbolos **`<!--`** y **`-->`**. El lector debe acostumbrarse al uso de comentarios puesto que un código bien comentado es un código fácil de leer y, por tanto, fácil de mantener.

El otro concepto es el de *atributo*, un atributo permite modificar el comportamiento de las etiquetas, el comportamiento puede ser modificado en algunos casos con sólo añadir el atributo, aunque en la mayor parte de los casos se requiere un valor que se asigna mediante el signo "=". Si una etiqueta admite varios atributos estos se indican como una lista separada por espacios en blanco. Por ejemplo:

```
<etiqueta atributo1 atributo2=valor2 atributo3=valor3>...</etiqueta>
```

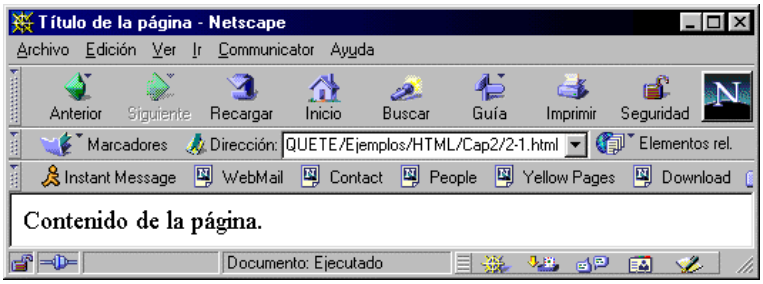
Como se puede ver la etiqueta está modificada por tres atributos, dos de los cuales toman valores.

Para terminar este punto se presenta un pequeño ejemplo en el que se emplea lo visto hasta ahora (ejemplo 2-1 y figura 2-3).

Ejemplo 2-1 (Un documento *HTML* mínimo)

```
<HTML>
  <HEAD>
    <TITLE>T&iacute;tulo de la p&aacute;gina</TITLE>
  </HEAD>
  <BODY>
    Contenido de la p&aacute;gina.
  </BODY>
</HTML>
```

Figura 2-3



2.2 Jerarquía del documento

Cuando se escribe un documento no se proporciona todo el contenido de forma continua sino que se estructura en capítulos, apartados y párrafos; cada una de estas partes puede llevar títulos con estilos diferentes para indicar un nivel distinto en la jerarquía del documento.

2.2.1 Títulos

En *HTML* se dispone de seis niveles para la jerarquía de títulos; para mostrar un título se encierra el texto del mismo entre las etiquetas **<Hn>**...**</Hn>** donde *n* varía entre 1 (el nivel más alto de la jerarquía) y 6 (el nivel más bajo). El texto delimitado por dichas etiquetas se muestra en negrita y el texto que lo sigue comienza en un párrafo nuevo.

A continuación se muestra un ejemplo de esto, ejemplo 2-2, y la forma en que se visualiza (figura 2-4).

Ejemplo 2-2 (La jerarquía de títulos en *HTML*)

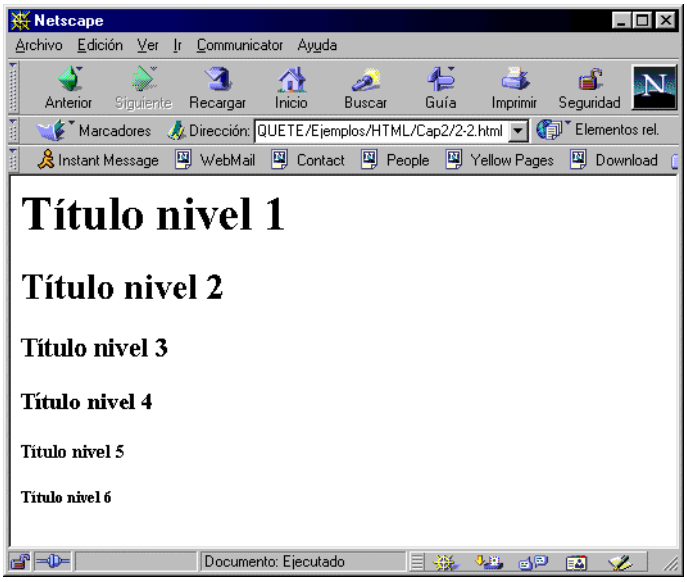
```
<H1>T&iacute;tulo nivel 1</H1>
<H2>T&iacute;tulo nivel 2</H2>
<H3>T&iacute;tulo nivel 3</H3>
<H4>T&iacute;tulo nivel 4</H4>
<H5>T&iacute;tulo nivel 5</H5>
<H6>T&iacute;tulo nivel 6</H6>
```

Estas etiquetas sólo se deben utilizar para títulos de apartados, es decir, con la única finalidad de estructurar el texto y nunca para cambiar el estilo de texto dentro de párrafos.

2.2.2 Saltos de línea y párrafos

Los navegadores al interpretar código *HTML* ignoran los saltos de línea y muestran el texto de forma continua cortando las líneas en función del ancho de la ventana de la aplicación. Cuando el autor de un documento *HTML* quiere introducir un salto de línea debe utilizar la etiqueta **
**, esta etiqueta no tiene cierre.

Figura 2-4



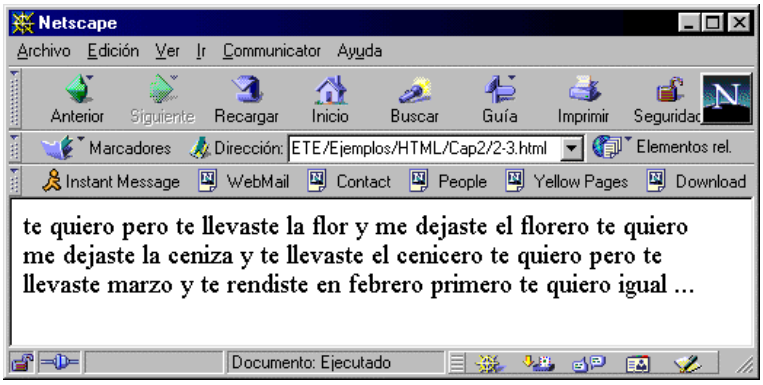
Esto quedará más claro con un ejemplo:

Ejemplo 2-3 (Saltos de línea en *HTML* I)

```
te quiero pero te llevaste la flor
y me dejaste el florero
te quiero me dejaste la ceniza
y te llevaste el cenicero
te quiero pero te llevaste marzo
y te rendiste en febrero
primero te quiero igual
...
```

Código anterior se visualiza de esta forma:

Figura 2-5

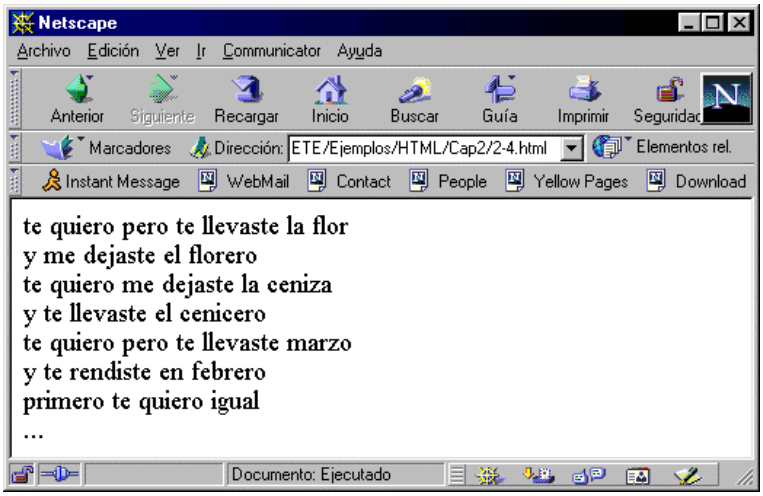


Evidentemente, esto no es lo que se pretende, para que haya un verso por línea hay que hacer lo que se muestra en el ejemplo 2-4; el resultado (ver figura 2-6) ahora sí es el apetecido.

Ejemplo 2-4 (Saltos de línea en *HTML* II)

```
te quiero pero te llevaste la flor<BR>
y me dejaste el florero<BR>
te quiero me dejaste la ceniza<BR>
y te llevaste el cenicero<BR>
te quiero pero te llevaste marzo<BR>
y te rendiste en febrero<BR>
primero te quiero igual<BR>
... <BR>
```

Figura 2-6



Ahora bien, de qué manera se puede agrupar el texto en párrafos dejando que el navegador se encargue de ajustar el tamaño de las líneas en función del tamaño de la ventana; para hacer esto existe las etiquetas **<P>**...**</P>** que colocan el texto que delimitan en un párrafo manteniendo un espacio a su alrededor para separarlo del texto que lo precede y lo sigue. Por ejemplo:

Ejemplo 2-5 (Uso de párrafos)

```
<P>
  This specification defines the HyperText Markup Language (HTML), version 4.0, the
  publishing language of the World Wide Web. In addition to the text, multimedia,
  and hyperlink features of the previous versions of HTML, HTML 4.0 supports more
  multimedia options, scripting languages, style sheets, better printing
  facilities, and documents that are more accessible to users with disabilities.
  HTML 4.0 also takes great strides towards the internationalization of documents,
  with the goal of making the Web truly World Wide.
</P>

<P>
  This document has been reviewed by W3C Members and other interested parties and
  has been endorsed by the Director as a W3C Recommendation. It is a stable
  document and may be used as reference material or cited as a normative reference
  from another document. W3C's role in making the Recommendation is to draw
  attention to the specification and to promote its widespread deployment. This
  enhances the functionality and interoperability of the Web.
</P>
```

La visualización del código se puede ver en la figura 2-7.

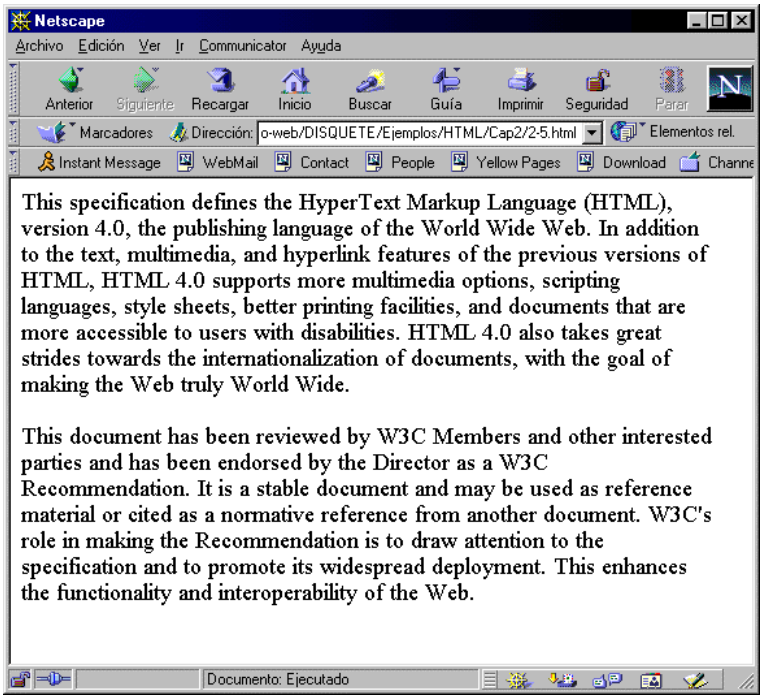
2.3 Utilización de estilos

El texto es el contenido fundamental de un documento *HTML* y un texto es más legible y tiene un mayor impacto visual si se utilizan diversos estilos para marcar de esta forma distintos matices; por ejemplo, **utilizar la negrita para marcar conceptos importantes** o *la cursiva para términos nuevos*, etc.

2.3.1 Estilos físicos y lógicos

En *HTML* se pueden emplear dos tipos de estilos, los llamados estilos físicos y los estilos lógicos. Los primeros son **negrita**, *cursiva*, subrayado y `teletipo`; dentro de los segundos se encuentran *cita*, *código*, *énfasis* y *fuerte*, entre otros. Los estilos lógicos se denominan así puesto que aunque el texto marcado con uno de esos estilos se muestra de forma diferente a como lo hace el texto normal el autor del documento desconoce la forma en que el navegador lo visualizará.

Figura 2-7



Las etiquetas a utilizar para cada estilo físico son las siguientes:

- **<B>**...**</B>**: para texto en negrita (*bold*).
- **<I>**...**</I>**: para texto en cursiva (*italic*).
- **<U>**...**</U>**: para texto subrayado (*underlined*), este estilo no es muy recomendable puesto que los *enlaces* se visualizan, generalmente, como texto subrayado.
- **<TT>**...**</TT>**: para texto teletipo (*teletype*).

En cuanto a los estilos lógicos:

- **<CITE>**...**</CITE>**: se utiliza para citar obras o referencias; generalmente, el tipo utilizado es cursiva.
- **<CODE>**...**</CODE>**: permite mostrar fragmentos de código informático; habitualmente en una fuente de tamaño fijo.
- **<VAR>**...**</VAR>**: se utiliza para mostrar variables de programas informáticos.
- **<KBD>**...**</KBD>**: permite simular entradas por teclado mediante el uso de fuentes no proporcionales.
- **<EM>**...**</EM>**: se usa para enfatizar el texto.
- **<STRONG>**...**</STRONG>**: para enfatizar fuertemente el texto; generalmente, se utiliza negrita.
- **<SAMP>**...**</SAMP>**: se usa para mostrar un texto de ejemplo.

A continuación se muestra código de ejemplo para dos de los estilos anteriores:

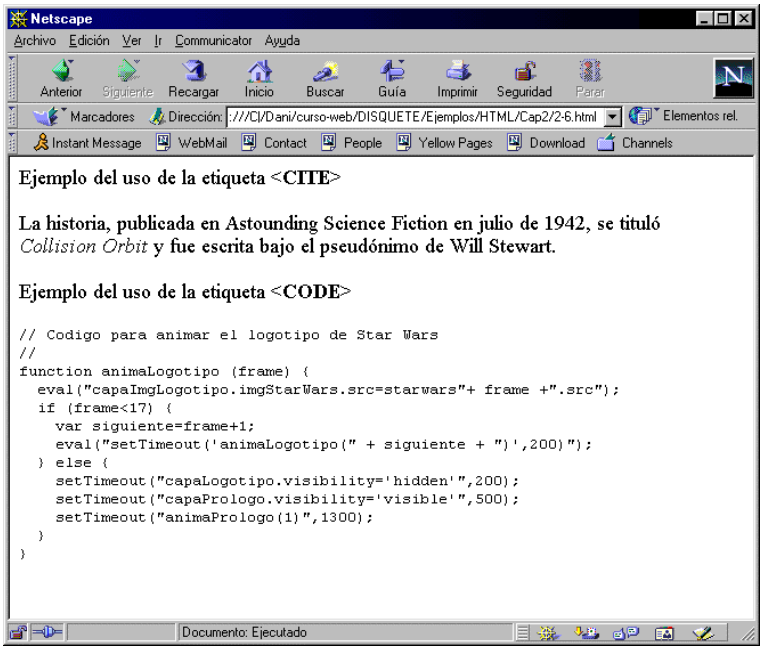
Ejemplo 2-6 (Uso de <CITE> y <CODE>)

```
Ejemplo del uso de la etiqueta <B>&lt;CITE&gt;</B>

<P>
  La historia, publicada en Astounding Science Fiction en julio de 1942, se tituló
  <CITE>Collision Orbit</CITE> y fue escrita bajo el pseud&oacute;nimo de Will
  Stewart.
</P>

Ejemplo del uso de la etiqueta <B>&lt;CODE&gt;</B>

<P>
  <CODE>
    <PRE>
// Codigo para animar el logotipo de Star Wars
//
```

```
function animaLogotipo (frame) {
  eval("capaImgLogotipo.imgStarWars.src=starwars"+ frame +".src");
  if (frame<17) {
    var siguiente=frame+1;
    eval("setTimeout('animaLogotipo(" + siguiente + ")',200)");
  } else {
    setTimeout("capaLogotipo.visibility='hidden'",200);
    setTimeout("capaPrologo.visibility='visible'",500);
    setTimeout("animaPrologo(1)",1300);
  }
}
    </PRE>
  </CODE>
</P>
```

El resultado de visualizar el código anterior es el siguiente:

Figura 2-8



2.3.2 Estilos de párrafo

En el ejemplo anterior el lector puede encontrar una etiqueta nueva, **<PRE>**, dicha etiqueta, junto con su correspondiente etiqueta de cierre, delimita el código de la función **animaLogotipo** logrando que se visualice exactamente igual a como fue escrito en el documento *HTML*; es decir, el navegador **no** ha ignorado los saltos de línea que hay en el código del documento sino que los ha utilizado. En definitiva, esta etiqueta permite *preformatear* el texto puesto que el autor del documento determina la forma en que aparecerá en pantalla.

La etiqueta **<PRE>** junto con las etiquetas **<ADDRESS>**, **<BLOCKQUOTE>** y alguna otra se conocen como etiquetas de estilo de párrafo; la primera ya ha sido comentada, con lo cual se pasará a hablar de las dos siguientes.

La etiqueta **<ADDRESS>**...**</ADDRESS>** se utiliza para delimitar la información que permita localizar al autor del documento; generalmente los navegadores muestran ese texto en cursiva.

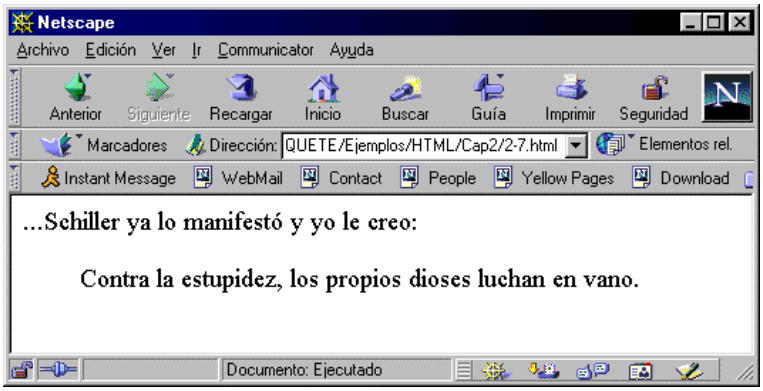
<BLOCKQUOTE>...**</BLOCKQUOTE>** permite citar texto de otros autores, el bloque de texto enmarcado por ambas etiquetas se coloca en un nuevo párrafo y aparece indentado. Por ejemplo el código 2-7.

Ejemplo 2-7 (Uso de <BLOCKQUOTE>)

```
...Schiller ya lo manifest&oacute; y yo le creo:

<BLOCKQUOTE>
  Contra la estupidez, los propios dioses luchan en vano.
</BLOCKQUOTE>
```

En estos momentos el lector ya puede formatear el texto de un documento, distribuirlo en secciones y darle diversos estilos; en el próximo capítulo aprenderá a preparar listas.

Figura 2-9



3. Creación de listas

Las listas son un elemento muy útil para presentar información de una forma clara y organizada. En *HTML* se pueden crear dos tipos distintos de listas: *descriptivas* y *regulares*; las primeras permiten mostrar una serie de conceptos junto con su descripción formando un glosario, las segundas se utilizan para realizar enumeraciones de elementos.

3.1 Listas ordenadas y no ordenadas

Las listas regulares pueden ser *ordenadas* y *no ordenadas*; el nombre no hace referencia a la existencia (o inexistencia) de un orden interno en la enumeración sino al ítem que precede a cada elemento de la lista: en el primer caso se trata de un número, generalmente, mientras que en el segundo aparece una viñeta.

La etiqueta **<LI>** es común a ambos tipos de lista, precediendo a cada elemento de la enumeración. La sintaxis general de las listas regulares es la siguiente:

```
<apertura de lista>
  <LI> elemento 1 de la lista
  <LI> elemento 2 de la lista
  ...
  <LI> elemento n de la lista
<cierre de lista>
```

La diferencia, obviamente, entre una lista ordenada y otra no ordenada reside simplemente en las etiquetas de apertura y cierre de la lista. Para el caso de una lista ordenada se emplean las etiquetas **<OL>**...**</OL>** (*ordered list*); mientras que para listas no ordenadas se utilizan **<UL>**...**</UL>** (*unordered list*).

El siguiente ejemplo ilustrará el uso de ambos tipos de lista (ver figura 3-1).

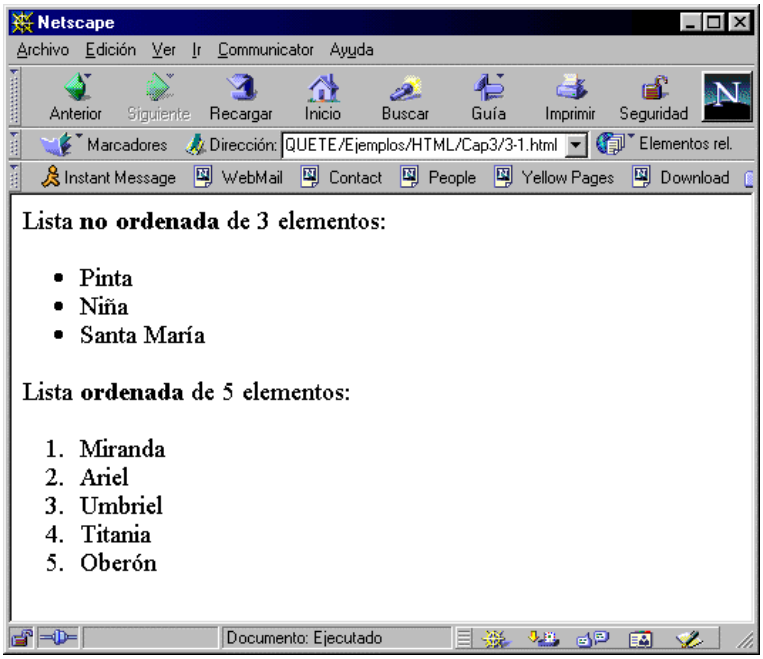
Ejemplo 3-1 (Listas no ordenadas y ordenadas)

```
Lista <B>no ordenada</B> de 3 elementos:

<UL>
  <LI>Pinta
  <LI>Ni&ntilde;a
  <LI>Santa Mar&iacute;a
</UL>

Lista <B>ordenada</B> de 5 elementos:

<OL>
  <LI>Miranda
  <LI>Ariel
  <LI>Umbriel
  <LI>Titania
  <LI>Ober&oacute;n
</OL>
```

3.2 Listas descriptivas

Como ya se dijo, las listas descriptivas permiten crear glosarios; por tanto, cada elemento de la lista tendrá dos partes, la primera será el elemento a describir y la segunda la descripción del mismo. Para poder hacer esto en *HTML* se precisan tres etiquetas: una para indicar que se trata de una lista descriptiva, **<DL>**, otra para el nombre del elemento a describir, **<DT>**, y finalmente otra para la descripción del mismo, **<DD>**.

Figura 3-1



A continuación se muestra un ejemplo de una de estas listas (ver figura 3-2).

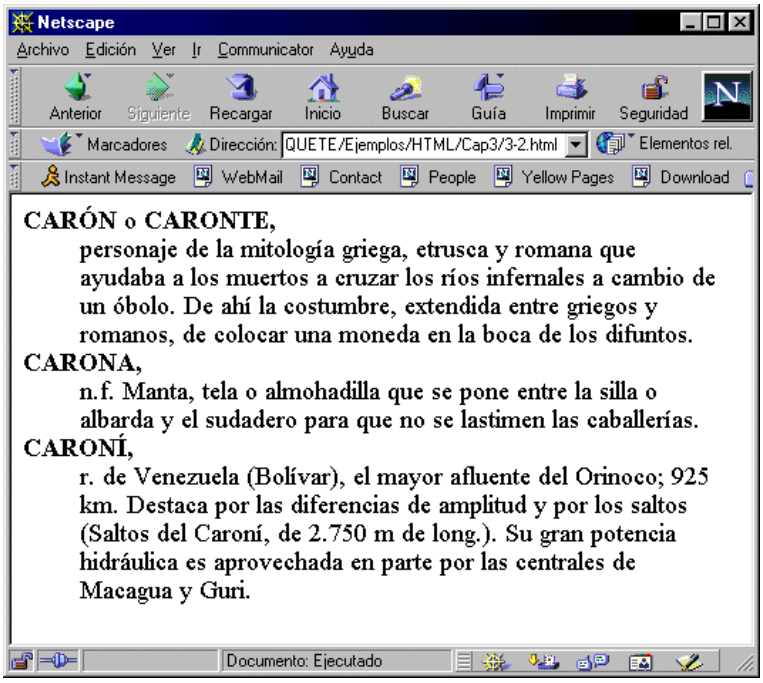
Ejemplo 3-2 (Lista descriptiva)

```
<DL>

<DT><B>CAR&Oacute;N</B> o <B>CARONTE,</B>
<DD>personaje de la mitolog&iacute;a griega, etrusca y romana que ayudaba a los
muertos a cruzar los r&iacute;os infernales a cambio de un &oacute;bolo. De
ah&iacute; la costumbre, extendida entre griegos y romanos, de colocar una moneda
en la boca de los difuntos.

<DT><B>CARONA,</B>
<DD>n.f. Manta, tela o almohadilla que se pone entre la silla o albarda y el
sudadero para que no se lastimen las caballer&iacute;as.

<DT><B>CARON&Iacute;,</B>
<DD>r. de Venezuela (Bol&iacute;var), el mayor afluente del Orinoco; 925 km.
Destaca por las diferencias de amplitud y por los saltos (Saltos del Caron&iacute;,
de 2.750 m de long.). Su gran potencia hidr&aacute;ulica es aprovechada en parte
por las centrales de Macagua y Guri.

</DL>
```

3.3 Anidamiento de listas

El lector ya habrá comprobado que las etiquetas *HTML* pueden anidarse de tal forma que varias pueden ser aplicadas a un mismo texto. Las listas no son una excepción y un caso típico de anidamiento de etiquetas se da en la creación de listas (ejemplo y figura 3-3).

Figura 3-2



Ejemplo 3-3 (Lista anidada)

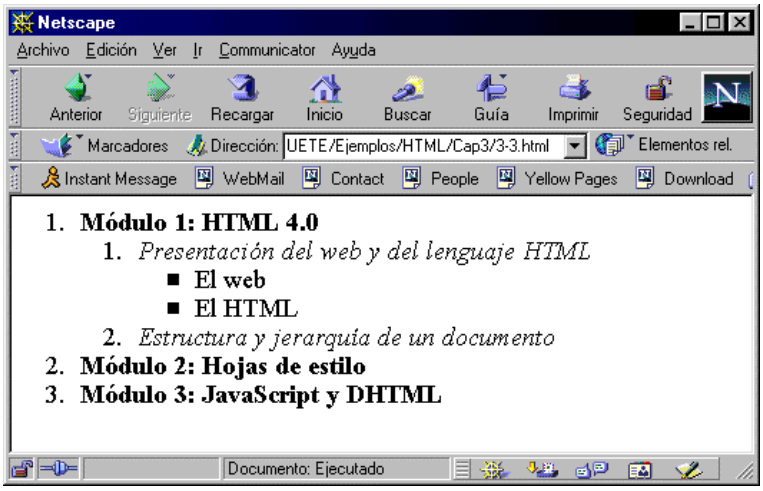
```
<OL>
  <LI><B>M&oacute;dulo 1: HTML 4.0</B>

  <OL>
    <LI><I>Presentaci&oacute;n del web y del lenguaje HTML</I>

    <UL>
      <LI>El web
      <LI>El HTML
    </UL>

    <LI><I>Estructura y jerarqu&iacute;a de un documento</I>
  </OL>

  <LI><B>M&oacute;dulo 2: Hojas de estilo</B>
  <LI><B>M&oacute;dulo 3: JavaScript y DHTML</B>
</OL>
```

3.4 Visualización de listas

Los navegadores tienen unas opciones por defecto a la hora de representar los ítems que preceden a las listas ordenadas y no ordenadas; sin embargo el autor de documentos *HTML* puede indicar sus preferencias mediante un *atributo* (ver la sección titulada “Estructura de un documento *HTML*” en la página 2-5) de las etiquetas **<UL>** y **<OL>** denominado **TYPE**. Dicho atributo puede tomar los siguientes valores para listas no ordenadas: **DISC**, **CIRCLE** y **SQUARE**; y para las listas ordenadas: **1**, **a**, **A**, **i** e **I**.

El ejemplo y la figura 3-4 muestran las posibilidades de visualización de listas que ofrece *HTML 4.0*.

Figura 3-3

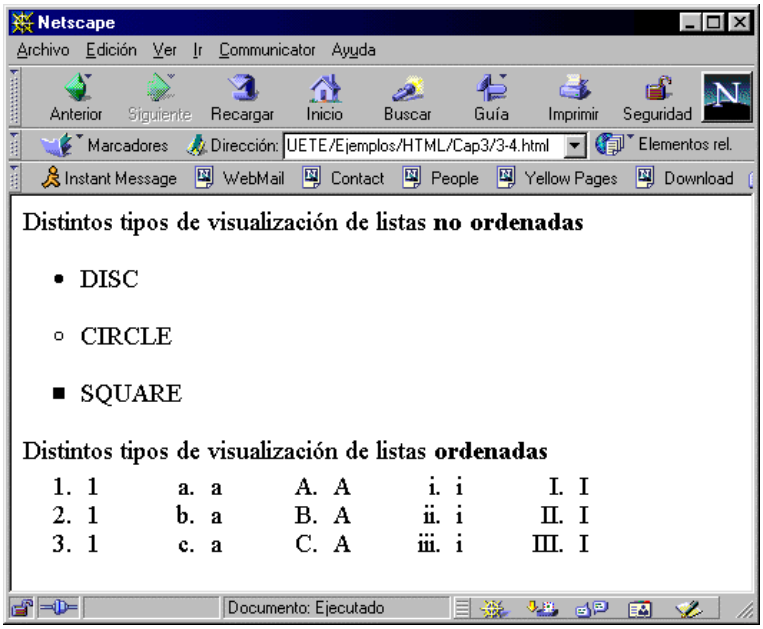


Ejemplo 3-4 (Visualización de listas)

```
Distintos tipos de visualizaci&oacute;n de listas <B>no ordenadas</B>

<UL TYPE=DISC>
  <LI>DISC
</UL>

<UL TYPE=CIRCLE>
  <LI>CIRCLE
</UL>

<UL TYPE=SQUARE>
  <LI>SQUARE
</UL>

Distintos tipos de visualizaci&oacute;n de listas <B>ordenadas</B>

<TABLE><TR>
<TD>
  <OL TYPE=1>
    <LI>1  <LI>1  <LI>1
  </OL>
</TD>
<TD>
  <OL TYPE=a>
    <LI>a  <LI>a  <LI>a
  </OL>
</TD>
<TD>
  <OL TYPE=A>
    <LI>A  <LI>A  <LI>A
  </OL>
</TD>
<TD>
  <OL TYPE=i>
    <LI>i  <LI>i  <LI>i
  </OL>
</TD>
<TD>
  <OL TYPE=I>
    <LI>I  <LI>I  <LI>I
  </OL>
</TD>
</TR></TABLE>
```

Como puede observar el lector, en el código anterior aparecen una serie de etiquetas nuevas; una de ellas, **<TABLE>,** indica que se está construyendo una tabla, el tema del siguiente capítulo; sin embargo, antes de pasar al mismo será mejor practicar lo visto hasta ahora y para ello nada mejor que hacer el primer ejercicio del curso.

Figura 3-4



4. Creación de tablas

Las tablas no sólo se emplean para tabular datos sino también para un diseño llamativo. *HTML* no es un lenguaje de diseño de páginas sino de documentos; sin embargo, esto no es obstáculo para que un autor de páginas *web* pueda lograr un buen diseño mediante un uso adecuado e imaginativo de las tablas.

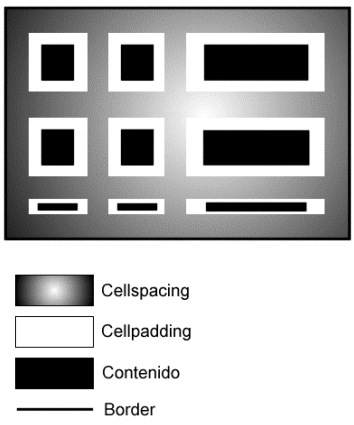
4.1 Etiquetas para construir tablas

La estructura de una tabla es análoga a la definición de una lista; se comienza con una marca de inicio de tabla, seguidamente se describe fila a fila y, finalmente, se indica la marca de fin de tabla. La descripción de cada fila se hace mediante las denominadas celdas (las casillas de la tabla).

Las celdas de una tabla pueden contener cualquier elemento *HTML*: texto, listas, imágenes (ver "Capítulo 8: Uso de imágenes") y, por supuesto, otra tabla.

La etiqueta **<TABLE>** permite la apertura de una tabla; el fin de tabla se especifica con **</TABLE>**. Una tabla tiene los siguientes atributos: **BORDER**, **CELLPADDING** y **CELLSPACING**; el valor de estos atributos se suele especificar en pixels.

Figura 4-1



Un par de atributos suplementarios permiten forzar[3] la tabla a ocupar un cierto porcentaje de la anchura (o altura) de la ventana del navegador o un número de pixels dado; se trata de los atributos **WIDTH** y **HEIGHT**.

<TABLE> admite, además, los atributos **VALIGN** para alineación vertical del contenido de la celda y **ALIGN** para la alineación horizontal. El primero de ellos admite los siguientes valores: **TOP** (superior), **BOTTOM** (inferior) y **MIDDLE** (centrado vertical). El segundo admite: **RIGHT** (derecha), **LEFT** (izquierda) y **CENTER** (centrado horizontal).

La etiqueta **<TR>** inicia una línea de la tabla que terminará con la etiqueta de cierre **</TR>**, mientras que la etiqueta **<TD>** con la correspondiente etiqueta de cierre **</TD>** define una celda. Ambas etiquetas también admiten los atributos **VALIGN** y **ALIGN**.

A continuación se muestra un ejemplo sencillo de tabla en *HTML* (ver figura 4-2).

Ejemplo 4-1 (Una tabla sencilla en *HTML*)

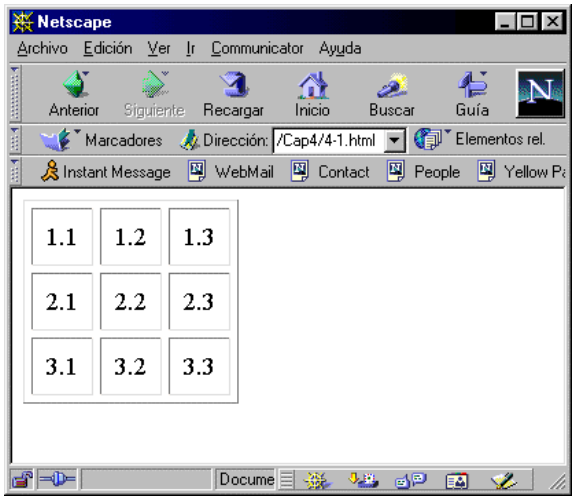
```
<TABLE BORDER=1 CELLSPACING=5 CELLPADDING=10>
  <TR>
    <TD>1.1</TD>
    <TD>1.2</TD>
    <TD>1.3</TD>
  </TR>
  <TR>
    <TD>2.1</TD>
    <TD>2.2</TD>
    <TD>2.3</TD>
  </TR>
  <TR>
    <TD>3.1</TD>
    <TD>3.2</TD>
    <TD>3.3</TD>
  </TR>
</TABLE>
```

[3] Siempre de forma aproximada.

4.2 Visualización de tablas

Las tablas en *HTML* aún permiten más posibilidades; en primer lugar, no todas las mismas filas tienen por qué tener el mismo número de columnas y viceversa todas las columnas el mismo número de filas, sino que se pueden fusionar celdas de varias columnas y/o de varias filas mediante los atributos **`COLSPAN`** y **`ROWSPAN`** de la etiqueta **`<TD>`**.

Figura 4-2

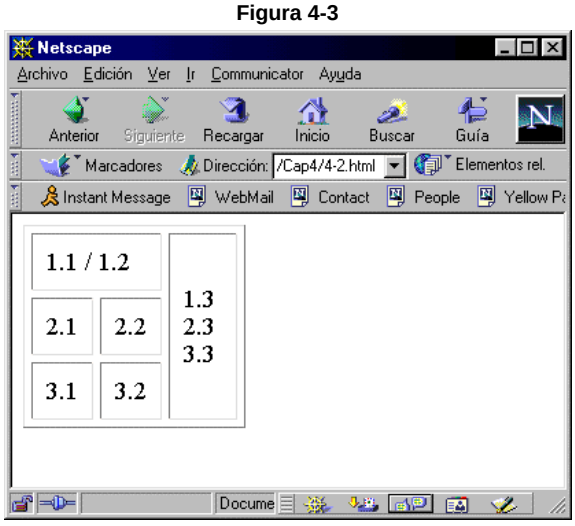


El siguiente ejemplo ilustra el uso de dichos atributos (ver figura 4-3).

Ejemplo 4-2 (Uso de COLSPAN y ROWSPAN)

```
<TABLE BORDER=1 CELLSPACING=5 CELLPADDING=10>
  <TR>
    <TD COLSPAN=2>1.1 / 1.2</TD>
    <TD ROWSPAN=3>1.3 <BR> 2.3 <BR> 3.3</TD>
  </TR>
  <TR>
    <TD>2.1</TD>
    <TD>2.2</TD>
  </TR>
  <TR>
    <TD>3.1</TD>
    <TD>3.2</TD>
  </TR>
</TABLE>
```

Figura 4-3



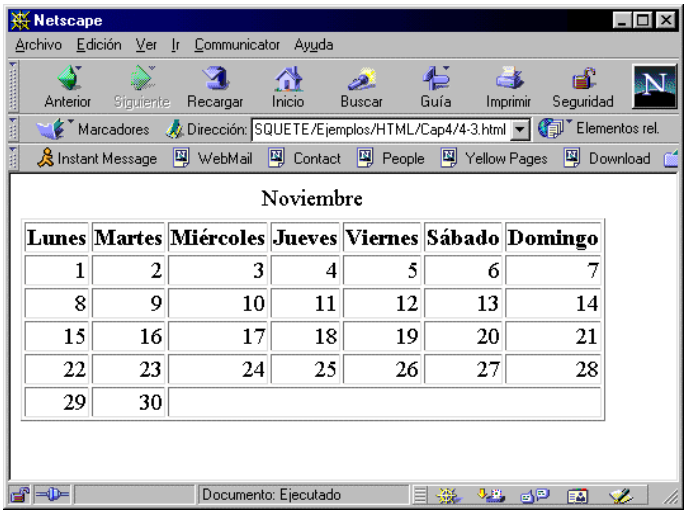
Por último[4], las etiquetas **<TH>** y **<CAPTION>** permiten presentar las tablas de forma mucho más adecuada. La primera se comporta exactamente igual a **<TD>** pero el texto de dichas celdas se considera como texto de cabecera con lo cual se centra automáticamente y se pone en negrita; por su parte, **<CAPTION>** permite poner un título a la tabla bien encima (atributo **ALIGN=TOP**) o debajo (atributo **ALIGN=BOTTOM**). A continuación se muestra un ejemplo:

Ejemplo 4-3 (Uso de <CAPTION> y <TH>)

```
<TABLE BORDER=1>
  <CAPTION ALIGN=TOP>Noviembre</CAPTION>

  <TR>
    <TH>Lunes</TH><TH>Martes</TH><TH>Mi&eacute;rcoles</TH>
    <TH>Jueves</TH><TH>Viernes</TH><TH>S&aacute;bado</TH><TH>Domingo</TH>
  </TR>

  <TR ALIGN=RIGHT>
    <TD>1</TD><TD>2</TD><TD>3</TD><TD>4</TD><TD>5</TD><TD>6</TD><TD>7</TD>
  </TR>

  <TR ALIGN=RIGHT>
    <TD>8</TD><TD>9</TD><TD>10</TD><TD>11</TD><TD>12</TD><TD>13</TD><TD>14</TD>
  </TR>

  <TR ALIGN=RIGHT>
    <TD>15</TD><TD>16</TD><TD>17</TD><TD>18</TD><TD>19</TD><TD>20</TD><TD>21</TD>
  </TR>

  <TR ALIGN=RIGHT>
    <TD>22</TD><TD>23</TD><TD>24</TD><TD>25</TD><TD>26</TD><TD>27</TD><TD>28</TD>
  </TR>

  <TR ALIGN=RIGHT>
    <TD>29</TD><TD>30</TD><TD COLSPAN=5> </TD>
  </TR>
</TABLE>
```

Código que se visualizaría de la siguiente manera:

Figura 4-4



[4] En realidad aún quedan por tratar varios aspectos de las tablas pero no se verán hasta la sección “7.3 Uso del color en tablas”.

5. Uso de enlaces

5.1 Introducción a los enlaces

Hasta el momento el lector ha visto cómo crear documentos textuales bastante ricos pero no sólo de texto vive el *web*; en este capítulo se pasará del texto al *hipertexto*, para ello se explicará cómo utilizar los enlaces *HTML* aunque, como se verá, su utilización no se reduce a establecer la red de documentos.

5.2 Utilización de los enlaces para apuntar recursos

El uso más habitual de los enlaces en un documento *HTML* es apuntar a recursos accesibles vía *web*; la mayor parte de tales recursos apuntados serán otros documentos *HTML* pero también pueden ser imágenes, vídeos, u otros recursos *Internet* aún más "exóticos".

Esto tipo de enlaces se basan en el uso de la etiqueta **<A>** (*anchor*) y su correspondiente etiqueta de cierre; dicha etiqueta tiene un atributo denominado **HREF** que toma como valor el *URI* (ver el punto "1.1.2 Introducción a los *URI's*") del recurso que se desea apuntar. En el siguiente punto se tratarán los distintos recursos accesibles desde un documento *HTML*.

5.3 Distintos tipos de recursos accesibles

A continuación se muestran una serie de ejemplos de enlaces *HTML* apuntando a distintos recursos.

Tabla 5-1

Esquema de acceso al recurso	Ejemplo de enlace

http	`<A HREF="http://giworks.uniovi.es/cursos/cursos.shtml">...</A>`
	`<A HREF="http://giworks.uniovi.es/imágenes/cursos.gif">...</A>`
	`<A HREF="http://giworks.uniovi.es/descarga/EAC.zip">...</A>`

ftp	`<A HREF="ftp://ftp.uniovi.es/linux/internet/apache.tar.gz">...</A>` `<A HREF="ftp://ftp.uniovi.es/unix/doc/man7.ps">...</A>`
news	`<A HREF="news:comp.lang.tcl">...</A>`
mailto	`<A HREF="mailto:zz97f019@centauro.aulario.uniovi.es">...</A>`

Como se puede ver, existen dos esquemas de acceso que permiten acceder a varios tipos de ficheros: mediante los protocolos *HTTP* y *FTP* se puede acceder a documentos *HTML*, imágenes, ficheros comprimidos (.ZIP, .TAR.GZ), documentos de texto en *PostScript* (.PS), etc. A efectos del usuario, las principales diferencias entre acceder a un recurso mediante *HTTP* o *FTP* son las siguientes:

- Al acceder mediante *HTTP* generalmente se visualizan los recursos accedidos siempre y cuando sean de un tipo conocido; en caso contrario se informa al usuario sobre la posibilidad de guardar el archivo en disco o abrirlo con otro programa.

- Al acceder mediante *FTP* el recurso no se visualiza sino que se guarda en disco; es importante hacer notar que para un acceso *FTP* no importa la naturaleza del archivo puesto que simplemente se transfiere[5].

Los otros dos esquemas de acceso, *news* y *mailto*, sirven para acceder a servicios *Internet* bastante más antiguos que el *web* pero tremendamente útiles. El primero permite acceder a un grupo de noticias (más bien un tablón de anuncios) dedicado a un tema específico para hacer o responder preguntas de otros usuarios interesados en el mismo tema. El segundo permite enviar correo electrónico a un usuario en *Internet*.

[5] El protocolo *FTP* no solo permite transferir ficheros desde un servidor al cliente (*download*, "bajar", descargar) sino también del cliente al servidor (*upload*, "subir").

De todos los esquemas de acceso a recursos *Internet* el más utilizado es el *http* por lo que se explicará con mayor detalle su uso, especialmente para acceder a documentos *HTML*.

5.3.1 Enlaces externos

Se dice que un enlace es externo cuando al activarse se carga un recurso distinto al actual; para convertir una zona del documento *HTML* en un enlace externo debe estar delimitada de la siguiente forma:

```
<A HREF=URI>zona de enlace</A>
```

El lector ya está familiarizado con el concepto de *URI* pero existe una cuarta sección dentro de un *URI* que aún no ha sido presentada por razones que se hacen obvias en el punto siguiente.

El siguiente es un *URI* completo

```
http://   giworks.uniovi.es   /cursos/cursos.shtml   #noviembre99
```

Préstese atención a la última parte del mismo, siguiendo al nombre del documento *HTML* apuntado hay un texto precedido del carácter "#", esa sección del *URI* indica lo que se denomina un enlace interno. Al activar un enlace con un *URI* de ese tipo no solo se carga el documento en cuestión sino que se avanza hasta alcanzar un punto en concreto del mismo (en este caso una zona del documento *HTML* con un enlace interno denominado **noviembre99**).

5.3.2 Enlaces internos

Los enlaces internos permiten desplazarse dentro de un documento *HTML*; para ello hay que emplear un enlace de llegada (el que marca una zona particular del documento) y un enlace de partida con un *URI* relativo (tratados en el punto siguiente).

Un enlace de llegada se crea delimitando la zona de llegada de la forma siguiente:

```
<A NAME="nombre_zona">zona de llegada</A>
```

Donde **nombre_zona** será el texto que seguirá al carácter "#" en el *URI* del enlace de partida.

5.3.3 *URI's* relativos

Los *URI's* de los enlaces no tienen por qué utilizarse siempre completos:

```
http://   máquina   path/documento   #sección
```

De hecho las formas más habituales son las siguientes:

`http://  máquina`	`http://giworks.uniovi.es`
`http://  máquina  path`	`http://giworks.uniovi.es/cursos`
`path`	`/cursos`
`path/documento`	`/cursos/cursos.shtml`
`#sección`	`#noviembre99`

En el primer caso se trataría de un enlace externo que cargaría la página principal de un servidor *web*; en el segundo se cargaría un documento especial (denominado generalmente **index.html**) de un *path* concreto; los casos tercero y cuarto son enlaces externos porque se cargaría un nuevo documento aunque dicho documento residiría en el mismo servidor que el documento actual y el último sería un *enlace interno* puesto que no se carga un nuevo documento sino que se va a una sección distinta del documento actual.

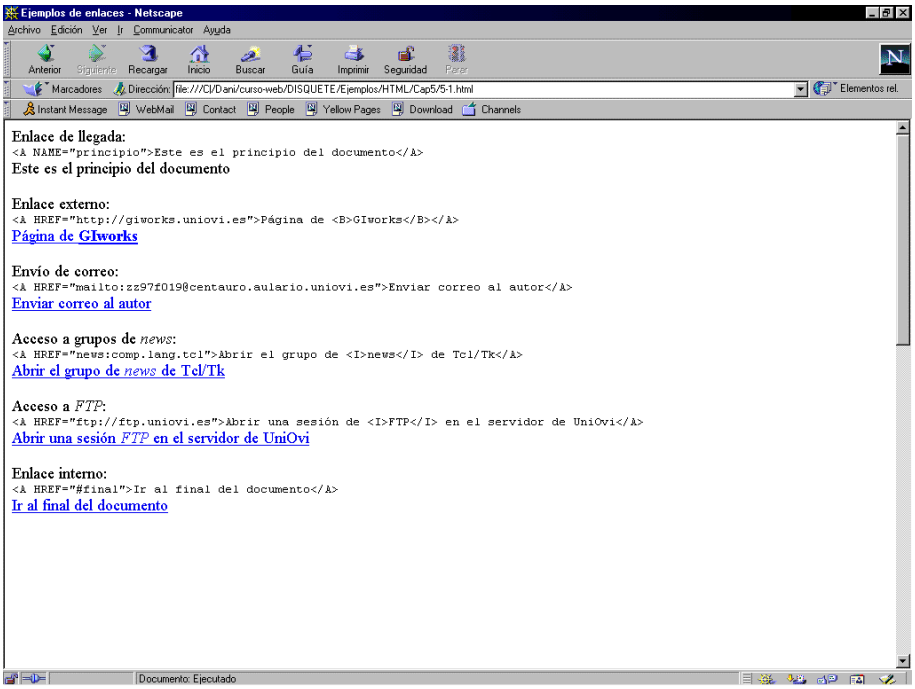
En las páginas siguientes se muestra código de ejemplo para los puntos anteriores (ejemplo 5-1) junto con su visualización (figura 5-1).

Ejemplo 5-1 (Uso de enlaces)

```
<HTML>
  <HEAD>
    <TITLE>Ejemplos de enlaces</TITLE>
  </HEAD>

  <BODY>
    <P>
    Enlace de llegada:<BR>
    <CODE>&lt;A NAME="principio"&gt;Este es el principio del
documento&lt;/A&gt;</CODE><BR>
    <A NAME="principio">Este es el principio del documento</A>
    </P>

    <P>
    Enlace externo:<BR>
    <CODE>&lt;A HREF="http://giworks.uniovi.es"&gt;P&aacute;gina de
&lt;B&gt;GIworks&lt;/B&gt;&lt;/A&gt;</CODE><BR>
    <A HREF="http://giworks.uniovi.es">P&aacute;gina de <B>GIworks</B></A>
    </P>

    <P>
    Env&iacute;o de correo:<BR>
    <CODE>&lt;A HREF="mailto:zz97f019@centauro.aulario.uniovi.es"&gt;Enviar correo
al autor&lt;/A&gt;</CODE><BR>
    <A HREF="mailto:zz97f019@centauro.aulario.uniovi.es">Enviar correo al autor</A>
    </P>

    <P>
    Acceso a grupos de <I>news</I>:<BR>
    <CODE>&lt;A HREF="news:comp.lang.tcl"&gt;Abrir el grupo de
&lt;I&gt;news&lt;/I&gt; de Tcl/Tk&lt;/A&gt;</CODE><BR>
    <A HREF="news:comp.lang.tcl">Abrir el grupo de <I>news</I> de Tcl/Tk</A><BR>
    </P>

    <P>
    Acceso a <I>FTP</I>:<BR>
    <CODE>&lt;A HREF="ftp://ftp.uniovi.es"&gt;Abrir una sesi&oacute;n de
&lt;I&gt;FTP&lt;/I&gt; en el servidor de UniOvi&lt;/A&gt;</CODE><BR>
    <A HREF="ftp://ftp.uniovi.es">Abrir una sesi&oacute;n <I>FTP</I> en el servidor
de UniOvi</A><BR>
    </P>

    <P>
    Enlace interno:<BR>
    <CODE>&lt;A HREF="#final"&gt;Ir al final del documento&lt;/A&gt;</CODE><BR>
    <A HREF="#final">Ir al final del documento</A><BR>
    </P>

    <TABLE HEIGHT=150%><TR><TD> </TD></TR></TABLE>

    <P>
    Enlace de llegada:<BR>
    <CODE>&lt;A NAME="final"&gt;Este es el final del documento&lt;/A&gt;</CODE><BR>
    <A NAME="final">Este es el final del documento</A><BR>
    </P>

    <P>
    Enlace interno:<BR>
    <CODE>&lt;A HREF="#principio"&gt;Volver al principio del
documento&lt;/A&gt;</CODE><BR>
    <A HREF="#principio">Volver al principio del documento</A><BR>
    </P>
  </BODY>
</HTML>
```

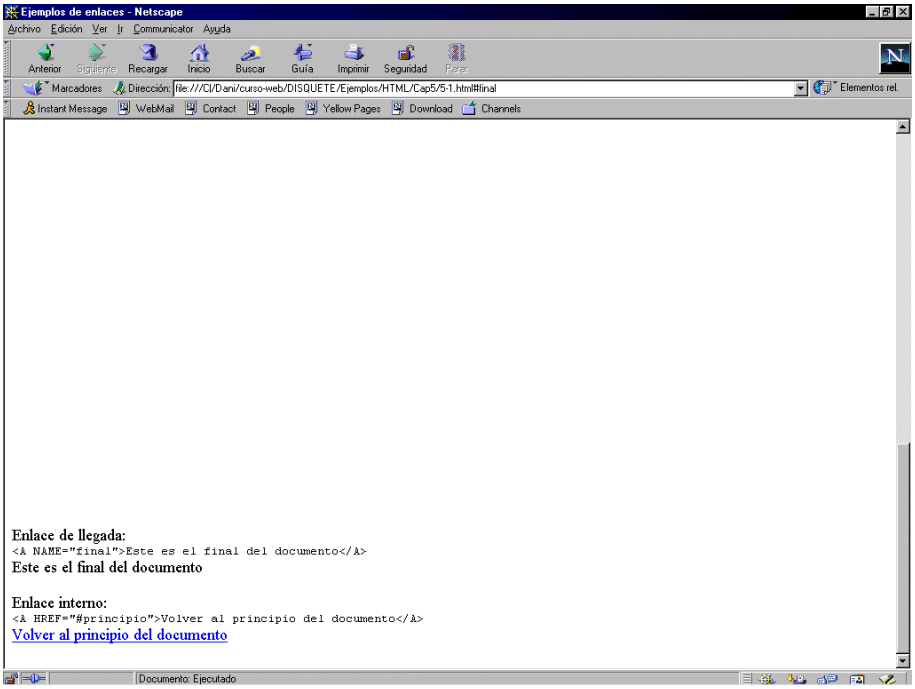
Al visualizar el código anterior se obtiene este resultado:

Figura 5-1



Si se activa el último enlace visible, **`Ir al final del documento`**, el navegador avanza a través del documento:

Figura 5-2



Ahora si se pulsa el enlace **`Volver al principio del documento`** se volvería a la situación anterior.

5.4 Utilización de los enlaces para indicar relaciones o incluir documentos

Existe otro tipo de enlaces que sólo pueden colocarse en la cabecera del documento *HTML* y que, por tanto, no se visualizarían sino que sirven para indicar relaciones entre o incluir documentos[6]; dichos enlaces no utilizan la etiqueta **<A>** sino otra distinta, **<LINK>**. Dicha etiqueta tiene los siguientes atributos:

- **HREF**: indica el documento al que "apuntaría" este enlace, es decir, aquel con el cual el documento actual tiene relación o debe incluir.
- **REL** y **REV**: indican relaciones (hacia delante y hacia atrás, respectivamente) aunque el más habitual es **REL** que puede tomar, entre otros, los siguientes valores: **NEXT**, **PREV** e **INDEX**.

Una posible utilización de estos "enlaces" sería para establecer relaciones jerárquicas entre documentos de tal manera que puedan ser utilizadas por los motores de búsqueda para generar información más completa sobre un tema; aunque el uso más habitual es para incluir documentos externos tales como hojas de estilo (ver "Módulo 2").

Y ahora, antes de pasar al siguiente capítulo, se recomienda al lector realizar el segundo ejercicio.

[6] Esta posibilidad se utilizará ampliamente con las hojas de estilo.

6. Creación de *frames*

6.1 Introducción a los frames

Los *frames* (en algunos lugares se traducen como “marcos”) permiten dividir el área visible del navegador en más de una zona, estas zonas tienen una serie de propiedades tales como: poder cargar su propio *URI* independientemente del resto de *frames*; puede ser nombrada y, por tanto, apuntada; cambia de tamaño dinámicamente en respuesta a cambios de tamaño del área visible del cliente y también se puede evitar que el usuario cambie manualmente su tamaño.

De esta forma se puede mantener parte de la información de forma estática (un menú de navegación, por ejemplo) y el resto en una vista con desplazamiento (*scroll*) que puede ser modificada desde la zona estática.

6.2 Cambios en la estructura del documento HTML

El empleo de *frames* implica que en cada vista que se divida la ventana del navegador aparecerá un documento *HTML* distinto; dichos documentos serán exactamente iguales a los vistos hasta ahora, la única diferencia que se produce entre preparar una página *web* corriente y otra con *frames* estriba en la necesidad de describir un documento *HTML* extra que describa la disposición de los *frames* en la ventana.

Un documento de *frames* tiene una estructura muy similar a la de un documento *HTML* salvo que carece de cuerpo, no se utiliza la etiqueta **<BODY>**, y tiene una zona donde se describen las divisiones (los *frames*), utilizando la etiqueta **<FRAMESET>**:

```
<HTML>
  <HEAD>
    <TITLE>...</TITLE>
  </HEAD>

  <FRAMESET>
    ...
  </FRAMESET>
</HTML>
```

La etiqueta **<FRAMESET>** tiene los atributos **ROWS** y **COLS** para determinar si la división se hará de forma horizontal o vertical. Ambos atributos toman como valor una lista de elementos separados por comas; cada elemento indicará una altura o anchura de una de las siguientes formas:

- **entero**: Se toma como un número de pixels.
- **1..100%**: Un porcentaje de las dimensiones de la ventana que acoge el documento de *frames*.
- **[entero]***: El uso de asteriscos permite crear *frames* de tamaño relativo; esto es, si se describen varios *frames* indicando sus dimensiones y uno de tamaño relativo éste ocupará el espacio restante, si se definen dos *frames* relativos se reparten a partes iguales dicho espacio, pero si se definieran de la forma ***,2*** entonces el primero ocuparía 1/3 del espacio libre y el otro los 2/3 del espacio restante.

Dentro de la sección enmarcada por **<FRAMESET>**...**</FRAMESET>** se colocarán etiquetas **<FRAME>** para describir cada división mediante los siguientes atributos (tantas como filas o columnas se hayan descrito implícitamente mediante los atributos **ROWS** o **COLS**):

- **SRC**: toma como valor el *URI* del documento a visualizar en este *frame* en particular.
- **NAME**: se utiliza para nombrar el *frame* y poder apuntarlo desde enlaces en otros documentos (típicamente en otros *frames* del mismo documento).
- **MARGINWIDTH**: el autor puede indicar un tamaño en pixels para el ancho del margen que tiene un *frame*.
- **MARGINHEIGHT**: análogo a **MARGINWIDTH**.
- **SCROLLING**: admite tres valores, **YES**, **NO** y **AUTO** para indicar si se obliga a que el *frame* tenga barra de *scroll*, se impide que la tenga o si se deja a elección del navegador.
- **NORESIZE**: un atributo sin valor que indica que no se puede cambiar el tamaño del *frame*.

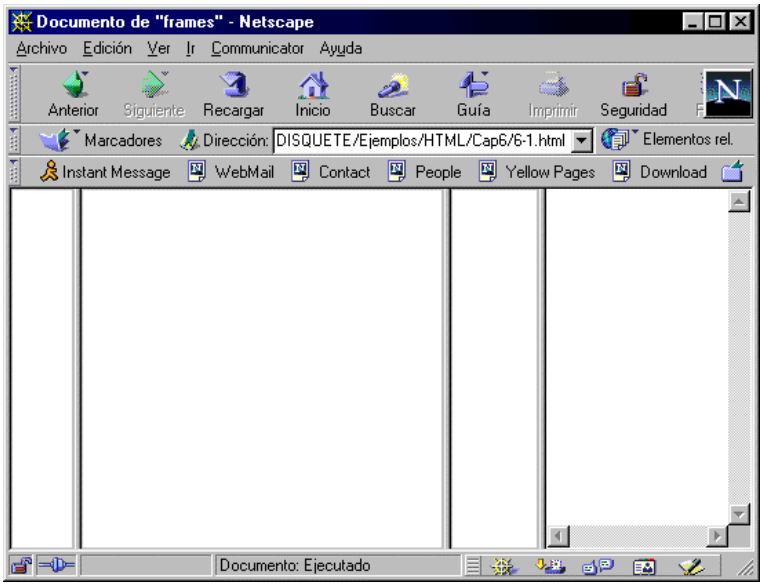
Un hecho que debe ser tenido en cuenta es la existencia de navegadores que no soportan el uso de *frames*; para ello conviene utilizar las etiquetas **<NOFRAMES>...</NOFRAMES>** encerrando entre ellas un texto que indique a tales usuarios la imposibilidad de mostrarles el documento (o posibilitar una navegación opcional). El funcionamiento de esta etiqueta es un tanto artificioso: los navegadores que no soportan *frames* no conocen la etiqueta **<NOFRAMES>** con lo cual de un documento de *frames* sólo podrían interpretar el texto delimitado en esa sección y lo visualizarían; mientras, los navegadores que soportan *frames* reconocen la etiqueta y saben que el texto delimitado no debe ser visualizado.

A continuación se muestra un ejemplo que ilustra lo visto hasta el momento[7]:

Ejemplo 6-1 (Ejemplo sencillo de *frames*)

```
<HTML>
  <HEAD>
    <TITLE>Documento de "frames"</TITLE>
  </HEAD>

  <FRAMESET COLS="50, 50%, *, 2*">
    <FRAME SRC="nada.htm">
    <FRAME SRC="nada.htm" SCROLLING=AUTO>
    <FRAME SRC="nada.htm" SCROLLING=NO>
    <FRAME SRC="nada.htm" SCROLLING=YES>
  </FRAMESET>
</HTML>
```

Figura 6-1



Como se puede observar en la figura 6-1, la primera columna tiene una anchura de 50 pixels, la segunda ocupa la mitad de la ventana y las otras dos se reparten el resto del espacio, la primera 1/3 del mismo y la segunda 2/3. El último *frame* es el único que tiene barras de *scroll*, pero inactivas porque se especificado la obligatoriedad de las mismas aún siendo innecesarias.

6.3 Anidamiento de frames

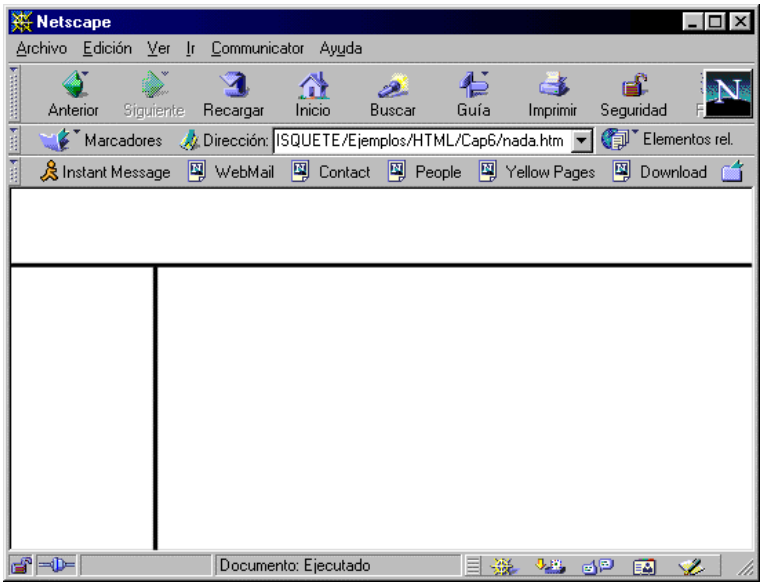
El lector estará pensando que no es excesivamente útil dividir en múltiples filas o columnas la ventana del navegador sino que resultará más interesante algo como lo que se muestra en la figura 6-2.

Para realizar este tipo de divisiones *HTML* permite anidar elementos **<FRAMESET>**, en el caso anterior se utilizaría un elemento **<FRAMESET>** para crear dos filas, en la primera se incluiría un *frame* con un

[7] A partir de ahora se usará en varias ocasiones el fichero **nada.htm**, se trata de un documento *HTML* sin contenido (tan sólo la estructura del documento, sin cuerpo).

documento *HTML* y en la segunda otro **<FRAMESET>** para crear dos columnas. El código tendría el aspecto que muestra el ejemplo 6-2.

Figura 6-2



Ejemplo 6-2 (Anidamiento de *frames*)

```
<HTML>
  <HEAD>
    <TITLE>Documento de "frames"</TITLE>
  </HEAD>

  <FRAMESET ROWS="100, *">
    <FRAME SRC="nada.htm">

    <FRAMESET COLS="150, *">
      <FRAME SRC="nada.htm">
      <FRAME SRC="nada.htm">
    </FRAMESET>
  </FRAMESET>
</HTML>
```

El resultado de visualizar dicho código se puede ver en la figura 6-3.

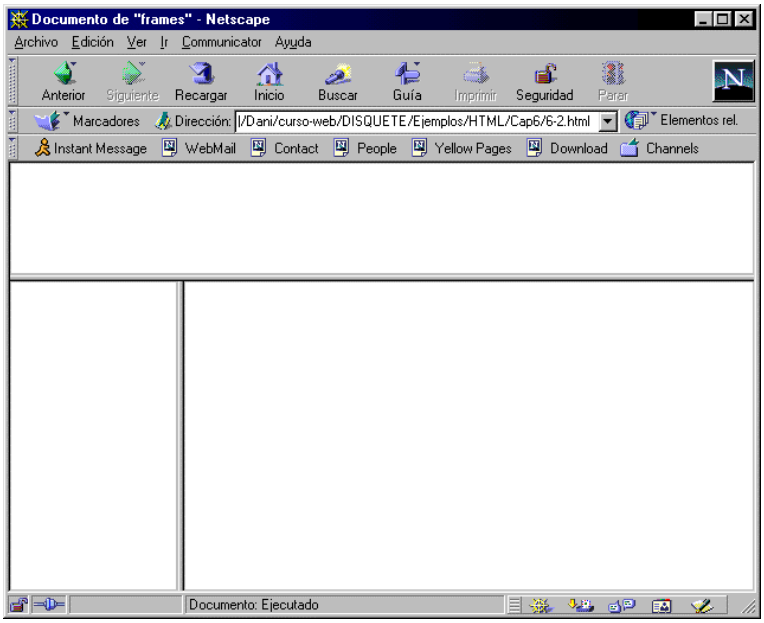
6.4 Empleo de enlaces y frames

El uso de enlaces y *frames* de forma simultánea permite introducir un nuevo atributo de la etiqueta **<A>**, el denominado **TARGET** (objetivo); este atributo toma como valor una cadena de texto que indica dónde (ventana del navegador o *frame*) va a cargarse el documento apuntado por el enlace. Al trabajar con *frames* ese atributo tomará, en la mayor parte de las ocasiones, el nombre que se le haya asignado al *frame* elegido para visualizar la información aunque existen una serie de nombres predefinidos que pueden resultar muy útiles:

- **_blank**: el recurso se carga en una nueva ventana.
- **_self**: el recurso se carga en el mismo *frame* en que se encuentra el enlace.
- **_parent**: el recurso se carga en el elemento **<FRAMESET>** padre del *frame* en que se encuentra el enlace; si no tiene padre equivale a _top.
- **_top**: el recurso se carga en la ventana original eliminando los *frames*.

A continuación se muestran dos ficheros *HTML*, el ejemplo 6-3 corresponde al documento de *frames* y el 6-4 al menú de navegación.

Figura 6-3



Ejemplo 6-3 (Documento de *frames*)

```
<HTML>
  <HEAD>
    <TITLE>Documento de "frames"</TITLE>
  </HEAD>

  <FRAMESET ROWS="100, *">
    <FRAME NAME="menu" SRC="menu.htm" NORESIZE SCROLLING=NO>
    <FRAME NAME="visualizacion" SRC="nada.htm">
  </FRAMESET>
</HTML>
```

Ejemplo 6-4 (Menú de navegación)

```
<HTML>
  <HEAD>
    <TITLE>El men&uacute;</TITLE>
  </HEAD>

  <BODY>
    <UL>
      <LI><A HREF="http://giworks.uniovi.es" TARGET="visualizacion">P&aacute;gina
de <B>GIworks</B></A>
      <LI><A HREF="http://www.uniovi.es" TARGET="visualizacion">Universidad de
Oviedo</A>
      <LI><A HREF="http://www.calamaro.com" TARGET="visualizacion">Andr&eacute;s
Calamaro</a>
    </UL>
  </BODY>
</HTML>
```

La visualización del código anterior puede apreciarse en la figura 6-4; en la figura 6-5 se puede ver cómo el *frame* de visualización se modifica al pulsar cada enlace mientras el menú permanece fijo.

6.5 Inline frames (sólo Internet Explorer)

HTML 4.0 ofrece una posibilidad más para crear *frames*, los denominados *inline frames* (una traducción libre sería la de "*frames* incrustados"); desgraciadamente, *Navigator* (al menos hasta la versión 4.5) no soporta el uso de dicha propuesta del estándar.

Figura 6-4

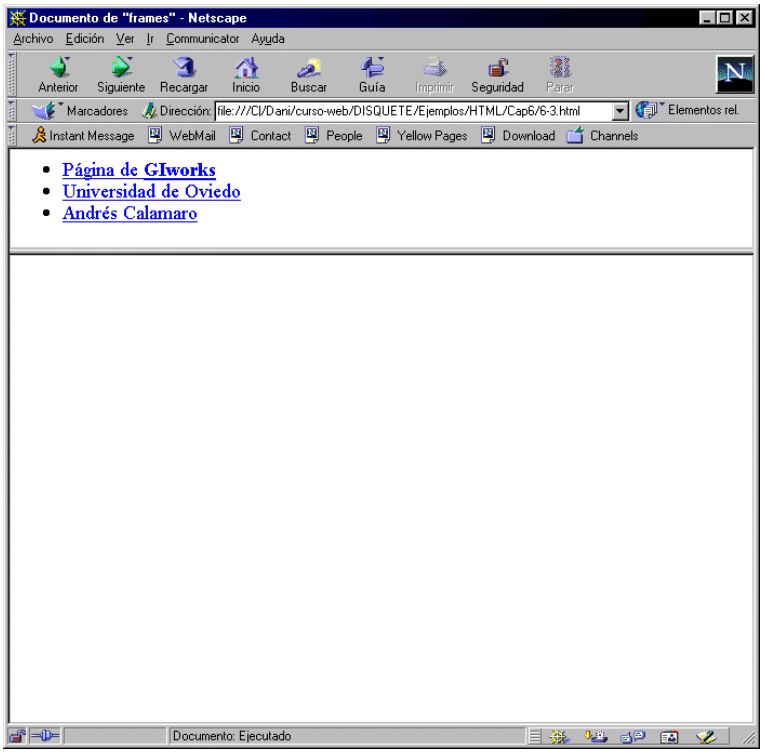
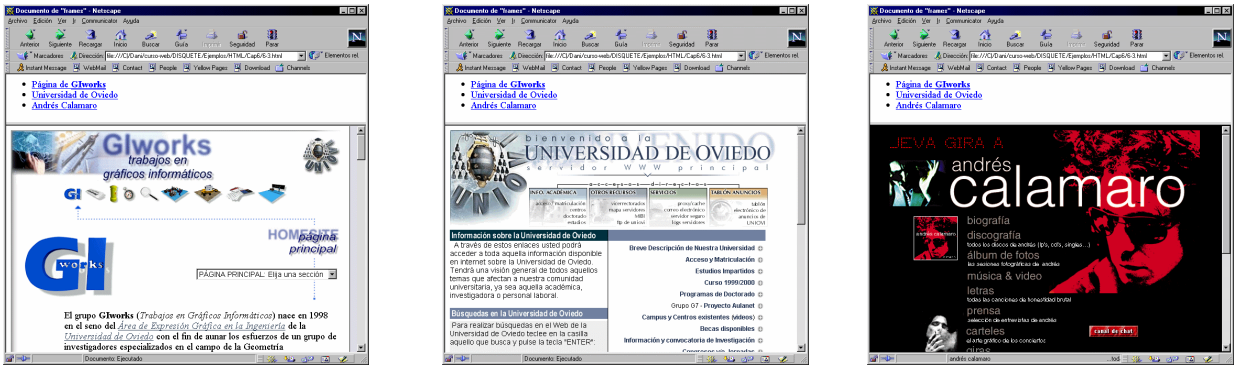


Figura 6-5



La etiqueta **<IFRAME>** admite los mismos atributos que **<FRAME>**, añadiendo los atributos **WIDTH** y **HEIGHT** para indicar la anchura y altura del *frame* incrustado. Al igual que con los *frames*, con los *inline frames* conviene indicar información alternativa para aquellos navegadores que no los soporten, para ello puede emplearse la etiqueta de cierre **</IFRAME>** y delimitar con ambas el texto explicativo.

A continuación se muestra un pequeño ejemplo, junto con la visualización del mismo en los dos navegadores más extendidos:

Ejemplo 6-5 (*Inline frames*)

```
<HTML>
  <HEAD>
    <TITLE>Ejemplo inline frames</TITLE>
  </HEAD>

  <BODY>
    <TABLE BORDER=0 CELLPADDING=5><TR>
      <TD><IFRAME SRC="http://www.calamaro.com" WIDTH=500 HEIGHT=500>Puedes visitar
esta p&aacute;gina en <A
HREF="http://www.calamaro.com">http://www.calamaro.com</A></IFRAME></TD>
      <TD>
        <P>
```

```
    Honestidad brutal es
    ... [TEXTO ELIMINADO]
    con frases tremendas que remiten al trillado caso de su separación del año
pasado.
        </P>
      </TD>
    </TR></TABLE>
  </BODY>
</HTML>
```

Figura 6-6

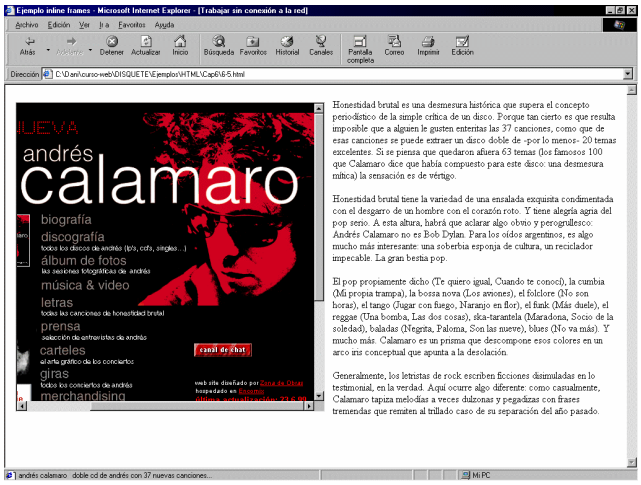
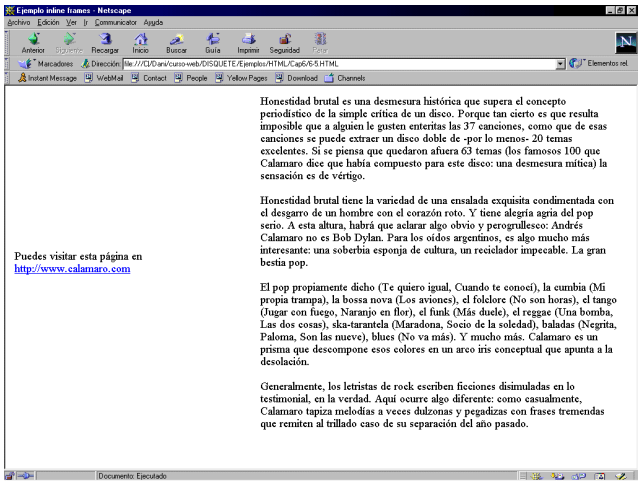


Figura 6-7



Para finalizar una cuestión estética que el lector puede haberse planteado ya: ¿cómo eliminar el borde de los *frames*? Para ello se dispone del atributo **`FRAMEBORDER`** que admite los valores **`1`** (se visualiza el borde) y **`0`** (no se visualiza). Y ahora... ¡Al ejercicio!.

7. Colores, alineación y fuentes de texto

7.1 Indicaciones de color para el cuerpo del documento

El lector ya puede en estos momentos crear documentos *HTML* de gran calidad pero desde el punto de vista del color bastante limitados; para suplir estas carencias cromáticas el lenguaje proporciona una serie de atributos para la etiqueta **<BODY>** que permiten indicar el color del fondo de la página, del texto y de los enlaces:

- **BGCOLOR**: color de fondo de la página.
- **TEXT**: color del texto.
- **LINK**: color de los enlaces no visitados.
- **VLINK**: color de los enlaces visitados.
- **ALINK**: color de los enlaces cuando son activados.

En todos estos casos el color puede indicarse de dos formas, mediante su nombre (en inglés) o mediante los componentes *RGB* del mismo en notación hexadecimal; para facilitar la tarea al futuro desarrollador *web* se proporciona una tabla con los colores *HTML* y se recomienda el uso de algún programa para determinar el código hexadecimal de aquellos colores no predefinidos[8].

Black	= "#000000"	Green	= "#008000"
Silver	= "#C0C0C0"	Lime	= "#00FF00"
Gray	= "#808080"	Olive	= "#808000"
White	= "#FFFFFF"	Yellow	= "#FFFF00"
Maroon	= "#800000"	Navy	= "#000080"
Red	= "#FF0000"	Blue	= "#0000FF"
Purple	= "#800080"	Teal	= "#008080"
Fuchsia	= "#FF00FF"	Aqua	= "#00FFFF"

A continuación se muestra un pequeño ejemplo en el que pueden apreciarse los efectos de los atributos **BGCOLOR**, **TEXT**, **LINK** y **VLINK** (ver figura 7-1).

Ejemplo 7-1 (Uso de BGCOLOR, TEXT, LINK y VLINK)

```
<HTML>
  <HEAD>
    <TITLE>Ejemplo de color</TITLE>
  </HEAD>

  <BODY BGCOLOR=BLACK TEXT=WHITE LINK=YELLOW ALINK=FUCHSIA VLINK=LIME>
    <P>Esto es texto normal y, por tanto, aparece en color blanco.</P>

    <P><A HREF="http://giworks.uniovi.es">Esto es un enlace y, por tanto, aparece
en color amarillo (a no ser que lo hayas visitado, entonces es color lima).</A></P>
  </BODY>
</HTML>
```

7.2 Control del texto

En estos momentos el lector ya tiene un mayor control sobre las características del documento *HTML* que genera pero aún puede controlar más aspectos de la visualización del texto de su documento haciéndolo más profesional. Para ello se utilizará una nueva etiqueta, **<FONT>**, y un atributo de una etiqueta ya conocida **<P>** (ver “2.2.2 Saltos de línea y párrafos”).

[8] El programa *Paint Shop Pro* 5.0 que ya se mencionó en el Módulo 0 permite mezclar un color y proporciona directamente el código *HTML* para el mismo.

Figura 7-1

7.2.1 Fuente y color

La etiqueta **<FONT>** permite mediante sus atributos **SIZE**, **FACE** y **COLOR** controlar el tamaño, la fuente y el color de un fragmento de texto.

- **SIZE**: el tamaño del texto puede indicarse de dos formas, absoluta o relativa. En el primer caso se indica un valor entero comprendido entre 1 y 7 que se corresponde a un tamaño fijo que dependerá del navegador. En el segundo caso se indica un incremento o un decremento en el tamaño del texto (por ejemplo, +2 o –1) que dará lugar a un valor también en el rango 1..7.
- **FACE**: este atributo permite indicar una fuente para el texto, debiendo elegir el navegador la más próxima (en cuanto a apariencia se refiere) a la seleccionada por el autor del documento[9].
- **COLOR**: el color del texto delimitado puede indicarse de forma análoga a como se explicó en la sección “7.1 Indicaciones de color para el cuerpo del documento”.

A continuación se muestra un ejemplo del uso de la etiqueta **<FONT>**:

Ejemplo 7-2 (Uso de <FONT>)

```
<HTML>
  <HEAD>
    <TITLE>Ejemplo del uso de &lt;FONT&gt;</TITLE>
  </HEAD>

  <BODY>
    <P>Ejemplo del uso del atributo <B><CODE>SIZE</CODE></B></P>
    <TABLE BORDER=0>
      <TR ALIGN=CENTER>
        <TD><FONT SIZE=1>1</FONT></TD>
        <TD><FONT SIZE=2>2</FONT></TD>
        <TD><FONT SIZE=3>3</FONT></TD>
        <TD><FONT SIZE=4>4</FONT></TD>
        <TD><FONT SIZE=5>5</FONT></TD>
        <TD><FONT SIZE=6>6</FONT></TD>
        <TD><FONT SIZE=7>7</FONT></TD>
      </TR>
      <TR ALIGN=CENTER>
        <TD>1</TD>
        <TD>2</TD>
        <TD>3</TD>
        <TD>4</TD>
        <TD>5</TD>
        <TD>6</TD>
        <TD>7</TD>
      </TR>
    </TABLE>

    <TABLE BORDER=0>
      <TR ALIGN=CENTER>
```

[9] Elija siempre fuentes comunes: *times*, *arial*, *courier*...

```
        <TD><FONT SIZE=-2>-2</FONT></TD>
        <TD><FONT SIZE=-1>-1</FONT></TD>
        <TD><FONT SIZE=+0>+0</FONT></TD>
        <TD><FONT SIZE=+1>+1</FONT></TD>
        <TD><FONT SIZE=+2>+2</FONT></TD>
        <TD><FONT SIZE=+3>+3</FONT></TD>
        <TD><FONT SIZE=+4>+4</FONT></TD>
      </TR>
      <TR ALIGN=CENTER>
        <TD>1</TD>
        <TD>2</TD>
        <TD>3</TD>
        <TD>4</TD>
        <TD>5</TD>
        <TD>6</TD>
        <TD>7</TD>
      </TR>
    </TABLE>

    <P>Ejemplo del uso del atributo <B><CODE>FACE</CODE></B></P>
    <FONT FACE=ARIAL>Esto es texto en ARIAL</FONT><BR>
    <FONT FACE=TIMES>Esto es texto en TIMES</FONT><BR>
    <FONT FACE=COURIER>Esto es texto en COURIER</FONT><BR>

    <P>Ejemplo del uso del atributo <B><CODE>COLOR</CODE></B></P>
    <B><FONT COLOR=RED>No</FONT> <FONT COLOR=PURPLE>es</FONT> <FONT
COLOR=FUCHSIA>buena</FONT> <FONT COLOR=GREEN>idea</FONT> <FONT
COLOR=OLIVE>usar</FONT> <FONT COLOR=NAVY>excesivos</FONT> <FONT
COLOR=BLUE>colores</FONT> <FONT COLOR=TEAL>en</FONT> <FONT COLOR=GRAY>el</FONT>
<FONT COLOR=MAROON>texto...</FONT></B>
  </BODY>
</HTML>
```

El resultado de visualizar el código anterior puede apreciarse en la figura 7-2.

El buen uso del tamaño, la fuente y el color del texto permiten al autor de documentos *HTML* mejorar el aspecto de los mismos; el abuso puede llevar a textos ilegibles y, por tanto, rechazados por los usuarios. Una serie de consejos habituales indican que no se deberían usar más de dos fuentes distintas (a lo sumo tres y bien conjuntadas), no usar tamaños absolutos sino relativos (y sólo cuando sea estrictamente necesario) y manejar el color con moderación (y buen gusto).

7.2.2 Alineación

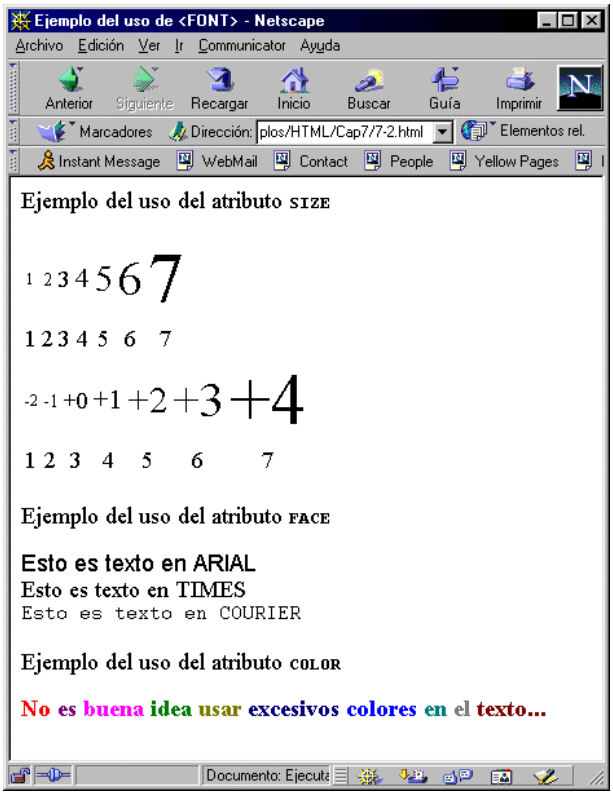
Con las posibilidades de alineación de texto el lector puede crear documentos *HTML* que no tienen nada que envidiar a documentos elaborados con un procesador de texto; mediante el atributo **`ALIGN`** de la etiqueta **`<P>`** el autor puede alinear el texto como desee: centrado (**`CENTER`**), justificado (**`JUSTIFY`**), alineado a derecha (**`RIGHT`**) o a izquierda (**`LEFT`**). Véase el código siguiente:

Ejemplo 7-3 (Uso de ALIGN)

```
<HTML>
  <HEAD>
    <TITLE>Ejemplo de alineado de texto</TITLE>
  </HEAD>

  <BODY>

    <H1>Centrado</H1>
    <P ALIGN=CENTER>Con las posibilidades... [TEXTO ELIMINADO]</P>

    <H1>Justificado</H1>
    <P ALIGN=JUSTIFY>Con las posibilidades... [TEXTO ELIMINADO]</P>

    <H1>Alineado a la izquierda</H1>
    <P ALIGN=LEFT> Con las posibilidades... [TEXTO ELIMINADO]</P>
```

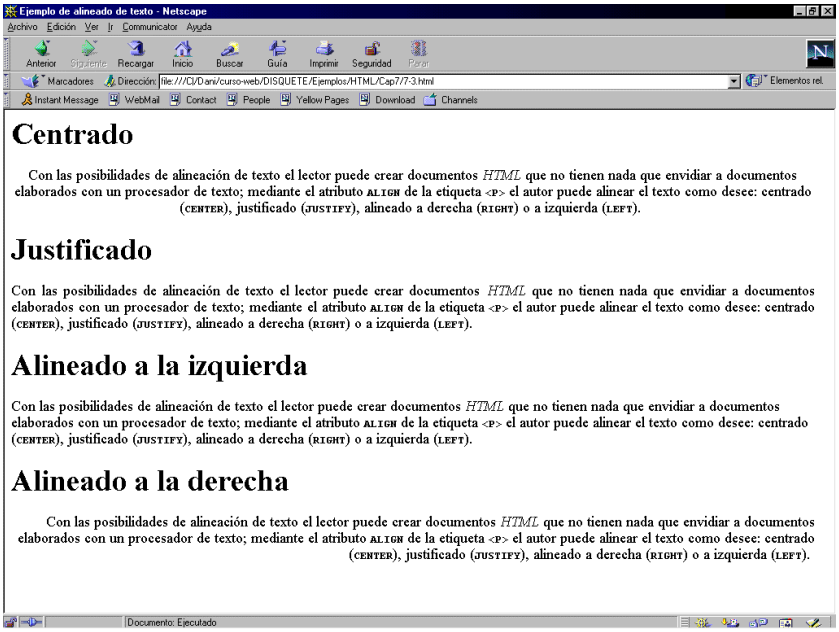
```
    <H1>Alineado a la derecha</H1>
    <P ALIGN=RIGHT> Con las posibilidades... [TEXTO ELIMINADO]</P>

  </BODY>
</HTML>
```

Figura 7-2



El resultado de visualizar el código del ejemplo 7-3 se puede apreciar en la siguiente figura:

Figura 7-3



7.3 Uso del color en tablas

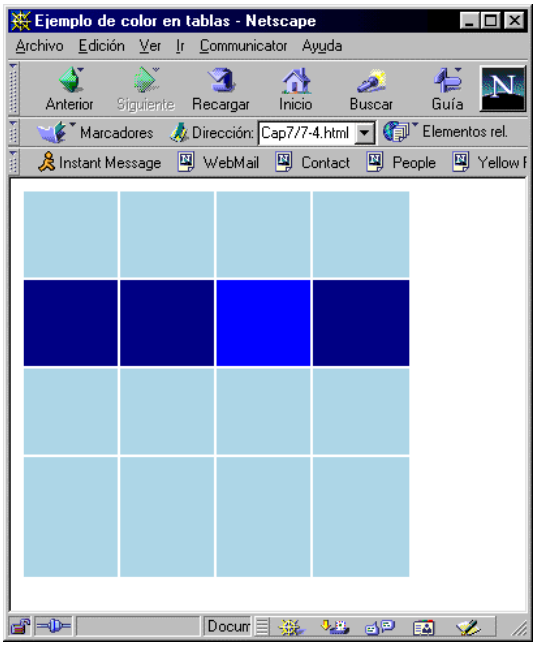
De la misma forma que se puede indicar el color de fondo de un documento *HTML* se puede indicar un color de fondo para una tabla, una fila de una tabla o una sola celda; empleando el atributo **BGCOLOR** de las etiquetas **<TABLE>**, **<TR>** y **<TD>**. A continuación se muestra un ejemplo:

Ejemplo 7-4 (Color en tablas)

```
<HTML>
  <HEAD>
    <TITLE>Ejemplo de color en tablas</TITLE>
  </HEAD>

  <BODY>
    <TABLE BGCOLOR=LIGHTBLUE BORDER=0 CELLPADDING=0 CELLSPACING=2 WIDTH=300
HEIGHT=300>
      <TR><TD> </TD><TD> </TD><TD> </TD><TD> </TD></TR>
      <TR BGCOLOR=DARKBLUE><TD> </TD><TD> </TD><TD
BGCOLOR=BLUE> </TD><TD> </TD></TR>
      <TR><TD> </TD><TD> </TD><TD> </TD><TD> </TD></TR>
      <TR><TD> </TD><TD> </TD><TD> </TD><TD> </TD></TR>
    </TABLE>
  </BODY>
</HTML>
```

El resultado de visualizar dicho código es el siguiente:

Figura 7-4



Antes de pasar al siguiente capítulo el lector debería realizar el ejercicio correspondiente a los últimos contenidos.

8. Uso de imágenes

8.1 Inclusión de imágenes

El *web* nació como una forma de transmitir información fundamentalmente textual que incluía, en ocasiones, algunos gráficos o fotografías... En la actualidad se utiliza más como una valla publicitaria fundamentalmente gráfica que incluye, en ocasiones, algo de texto... Sin embargo, la virtud es el término medio entre dos extremos viciosos, cara y cruz del mismo concepto, en este caso la etiqueta **<IMG>**.

Cualquier documento gana en vistosidad con una utilización adecuada de la imagen, ahora bien, el autor debe ser siempre consciente de cuál es la naturaleza y finalidad del documento que está desarrollando para determinar qué nivel de uso de gráficos debe hacer.

Los atributos fundamentales de la etiqueta **<IMG>** son los siguientes:

- **SRC**: atributo que especifica el *URI* del fichero gráfico que se quiere visualizar.
- **ALIGN**: admite los valores **TOP**, **MIDDLE** y **BOTTOM** para alinear la imagen respecto al texto que la rodea.
- **ALT**: atributo que toma como valor un texto explicativo de la imagen, **debe utilizarse siempre**[10].
- **WIDTH** y **HEIGHT**: permiten especificar las dimensiones de la imagen (de forma absoluta, mediante un valor en pixels, o relativa, como un porcentaje de las dimensiones de la ventana del navegador).
- **BORDER**: permite establecer (o eliminar) un borde alrededor de la imagen.

A continuación se muestra unos ejemplos del uso de la etiqueta **<IMG>**:

Ejemplo 8-1 (Alineamiento de imágenes)

```
<HTML>
  <HEAD>
    <TITLE>Ejemplo del uso de ALIGN en im&aacute;genes</TITLE>
  </HEAD>

  <BODY>
    <P>
      <IMG SRC="garfield.gif" ALIGN=TOP>
      Garfield alineado con el texto por la parte superior.
    </P>

    <P>
      <IMG SRC="garfield.gif" ALIGN=MIDDLE>
      Garfield centrado respecto al texto.
    </P>

    <P>
      <IMG SRC="garfield.gif" ALIGN=BOTTOM>
      Garfield alineado con el texto por la parte inferior.
    </P>
  </BODY>
</HTML>
```

El resultado de visualizar este código puede apreciarse en la figura 8-1.

Ejemplo 8-2 (Modificando el tamaño de las imágenes)

```
<HTML>
  <HEAD>
    <TITLE>Ejemplo del uso de WIDTH en im&aacute;genes</TITLE>
  </HEAD>
  <BODY>
```

[10] A no ser que la imagen sea meramente decorativa pero únicamente en ese caso, si la imagen proporciona información entonces el uso de **ALT** debería considerarse obligatorio.

```
    <IMG SRC="garfield.gif" WIDTH=83 ALIGN=MIDDLE>
    <IMG SRC="garfield.gif" WIDTH=130 ALIGN=MIDDLE>
    <IMG SRC="garfield.gif" WIDTH=200 ALIGN=MIDDLE>
    <IMG SRC="garfield.gif" WIDTH=310 ALIGN=MIDDLE>
  </BODY>
</HTML>
```

La figura 8-2 muestra el resultado de visualizar este último ejemplo.

Figura 8-1

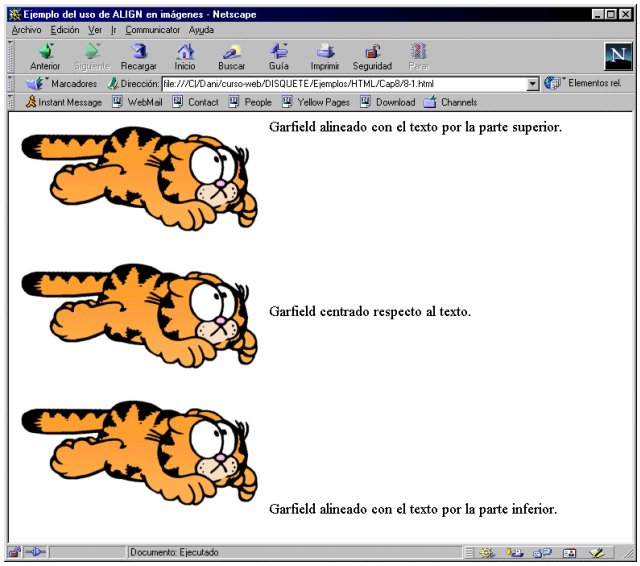
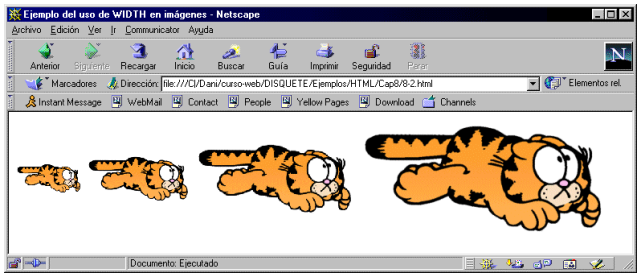


Figura 8-2



8.2 Utilización de imágenes como enlaces e imágenes mapa

Una imagen en un documento *HTML* puede utilizarse como enlace simplemente delimitándola con las etiquetas **<A>**...**</A>**. Al emplear de este modo una imagen ésta aparece enmarcada para indicar que puede ser activada. En la práctica general se emplea el atributo **BORDER** con un valor **0** para eliminar dicho marco; hubo una época en que esto no se consideraba correcto pero en la actualidad ninguna página medianamente cuidada deja que aparezca el borde alrededor de las imágenes activables.

En ocasiones el autor de un documento *HTML* no quiere utilizar toda la imagen como un único enlace sino que distintas regiones de dicha imagen deberían apuntar a recursos diferentes; esto es posible mediante el uso de *imágenes mapa*.

Existen dos posibilidades a la hora de crear una imagen mapa en función de "quién" gestiona dicho mapa. Se habla de *mapas en el servidor* cuando es el servidor *web* el que determina qué zona del mapa ha sido activada y qué enlace *HTML* tiene asociado; por otro lado, se tienen los *mapas en el cliente* que son interpretados por el navegador. En la actualidad este tipo de mapas son los más utilizados por lo cual serán los únicos que se traten en este curso.

El atributo **USEMAP** permite indicar el *URI* donde se encuentra el mapa de la forma siguiente:

```
<IMG SRC="nombre_imagen" USEMAP="[documento_html]#nombre_mapa">
```

El mapa propiamente dicho se define en *HTML* mediante las etiquetas **<MAP>** y **<AREA>**. La primera solo tiene un atributo, **NAME**, que sirve para indicar el nombre del mapa y poder referirse a él mediante el atributo **USEMAP** de **<IMG>**. La etiqueta **<AREA>** se emplea para describir las zonas activables del mapa y debe ir enmarcada por **<MAP>**...**</MAP>**, tiene los siguientes atributos:

- **SHAPE**: la forma de la zona activable, puede tomar los valores **RECT** (rectángulo), **CIRCLE** (círculo) y **POLY** (polígono).
- **COORDS**: lista de coordenadas separadas por comas que describen la zona activable. Dependen de la forma de la misma, si se trata de un rectángulo se da como **"lado izquierdo, lado**

superior, lado derecho, lado inferior”, si es un círculo **“centro-x, centro-y, radio”** y si es un polígono **“x1, y1, x2, y2, ..., xN, yN”**.

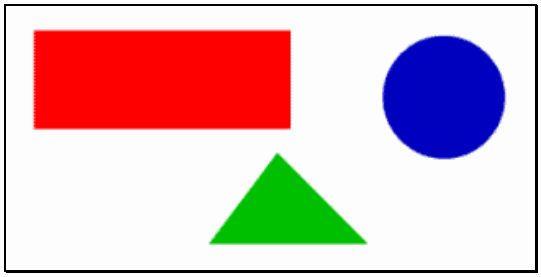
- **HREF**: el *URI* al que apunta la zona activable definida.
- **NOHREF**: atributo sin valor que indica que la zona no tiene asociado ningún enlace.

A continuación se muestra un sencillo ejemplo de imagen mapa correspondiente a la imagen de la figura 8-3:

Ejemplo 8-3 (Imágenes mapa)

```
<HTML>
  <HEAD>
    <TITLE>Ejemplo del uso de im&aacute;genes mapa</TITLE>
  </HEAD>

  <BODY>
    <MAP NAME="ejemplo_mapa">
      <AREA SHAPE=RECT COORDS="18,16,171,73" HREF="rectangulo.htm">
      <AREA SHAPE=CIRCLE COORDS="263,55,36" HREF="circulo.htm">
      <AREA SHAPE=POLY COORDS="163,91,213,141,127,141" HREF="triangulo.htm">
    </MAP>

    <IMG SRC="mapa.gif" USEMAP="#ejemplo_mapa" BORDER=0>
  </BODY>
</HTML>
```

Figura 8-3



Para realizar imágenes mapa se suelen utilizar programas especializados que permiten “dibujar” sobre la imagen las distintas zonas activables y generan el código *HTML* para dicho mapa; el lector interesado en realizar imágenes mapa complejas debería consultar en la referencia acerca de este tipo de aplicaciones.

8.3 Uso de imágenes como fondos de página y de tabla

Para finalizar el capítulo dedicado a imágenes se explicará la forma de emplear imágenes como fondos de página y de tabla; para ello se introduce un nuevo atributo, **BACKGROUND**, que toma como valor el *URI* de la imagen a emplear y que se emplea de forma análoga a como se utilizaba **BGCOLOR**[11] (ver sección “7.1 Indicaciones de color para el cuerpo del documento”). A continuación se muestra un ejemplo que muestra el uso de dicho atributo (ver figura 8-4).

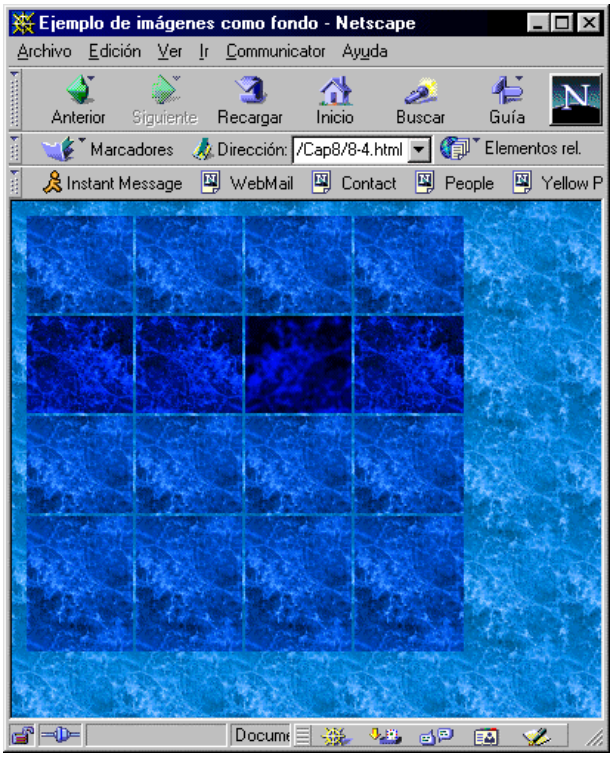
Ejemplo 8-4 (Uso de imágenes como fondos)

```
<HTML>
  <HEAD>
    <TITLE>Ejemplo de im&aacute;genes como fondo</TITLE>
  </HEAD>
  <BODY BACKGROUND="f-pagina.gif">
```

[11] **BGCOLOR** y **BACKGROUND** deberían utilizarse de forma simultanea, asignando como color de fondo uno similar al de la imagen; así, mientras se carga dicha imagen, el usuario puede ir leyendo el texto del documento con comodidad y no se produce un cambio brusco en la apariencia del mismo al visualizar la imagen el navegador.

```
    <TABLE BACKGROUND="f-tabla.gif" BORDER=0 CELLPADDING=0 CELLSPACING=2 WIDTH=300
HEIGHT=300>
      <TR><TD> </TD><TD> </TD><TD> </TD><TD> </TD></TR>
      <TR BACKGROUND="f-fila.gif"><TD> </TD><TD> </TD>
<TD BACKGROUND="f-celda.gif"> </TD><TD> </TD></TR>
      <TR><TD> </TD><TD> </TD><TD> </TD><TD> </TD></TR>
      <TR><TD> </TD><TD> </TD><TD> </TD><TD> </TD></TR>
    </TABLE>
  </BODY>
</HTML>
```

Figura 8-4



Con el próximo capítulo se finaliza el módulo dedicado al lenguaje *HTML*, pero, antes de pasar a él, el lector debería realizar el quinto ejercicio.

9. Conceptos avanzados

Este último capítulo es una miscelánea de conceptos no relacionados entre sí pero que tienen en común el hecho de no ser tratados en la mayor parte de introducciones al *HTML*; se trata de la inclusión de objetos, el uso de metainformación y el empleo de la versión del lenguaje empleado en un documento.

9.1 Objetos incrustados

HTML 4.0 especifica una etiqueta que permite incrustar cualquier tipo de objeto en un documento *HTML*, la etiqueta **<OBJECT>**, desgraciadamente *Navigator* (versión 4.5) no la soporta e *Internet Explorer* (versión 4.72) lo hace de una forma un tanto peculiar.

Para incrustar objetos, saltándose el estándar, existe la extensión **<EMBED>**, soportada por ambos navegadores. Mediante la misma podemos incluir en las páginas *web* sonido, vídeo[12], realidad virtual, etc.

A continuación se muestra un pequeño ejemplo con tres objetos incrustados: un archivo de sonido *MIDI*, un vídeo y un mundo en *VRML*; cuya visualización en *Internet Explorer* puede verse en la figura 9-1.

Ejemplo 9-1 (Objetos incrustados)

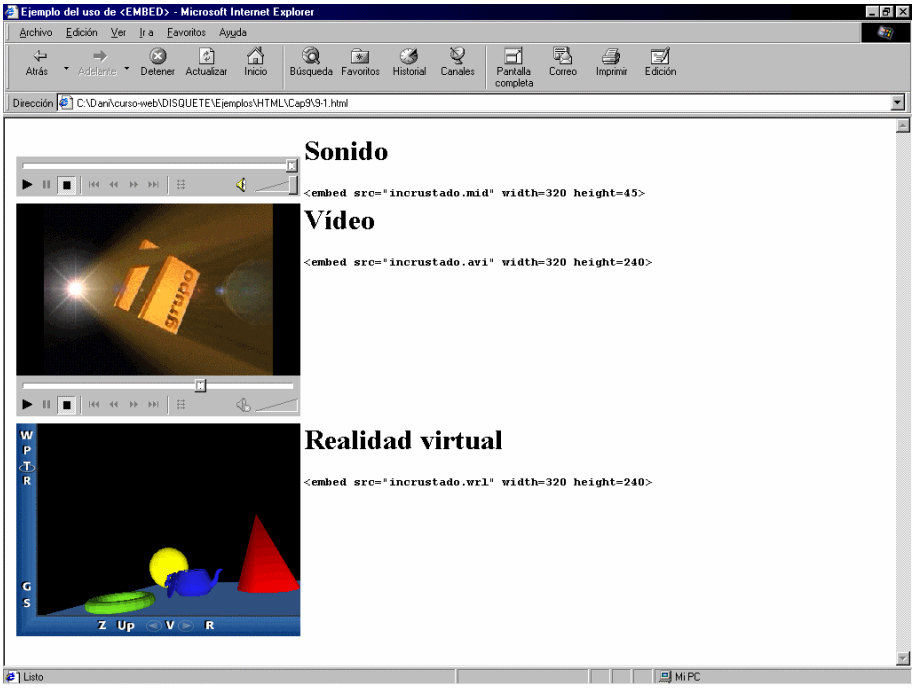
```
<HTML>
  <HEAD>
    <TITLE>Ejemplo del uso de &lt;EMBED&gt;</TITLE>
  </HEAD>
  <BODY>
    <TABLE BORDER=0>
      <TR VALIGN=BOTTOM>
        <TD>
          <embed src="incrustado.mid" width=320 height=45>
        </TD>
        <TD>
          <H1>Sonido</H1>
          <B><CODE>&lt;embed src="incrustado.mid" width=320
height=45&gt;</CODE></B>
        </TD>
      </TR>
      <TR VALIGN=TOP>
        <TD>
          <embed src="incrustado.avi" width=320 height=240>
        </TD>
        <TD>
          <H1>V&iacute;deo</H1>
          <B><CODE>&lt;embed src="incrustado.avi" width=320
height=240&gt;</CODE></B>
        </TD>
      </TR>
      <TR VALIGN=TOP>
        <TD>
          <embed src="incrustado.wrl" width=320 height=240>
        </TD>
        <TD>
          <H1>Realidad virtual</H1>
          <B><CODE>&lt;embed src="incrustado.wrl" width=320
height=240&gt;</CODE></B>
        </TD>
      </TR>
    </TABLE>
  </BODY>
</HTML>
```

[12] Incrustar ficheros de gran tamaño (vídeo o audio de gran calidad) no es una buena idea, es mejor elegir tamaños y/o formatos de poco “peso” y, en casos de extrema necesidad, poner un enlace para su descarga.

Cada navegador (*Communicator* e *Internet Explorer*) dispone de su propio conjunto de atributos para la etiqueta **<EMBED>**, siendo los siguientes los únicos comunes a ambos:

- **SRC**: especifica el *URI* del objeto a incrustar.
- **WIDTH** y **HEIGHT**: permiten especificar las dimensiones del objeto incrustado.
- **ALIGN**: análogo al mismo atributo de la etiqueta **<IMG>**.
- **NAME**: permite dar un nombre al objeto incrustado (útil para acceder a él desde un *script*[13]).
- **HIDDEN**: admite los valores **FALSE** (el objeto es visible) y **TRUE** (el objeto está oculto).

Figura 9-1



9.2 Indicación de la versión de HTML utilizada

Como ya se comentó con anterioridad (punto "1.2.2 Breve historia del *HTML*"), el lenguaje *HTML* no es estático sino que evoluciona con el tiempo y va pasando por sucesivas revisiones. Por ello, es conveniente (según el estándar obligatorio) indicar en los documentos *HTML* que versión del lenguaje se está utilizando; de esta manera la estructura de un documento *HTML* tendría tres partes:

- **Una línea con la información de versión del lenguaje utilizado:**
  - Si se emplea el *HTML 4.0* tal y como se ha visto hasta ahora:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN"
                "http://www.w3.org/TR/REC-html40/loose.dtd">
```

  - Si se emplea el *HTML 4.0* estricto (tal y como se verá tras acabar el Módulo 2):

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0//EN"
                "http://www.w3.org/TR/REC-html40/strict.dtd">
```

  - Si se emplea *HTML 4.0* con *frames*:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Frameset//EN"
                "http://www.w3.org/TR/REC-html40/frameset.dtd">
```

- **La cabecera** (**<HEAD>**...**</HEAD>**)
- **El cuerpo** (**<BODY>**...**</BODY>** o **<FRAMESET>**...**</FRAMESET>**)

Con esa línea se le indicaría al navegador qué lenguaje se está usando y éste podría obtener del *URI* indicado las características del mismo, aunque por el momento no parece que *Internet Explorer* ni *Navigator* hagan el más mínimo caso a estas indicaciones.

[13] Aunque en este curso se hará una breve introducción a *JavaScript* y se programará en este lenguaje en la parte dedicada a *DHTML*, el acceso mediante *scripts* a objetos incrustados va más allá de los objetivos de este curso. Se recomienda al lector interesado en el tema que consulte las referencias.

9.3 Uso de metadatos

Por metadatos (o metainformación) se entiende información sobre la información; es decir, se trata de datos que informan sobre el documento y/o sobre su contenido. Para ello se emplea la etiqueta **<META>** siempre en la cabecera del documento. La metainformación más habitual incluye idioma, descripción del documento, palabras clave, autor, organización, etc.

A continuación se muestra un ejemplo:

Ejemplo 9-2 (Uso de metainformación)

```
<meta name="lang" content="es"> <!--Idioma del documento (Español) -->
<meta name="description" content="P&aacute;gina principal de GIworks">
<meta name="keywords" content="giworks, presentacion, recursos, objetivos, home,
homesite, sitio web">
<meta name="author" content="Daniel Gayo Avello">
<meta name="organization" content="GIworks - Trabajos en Graficos Informaticos">
<meta name="locality" content="Gijon, Asturias, Espana">
```

De todas las cabeceras anteriores hay una que es realmente importante, se trata de la que lleva la lista de palabras clave (*keywords*); esa lista incluye términos o conceptos importantes que son tratados en el documento *HTML* y que son de gran utilidad para los motores de búsqueda, dichas palabras clave son las que se utilizan para determinar si un documento *HTML* puede servir a un usuario que plantea una consulta.

Con el sexto ejercicio finaliza el Módulo 1 y se puede pasar al segundo, Hojas de estilo.

Índice Módulo 2 (Hojas de estilo)

10. Introducción a las hojas de estilo

10.1 Razón de ser y utilidad de las hojas de estilo

En el módulo anterior el lector aprendió a desarrollar documentos *HTML*; como recordará, gran parte de las etiquetas y atributos tenían como única finalidad establecer colores, fuentes, tamaños de letra, estilos de texto... En definitiva, el documento contenía la información, la estructura de la misma y cuestiones estéticas respecto a la visualización de dicha información.

Esta forma de preparar documentos no presenta grandes problemas cuando se trata de páginas pequeñas pero se convierte en un quebradero de cabeza cuando hay que desarrollar un documento *HTML* complejo. Por ello, se hace muy necesario separar por un lado el contenido y por otro la forma; de esta manera el autor puede ocuparse de estructurar y organizar la información de la forma más conveniente y, posteriormente, ocuparse de las cuestiones de forma (que por estar separadas del contenido son mucho más sencillas de mantener y cambiar).

La principal utilidad que se da a las hojas de estilo es eliminar del contenido del documento todas las cuestiones de forma (fuentes, colores, alineación del texto, etc.), el autor especificará en una o más hojas de estilo todo lo relacionado con la visualización del documento y sólo tendrá que hacer uso de dicha hoja en todos los documentos *HTML* para que tengan el mismo aspecto. Sin embargo, las hojas de estilo ofrecen muchas más posibilidades puesto que pueden emplearse para desarrollar páginas *web* dinámicas.

10.2 Un ejemplo sencillo

La mejor forma de demostrar el movimiento es andando así que a continuación se muestra un ejemplo sencillo junto con su visualización en *Internet Explorer* y *Navigator*:

Ejemplo 10-1 (Un ejemplo sencillo de hojas de estilo)

```
<HTML>
  <HEAD>
    <TITLE>Ejemplo de uso de hojas de estilo</TITLE>
  </HEAD>

  <!-- Esta es una hoja de estilo incrustada en el codigo HTML -->

  <STYLE TYPE=TEXT/CSS>
    BODY {
      margin: 2em;
      background-color: lightyellow;

      color: black;

      font-size: 12pt;
      font-family: arial;

      text-align: justify;
    }

    H1 {
      color: darkblue;

      font-size: 18pt;
      font-style: italic;

      text-align: right;
      text-decoration: underline;
      letter-spacing: 8px;
    }

    P {
      text-indent: 3em;
    }
```

```
    P.FIGURA {
      font-size: 10pt;
      font-weight: bold;

      text-indent: 0em;
      text-align: center;
    }

    IMG {
      text-indent: 0em;
      text-align: center;
      align: center;

      border: solid;
      border-width: thin;
      border-color: black;
    }
  </STYLE>

  <BODY>
    <H1>Introducci&oacute;n a las hojas de estilo</H1>

    <P>
En el m&oacute;dulo anterior el lector aprendi&oacute; a desarrollar documentos
HTML, como recordar&aacute; gran parte de las etiquetas y atributos tenían como
&uacute;nica finalidad establecer colores, fuentes y tamaños de letra, estilos de
texto... En definitiva, el documento conten&iacute;a la informaci&oacute;n, la
estructura de la misma y cuestiones est&eacute;ticas respecto a la
visualizaci&oacute;n de dicha informaci&oacute;n.
    </P>

    <P CLASS=FIGURA>Figura 10-1</P>

    <IMG SRC="../../HTML/Cap8/garfield.gif">

  </BODY>
</HTML>
```

Figura 10-1

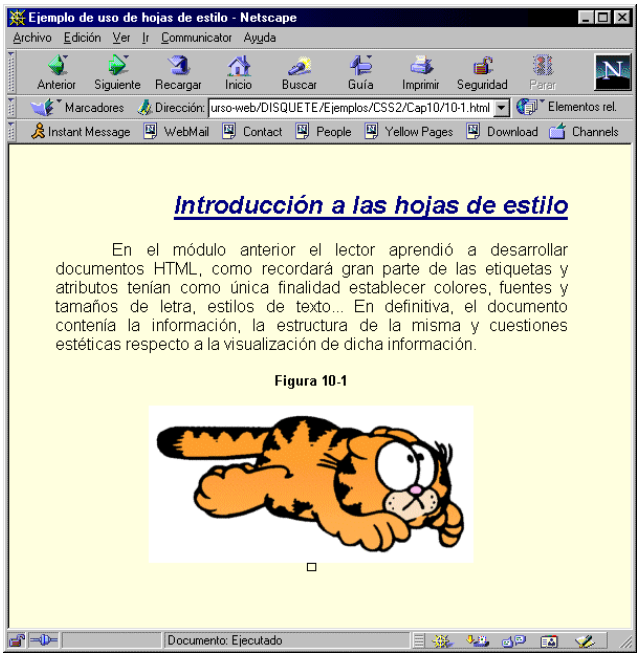
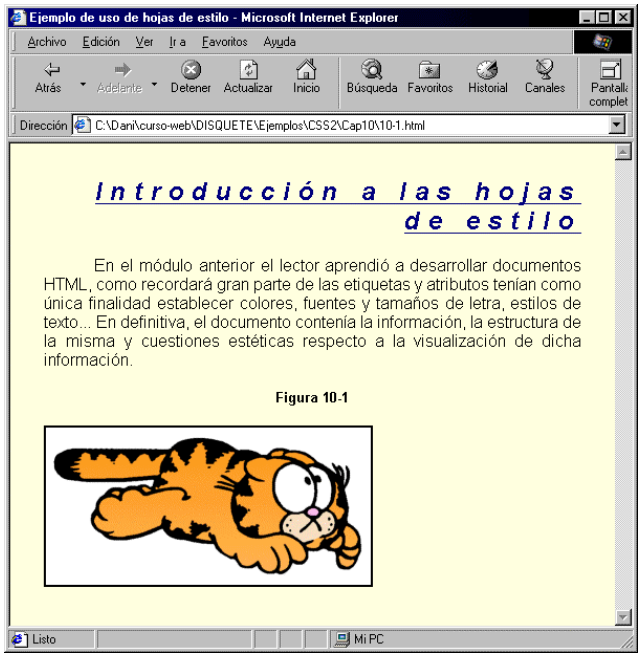


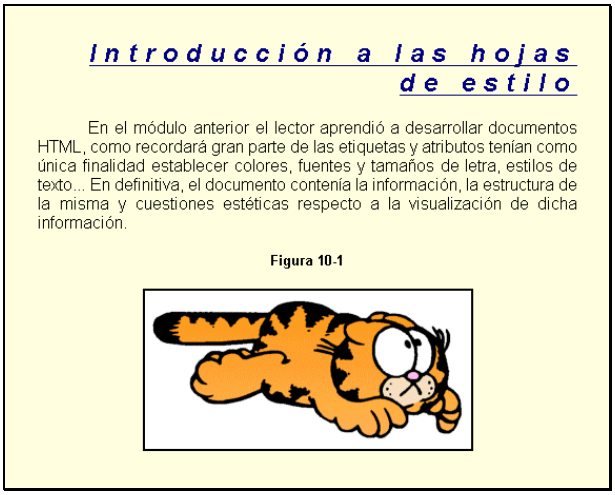
Figura 10-2



Como puede observar el lector, el aspecto de ambas páginas es diferente entre sí, eso no es lo peor, lo cierto es que ninguna de las dos páginas visualiza el código correctamente; la figura 10-3 muestra

el aspecto que tendría en un navegador que implementase correctamente la especificación de hojas de estilo.

Figura 10-3



Al comparar las tres imágenes se llega a las siguientes conclusiones:

- Ambos navegadores interpretan correctamente los aspectos relacionados con el color del documento y del texto, así como con la forma de alineación y las fuentes.
- Sólo *Internet Explorer* soporta la separación entre caracteres y visualiza los márgenes del documento de forma correcta.
- Ni *Internet Explorer* ni *Navigator* interpretan correctamente las propiedades que se especificaron para la imagen: el primero la enmarca correctamente pero no la centra, el segundo la centra pero no la enmarca correctamente (de hecho dibuja un marco pero *tras* la imagen no a su alrededor).

En definitiva, al igual que con *HTML 4.0* había discrepancias entre los navegadores y una implementación parcial de la especificación, con hojas de estilo sigue habiendo discrepancias y una mala implementación de la propuesta del estándar; sin embargo, todo lo que se explique a partir de ahora sobre hojas de estilo estará soportado (correctamente) por ambos navegadores.

El código anterior servirá también para presentar al lector la sintaxis que usan las hojas de estilo. Como se puede ver, la hoja de estilo contiene una serie de *reglas*, por ejemplo

```
background-color: lightyellow;
```

que especifica el color de fondo del documento; las reglas, en el caso general y más sencillo, están formadas siguiendo la estructura:

```
propiedad: valor;
```

Las reglas, por tanto, modifican alguna propiedad de un elemento, para ello se colocan en un bloque encerradas entre llaves, el bloque es precedido por el nombre del elemento; así

```
P {color: black;}
```

indica que el color del texto delimitado por las etiquetas **<P>**...**</P>** será de color negro.

Por último si un elemento puede ser incluido dentro de otro, por ejemplo los párrafos dentro del cuerpo del documento, el primero hereda las propiedades del segundo a no ser que las redefina; por ejemplo si se tiene la siguiente hoja de estilo:

```
BODY {background-color: white; color: black;}
H1 {color: red;}
```

El color del texto será negro, excepto el de los títulos de nivel 1 que lo tienen redefinido a rojo; sin embargo, el color de fondo será blanco puesto que se define a nivel del elemento **<BODY>** y no está redefinido con lo cual se hereda.

10.3 Inclusión de hojas de estilo

Para terminar este capítulo introductorio se explicará la forma en que se suele trabajar con hojas de estilo descritas en ficheros externos. Como se dijo en la sección “5.4 Utilización de enlaces para indicar relaciones o incluir documentos” se puede utilizar la etiqueta **<LINK>** para apuntar a un fichero que será utilizado para la visualización del documento *HTML* que la incluya. De esta manera las hojas de estilo se definen en ficheros aparte (generalmente con la extensión **.css**) y son incluidos mediante un enlace como este:

```
<LINK rel="stylesheet" href="hoja_de_estilo.css" type="text/css">
```

Así se pueden definir en un único fichero todas las características de estilo para una serie de documentos y utilizarlas en todos ellos; con cambiar ese fichero se cambiaría el aspecto de *todos* los documentos de forma instantánea (si el lector piensa en el enorme esfuerzo que supondría modificar de forma manual *todo* el código *HTML* de *todos* los documentos se hará una idea de lo útiles que le pueden resultar las hojas de estilo).

Antes de pasar al siguiente capítulo, sería aconsejable que el lector experimentase con el ejemplo anterior; una idea: trate de lograr que *Internet Explorer* centre la imagen y que *Navigator* la enmarque correctamente; por cierto... ¡Para ello no puede utilizar *HTML*!

11. Manejando color y texto

En *HTML 4.0 transitorio* se podían especificar colores, fuentes, emplear imágenes como fondos, etc. Con un uso estricto del lenguaje esto ya no se puede hacer puesto que se trata de aspectos de forma y de eso se encargan las hojas de estilo; en esta sección se explicará al lector como hacer con hojas de estilo lo que hacía con *HTML*... Y, además, a hacer cosas que no se pueden hacer con *HTML*.

11.1 Colores y fondos

11.1.1 Color de fondo y de primer plano

Mediante hojas de estilo se puede especificar el color de fondo y el color de primer plano (el color del texto); pero no sólo para el cuerpo, **<BODY>**, sino para cualquier etiqueta. El valor del color puede especificarlo exactamente igual que en *HTML*, mediante el nombre o mediante el valor del color. A continuación se muestra un ejemplo sencillo.

Ejemplo 11-1 (Uso del color)

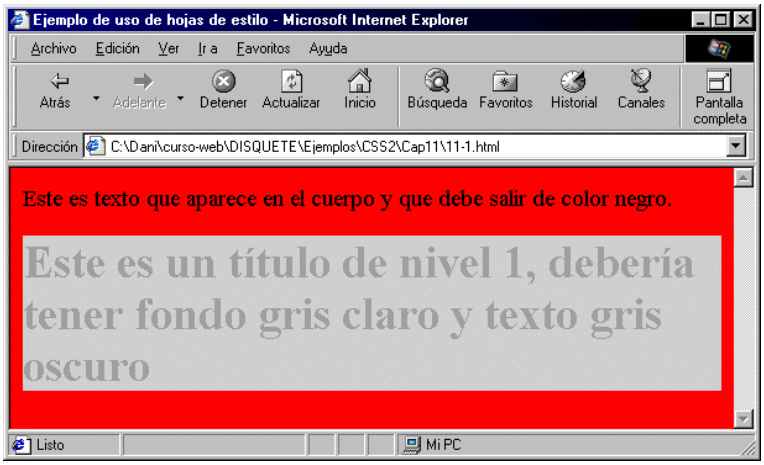
```
<HTML>
  <HEAD>
    <TITLE>Ejemplo de uso de hojas de estilo</TITLE>
  </HEAD>

  <!-- Esta es una hoja de estilo incrustada en el codigo HTML -->

  <STYLE TYPE=TEXT/CSS>
    BODY {
      background-color: red;
      color: black;
    }

    H1 {
      background-color: lightgrey; /* Notese que no es lightgrAy */
      color: darkgray;             /* Notese que no es darkgrEy  */
    }

  </STYLE>

  <BODY>
    Este es texto que aparece en el cuerpo y que debe salir de color negro.

    <H1>Este es un t&iacute;tulo de nivel 1, deber&iacute;a tener fondo gris claro
y texto gris oscuro</H1>

  </BODY>
</HTML>
```

En la figura 11-1 se puede ver el aspecto que debería tener el documento al visualizarlo en un navegador.

11.1.2 Uso de imágenes como fondos

De la misma forma que se puede especificar un color de fondo para cualquier etiqueta, se puede especificar una imagen para el fondo ¡de cualquier elemento de una página *web*! mediante el uso de **background**. En el ejemplo 11-2 hay código que muestra cómo hacerlo y la figura 11-2 muestra el resultado.

Figura 11-1



Ejemplo 11-2 (Uso de imágenes como fondos)

```
<HTML>
  <HEAD>
    <TITLE>Ejemplo de uso de im&aacute;genes como fondos</TITLE>
  </HEAD>

  <!-- Esta es una hoja de estilo incrustada en el codigo HTML -->

  <STYLE TYPE=TEXT/CSS>
    BODY {
      background: url("f-pagina.gif");
    }

    H1 {
      background: url("f-celda.gif");
      color: white;
    }

  </STYLE>

  <BODY>
    <H1>T&iacute;tulo con una imagen de fondo</H1>
  </BODY>
</HTML>
```

Figura 11-2

Está claro que algo como lo que se muestra en la imagen anterior es imposible de conseguir sólo mediante el uso de *HTML*.

11.2 Fuentes de letra

En *HTML* existía una etiqueta que permitía seleccionar la fuente y el tamaño de un texto; mediante hojas de estilo pueden especificarse las siguientes propiedades para una fuente:

- **`font-family`**: puede equipararse, *grosso modo*, con el atributo **`FACE`** de la etiqueta **`<FONT>`**.
- **`font-style`**: permite elegir el estilo que se aplicará a la fuente: **`normal`**, **`italic`** u **`oblique`**. Los dos últimos son estilos en cursiva y las diferencias entre ambos son suficientemente sutiles como para poder usarlos indistintamente[1].
- **`font-variant`**: propiedad que admite dos variantes **`normal`** o **`small-caps`**, la diferencia radica en que con la segunda variante el texto aparece todo en mayúsculas pero las letras que corresponderían a minúsculas se visualizan en un tamaño menor[2].
- **`font-weight`**: con *HTML* existía la **negrita**, con hojas de estilo hay una gama de valores o pesos que van desde **`100`** a **`900`** (saltando de 100 en 100); hay cuatro valores predefinidos: **`normal`** (400), **`bold`** (700), **`bolder`** (+100 o 900 si el peso actual es 900) y **`lighter`** (-100 o 100 si el peso actual es 100).
- **`font-size`**: equivalente al atributo **`SIZE`** de la etiqueta **`<FONT>`**.

A continuación se muestra un ejemplo que emplea todas las propiedades anteriores, en la figura 11-3 se puede apreciar el resultado.

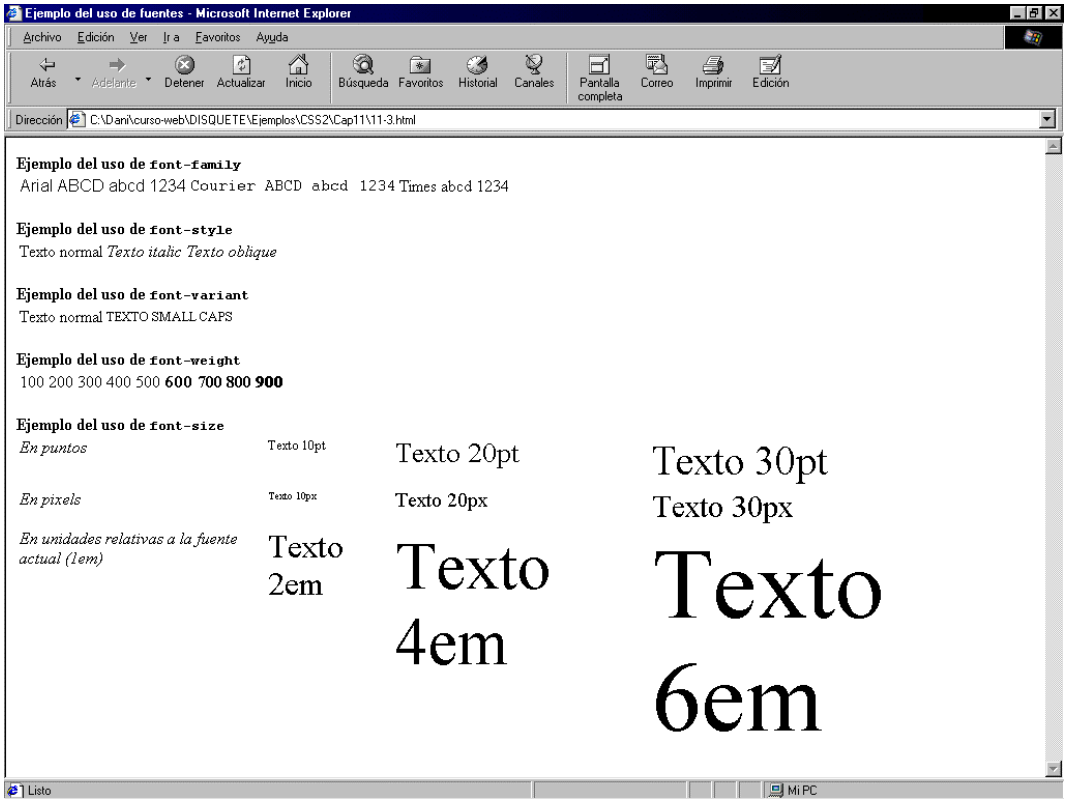
Ejemplo 11-3 (Uso de fuentes)

```
<HTML>
  <HEAD>
    <TITLE>Ejemplo del uso de fuentes</TITLE>
  </HEAD>

  <!-- Esta es una hoja de estilo incrustada en el codigo HTML -->
  <STYLE TYPE=TEXT/CSS>
    DIV.arial {font-family: arial;}
    DIV.courier {font-family: courier;}
    DIV.times {font-family: times;}

    DIV.snormal {font-style: normal;}
    DIV.sitalic {font-style: italic;}
    DIV.soblique {font-style: oblique;}

    DIV.vnormal {font-variant: normal;}
    DIV.vsmall {font-variant: small-caps;}

    DIV.w100 {font-weight: 100;}
    DIV.w200 {font-weight: 200;}
    DIV.w300 {font-weight: 300;}
    DIV.w400 {font-weight: 400;}
    DIV.w500 {font-weight: 500;}
    DIV.w600 {font-weight: 600;}
    DIV.w700 {font-weight: 700;}
    DIV.w800 {font-weight: 800;}
    DIV.w900 {font-weight: 900;}

    DIV.s10pt {font-size: 10pt;}
    DIV.s20pt {font-size: 20pt;}
    DIV.s30pt {font-size: 30pt;}

    DIV.s10px {font-size: 10px;}
    DIV.s20px {font-size: 20px;}
    DIV.s30px {font-size: 30px;}

    DIV.s2em {font-size: 2em;}
    DIV.s4em {font-size: 4em;}
```

[1] De hecho es aconsejable utilizar solo **`italic`** puesto que **`oblique`** no es reconocido por *Navigator* (versión 4.5).
[2] No soportado por *Communicator*, *Internet Explorer* lo soporta parcialmente, visualiza *todo* en letras mayúsculas pequeñas.

```
    DIV.s6em {font-size: 6em;}
  </STYLE>
  <BODY>
    <P>
    <B>Ejemplo del uso de <CODE>font-family</CODE></B>
    <TABLE><TR>
      <TD><DIV CLASS=arial>Arial ABCD abcd 1234</DIV></TD>
      <TD><DIV CLASS=courier>Courier ABCD abcd 1234</DIV></TD>
      <TD><DIV CLASS=times>Times abcd 1234</DIV></TD>
    </TR></TABLE>
    </P>
    <P>
    <B>Ejemplo del uso de <CODE>font-style</CODE></B>
    <TABLE><TR>
      <TD><DIV CLASS=snormal>Texto normal</DIV></TD>
      <TD><DIV CLASS=sitalic>Texto italic</DIV></TD>
      <TD><DIV CLASS=soblique>Texto oblique</DIV></TD>
    </TR></TABLE>
    </P>
    <P>
    <B>Ejemplo del uso de <CODE>font-variant</CODE></B>
    <TABLE><TR>
      <TD><DIV CLASS=vnormal>Texto normal</DIV></TD>
      <TD><DIV CLASS=vsmall>Texto Small Caps</DIV></TD>
    </TR></TABLE>
    </P>
    <P>
    <B>Ejemplo del uso de <CODE>font-weight</CODE></B>
    <TABLE><TR>
      <TD><DIV CLASS=w100>100</DIV></TD>
      <TD><DIV CLASS=w200>200</DIV></TD>
      <TD><DIV CLASS=w300>300</DIV></TD>
      <TD><DIV CLASS=w400>400</DIV></TD>
      <TD><DIV CLASS=w500>500</DIV></TD>
      <TD><DIV CLASS=w600>600</DIV></TD>
      <TD><DIV CLASS=w700>700</DIV></TD>
      <TD><DIV CLASS=w800>800</DIV></TD>
      <TD><DIV CLASS=w900>900</DIV></TD>
    </TR></TABLE>
    </P>
    <P>
    <B>Ejemplo del uso de <CODE>font-size</CODE></B>
    <TABLE>
      <TR VALIGN=TOP>
        <TD><I>En puntos</I></TD>
        <TD><DIV CLASS=s10pt>Texto 10pt</DIV></TD>
        <TD><DIV CLASS=s20pt>Texto 20pt</DIV></TD>
        <TD><DIV CLASS=s30pt>Texto 30pt</DIV></TD>
      </TR>
      <TR VALIGN=TOP>
        <TD><I>En pixels</I></TD>
        <TD><DIV CLASS=s10px>Texto 10px</DIV></TD>
        <TD><DIV CLASS=s20px>Texto 20px</DIV></TD>
        <TD><DIV CLASS=s30px>Texto 30px</DIV></TD>
      </TR>
      <TR VALIGN=TOP>
        <TD><I>En unidades relativas a la fuente actual (1em)</I></TD>
        <TD><DIV CLASS=s2em>Texto 2em</DIV></TD>
        <TD><DIV CLASS=s4em>Texto 4em</DIV></TD>
        <TD><DIV CLASS=s6em>Texto 6em</DIV></TD>
      </TR>
    </TABLE>
  </BODY>
</HTML>
```

Figura 11-3



El ejemplo anterior, introduce varios conceptos nuevos para el lector:

- **El uso de clases:** todas las reglas *CSS* son del estilo **`etiqueta.nombre_clase {...}`**, empleando luego en el código *HTML* la siguiente construcción **`<etiqueta CLASS=nombre_clase>`**; en la sección "13.1 Utilización de clases e identificadores" se trata con mayor detalle el empleo de clases, baste aquí decir que las clases permiten especificar reglas distintas para una misma etiqueta que se comportará de forma distinta en función de la clase o "subtipo" al que pertenezca dicha etiqueta.
- **La etiqueta `<DIV>`:** una etiqueta *HTML* que permite especificar divisiones dentro del documento, su finalidad es sólo esa, permitir distinguir una zona del documento de otra, por ello el atributo **`CLASS`** es tan importante puesto que en función de su valor esa zona delimitada por **`<DIV>`**...**`</DIV>`** va a ser de un tipo o de otro y actuar en consecuencia.
- **Unidades de medida:** *CSS* proporciona una serie de unidades de medida, unas relativas y otras absolutas. Las unidades de medida relativas son las siguientes:
  - *em:* el tamaño de la fuente base (la especificada por el usuario en el navegador).
  - *ex:* la altura en pixels de la fuente base.
  - *px:* pixels (es relativa porque el aspecto dependerá de la resolución del monitor, no es lo mismo un tipo de letra de 10 pixels en un monitor a baja resolución –640x480– que en un monitor en alta resolución –1024x768–).

  Las unidades de medida absolutas son las siguientes:
  - *in:* pulgadas (2,54 cm).
  - *cm:* centímetros.
  - *mm:* milímetros.
  - *pt:* puntos (1/72 de pulgada).
  - *pc:* picas (12 puntos).

11.3 Características del texto

Para poder controlar totalmente el texto ya sólo falta gestionar la alineación del mismo, la decoración (subrayado, tachado, etc.), el control del espacio y algunos aspectos más.

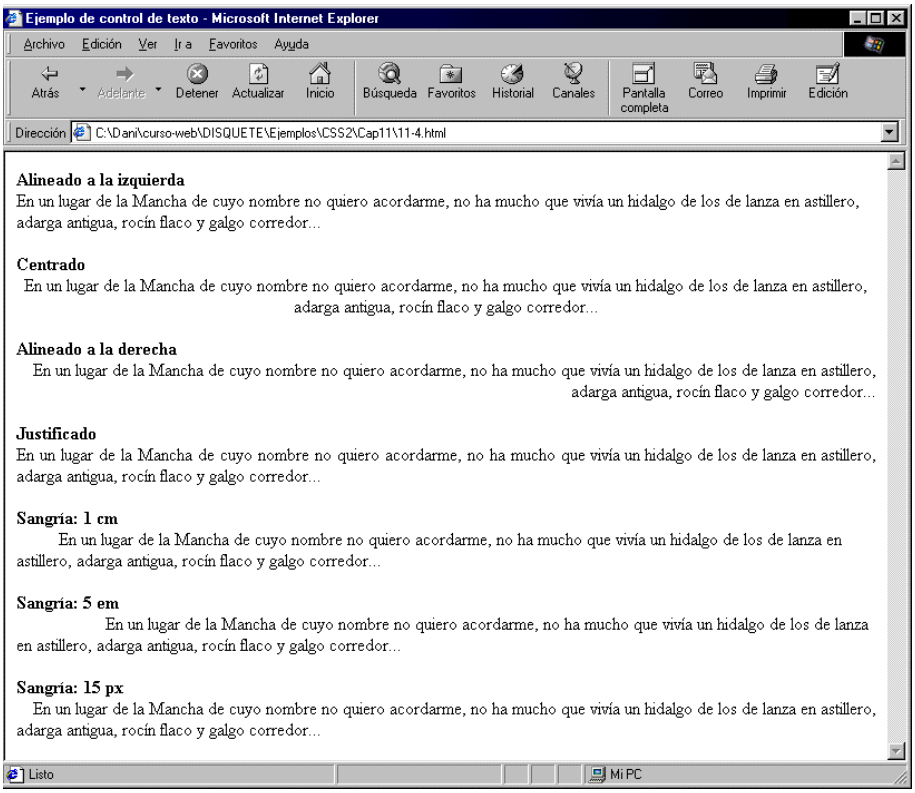
HTML 4.0 ya permitía indicar la alineación del texto (justificado, centrado, etc.), sin embargo, el indentado o sangrado en la primera línea no se podía hacer; para hacer esto *CSS* añade la propiedad **`text-indent`** que toma como valor una medida (en una de las unidades vistas anteriormente) que será la distancia respecto al margen a la que comenzará la primera línea del texto en cada párrafo. A continuación se muestra un ejemplo que maneja esta propiedad junto con la alineación de texto, **`text-align`**. El resultado se puede ver en la figura 11-4.

Ejemplo 11-4 (Alineado y tamaño del texto)

```
<HTML>
  <HEAD>
    <TITLE>Ejemplo de control de texto</TITLE>
  </HEAD>

  <!-- Esta es una hoja de estilo incrustada en el codigo HTML -->

  <STYLE TYPE=TEXT/CSS>
    DIV.left    {text-align: left;}
    DIV.center  {text-align: center;}
    DIV.right   {text-align: right;}
    DIV.justify {text-align: justify;}

    DIV.i1cm    {text-indent: 1cm;}
    DIV.i5em    {text-indent: 5em;}
    DIV.i15px    {text-indent: 15px;}
  </STYLE>

  <BODY>
    <P>
      <B>Alineado a la izquierda</B><BR>
      <DIV CLASS=left>
        En un lugar de la Mancha de cuyo nombre no quiero acordarme, no ha mucho
que viv&iacute;a un hidalgo de los de lanza en astillero, adarga antigua,
roc&iacute;n flaco y galgo corredor...
      </DIV>
    </P>
    <P>
      <B>Centrado</B><BR>
      <DIV CLASS=center>
        En un lugar de la Mancha de cuyo nombre no quiero acordarme, no ha mucho
que viv&iacute;a un hidalgo de los de lanza en astillero, adarga antigua,
roc&iacute;n flaco y galgo corredor...
      </DIV>
    </P>
    <P>
      <B>Alineado a la derecha</B><BR>
      <DIV CLASS=right>
        En un lugar de la Mancha de cuyo nombre no quiero acordarme, no ha mucho
que viv&iacute;a un hidalgo de los de lanza en astillero, adarga antigua,
roc&iacute;n flaco y galgo corredor...
      </DIV>
    </P>
    <P>
      <B>Justificado</B><BR>
      <DIV CLASS=justify>
        En un lugar de la Mancha de cuyo nombre no quiero acordarme, no ha mucho
que viv&iacute;a un hidalgo de los de lanza en astillero, adarga antigua,
roc&iacute;n flaco y galgo corredor...
      </DIV>
    </P>
    <P>
```

```
      <B>Sangr&iacute;a: 1 cm</B>
      <DIV CLASS=i1cm>
        En un lugar de la Mancha de cuyo nombre no quiero acordarme, no ha mucho
que viv&iacute;a un hidalgo de los de lanza en astillero, adarga antigua,
roc&iacute;n flaco y galgo corredor...
      </DIV>
    </P>

    <P>
      <B>Sangr&iacute;a: 5 em</B>
      <DIV CLASS=i5em>
        En un lugar de la Mancha de cuyo nombre no quiero acordarme, no ha mucho
que viv&iacute;a un hidalgo de los de lanza en astillero, adarga antigua,
roc&iacute;n flaco y galgo corredor...
      </DIV>
    </P>
    <P>
      <B>Sangr&iacute;a: 15 px</B>
      <DIV CLASS=i15px>
        En un lugar de la Mancha de cuyo nombre no quiero acordarme, no ha mucho
que viv&iacute;a un hidalgo de los de lanza en astillero, adarga antigua,
roc&iacute;n flaco y galgo corredor...
      </DIV>
    </P>
  </BODY>
</HTML>
```

Figura 11-4



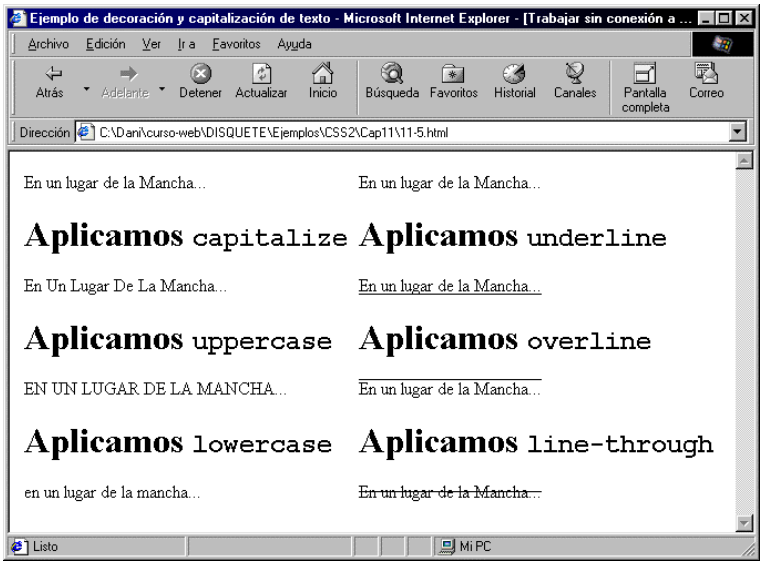
Para terminar con el texto, y con este capítulo, se presentarán las propiedades de capitalización y decoración. La primera, **`text-transform`**, permite manejar el uso de mayúsculas y minúsculas independientemente del texto escrito; así, el valor **`capitalize`** convierte a mayúscula la primera letra de cada palabra, **`uppercase`** escribe todas las letras en mayúsculas, **`lowercase`** en minúsculas y **`none`** no causa ningún efecto. El ejemplo 11-5 muestra el uso de dicha propiedad y la figura 11-5 los resultados.

La segunda propiedad, **`text-decoration`**, admite los siguientes valores: **`underline`** (subrayado), **`overline`** (“superrayado”), **`line-through`** (tachado) y **`blink`** (parpadeo).

Ejemplo 11-5 (Decoración y capitalización de texto)

```
<HTML>
  <HEAD>
    <TITLE>Ejemplo de decoraci&oacute;n y capitalizaci&oacute;n de texto</TITLE>
  </HEAD>


  <!-- Esta es una hoja de estilo incrustada en el codigo HTML -->

  <STYLE TYPE=TEXT/CSS>
    DIV.tcapitalize {text-transform: capitalize;}
    DIV.tuppercase  {text-transform: uppercase;}
    DIV.tlowercase  {text-transform: lowercase;}

    DIV.dunderline  {text-decoration: underline;}
    DIV.doverline  {text-decoration: overline;}
    DIV.dline-through  {text-decoration: line-through;}
  </STYLE>

  <BODY>
    <TABLE WIDTH=100%><TR>
      <TD>
        <DIV>En un lugar de la Mancha...</DIV>

        <H1>Aplicamos <CODE>capitalize</CODE></H1>
        <DIV CLASS=tcapitalize>En un lugar de la Mancha...</DIV>

        <H1>Aplicamos <CODE>uppercase</CODE></H1>
        <DIV CLASS=tuppercase>En un lugar de la Mancha...</DIV>

        <H1>Aplicamos <CODE>lowercase</CODE></H1>
        <DIV CLASS=tlowercase>En un lugar de la Mancha...</DIV>
      </TD>
      <TD>
        <DIV>En un lugar de la Mancha...</DIV>

        <H1>Aplicamos <CODE>underline</CODE></H1>
        <DIV CLASS=dunderline>En un lugar de la Mancha...</DIV>

        <H1>Aplicamos <CODE>overline</CODE></H1>
        <DIV CLASS=doverline>En un lugar de la Mancha...</DIV>

        <H1>Aplicamos <CODE>line-through</CODE></H1>
        <DIV CLASS=dline-through>En un lugar de la Mancha...</DIV>
      </TD>
    </TR></TABLE>
  </BODY>
</HTML>
```

Figura 11-5



Antes de pasar al siguiente capítulo el lector debería realizar el ejercicio correspondiente a la última materia explicada.

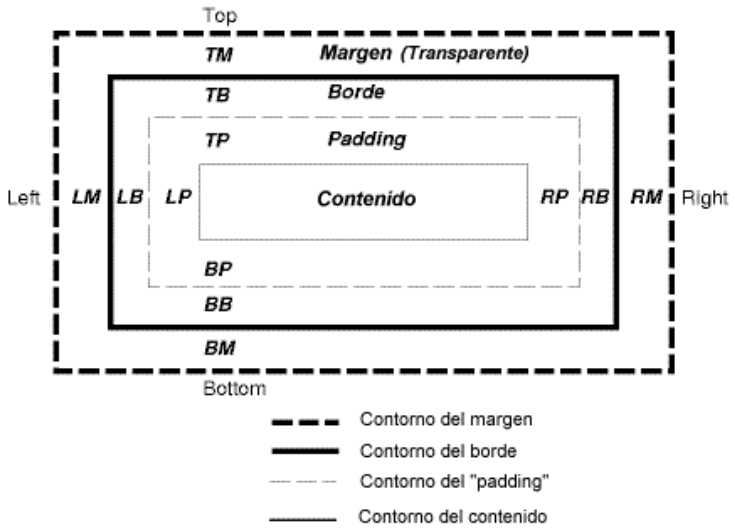
12. Manejando capas

12.1 El modelo de cajas

Cuando se visualizan elementos en un documento *HTML* (encabezados, párrafos, imágenes, etc.) se genera una caja por cada uno, caja que lo contiene y cuyas características pueden ser modificadas mediante hojas de estilo (el lector, de hecho, ya lo ha estado haciendo; si observa las figuras 11-1 y 11-2 podrá ver la caja que enmarca los encabezamientos).

Cada caja se divide en una serie de zonas (ver figura 12-1) que se van colocando alrededor del contenido. Son las siguientes: *padding* (distancia desde el *contenido* al *borde*), *borde* (un marco alrededor del *contenido* y el *padding*) y *margen* (distancia desde el *borde* hasta el resto de cajas del documento que rodean a ésta).

Figura 12-1



Las propiedades **`background`** y **`background-color`** ya vistas afectan al área ocupada por *contenido* y *padding*. La especificación de distancias de *padding*, así como las propiedades del *borde* junto con las de los *márgenes* se verán a continuación.

12.1.1 Los márgenes

Se comienza estudiando los márgenes porque son muy sencillos, al ser transparentes solo pueden indicarse sus dimensiones; para controlar los márgenes se dispone de las siguientes propiedades:

- **`margin-top`**, **`margin-right`**, **`margin-bottom`** y **`margin-left`**: controlan el margen superior, derecho, inferior e izquierdo, respectivamente; toman como valor una distancia en cualquiera de las unidades conocidas por el lector.
- **`margin`**: puede recibir de 1 a 4 valores; si recibe sólo uno se aplica el mismo margen en todos los lados de la caja; si recibe dos valores el primero se aplica a los lados superior e inferior y el segundo a los lados izquierdo y derecho; no es aconsejable especificar tres valores puesto que la forma de asignarlos no es excesivamente lógica y, por tanto, es difícil de recordar; si recibe cuatro valores se asignan en el orden de aplicación siguiente: superior, derecha, inferior e izquierda.

12.1.2 El *padding*

Se comporta de forma totalmente análoga a las propiedades para márgenes; teniendo las siguientes propiedades relativas a distancias: **`padding-top`**, **`padding-right`**, **`padding-bottom`**, **`padding-left`** y **`padding`**.

12.1.3 Los bordes

Los bordes de las cajas son los que más propiedades tienen, se puede controlar la anchura, **–width**, el color, **–color** y el estilo, **–style**, de forma análoga a como se controlan las dimensiones de márgenes y *padding*. Así, propiedades de la forma **border-top?** (donde **?** puede ser **–width**, **–color** o **-style**) controlarán una característica dada para el borde superior; lo mismo es aplicable al resto de bordes (**border-right?**, **border-bottom?** y **border-left?**) y al conjunto de todos los bordes (**border?**).

La anchura puede tomar como valor una dimensión o un valor predefinido de los tres siguientes: **thin** (fino), **medium** (medio) y **thick** (grueso); el color se especifica como se viene haciendo hasta ahora en todas las propiedades relativas al color y el estilo[3] puede tomar uno de los siguientes valores: **none** (invisible), **hidden** (también invisible), **dotted** (punteado), **dashed** (rayado), **solid** (trazo continuo), **double** (doble trazo continuo), **groove** (grabado), **ridge** (resaltado), **inset** (caja hundida) y **outset** (caja elevada).

El ejemplo 12-1 muestra todas las propiedades anteriores en uso y la figura 12-2 el resultado de visualizar ese código.

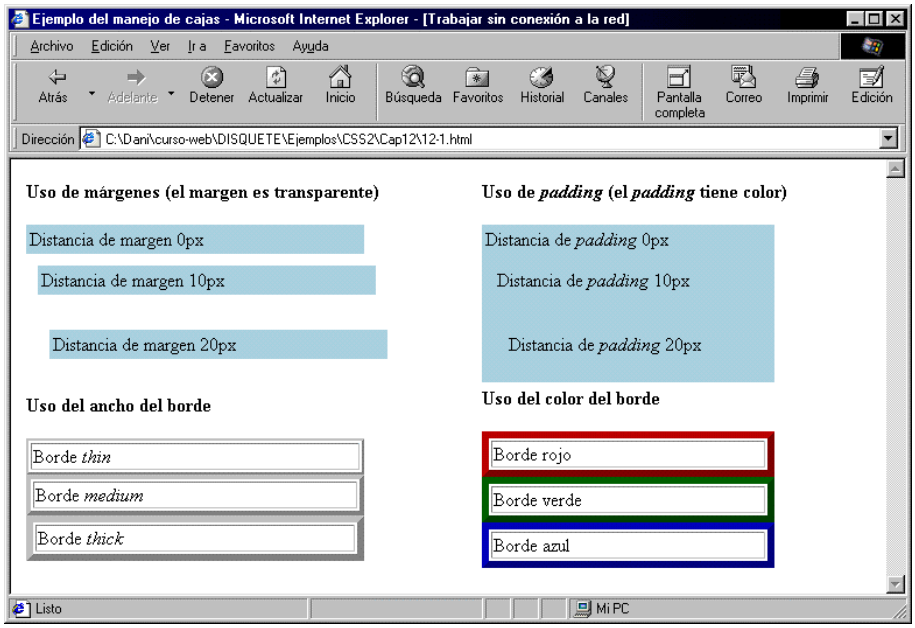
Ejemplo 12-1 (Muestra de propiedades de las cajas)

```
<HTML>
  <HEAD>
    <TITLE>Ejemplo del manejo de cajas</TITLE>
  </HEAD>

  <!-- Esta es una hoja de estilo incrustada en el codigo HTML -->

  <STYLE TYPE=TEXT/CSS>
    TABLE.m0  {background-color: lightblue; margin: 0px;}
    TABLE.m10 {background-color: lightblue; margin: 10px;}
    TABLE.m20 {background-color: lightblue; margin: 20px;}

    TABLE.p0  {background-color: lightblue; padding: 0px;}
    TABLE.p10 {background-color: lightblue; padding: 10px;}
    TABLE.p20 {background-color: lightblue; padding: 20px;}

    TABLE.bthin   {border-width: thin;}
    TABLE.bmedium {border-width: medium;}
    TABLE.bthick  {border-width: thick;}

    TABLE.bred   {border-color: red; border-width:thick;}
    TABLE.bgreen {border-color: green; border-width:thick;}
    TABLE.bblue  {border-color: blue; border-width:thick;}
  </STYLE>

  <BODY>
    <TABLE WIDTH=100%>
    <TR>
    <TD>
    <P><B>Uso de m&aacute;rgenes (el margen es transparente)</B></P>
    <TABLE CLASS=m0 WIDTH=75%><TR><TD>Distancia de margen 0px</TD></TR></TABLE>
    <TABLE CLASS=m10 WIDTH=75%><TR><TD>Distancia de margen 10px</TD></TR></TABLE>
    <TABLE CLASS=m20 WIDTH=75%><TR><TD>Distancia de margen 20px</TD></TR></TABLE>
    </TD>
    <TD>
    <P><B>Uso de <I>padding</I> (el <I>padding</I> tiene color)</B></P>
    <TABLE CLASS=p0 WIDTH=75%><TR><TD>Distancia de <I>padding</I>
0px</TD></TR></TABLE>
    <TABLE CLASS=p10 WIDTH=75%><TR><TD>Distancia de <I>padding</I>
10px</TD></TR></TABLE>
    <TABLE CLASS=p20 WIDTH=75%><TR><TD>Distancia de <I>padding</I>
20px</TD></TR></TABLE>
```

[3] **border-style** es una propiedad que no está soportada por *Internet Explorer* (versión 4.72) y que *Navigator* (versión 4.5) soporta parcialmente (no soporta todos los estilos y los estilos que soporta tiene un aspecto bastante pobre). Esto tampoco quiere decir que el resto de propiedades estén total ni correctamente implementadas en ambos navegadores.

```
    </TD>
    </TR>
    <TR>
    <TD>
    <P><B>Uso del ancho del borde</B></P>
    <TABLE BORDER CLASS=bthin WIDTH=75%><TR><TD>Borde <I>thin</I></TD></TR></TABLE>
    <TABLE BORDER CLASS=bmedium WIDTH=75%><TR><TD>Borde
<I>medium</I></TD></TR></TABLE>
    <TABLE BORDER CLASS=bthick WIDTH=75%><TR><TD>Borde
<I>thick</I></TD></TR></TABLE>
    </TD>
    <TD>
    <P><B>Uso del color del borde</B></P>
    <TABLE BORDER CLASS=bred WIDTH=75%><TR><TD>Borde rojo</TD></TR></TABLE>
    <TABLE BORDER CLASS=bgreen WIDTH=75%><TR><TD>Borde verde</TD></TR></TABLE>
    <TABLE BORDER CLASS=bblue WIDTH=75%><TR><TD>Borde azul</TD></TR></TABLE>
    </TD>
    </TR>
    </TABLE>
  </BODY>
</HTML>
```

Figura 12-2



12.2 Posicionamiento

En estos momentos el lector ya puede controlar gran cantidad de aspectos de las cajas y en esta sección aprenderá a colocarlas dentro del documento. El posicionamiento de las cajas puede ser flotante (**`float`**), absoluto (**`absolute`**) o relativo (**`relative`**), este último no será tratado puesto que es muy similar al absoluto, se recomienda al lector que quiera profundizar en el tema consulte las referencias.

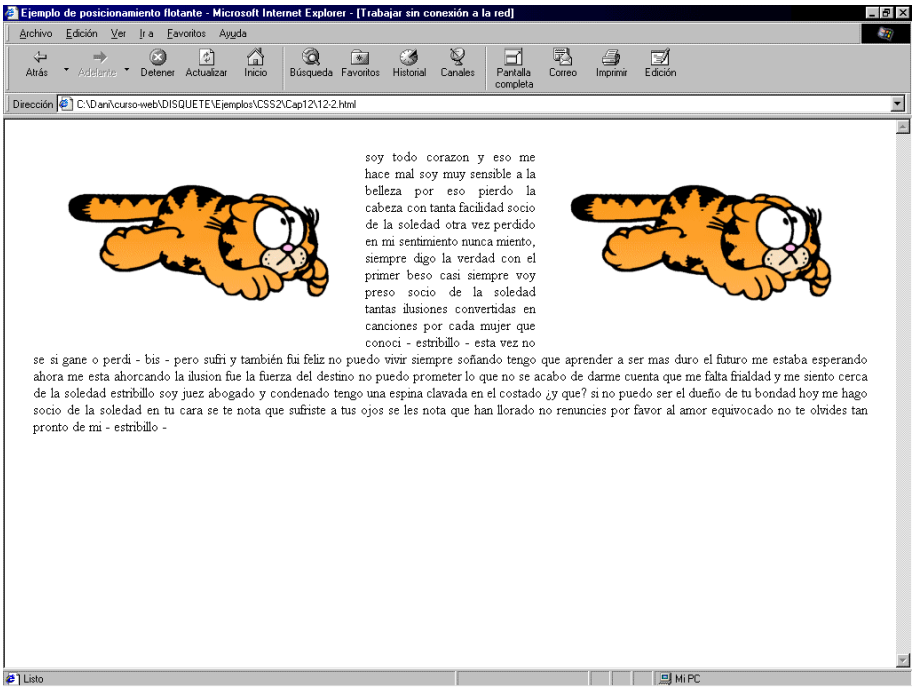
12.2.1 Posicionamiento flotante

El posicionamiento flotante se comporta de forma similar a como *HTML* coloca los elementos en la página, un objeto flotante es colocado por el navegador y el texto (u otros objetos) fluyen a su alrededor; pueden especificarse aspectos tales como los márgenes pero la visualización siempre depende en última instancia de la configuración de la plataforma donde se vaya a ver el documento. La propiedad **`float`** admite tres valores: **`left`** (flota a la izquierda del documento), **`right`** (flota a la derecha del documento), **none** (no flota, está incrustado *dentro* del flujo del documento). El ejemplo 12-2 y la figura 12-3 muestran la forma en que se comportan las cajas flotantes.

Ejemplo 12-2 (Posicionamiento flotante)

```
<HTML>
  <HEAD>
    <TITLE>Ejemplo de posicionamiento flotante</TITLE>
  </HEAD>

  <!-- Esta es una hoja de estilo incrustada en el codigo HTML -->

  <STYLE TYPE=TEXT/CSS>
      IMG.left {float: left;}
      IMG.right {float: right;}

      BODY, P, IMG {margin: 2em;}

      BODY {text-align:justify;}
  </STYLE>

  <BODY>
    <IMG CLASS=left SRC="../../html/cap8/garfield.gif">
    <IMG CLASS=right SRC="../../html/cap8/garfield.gif">
soy todo corazon y eso me hace mal
... [TEXTO ELIMINADO]
no renuncies por favor
al amor equivocado
no te olvides tan pronto de mi
- estribillo -
  </BODY>
</HTML>
```

Figura 12-3



Como se puede ver en la figura 12-3, el navegador coloca las imágenes a derecha e izquierda del documento, manteniendo los márgenes indicados para cuerpo, párrafo e imagen (observe el lector cómo se ha especificado en el código una misma propiedad para varios elementos de forma simultánea: **`BODY, P, IMG {margin: 2em;}`**), así como el justificado del texto. Hay que decir que *Navigator* organiza un verdadero desastre con el documento (ver figura 12-4), aparentemente por culpa de los márgenes puesto que si se comenta[4] la regla que establece los distintos márgenes el resultado es el de la figura 12-5.

[4] Para comentar algo en *CSS* basta con colocarlo entre los símbolos **`/*`** y **`*/`**, análogos a **`<!--`** y **`-->`**.

Figura 12-4

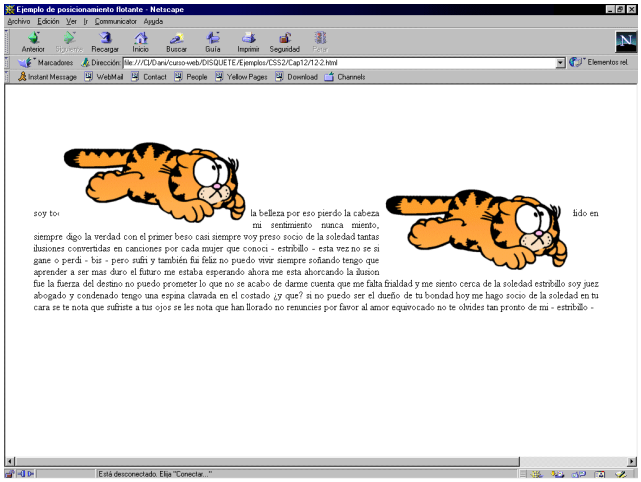
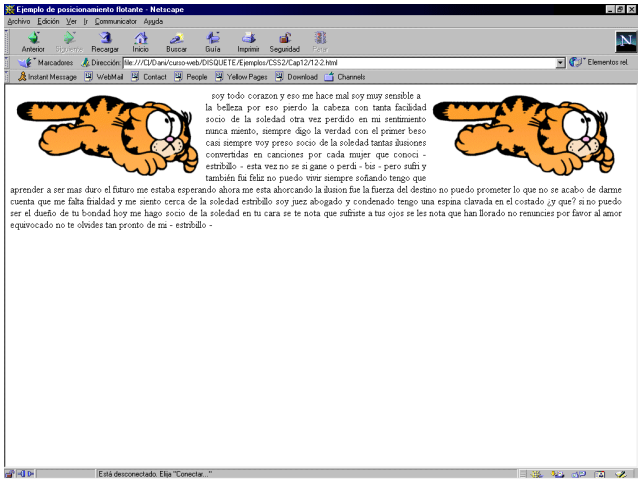


Figura 12-5



A la vista de estos resultados, resulta bastante claro que la implementación de los estándares es bastante más correcta en *Internet Explorer* que en *Communicator*, esperemos que para la próxima versión todos estos problemas hayan sido subsanados para permitir a los usuarios emplear toda la potencia de las especificaciones de *HTML* y *CSS*.

12.2.2 Posicionamiento absoluto

Para emplear un posicionamiento absoluto hay que emplear, en primer lugar, la propiedad **`position`** con el valor **`absolute`**; en segundo lugar hay que colocar la caja mediante las propiedades **`top`** (distancia respecto al borde superior del documento), **`bottom`** (respecto al borde inferior), **`left`** (respecto al borde izquierdo) y **`right`** (respecto al borde derecho).

El uso de posicionamiento absoluto implica que la caja estará siempre colocada en esa posición, independientemente del flujo del resto de elementos. En el ejemplo 12-3 hay una caja con posicionamiento absoluto (enmarcada para verla de forma más clara) y otra con posicionamiento flotante; en las figuras 12-6 y 12-7 puede apreciarse como una “fluye” con el texto mientras la otra permanece colocada en su posición del documento.

Ejemplo 12-3 (Posicionamiento absoluto)

```
<HTML>
  <HEAD>
    <TITLE>Ejemplo de posicionamiento absoluto</TITLE>
  </HEAD>

  <!-- Esta es una hoja de estilo incrustada en el codigo HTML -->
  <STYLE TYPE=TEXT/CSS>
      IMG.absolute {position: absolute; top: 100px; left: 100px; border-width:
thin;}
      IMG.right {float: right;}
      BODY, P {margin: 2em;}
      BODY {text-align:justify;}
  </STYLE>

  <BODY>
    <IMG CLASS=absolute SRC="../../html/cap8/garfield.gif" BORDER>
    <IMG CLASS=right SRC="../../html/cap8/garfield.gif">
soy todo corazon y eso me hace mal
... [TEXTO ELIMINADO]
no renuncies por favor
al amor equivocado
no te olvides tan pronto de mi
- estribillo -
  </BODY>
</HTML>
```

Figura 12-6

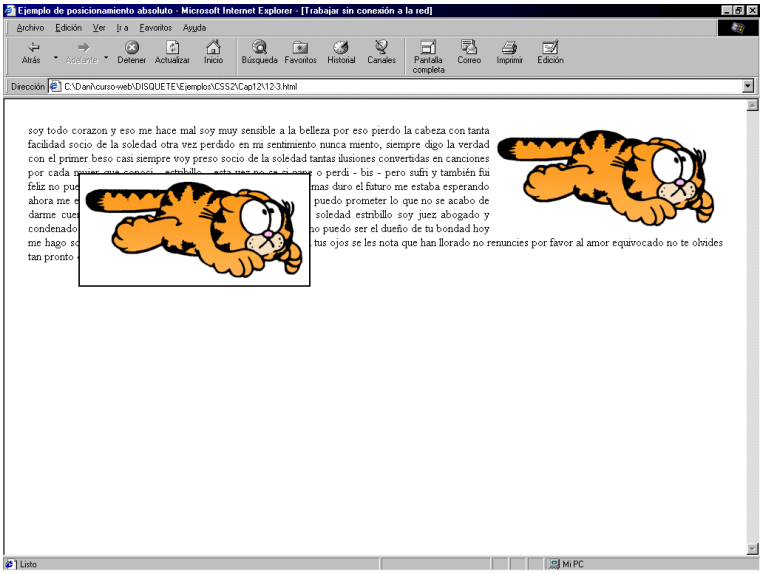
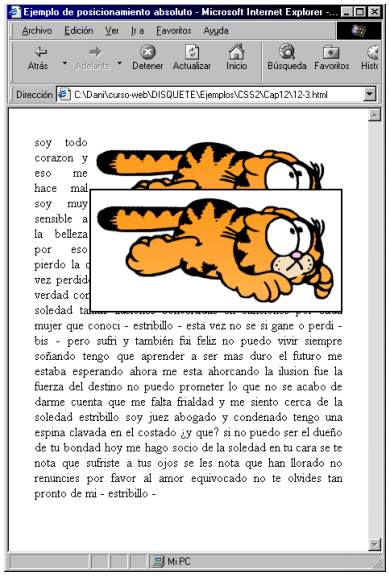


Figura 12-7



En las imágenes anteriores el usuario puede ver cómo la imagen enmarcada de *Garfield* permanece inmóvil respecto al documento (a 100 pixels del borde superior y 100 pixels del izquierdo), mientras que la otra imagen y el texto “fluyen”, obsérvese también como la imagen con posicionamiento absoluto está *por encima* del resto del documento; este es un concepto importante que se verá en la siguiente sección.

12.3 Uso de capas

En la sección anterior el lector vio como los elementos de un documento *HTML* pueden, mediante el uso de *CSS*, montarse unos por encima de otros; esto es lo que se suele denominar manejo de capas, cada elemento puede estar en una capa distinta y las capas flotan unas sobre otras. Para controlar la “profundidad” de cada capa se tiene una nueva propiedad que se denomina **`z-index`**; cuanto mayor sea este valor, más próxima al usuario estará la capa (por encima del resto), y cuanto menor más alejada (por debajo); el valor más pequeño posible es 0.

Un último comentario respecto al uso de capas: aunque cualquier elemento (léase etiqueta) de *HTML* puede emplearse como capa (ver ejemplo 12-3 en el que se emplea una imagen), lo más habitual es utilizar la etiqueta **`<DIV>`**[5] para delimitar todos los elementos que se desee colocar en una capa dada y no hacer uso de las clases (atributo **`CLASS`** de las etiquetas *HTML*) sino de identificadores[6] (atributo **`ID`**), la diferencia radica en que para especificar el estilo de un identificador se hace mediante reglas del tipo

```
#identificador {regla 1; ... regla N;}
```

El ejemplo 12-4 es ilustrativo de esta forma de trabajo[7]. Especifica tres capas con niveles de **`z-index`** distintos, de tal manera que se superponen. La más profunda contiene una imagen de *Garfield* y está situada a 250 pixels de distancia del borde superior del documento y a 250 pixels del izquierdo. La siguiente está situada a 50 pixels del lado superior y a 50 del izquierdo y contiene una imagen flotante y un texto justificado; al estar por encima de la anterior, el texto se escribe sobre la imagen de la capa inferior. Por último, la última capa está situada a 150 y 150 pixels de la esquina superior izquierda y sólo muestra

[5] La etiqueta **`<DIV>`** se comenta en la sección “11.2 Fuentes de letra”.

[6] Un identificador es un nombre único que se dará a un elemento de un documento *HTML* para darle un estilo único, generalmente ese elemento será una capa y el estilo único las características de la misma de posicionamiento y **`z-index`**. Los identificadores se tratan en el punto “13.1.2 Identificadores”.

[7] En esto están de acuerdo tanto *Internet Explorer* como *Navigator* puesto que ambos interpretan este tipo de código de forma correcta; esto no es extraño, puesto que los aspectos de posicionamiento y trabajo con capas son aquellos que más gente utiliza para desarrollar páginas dinámicas.

una imagen que aparece superpuesta al resto de capas. El resultado del ejemplo se puede ver en la figura 12-8 y en la figura 12-9 un esquema con la estructura “3D” que describe el documento.

Ejemplo 12-4 (Uso de capas)

```
<HTML>
  <HEAD>
    <TITLE>Ejemplo del uso de capas</TITLE>
  </HEAD>

  <!-- Esta es una hoja de estilo incrustada en el codigo HTML -->

  <STYLE TYPE=TEXT/CSS>
      P {text-align: justify;}

      IMG.float {float:left;}

      #uno {position: absolute; top: 50px; left: 50px; z-index: 1;}
      #dos {position: absolute; top: 150px; left: 150px; z-index: 2;}
      #tres {position: absolute; top: 250px; left: 250px; z-index: 0;}
  </STYLE>

  <BODY>
    <DIV ID=uno>
      <IMG CLASS=float SRC="../../html/cap8/garfield.gif" BORDER=2>
      <P>
soy todo corazon y eso me hace mal
... [TEXTO ELIMINADO]
no renuncies por favor
al amor equivocado
no te olvides tan pronto de mi
- estribillo -
      </P>
    </DIV>
    <DIV ID=dos><IMG SRC="../../html/cap8/garfield.gif" BORDER=2></DIV>
    <DIV ID=tres><IMG SRC="../../html/cap8/garfield.gif" BORDER=2></DIV>

  </BODY>
</HTML>
```

Figura 12-8

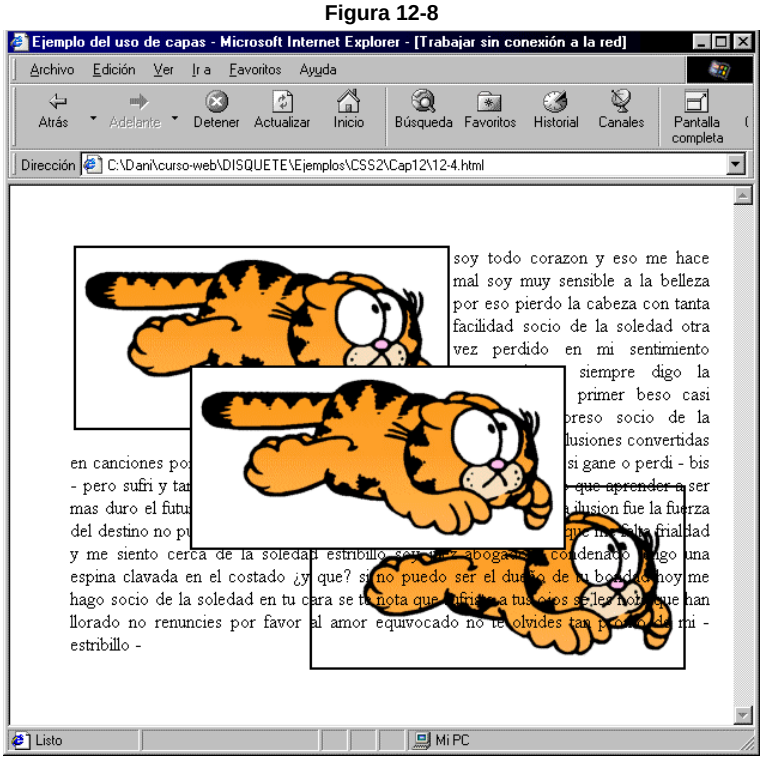
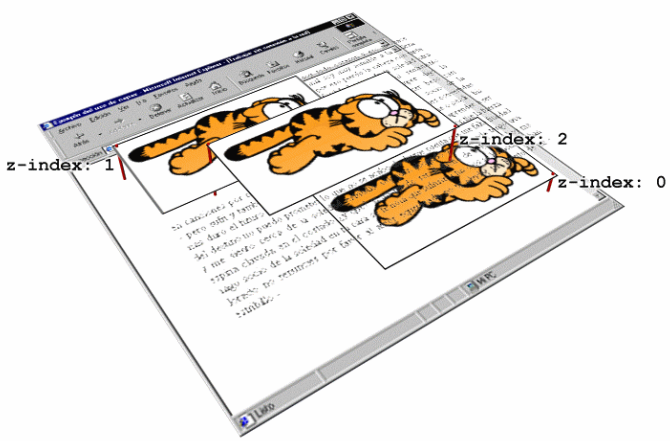


Figura 12-9



12.4 Controlando la visualización

Para terminar el capítulo de capas ya sólo restan dos conceptos que debe manejar el futuro desarrollador: se trata de las propiedades de **`overflow`** y **`clip`**. Ambas propiedades son muy simples y están relacionadas con las dimensiones de las cajas.

12.4.1 La propiedad `overflow`

Hasta ahora, las cajas (o capas) simplemente se han colocado en el documento sin indicar sus dimensiones de tal forma que se expandían lo necesario para admitir su contenido; sin embargo, es posible definir para cada capa unas dimensiones mediante los atributos **`width`** y **`height`**. La cuestión es la siguiente: dada una capa con unas dimensiones dadas, ¿que sucede si el contenido no cabe sino que rebosa? ¿Se visualiza o no? Eso es lo que responde la propiedad **`overflow`**; ésta admite los valores **`visible`** (se visualiza), **`hidden`** (no se visualiza sino que se recorta), **`scroll`** (se establece, si es posible, un mecanismo de *scroll* para poder ver todo el contenido en la caja) y **`auto`** (en caso de desbordamiento generaría barras de desplazamiento). A continuación se muestra un ejemplo para esta propiedad junto con la visualización en *Internet Explorer* y *Navigator* (figuras 12-10 y 12-11).

Ejemplo 12-5 (La propiedad `overflow`)

```
<HTML>
  <HEAD>
    <TITLE>Ejemplo del uso de overflow</TITLE>
  </HEAD>

  <!-- Esta es una hoja de estilo incrustada en el codigo HTML -->
  <STYLE TYPE=TEXT/CSS>
      #uno  {position: absolute; top: 50px; left: 50px; background-color:
lightblue; width: 320px; height: 50px; overflow: auto;}
      #dos  {position: absolute; top: 150px; left: 50px; background-color:
lightblue; width: 320px; height: 50px; overflow: hidden;}
      #tres  {position: absolute; top: 250px; left: 50px; background-color:
lightblue; width: 320px; height: 50px; overflow: scroll;}
      #cuatro {position: absolute; top: 350px; left: 50px; background-color:
lightblue; width: 320px; height: 50px; overflow: visible;}
  </STYLE>

  <BODY>
    <DIV ID=uno><IMG CLASS=float SRC="../../html/cap8/garfield.gif" BORDER=2></DIV>
    <DIV ID=dos><IMG CLASS=float SRC="../../html/cap8/garfield.gif" BORDER=2></DIV>
    <DIV ID=tres><IMG CLASS=float SRC="../../html/cap8/garfield.gif"
BORDER=2></DIV>
    <DIV ID=cuatro><IMG CLASS=float SRC="../../html/cap8/garfield.gif"
BORDER=2></DIV>
  </BODY>
</HTML>
```

Figura 12-10

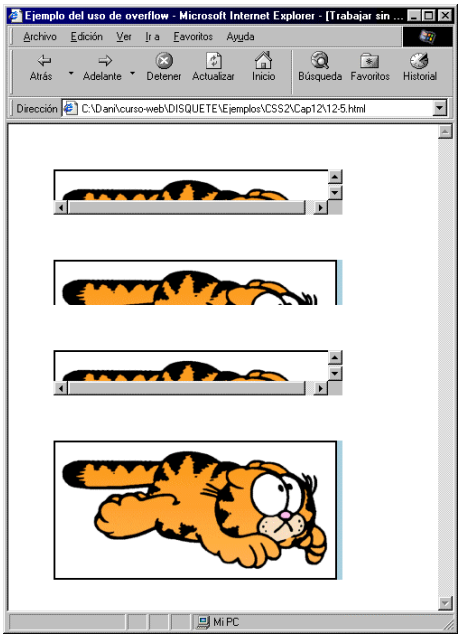
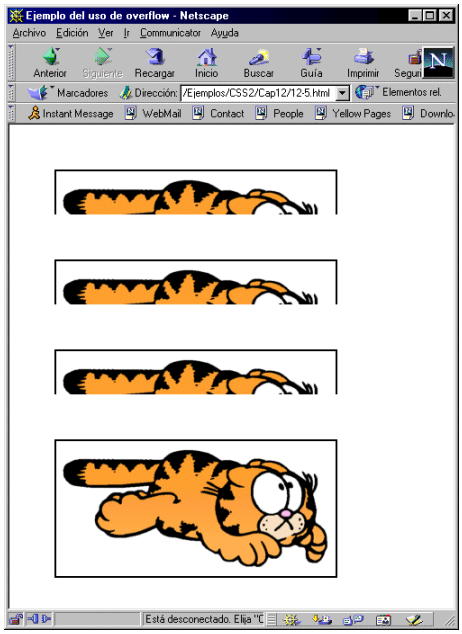


Figura 12-11



Como siempre, hay discrepancias entre un navegador y otro: *Internet Explorer* cumple escrupulosamente las exigencias de la especificación y *Communicator*, de no ser por el color de fondo que no lo visualiza y la implementación de **`overflow:auto`**, también...

- La caja de la parte superior del documento tiene especificadas unas dimensiones menores que la imagen que contiene y un valor para **`overflow`** de **`auto`**, esto quiere decir que el navegador debería actuar como en el caso de **`scroll`** porque hay desbordamiento. Ambos lo cumplen a su manera, *Internet Explorer* añade barras de *scroll* y *Navigator* se comporta como en su implementación de **`scroll`** que es exactamente igual a **`hidden`**.
- La segunda caja tiene un valor para **`overflow`** de **`hidden`**, ambos recortan el contenido.
- La tercera caja tiene un valor de **`scroll`**, *Internet Explorer* añade barras de *scroll* y *Communicator* recorta el contenido.
- La cuarta caja tiene un valor de **`visible`** para **`overflow`** así que ambos estiran la caja hasta ajustarse al contenido.

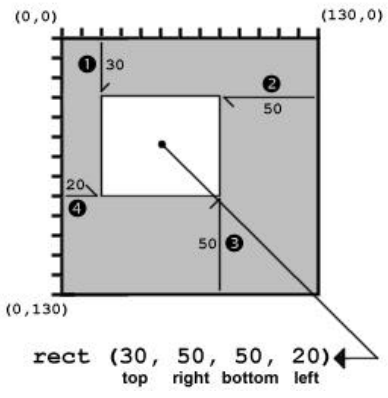
12.4.2 La propiedad `clip`

Esta propiedad permite indicar qué porción de una caja se visualiza, es decir puede recortarse de tal forma que sólo se muestre el contenido que se encuentra dentro de dicha zona. Admite dos valores: **`auto`** (el área visible coincide con todo el área de la caja) o una forma que, en la versión actual, sólo puede ser un rectángulo indicado de la forma:

```
rect(alto, derecha, bajo, izquierda)
```

donde cada uno de esos valores indica un desplazamiento desde el correspondiente lado de la caja.

Figura 12-12

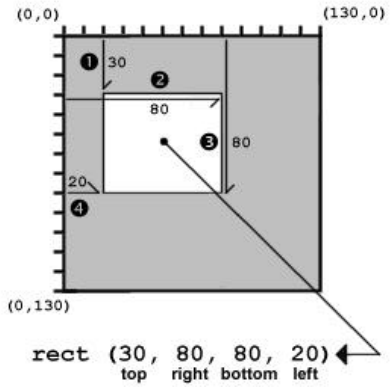


Esto es puramente teórico porque tanto *Internet Explorer* como *Navigator* **¡ignoran el estándar y especifican las coordenadas de una forma totalmente distinta[8]!**, puesto que, en lugar de indicar

[8] Un nuevo cargo que añadir a la lista de delitos de ambas compañías, no sólo implementan de forma parcial los estándares, no sólo lo hacen de manera incorrecta e inconsistente sino que son capaces de ponerse de acuerdo en la forma de utilizar una propiedad descrita en una especificación justo de forma opuesta a como se describe.

desplazamiento desde el borde, simplemente dan las coordenadas cartesianas del rectángulo tomando como origen la esquina superior izquierda de la capa. La figura 12-13 ilustra la forma de especificar el rectángulo de corte de la figura 12-12 en ambos navegadores.

Figura 12-13



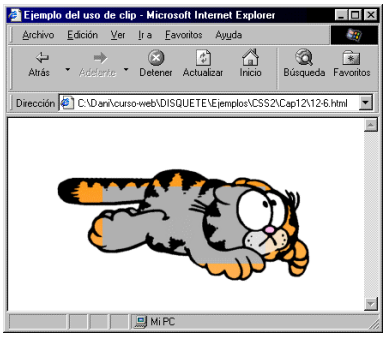
El código que se muestra a continuación ilustra la forma en que se puede emplear la propiedad **clip**, aunque al estilo *Explorer* y *Communicator*; se trata del único ejemplo del documento que no es acorde con el estándar pero no se puede luchar contra los elementos.

En la figura 12-14 se muestra la visualización de ese código; se trata de dos capas con una misma imagen, la inferior en color y la superior en escala de grises, esta última está recortada de tal forma que el usuario ve una única imagen en color con una zona rectangular central en blanco y negro.

Ejemplo 12-6 (La propiedad `clip`)

```
<HTML>
  <HEAD>
    <TITLE>Ejemplo del uso de clip</TITLE>
  </HEAD>

  <!-- Esta es una hoja de estilo incrustada en el codigo HTML -->

  <STYLE TYPE=TEXT/CSS>
      #color {position: absolute; top: 50px; left: 50px;}
      #gris  {position: absolute; top: 50px; left: 50px; clip:
rect(30px,250px,120px,60px);}
  </STYLE>

  <BODY>
    <DIV ID=color><IMG CLASS=float SRC="../../Html/Cap8/garfield.gif"></DIV>
    <DIV ID=gris><IMG CLASS=float SRC="garfield-gris.gif"></DIV>
  </BODY>
</HTML>
```

Figura 12-14



Hay aún otra propiedad relacionada con la visualización, se denomina **visibility** y permite indicar si una capa es visible o no; admite los valores **visible** (la capa se visualiza) y **hidden** (la capa permanece oculta).

Y ahora, antes de pasar al siguiente capítulo, el lector debería realizar el ejercicio correspondiente a la materia explicada.

13. Conceptos avanzados

Al igual que en la primera parte, también este módulo tiene un último capítulo dedicado a cuestiones varias poco tratadas en otros documentos y tutoriales sobre *CSS*; los aspectos que trata este capítulo son los siguientes: el uso de clases e identificadores, los conceptos de herencia y cascada, el uso de pseudoelementos y las etiquetas *HTML* que deberían abandonarse y sustituirse por hojas de estilo.

13.1 Utilización de clases e identificadores

En realidad este tema ya se tocó de manera informal con anterioridad puesto que en numerosos ejemplos se ha hecho uso de clases y en los últimos, dedicados a capas, de identificadores; sin embargo, es conveniente dejar recogido, aunque sea brevemente, lo fundamental del concepto.

13.1.1 Clases

En *CSS*, y por extensión en *HTML 4.0*, una clase es una especialización de un elemento, un caso particular, que puede utilizarse en múltiples ocasiones. Se explicará esto utilizando este documento como ejemplo; este texto está dividido en párrafos, pero estos párrafos tienen características distintas, hay unos como el que está leyendo ahora, otros que se utilizan para representar el código de los ejemplos, etc. Esto quiere decir que, aunque todos son párrafos, están especializados, separados en clases. Para describir las reglas para una clase de un elemento dado se emplea una estructura como la siguiente:

```
elemento.clase {regla1; ... reglaN;}
```

Después, cuando en el código *HTML* se pretende asignar a un elemento el estilo que le corresponde a su clase se hace de la forma siguiente:

```
<elemento CLASS=clase>...</elemento>
```

A continuación se muestra un ejemplo sencillo que muestra las clases necesarias para representar en un documento *HTML* parte del texto de esta página (ver figura 13-1).

Ejemplo 13-1 (Uso de clases)

```
<HTML>
  <HEAD>
    <TITLE>Ejemplo del uso de clases</TITLE>
  </HEAD>

  <!-- Esta es una hoja de estilo incrustada en el codigo HTML -->

  <STYLE TYPE=TEXT/CSS>
      BODY {
        background-color: white;
        color: black;
        margin-top: 2.5cm;
        margin-left: 2cm;
        margin-right: 2cm;
      }

      H1 {
        text-align: right;
        text-decoration: underline;
        font-size: 14pt;
        font-family: arial;
        font-weight: bold;
      }

      H2 {
        text-align: left;
        font-size: 12pt;
        font-family: arial;
```

```
        font-style: italic;
        font-weight: bold;
      }

      H3 {
        text-align: left;
        font-size: 11pt;
        font-family: arial;
        font-weight: bold;
      }

      P.estilo1 {
        text-align: justify;
        text-indent: 1.5cm;
        font-size: 10pt;
        font-family: arial;
      }

      P.codigo {
        text-align: center;
        font-size: 9pt;
        font-family: courier;
        font-weight: bold;
      }
  </STYLE>

  <BODY>

    <H1>Conceptos avanzados</H1>

    <P CLASS=estilo1>
      Al igual que en la primera parte... [TEXTO ELIMINADO]
    </P>

    <H2>Utilizaci&oacute;n de clases e identificadores</H2>

    <P CLASS=estilo1>
      En realidad este tema... [TEXTO ELIMINADO]
    </P>

    <H3>Clases</H3>

    <P CLASS=estilo1>
      En <I>CSS</I>... [TEXTO ELIMINADO]
    </P>

    <P CLASS=codigo>
      elemento.clase {regla1; ... reglaN;}
    </P>

  </BODY>
</HTML>
```

13.1.2 Identificadores

Los identificadores permiten describir estilos que se aplicarán a un único elemento del documento, elemento que tendrá un nombre único que será el del identificador. Los identificadores se emplean, generalmente, para crear y colocar capas en una página *web*. Para describir las reglas para un identificador se emplea una estructura como la siguiente:

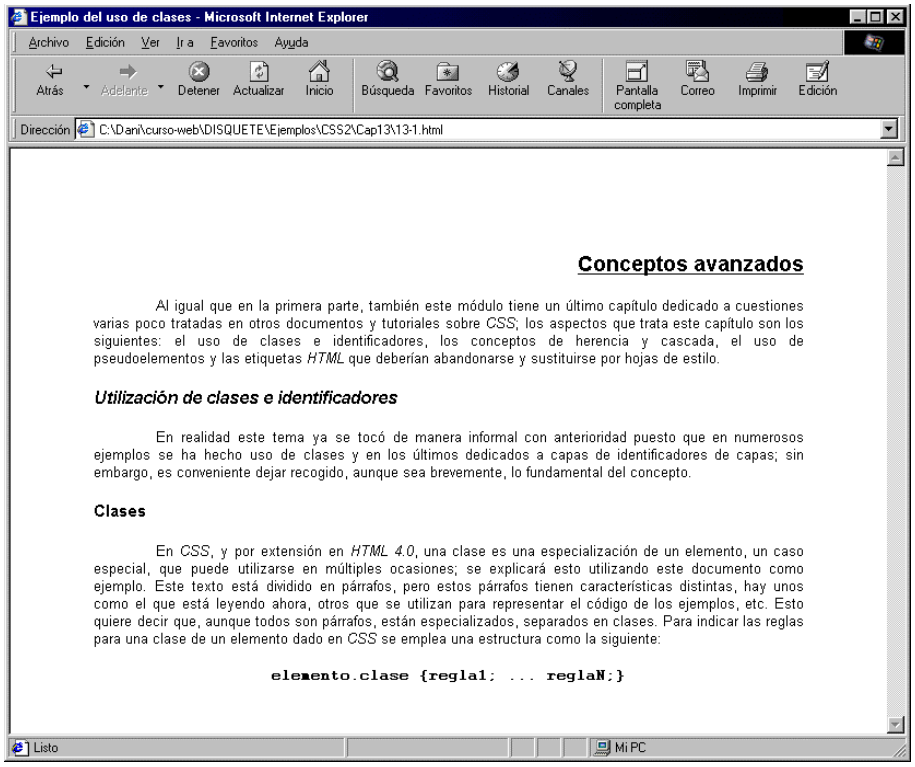
```
#identificador {regla1; ... reglaN;}
```

Después, cuando en el código *HTML* se pretende crear el contenido del elemento que tenga las características especificadas por ese identificador se hace lo siguiente:

```
<elemento ID=identificador>...</elemento>
```

En el capítulo anterior hay abundantes ejemplos del uso de identificadores (ejemplos 12-4, 12-5 y 12-6).

Figura 13-1



13.2 La herencia y el concepto de cascada

A través de los ejemplos de este módulo el lector ya debería tener bastante claro el concepto de herencia pero es demasiado importante como para no tratarlo de forma explícita; además de la herencia, en esta sección también se explica el concepto de cascada que da nombre a la especificación (recuerde que *CSS* significa *Hojas de Estilo en Cascada* y, hasta ahora, en ningún momento se ha dicho por qué).

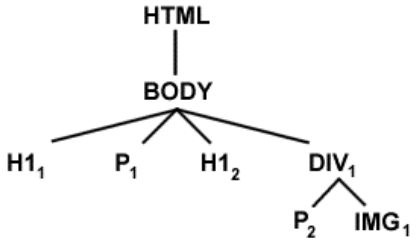
13.2.1 La herencia

Los navegadores manejan un documento *HTML* como si fuera un árbol, donde la raíz sería el elemento **<HTML>**, seguido de **<BODY>**; por cada elemento se iría creando una rama o una hoja en función de si el elemento contiene o no más elementos en su interior. Sea el siguiente documento *HTML*:

```
<HTML>
  <HEAD><TITLE>Documento de ejemplo</TITLE></HEAD>
  <BODY>
    <H1>Un t&iacute;tulo 1</H1>
    <P>Texto...</P>
    <H1>Otro t&iacute;tulo 1</H1>
    <DIV>
      <P>Esto es una capa que contiene texto y una imagen...</P>
      <IMG SRC=”imagen.gif”>
    </DIV>
  </BODY>
</HTML>
```

Dicho documento puede representarse mediante el árbol de la figura 13-2.

Figura 13-2

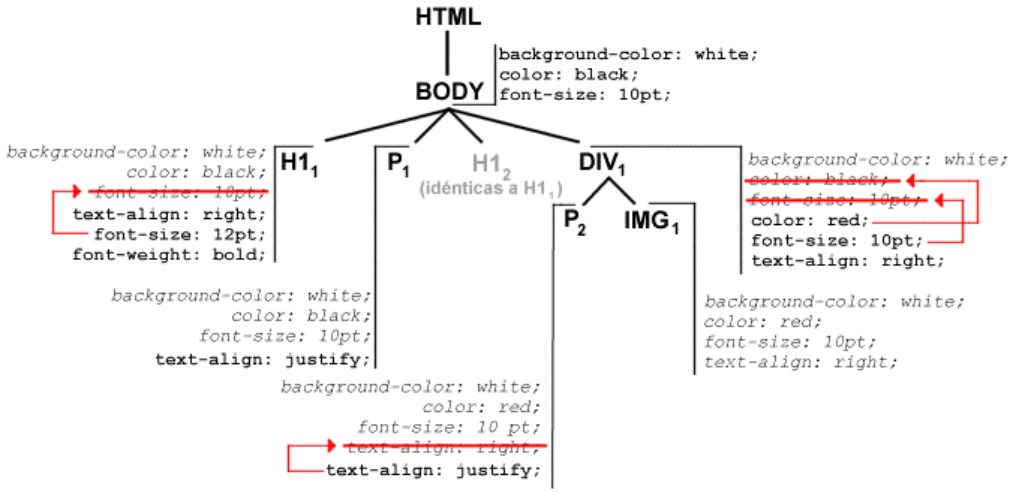


Si dicho documento tiene definida la siguiente hoja de estilo

```
<STYLE TYPE=TEXT/CSS>
  BODY {
    background-color: white;
    color: black;
    font-size: 10pt;
  }

  H1 {
    text-align: right;
    font-size: 12pt;
    font-weight: bold;
  }

  P {text-align: justify;}

  DIV {
    color: red;
    font-size: 10pt;
    text-align: right;
  }
</STYLE>
```

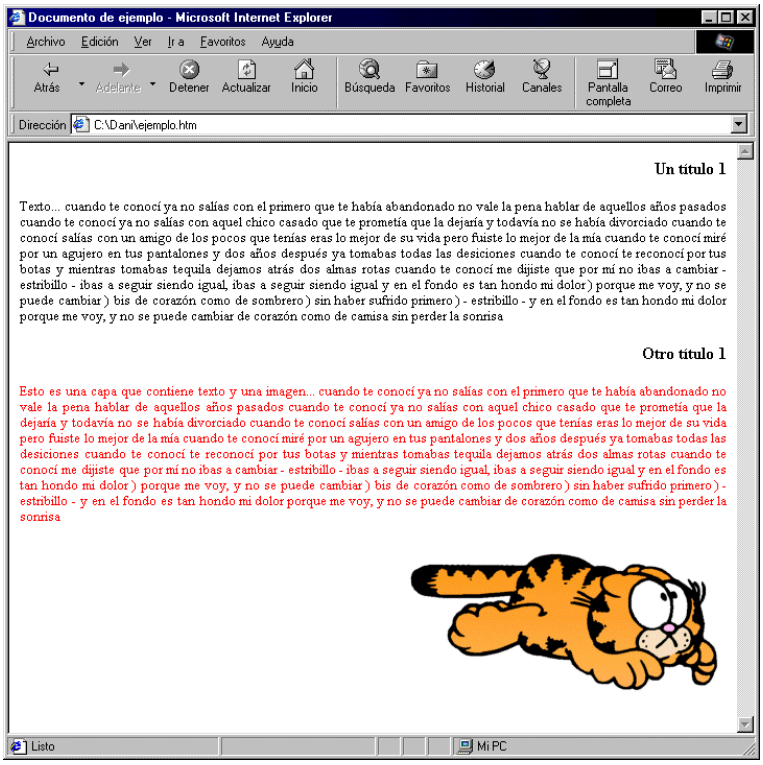
cada elemento de la página tendría las propiedades que le asigna el árbol de la figura 13-3[9] (la visualización del documento se muestra en la figura 13-4).

Figura 13-3



[9] Se muestran en negrita las propiedades indicadas por reglas del elemento, en cursiva las propiedades heredadas y tachadas las propiedades redefinidas, indicando qué regla las redefinió.

Figura 13-4



13.2.2 El concepto de cascada

Todas las hojas de estilo vistas hasta ahora tienen el mismo origen, el autor del documento, sin embargo, este no es el único puesto que tanto el navegador como el usuario pueden tener sus propias hojas de estilo:

- El navegador puede permitirle al usuario indicar una hoja de estilo o introducir mediante un interfaz sus preferencias de tal manera que se genere una hoja de estilo o el navegador se comporte como si ésta existiera.
- El navegador debería tener, o comportarse como si tuviera, una hoja de estilo por defecto para visualizar documentos *HTML* de una forma adecuada.

Los tres tipos de hojas de estilo modifican la representación del documento de una forma determinada por las prioridades que asigna la cascada a cada hoja y por el peso que tiene cada regla; cuando para un mismo elemento son aplicables varias reglas se emplea aquella de más peso. Las reglas de las hojas de autor tienen más peso que las reglas del usuario que, a su vez, tienen más peso que las reglas del navegador. Además del peso de las reglas hay que tener en cuenta la especificidad (el nivel de detalle) de las reglas, así si a un elemento se pueden aplicar varias reglas, la más específica (léase restrictiva) es la que se aplica finalmente.

Sin embargo, el lector no debe preocuparse en exceso por estos aspectos puesto que lo más aconsejable es especificar los estilos que necesite para cada elemento y emplear la herencia de forma adecuada.

13.3 Pseudoelementos y pseudoclases

Una de las características más potentes de la especificación *CSS* es la de pseudoelementos y pseudoclases. Hasta el momento se puede modificar el estilo del documento basándose en la naturaleza de cada elemento, su posición en el árbol del documento y su clase o identificador; sin embargo hay ciertas características de los documentos que no pueden lograrse aún, características que se pueden ver en las figuras 13-5, 13-6 y 13-7.

Figura 13-5

EN UN LUGAR DE LA MANCHA de cuyo nombre no quiero acordarme, no ha mucho que vivía un hidalgo de los de lanza en astillero, adarga antigua, rocín flaco y galgo corredor...

Figura 13-6

EN UN LUGAR DE LA MANCHA DE CUYO nombre no quiero acordarme, no ha mucho que vivía un hidalgo de los de lanza en astillero, adarga antigua, rocín flaco y galgo corredor...

Figura 13-7

En un lugar de la Mancha de cuyo nombre no quiero acordarme, no ha mucho que vivía un hidalgo de los de lanza en astillero, adarga antigua, rocín flaco y galgo corredor...

Las figuras 13-5 y 13-6 corresponden al mismo texto con el mismo estilo, justificado y primera línea en mayúsculas. La figura 13-7 es otro estilo distinto, justificado y primera letra capital ocupando dos líneas. Para poder crear efectos como estos, y otros, es para lo que se han especificado las pseudoclases y pseudoelementos.

Sin embargo, ni *Internet Explorer* ni *Navigator* lo implementan, tan sólo lo hacen para las pseudoclases de enlaces[10] que permiten modificar el estilo de los mismos. *Internet Explorer*, define además las pseudoclases dinámicas; sin embargo, es más que probable que el mayor interés que despertaran estas pseudoclases en *Microsoft* fuera el de poder cambiar de forma dinámica el aspecto de los enlaces: seguramente el lector habrá visto enlaces en algunas páginas *web* que aparecen sin el típico subrayado y que lo muestran al colocarse encima el puntero del ratón; a ese pequeño y anecdótico truco queda reducida una característica tan interesante de *CSS* en el navegador de Gates. El ejemplo 13-3 muestra la forma de crear ese efecto; el 13-2 muestra el uso de las pseudoclases de enlaces y dinámicas.

Ejemplo 13-2 (Pseudoclases de enlaces y dinámicas)

```
<HTML>
 <HEAD>
  <TITLE>Pseudoclases de enlaces y din&aacute;micas</TITLE>
    <STYLE TYPE=text/css>
      A:link    {color: blue;}
      A:visited {color: darkblue;}
      A:hover   {color: red;}
      A:active  {color: red;}
    </STYLE>
 </HEAD>

 <BODY>
   <A HREF="http://giworks.uniovi.es">P&aacute;gina <I>web</I> de
<B>GIworks</B></A>
 </BODY>
</HTML>
```

La visualización del código anterior puede apreciarse en la serie de figuras 13-8, 13-9 y 13-10.

Figura 13-8 (Enlace no visitado)

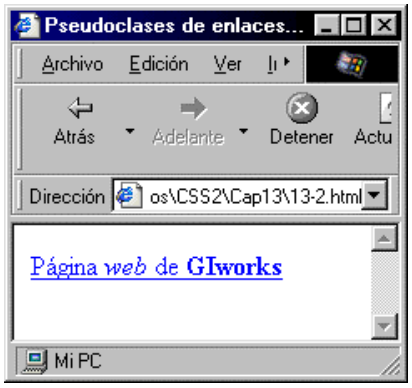


Figura 13-9 (Enlace visitado)

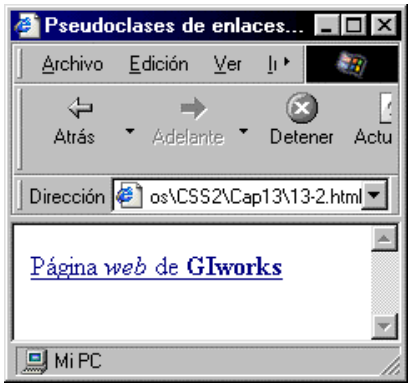
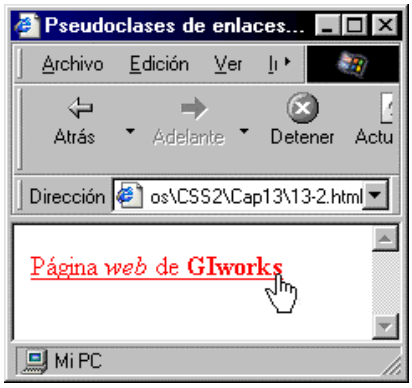


Figura 13-10 (Enlace con cursor encima)



[10] La implementación de esas pseudoclases era obligatoria puesto que, de otro modo, no sería posible especificar los colores de los enlaces mediante *HTML 4.0* estricto (las propiedades **`LINK`** y **`VLINK`** de *HTML 4.0* pertenecen a la versión transitoria del lenguaje).

Antes de mostrar el código que permite modificar el estilo de un enlace hay que decir que la implementación de pseudoclases dinámicas que hace *Internet Explorer* permite que se apliquen estilos distintos de forma simultanea a un mismo elemento (ver ejemplo 13-4); a la luz de la especificación no está muy claro si esto debe o no ser así, por el momento se le concederá el beneficio de la duda al navegador de *Microsoft*.

Ejemplo 13-3 (Modificación del estilo de los enlaces de forma dinámica I)

```
<HTML>
 <HEAD>
  <TITLE>Modificando din&aacute;micamente el estilo de los enlaces en Internet
Explorer</TITLE>
    <STYLE TYPE=text/css>
      A {text-decoration: none; color: blue;}
      A:hover {text-decoration: underline;}
    </STYLE>
 </HEAD>

 <BODY>
   <A HREF="http://giworks.uniovi.es">P&aacute;gina <I>web</I> de
<B>GIworks</B></A>
 </BODY>
</HTML>
```

Figura 13-11

Figura 13-12

Ejemplo 13-4

(Modificación del estilo de los enlaces de forma dinámica II)

```
<HTML>
 <HEAD>
  <TITLE>Modificando
din&aacute;micamente el estilo de los
enlaces en Internet Explorer</TITLE>
    <STYLE TYPE=text/css>
      A {text-decoration: none; color:
blue;}
      A:hover {text-decoration:
underline;}
      A:hover {text-decoration:
overline;}
    </STYLE>
 </HEAD>

 <BODY>
   <A
HREF="http://giworks.uniovi.es">P&aacu
te;gina <I>web</I> de
<B>GIworks</B></A>
 </BODY>
</HTML>
```

Figura 13-13

13.4 Uso de HTML 4.0 estricto

El lector tiene ahora un buen conocimiento de las capacidades de *HTML 4.0* y de *CSS 2*; ha llegado el momento de abandonar el *HTML 4.0* transitorio y pasar a la versión estricta; la forma de hacer esto es abandonando las etiquetas desaprobadas (las referentes a cuestiones de forma del documento) e indicar dichos aspectos mediante hojas de estilo. Para ayudar al futuro desarrollador *web* a realizar esta transición se muestra a continuación un listado de las etiquetas y atributos[11] desaprobados que desaparecen en la versión estricta del lenguaje junto con las propiedades *CSS* relacionadas con la funcionalidad de dicha etiqueta o atributo.

Tabla 13-1

Etiquetas o atributos desaprobados	Propiedades *CSS* relacionadas o equivalentes
<FONT>	**font-family** **font-size** **color**
<U>	**text-decoration underline**
ALIGN Se mantiene en las etiquetas **<TD>**, **<TH>** y **<TR>**	**text-align**
ALINK	**A:active**
BACKGROUND	**Background**
BGCOLOR	**Background-color**
BORDER Se mantiene en la etiqueta **<TABLE>**	**border-width** **border-style**
COLOR	**color**
FACE	**font-family**
HEIGHT Se mantiene en las etiquetas **<IFRAME>**, **<IMG>** y **<OBJECT>**	**height**
LINK	**A:link**
SIZE	**font-size**
TEXT	**BODY {color}**
TYPE	**list-style-type**
VLINK	**A:visited**
WIDTH Se mantiene en las etiquetas **<IMG>** y **<TABLE>**	**Width**

Con este capítulo termina el Módulo 2 y se pasa a la tercera y última parte del curso dedicada a *HTML* dinámico.

[11] Etiquetas y atributos vistos en este documento; para un listado completo consultar las referencias.

Índice Módulo 3 (*JavaScript* y *DHTML*)

14. Introducción al *JavaScript*

14.1 Presentación del lenguaje

JavaScript es un lenguaje empleado fundamentalmente en el desarrollo de páginas *web* dinámicas. *Netscape* lo desarrolló e introdujo con la versión 2.0 del *Navigator*; posteriormente, *Microsoft* introdujo un clónico, denominado *JScript*, en la versión 3.0 del *Internet Explorer*. Las implementaciones iniciales de *JScript* en *Explorer* no eran excesivamente fiables, además de no ser totalmente compatibles con el código *JavaScript* escrito para *Navigator*.

En la actualidad *Netscape* y la *ECMA* (Asociación Europea de Fabricantes de Ordenadores) trabajan en el desarrollo de un lenguaje estándar basado en *JavaScript*, denominado *ECMAScript*, cuya especificación se recoge en el documento *ECMA-262*, especificación que tiene su equivalente *ISO* en la *ISO-16262*. Tanto *Navigator* como *Internet Explorer* en sus versiones 4.0 y superiores soportan esta versión estandarizada del lenguaje.

Después de aburrir al lector con los dos párrafos anteriores puede que sea necesario recordarle la razón por la que quería aprender *JavaScript*: mediante este lenguaje sus páginas pueden reconocer y responder a eventos de usuario, facilitar la navegación en un sitio *web*, verificar los datos de los formularios, manipular capas, crear animaciones, desarrollar multimedia... ¿Ya está despierto? Entonces ya se puede entrar en materia, en el siguiente punto se explica la forma en que puede incrustar *scripts* en sus páginas *web*.

14.2 Incrustando código JavaScript en un documento HTML

El código *JavaScript* siempre está inmerso en el código *HTML* de las páginas *web*, para ello se emplea la etiqueta **<SCRIPT>** que junto con su correspondiente etiqueta de cierre delimita la porción de código que se quiere incluir en un documento. El aspecto del código *JavaScript* sería similar a este:

```
<SCRIPT LANGUAGE="JavaScript">
  <!--
    [AQUÍ CÓDIGO JAVASCRIPT]
  //-->
</SCRIPT>
```

Obsérvese el atributo **LANGUAGE** que tiene la etiqueta **<SCRIPT>**, es opcional pero muy recomendable puesto que *JavaScript* no es el único lenguaje de *script* que puede soportar un navegador (*Internet Explorer*, por ejemplo, soporta también *VBScript*, un lenguaje propio de *Microsoft*). Un uso común de dicho atributo es para especificar qué versión del lenguaje se desea emplear; por ejemplo:

```
<SCRIPT LANGUAGE="JavaScript1.2">
```

Indica que para interpretar ese código el navegador debe soportar la última versión del lenguaje[1], la correspondiente, más o menos, al estándar anteriormente mencionado.

Tras la etiqueta de apertura del *script* aparece el símbolo de apertura de comentario en *HTML*, ¿qué hace ahí? La razón es muy simple, para evitar que los navegadores que no soportan la etiqueta **<SCRIPT>** visualicen el código del programa dentro del documento. Si se fija, el comentario se cierra justo antes de la etiqueta **</SCRIPT>** y va precedido de dos barras inclinadas hacia la derecha; ése es un tipo de comentario en *JavaScript* aunque sin cierre, puesto que ocupa toda la línea. La cuestión es la siguiente, ¿por qué se comenta en *JavaScript* el comentario *HTML*? Respuesta, para que no cause un error al interpretarse el código *JavaScript*[2].

[1] Que es la que se verá en este curso.

[2] Lo cierto es que esta forma de enmarcar el código *JavaScript* más que una práctica puede considerarse una tradición; después de todo, es realmente difícil que alguien siga utilizando versiones de *Navigator* anteriores a la 2.0 o de *Explorer* anteriores a la 3.0. Es cierto que hay navegadores en modo texto, pero esos ni siquiera visualizan tablas... Por todo ello, lo más recomendable para el futuro desarrollador *web* es que se dirija a un público que va a emplear las últimas

Lo cierto es que esta mezcla de lenguajes en un mismo fichero resulta un tanto confusa y sólo es práctica (sería más correcto decir "poco problemática") cuando los *scripts* son sencillos. Cuando el código *JavaScript* es complejo (por ejemplo, una pequeña página *web* con *HTML* dinámico y algunas capas) conviene separarlo del código del documento; para ello se escribe todo el código *JavaScript* en un fichero con extensión **`.js`** y se emplea la estructura siguiente:

```
<SCRIPT LANGUAGE="JavaScript" SRC="fichero.js"></SCRIPT>
```

Esta última forma de incrustar código *JavaScript*, es la más cómoda y la más mantenible; si las hojas de estilo a emplear se colocan también en ficheros externos, entonces el mantenimiento futuro de ese sitio *web* se facilita muchísimo.

14.3 Valores, variables y literales

14.3.1 Valores

JavaScript reconoce los siguientes tipos de valores:

- *Números:* cualquier valor numérico expresado en un formato válido (12, 3.141592, -4.5E-99, etc.)
- *Valores lógicos:* **`true`** (verdadero) y **`false`** (falso).
- *Cadenas:* series de caracteres encerrados entre comillas simples, **`''`**, o dobles, **`""`**.
- **`null`** y **`undefined`**: que son dos valores especiales. El primero indica que una propiedad no tiene un valor asignado; por ejemplo, no es lo mismo que una variable valga 0 o valga **`null`**: en el primer caso contiene "algo" puesto que se puede trabajar con ella ("Multiplícate por cero"), en el segundo no se puede hacer nada, tan sólo saber que no tiene ningún valor asignado. **`undefined`** es, si cabe, aún más extraño; indica que una propiedad de un objeto (ver la sección "14.7 Objetos") no está definida, no existe. Suponga el lector que alguien le hace la "típica" pregunta: "¿A qué huele la risa?" La respuesta más habitual es decir que a nada (**`risa.olor=null`**) pero eso es *falso*: el agua no huele a nada (**`agua.olor=null`**) pero la risa NO huele (**`risa.olor=undefined`**), es decir, la propiedad olor no está definida para la risa.

Antes de proseguir es necesario señalar una cuestión fundamental, en *JavaScript* el uso de mayúsculas y minúsculas es importante; así, **`null`** es una palabra que tiene un significado, mientras que **`NULL`**, **`Null`** o cualquier variación distinta de **`null`** no significan nada en absoluto para el intérprete de *JavaScript*.

14.3.2 Variables

Las variables se utilizan para almacenar valores en un programa; cada variable tiene un nombre que permite referenciarla, nombre que se da en base a unas reglas. En *JavaScript* un nombre o identificador debe comenzar siempre con una letra o un subrayado; los siguientes caracteres pueden ser dígitos o letras; y nunca se puede usar una palabra reservada como identificador (por el momento el lector ya conoce cuatro: **`true`**, **`false`**, **`null`** y **`undefined`**).

Las variables tienen un ámbito, unas zonas del *script* donde son visibles y utilizables; en *JavaScript* una variable puede ser de ámbito:

- *Global*: se puede utilizar en cualquier punto del código.
- *Local*: se declara y utiliza sólo dentro de una función (ver la sección "14.6 Funciones").

Las variables locales sólo pueden declararse mediante el uso de la palabra reservada **`var`**:

```
var variable_local="valor"
```

El uso de **`var`** con variables globales es opcional, así, **`var variable_global="valor"`** y **`variable_global="valor"`** son equivalentes.

---

versiones de los navegadores del *duopolio* y que en sus trabajos se ciña lo más posible a las últimas especificaciones de los estándares.

Como puede ver el lector, al declarar una variable no se indica el tipo de la misma; de hecho éste va a cambiar en función del *literal* que se le asigne. En general, en *expresiones* en las que intervengan cadenas alfanuméricas y números, *JavaScript* convertirá los números a cadenas para su manipulación.

14.3.3 Literales

Los literales permiten representar valores en *JavaScript*; los literales pueden ser *enteros*, *flotantes*, *valores lógicos* y *cadenas de caracteres*.

Los enteros pueden expresarse en tres bases distintas, decimal (base 10), hexadecimal (base 16) y octal (base 8). Un entero decimal es una secuencia de dígitos que no empiece por 0; un entero que empieza por 0 está en base 8 y si empieza por 0x o 0X entonces está en hexadecimal. Los enteros hexadecimales pueden incluir los dígitos del 0 al 9 y las letras desde la "a" a la "f" y desde la "A" a la "F". A continuación se muestran algunos literales enteros: 99 (decimal, valor: $9 \cdot 10^1 + 9 \cdot 10^0 = 99$), 0143 (octal, valor: $1 \cdot 8^2 + 4 \cdot 8^1 + 3 \cdot 8^0 = 99$), 0x63 (hexadecimal, valor: $6 \cdot 16^1 + 3 \cdot 16^0 = 99$).

Los flotantes se dividen en una parte entera (un entero decimal), un punto, una parte fraccionaria (otro entero decimal) y un exponente; la parte del exponente comienza por una "e" o "E" seguida por un entero (con o sin signo). Un flotante debe tener al menos un dígito seguido por un punto o un exponente.

Los valores lógicos son los ya conocidos **`true`** y **`false`**.

Las cadenas de caracteres son series de cero o más caracteres encerrados por comillas simples o dobles.

14.4 Expresiones y operadores

14.4.1 Expresiones

Una expresión es cualquier conjunto válido de literales, variables, operadores y otras expresiones que evalúa a un único valor. El valor puede ser un entero, un flotante, un valor lógico o una cadena de caracteres. Conceptualmente, hay dos tipos de expresiones: las que tienen un valor (**`2+2`**) y las que asignan un valor a una variable[3] (**`cuatro=4`**); las primeras usan *operadores* mientras que las segundas usan *operadores de asignación*.

En base a su naturaleza se distinguen tres tipos de expresiones en *JavaScript*:

- Aritméticas: evalúan a un entero o a un flotante.
- Cadenas: evalúan a una cadena de caracteres.
- Lógicas: evalúan a **`true`** o **`false`**.

14.4.2 Operadores

En la tabla 14-1 se listan los operadores más habituales de *JavaScript*, se muestra su sintaxis, significado y se da un ejemplo.

14.5 Sentencias

Una *sentencia* puede definirse como un fragmento de código que indica al intérprete de *JavaScript* una serie de operaciones a realizar. En *JavaScript* existen varios tipos de sentencias: *condicionales*, *bucles* y *comentarios*.

[3] Que además de asignar el valor a la variable evalúan a dicho valor, así la expresión **`cuatro=4`**, asigna a la variable **`cuatro`** el valor 4 y ella misma evalúa a 4.

14.5.1 Sentencias condicionales

Las sentencias condicionales permiten realizar ciertas acciones basándose en una condición lógica. Se especifica una condición a comprobar y las tareas a ejecutar si la condición es verdadera. *JavaScript* dispone de una sentencia condicional, **`if`**, pero tanto *Navigator* como *Internet Explorer* en sus versiones 4.0 y superiores soportan la sentencia **`switch`**.

Tabla 14-1 (Operadores más habituales en *JavaScript*)

Operadores		Sintaxis	Significado	Ejemplo
Asignación	`=`	`a=b`	Asigna a **a** el valor de **b**	`a=5`
	`+=`	`a+=b` equivale a `a=a+b`	Asigna a **a** el valor de **a+b**	`a+=5`
	`-=`	`a-=b` equivale a `a=a-b`	Asigna a **a** el valor de **a-b**	`a-=5`
	`*=`	`a*=b` equivale a `a=a*b`	Asigna a **a** el valor de **a*b**	`a*=5`
	`/=`	`a/=b` equivale a `a=a/b`	Asigna a **a** el valor de **a/b**	`a/=5`
Aritméticos	`+`	`a+b`	Evalúa al valor de **a+b**	`2+2`
	`-`	`a-b`	Evalúa al valor de **a-b**	`2-2`
	`*`	`a*b`	Evalúa al valor de **a*b**	`2*2`
	`/`	`a/b`	Evalúa al valor de **a/b**	`2/2`
	`++`	`++a`	Suma 1 a **a** y evalúa al valor de **a+1**	`++2`
		`a++`	Suma 1 a **a** y evalúa al valor de **a**	`2++`
	`--`	`--a`	Resta 1 a **a** y evalúa al valor de **a-1**	`--2`
		`a--`	Resta 1 a **a** y evalúa al valor de **a**	`2--`
	`-`	`-a`	Evalúa al valor de **–a**	`-2`
Lógicos	`&&`	`a&&b`	*Y lógico*, evalúa a **`true`** si ambos valores son verdaderos y a **`false`** en caso contrario.	
	`||`	`a||b`	*O lógico*, evalúa a **`false`** si ambos valores son falsos y a **`true`** en caso contrario	
	`!`	`!a`	*No lógico*, evalúa a **`false`** si el operando es verdadero y a **`true`** en caso contrario.	
Comparación	`==`	`a==b`	Evalúa a **`true`** si **a** y **b** son iguales y a **`false`** en caso contrario.	`true==!false`
	`>`	`a>b`	Evalúa a **`true`** si **a** es estrictamente mayor que **b** y a **`false`** en caso contrario.	`edad>65`
	`>=`	`a>=b`	Evalúa a **`true`** si **a** es mayor o igual que **b** y a **`false`** en caso contrario.	`edad>=18`
	`<`	`a<b`	Evalúa a **`true`** si **a** es estrictamente menor que **b** y a **`false`** en caso contrario.	`volumen<25.7`
	`<=`	`a<=b`	Evalúa a **`true`** si **a** es menor o igual que **b** y a **`false`** en caso contrario.	`x<=y`
	`!=`	`a!=b`	Evalúa a **`true`** y **a** y **b** son distintos y a **`false`** en caso contrario.	`anno!=2000`

14.5.1.1 Sentencia `if`

Tiene la estructura siguiente:

```
if (condición) {
  sentencias
}
[else {
  otras sentencias
}]
```

Si la **`condición`** es verdadera entonces se ejecutan las **`sentencias`** y si es falsa entonces se ejecutan **`otras sentencias`**.

A continuación se muestran dos ejemplos muy sencillos del uso de esta sentencia.

```
if (llueve)
  coger_paraguas();
```

```
if (lunes)
  ver_cine();
else
  ver_tv();
```

14.5.1.2 Sentencia **switch**

Suponga el lector que tiene una variable que toma como posibles valores los días de la semana, debiendo ejecutar unas sentencias distintas para cada día, ¿cómo se podría expresar esta situación mediante la sentencia **if**? Más o menos de esta forma:

```
if (dia==lunes) {
  hacer_cosas_lunes();
} else {
  if (dia==martes) {
    hacer_cosas_martes();
  } else {
    if (dia==miercoles) {
      hacer_cosas_miercoles();
    } else...
  }
}
```

La solución es, como poco, confusa; para evitar estas situaciones se introduce la sentencia **switch** que sigue esta estructura:

```
switch (expresión) {
  case etiqueta:
    sentencias
    break;
  case etiqueta:
    sentencias
    break;
  ...
  default:
    sentencias
}
```

Esta construcción evalúa la expresión y compara dicho valor con las etiquetas ejecutando las sentencias que se encuentre desde la etiqueta coincidente hasta el primer **break** (esto es muy importante); en caso de que ninguna etiqueta coincida con el valor de la expresión ejecutaría las sentencias tras la etiqueta **default** si existe o no haría nada en caso de no existir. El ejemplo de los días de la semana quedaría así:

```
switch (dia) {
  case “lunes”:
    hacer_cosas_lunes();
    break;
  case “martes”:
    hacer_cosas_marte();
    break;
  case “miercoles”:
    hacer_cosas_miercoles();
    break;
  case “jueves”:
    hacer_cosas_jueves();
    break;
  case “viernes”:
    hacer_cosas_viernes();
    break;
}
```

Como se puede ver, el código es muchísimo más claro.

14.5.2 Sentencias de bucles

Un bucle es un conjunto de sentencias que se ejecutan repetidamente hasta que se cumple una condición específica. *JavaScript* soporta dos estructuras repetitivas: **`for`** y **`while`**; por su parte, *Navigator* e *Internet Explorer* añaden una tercera, el **`do...while`**.

14.5.2.1 Sentencia **`for`**

Un bucle **`for`** se repite hasta que una condición previamente especificada evalúa a falso. Su estructura es la siguiente:

```
for ([expresión inicial;] [condición;] [expresión incremental]} {
  sentencias
}
```

El intérprete de *JavaScript* hace lo siguiente cuando se encuentra una sentencia for:

1. Ejecutar la **`expresión inicial`**.
2. Comprobar la **`condición`**.
3. Si es falsa saltar a 7.
4. Ejecutar las **`sentencias`**.
5. Ejecutar la **`expresión incremental`**.
6. Volver a 2.
7. Salir del bucle.

Generalmente, la **`expresión inicial`** sirve para establecer un valor de partida de un contador, cuyo valor se comprueba en la **`condición`** y se incrementa en la **`expresión incremental`**; así un bucle for típico es el siguiente:

```
for (var i=0; i<5; i++) {
  hacer_algo();
}
```

14.5.2.2 Sentencia **`while`**

Una sentencia while repite un bucle hasta que una condición específica evalúa a falso. Su estructura es la siguiente:

```
while (condición) {
  sentencias
}
```

Si la **`condición`** se hace falsa, las **`sentencias`** del interior del bucle dejan de ejecutarse y el control pasa a la siguiente sentencia tras el bucle.

La **`condición`** sólo se comprueba cuando las **`sentencias`** han sido ejecutadas y el bucle va a repetirse; es decir, la comprobación no es continua sino que se lleva a cabo una vez al principio del bucle y de nuevo justo después de la última de las **`sentencias`**.

Un ejemplo típico de bucle **`while`** es el siguiente:

```
while (semaforo==rojo) {
  esperar();
}
```

14.5.2.3 Sentencia **`do...while`**

La sentencia **`do...while`** es muy similar a la sentencia **`while`**, con la diferencia de que la condición se comprueba sólo tras la ejecución de las sentencias; así una sentencia **`while`** se ejecuta 0 o más veces, mientras que una sentencia **`do...while`** se ejecuta 1 o más veces. Su estructura es la siguiente:

```
do {
  sentencias
} while (condicion);
```

14.5.2.4 Sentencias **`break`** y **`continue`**

La sentencia **`break`** termina el bucle actual y transfiere el control del programa a la sentencia que sigue al bucle.

Una sentencia **`continue`** termina la ejecución del bloque de sentencias en un bucle y continua la ejecución del mismo en la siguiente iteración. En contraste con la sentencia **`break`**, **`continue`** no termina la ejecución de un bucle sino que en el caso de un bucle **`while`** vuelve a la condición y en un bucle **`for`** a la expresión incremental.

14.5.3 Comentarios

El lector ya está familiarizado con los comentarios (ya sabe como escribirlos en *HTML* y *CSS*) y se le ha hablado de los comentarios en *JavaScript* al principio de este capítulo; sin embargo, no está de más insistir.

En *JavaScript* se pueden escribir comentarios de dos tipos: de una sola línea, van precedidos por dos barras inclinadas hacia la derecha (`//`) y comentarios que pueden ocupar varias líneas, delimitados por los símbolos **`/*`** y **`*/`**.

En estos momentos el lector ya tiene una comprensión bastante buena de los fundamentos del lenguaje *JavaScript*; es hora pues de hacer un ejercicio.

14.6 Funciones

Una función es un conjunto de sentencias que llevan a cabo una tarea específica y puede ser invocada mediante su nombre. Antes de poder llamar a una función es necesario definirla; la definición de una función consiste en la palabra reservada **`function`** seguida de:

- El nombre de la función.
- La lista de argumentos que recibe la función, separados por comas y encerrados entre paréntesis.
- Las sentencias que definen la función encerradas entre llaves (estas sentencias pueden incluir llamadas a otras funciones, o a la misma función, definidas en la misma aplicación).

Las funciones pueden, opcionalmente, devolver un valor.

A continuación se muestra una pequeña función que retorna el cuadrado del valor que se le pasa como argumento:

```
function cuadrado (valor) {
  return valor*valor;
}
```

Dicha función puede invocarse de la forma siguiente:

```
resultado=cuadrado(2);
```

14.6.1 Funciones con un número variable de argumentos

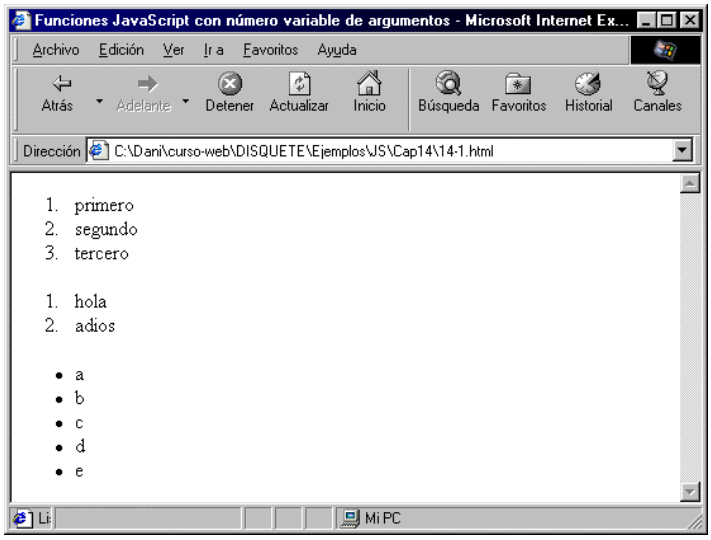
Se le pueden pasar a una función más argumentos de los declarados. Esto es muy útil si no se sabe de antemano cuántos van a ser necesarios. Para poder acceder desde el código de la función a los mismos se utiliza una *propiedad* de las funciones denominada **`arguments`**; dicha propiedad es un vector que almacena los argumentos recibidos. El número de argumentos pasados a la función se puede terminar consultando la propiedad **`arguments.length`**. Para dejar más claro este concepto se muestra el siguiente código que genera listas ordenadas y no ordenadas de cualquier número de elementos.

Ejemplo 14-1 (Funciones con un número variable de argumentos)

```
<HTML>
  <HEAD>
    <TITLE>Funciones JavaScript con n&uacute;mero variable de argumentos</TITLE>

    <SCRIPT LANGUAGE="JavaScript">
      <!--
      function creaLista (tipo) {
        switch (String(tipo).toLowerCase()) {
          case "ol":
            document.write("<OL>\n");
            break;
          case "ul":
          default:
            document.write("<UL>\n");
        }

        for (var elemento=1; elemento<creaLista.arguments.length; elemento++)
          document.write("<LI>" + creaLista.arguments[elemento]);

        switch (String(tipo).toLowerCase()) {
          case "ol":
            document.write("</OL>\n");
            break;
          case "ul":
          default:
            document.write("</UL>\n");
        }
      }
      //-->
    </SCRIPT>
  </HEAD>

  <BODY>
    <SCRIPT LANGUAGE="JavaScript">
      creaLista("ol","primero","segundo","tercero");
      creaLista("Ol","hola","adios");
      creaLista("ul","a","b","c","d","e");
    </SCRIPT>
  <BODY>
</HTML>
```

El resultado puede observarse en la figura 14-1.

Figura 14-1



14.7 Objetos

El código anterior tiene algunas líneas que resultarán nuevas para el lector:

```
String(tipo).toLowerCase()
document.write("<UL>\n");
document.write("<LI>" + creaLista.arguments[elemento]);
document.write("</UL>\n");
```

Aunque seguramente se haga una idea de lo que significan: la primera crea una cadena a partir del argumento recibido y la pasa a minúsculas; las otras tres escriben código *HTML* en el documento.

Como puede ver parecen llamadas a funciones, pero también se parecen algo a la forma de acceder a la propiedad que almacena el número de argumentos de una función, **`creaLista.arguments.length`**. Ciertamente tienen algo en común, puesto que todo ese código está trabajando con objetos; lo que parecen funciones se denominan *métodos* y es código definido para el objeto y que trabaja con (o sobre) propiedades del mismo, propiedades que se denominan *atributos*.

Sin embargo, no es el objetivo de este documento entrar en la discusión de las características de los objetos de *JavaScript* sino explicar lo fundamental del lenguaje para poder hacer frente al *HTML* dinámico; al lector interesado se le recomienda que consulte en las referencias para poder profundizar más en el tema. Baste decir, que los navegadores proporcionan una serie de objetos predefinidos que permiten entre otras cosas acceder al documento *HTML* y modificarlo; así como realizar operaciones con ventanas y *frames*

14.8 Objetos predefinidos

Cuando se carga una página en un navegador se crean una serie de objetos correspondientes a la página, sus contenidos y otra información relacionada con el documento. Cada página genera al menos los siguientes objetos:

- **`window`**: el objeto de más alto nivel de la jerarquía; contiene propiedades que se aplican a la ventana completa. También existe un objeto **`window`** por cada hijo en un documento de *frames*.
- **`location`**: contiene información sobre el *URI* actual.
- **`history`**: objeto con atributos que representan *URI's* previamente visitados por el usuario.
- **`document`**: objeto que controla las propiedades del documento *HTML* visualizado.

A continuación se mostrará la forma de usar a un nivel muy básico los dos objetos del navegador más importantes, **`document`** y **`window`**.

14.8.1 Utilización del objeto `document`

El objeto **`document`** posee una serie de métodos y propiedades muy interesantes; sin embargo, tan solo se verán en este documento dos métodos, **`write`** y **`writeln`**[4], cuya función es escribir código *HTML* en el cuerpo del documento.

El uso de tales métodos es muy sencillo como ya se vio en el ejemplo 14-1, el argumento que reciben es una cadena (o concatenación de cadenas) que representen código *HTML* válido que se añade al documento en el punto donde se encuentra incrustado el *script*.

Un uso habitual de este tipo de métodos es el de adecuar parte del contenido de un documento al navegador utilizado para visualizarlo.

14.8.2 Utilización del objeto `window`

El objeto **`window`** tiene una serie de métodos útiles para crear nuevas ventanas y diálogos *pop-up*:

- **`open`** y **`close`**: permiten abrir y cerrar ventanas del navegador.
- **`alert`**: muestra una caja de diálogo con un mensaje de alerta.
- **`confirm`**: muestra un diálogo de confirmación.

14.8.2.1 El método **`open`**

Este método permite crear nuevas ventanas; la sintaxis es la siguiente:

```
open(URI, nombre_ventana [,características_ventana])
```

`URI` es el *URI* del documento a visualizar en la nueva ventana; **`nombre_ventana`** es un identificador que después podrá ser empleado en el atributo **`TARGET`** de enlaces e imágenes mapa; en cuanto a **`características_ventana`**, se trata de una lista separada por comas de atributos aplicables a la nueva ventana:

- **`height`**: especifica la altura de la ventana en pixels.
- **`resizable`**: admite los valores **`yes`** (se puede cambiar el tamaño de la ventana) y **`no`** (no se puede cambiar el tamaño de la ventana).
- **`screenX`**: coordenada X de la pantalla , en pixels, donde se colocará la esquina superior izquierda de la nueva ventana (en *Internet Explorer* se denomina **`left`**).
- **`screenY`**: coordenada Y de la pantalla , en pixels, donde se colocará la esquina superior izquierda de la nueva ventana (en *Internet Explorer* se denomina **`top`**).
- **`scrollbars`**: admite los valores **`yes`** (se muestran barras de *scroll* si el contenido es mayor que la ventana) y **`no`** (no se muestran barras de *scroll*).
- **`toolbar`**: admite los valores **`yes`** (se muestra la barra de botones y el menú) y **`no`** (no se muestra).
- **`width`**: especifica la anchura de la ventana en pixels.

A continuación se muestra código que abrirá la página *web* de ***GIworks*** en una ventana de 320x240 pixels situada en las coordenadas 100,100 de la pantalla, sin ninguna decoración; obsérvese como se emplean simultaneamente las propiedades **`screenX`**, **`screenY`**, **`left`** y **`top`** para garantizar el correcto funcionamiento tanto en *Navigator* como en *Internet Explorer*. En las figuras 14-2 y 14-3 puede apreciarse el funcionamiento del mismo.

[4] La única diferencia entre **`write`** y **`writeln`** es que el último añade un retorno de carro al final de la línea; dado que los saltos de línea se ignoran en *HTML* esto solo afecta al texto preformateado (mediante la etiqueta **`<PRE>`**).

Ejemplo 14-2 (Apertura de una nueva ventana)

```
<HTML>
  <HEAD>
    <TITLE>Uso del m&eacute;todo open del objeto window</TITLE>

    <SCRIPT LANGUAGE="JavaScript">
      <!--
      function abreGIworks () {
        window.open("http://giworks.uniovi.es",
                    "giworks",”scrollbars=no, toolbar=no, height=240, width=320,
resizable=no, screenX=100, screenY=100, left=100, top=100");
      }
      //-->
    </SCRIPT>
  </HEAD>

  <BODY>
    <A HREF="javascript:abreGIworks()">Abre la p&aacute;gina <I>web</I> de
<B><I>GIworks</I></B></A>
  <BODY>
</HTML>
```

Figura 14-2

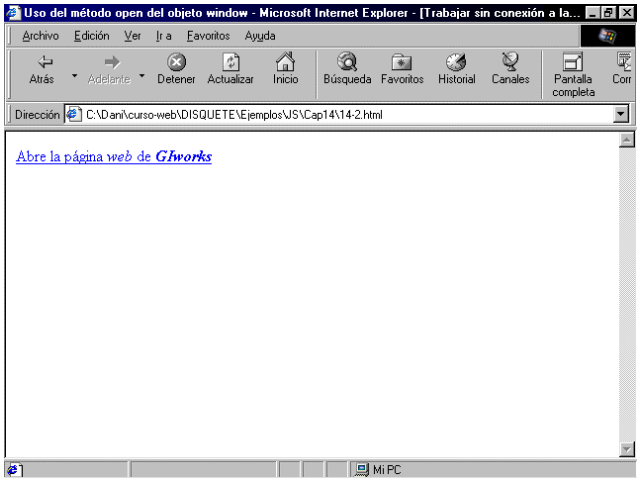
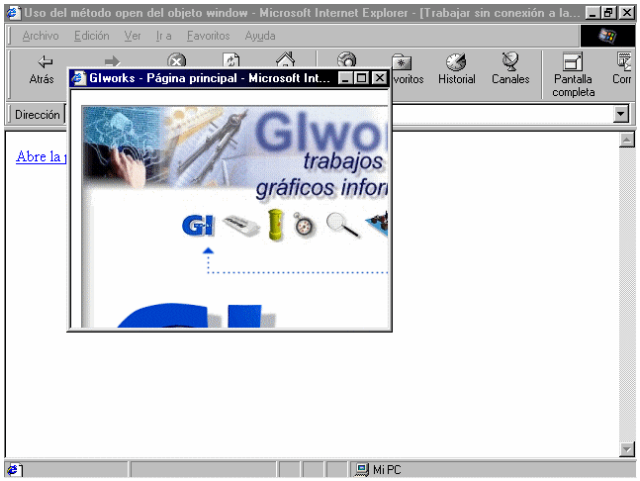


Figura 14-3



14.8.2.2 El método `close`

El método **`close`** cierra una ventana; si se quiere cerrar la propia ventana bastaría con emplear una de las siguientes sentencias:

```
window.close();
self.close();
close();
```

En caso de querer cerrar otra ventana habría que haber recogido el valor de retorno del método **`open`** empleado para crearla; así el código

```
ventana.close();
```

cerraría una ventana creada de la forma

```
ventana=window.open(...);
```

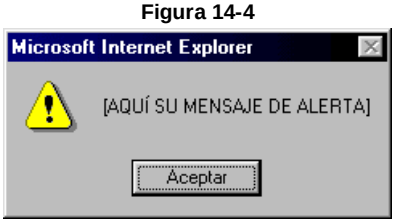
14.8.2.3 El método `alert`

Este método permite mostrar un mensaje de alerta en una ventana; la sintaxis es la siguiente:

```
window.alert(mensaje);
```

Que daría lugar a un resultado similar al de la figura 14-4.

Figura 14-4



14.8.2.4 El método `confirm`

Este método visualiza un diálogo de confirmación con un mensaje explicativo y dos botones, uno para aceptar y otro para cancelar la operación. La sintaxis es semejante a la de **`alert`**:

```
window.confirm(mensaje);
```

Con la diferencia de que este método devuelve un valor lógico, **`true`** si el usuario decide aceptar y **`false`** si decide cancelar. El ejemplo 14-3 muestra código que emplea este método.

Ejemplo 14-3 (El método `confirm` del objeto `window`)

```
<HTML>
  <HEAD>
    <TITLE>Uso del m&eacute;todo confirm del objeto window</TITLE>
  </HEAD>

  <BODY>
    <SCRIPT LANGUAGE="JavaScript">
      <!--
      if (window.confirm("El área de datos..."))
        window.alert("El desfrinkado ha eliminado...");
      else
        window.alert("El modo no estable...");
      //-->
    </SCRIPT>
  <BODY>
  </BODY>
</HTML>
```

El resultado de visualizar este código junto con los efectos de elegir una u otra opción se puede ver en las figuras 14-5, 14-6 y 14-7.

Figura 14-5

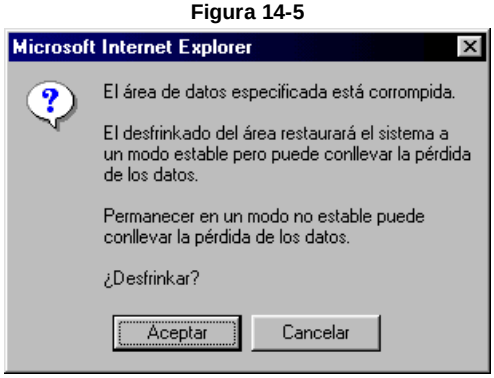


Figura 14-6

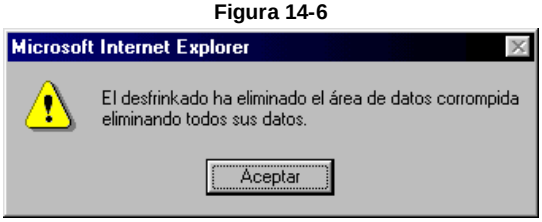


Figura 14-7

14.9 Empleo de eventos

Para terminar este capítulo y antes de pasar al dedicado a *HTML* dinámico hay un último concepto del lenguaje *JavaScript* que debería conocer el lector, se trata de los *eventos* y sus correspondientes manejadores.

Un evento es algo que sucede en un momento dado, generalmente debido a una acción de usuario; por ejemplo, pulsar un botón, cargar una página, colocar el ratón sobre un enlace, etc. *JavaScript* permite gestionar esos eventos mediante sus manejadores, que capturan el evento y ejecutan cuando éste se produce el código que el desarrollador desee; dichos manejadores no son más que atributos de etiquetas *HTML* cuyo valor es el nombre de una función *JavaScript* a ejecutar cuando el evento que manejan se dispara. Por ejemplo:

```
<A HREF="http://giworks.uniovi.es" onClick="window.alert('Bien
hecho!');">P&aacute;gina <I>web</I> de <I><B>GIworks</B></I></B>
```

El código anterior declara un manejador de eventos para el enlace, manejador que se ocupa del evento **`Click`** (pulsar el enlace) y que ejecuta el código *JavaScript* que se le ha indicado, en este caso la visualización de un mensaje.

En la tabla 14-2 se muestran los eventos más comunes, los elementos[5] a que se aplican, cuándo ocurren y cuál es el correspondiente manejador.

Tabla 14-2

Evento	Elementos a que se aplica	Cuándo ocurre	Manejador
`Click`	Enlaces	El usuario activa el enlace.	**`onClick`**
`Load`	Cuerpo del documento	Se carga la página en el navegador.	**`onLoad`**
`MouseOut`	Enlaces	El usuario saca el ratón de un enlace.	**`onMouseOut`**
`MouseOver`	Enlaces	El usuario coloca el ratón encima de un enlace.	**`onMouseOver`**
`Resize`	Ventanas	La ventana cambia de tamaño.	**`onResize`**
`UnLoad`	Cuerpo del documento	El usuario abandona la página.	**`onUnLoad`**

El ejemplo 14-4 muestra todos los eventos anteriores en acción.

Ejemplo 14-4 (Manejo de eventos)

```
<HTML>
  <HEAD>
    <TITLE>Uso de manejadores de eventos</TITLE>
  </HEAD>

  <BODY>
    <SCRIPT LANGUAGE="JavaScript">
      <!--
      function miOnClick () {
        window.alert("Se ha disparado Click");
      }

      function miOnLoad () {
        window.alert("Se ha cargado la página\nSe ha disparado Load");
      }

      function miMouseOut () {
        window.alert("El cursor ha salido\nSe ha disparado MouseOut");
      }

      function miMouseOver () {
        window.alert("El cursor ha entrado\nSe ha disparado MouseOver");
      }

      function miResize () {
        window.alert("La ventana ha cambiado de tamaño\nSe ha disparado Resize");
      }
```

[5] Elementos *HTML* vistos durante el curso, para mayor información consultar la referencia.

```
      function miUnLoad () {
        window.alert("Te vas de la página\nSe ha disparado UnLoad");
      }

      // Se asigna el manejador de cambio de tamaño a la ventana
      //
      window.onResize=miResize;

      //-->
    </SCRIPT>
  <BODY onLoad="miOnLoad()" onUnLoad="miUnLoad()">
    <A HREF="http://giworks.uniovi.es" onClick="miOnClick()"
onMouseOver="miMouseOver()" onMouseOut="miMouseOut()">P&aacute;gina <I>web</I> de
<I><B>GIworks</B></I></A>
  </BODY>
</HTML>
```

Antes de pasar al capítulo dedicado a *DHTML* el lector debería hacer el ejercicio correspondiente a la última materia vista sobre *JavaScript*.

15. *HTML* dinámico

15.1 Introducción

Existe bastante confusión respecto a lo que es realmente el *HTML* dinámico, mencionado en muchos lugares como *DHTML*; a esa confusión generalizada contribuyen, como es habitual, *Microsoft* y *Netscape*, puesto que cada uno de ellos entiende el *HTML* dinámico de forma ligeramente distinta; incluyendo, en el caso de *Netscape*, la introducción de una nueva etiqueta que sirve para hacer lo mismo que ya se puede hacer con *CSS*.

Lo que debe quedarle claro al lector es que no existe ningún lenguaje denominado *DHTML*, ni se trata de una extensión del *HTML*, *HTML* dinámico es la conjunción de *HTML 4.0*, hojas de estilo y *JavaScript*. El uso combinado de estas tres tecnologías permite introducir dinamismo en las páginas *web*; dinamismo que permite el desarrollo de atractivos interfaces, complejas animaciones y multitud de efectos y capacidades que hacen las páginas *web* no sólo más vistosas sino mucho más útiles para el usuario.

El lector debe tener en cuenta que desarrollar una página *web* con *HTML* dinámico no es trivial, por ello es preciso que prepare unos bocetos iniciales con lo que desea hacer, la estructura que quiere dar el interfaz para después pasar a determinar cuál será la mejor forma de implementarlo; implementación que implicará el desarrollo de las imágenes del interfaz, la creación de las capas mediante *HTML* y *CSS* y el desarrollo del código *JavaScript* necesario para gestionarlo todo.

Un último consejo para el futuro desarrollador *web*: la mejor forma de aprender es viendo cómo lo hacen los demás, siga tutoriales pero sobre todo estudie el código de otros autores.

15.2 Detección del navegador y características de la plataforma

En primer lugar, una página *web* dinámica debería determinar no sólo el navegador, sino también versión y plataforma[6]. Para ello se debería usar una función que detecte todos esos parámetros; a continuación se muestra un código muy bueno desarrollado por *Freefall Web Design Studio* (**http://www.htmlguru.com**):

```
function Is() {
  var agent = navigator.userAgent.toLowerCase();
  this.major = parseInt(navigator.appVersion);
  this.minor = parseFloat(navigator.appVersion);
  this.ns = ((agent.indexOf('mozilla')!=-1) &&
((agent.indexOf('spoofer')==-1) && (agent.indexOf('compatible')
== -1)));
  this.ns2 = (this.ns && (this.major == 3));
  this.ns3 = (this.ns && (this.major == 3));
  this.ns4b = (this.ns && (this.minor < 4.04));
  this.ns4 = (this.ns && (this.major >= 4));
  this.ie = (agent.indexOf("msie") != -1);
  this.ie3 = (this.ie && (this.major == 2));
  this.ie4 = (this.ie && (this.major >= 4));
  this.op3 = (agent.indexOf("opera") != -1);
  this.win = (agent.indexOf("win")!=-1);
  this.mac = (agent.indexOf("mac")!=-1);
  this.unix = (agent.indexOf("x11")!=-1);
}

var is = new Is();
```

[6] Las diferencias entre navegadores ya han debido quedar patentes para el lector, pero debe ser consciente de que las diferencias entre distintas versiones del mismo navegador (especialmente las primeras versiones "menores", por ejemplo 4.1 frente a 4.5) o las diferencias entre una misma versión de un navegador en función del sistema operativo pueden dar muchos quebraderos de cabeza al desarrollador.

Empleando este código al principio del *script* se dispone de un objeto, **is**, que tiene una serie de atributos que indican navegador, versión y operativo. Por ejemplo si **is.ns4** es verdadero se trata de *Navigator* 4.0 o superior, mientras que si **is.ie4** es verdadero se trata de *Internet Explorer* 4.0 o superior; **is.win** indica que la plataforma es *Windows*, **is.mac** *Macintosh*, etc. Es muy recomendable que el usuario emplee este código para sus propias páginas.

15.3 Código genérico multinavegador

Como se dijo anteriormente, el *HTML* dinámico se basa por igual en la creación de capas mediante *CSS*[7] y en su gestión mediante *JavaScript*. Como puede suponer el lector, para ello se emplea el modelo de objetos del navegador que permite acceder a las diversas propiedades de las capas; esto supone un problema puesto que aunque las diferencias entre el *JavaScript* y el *JScript* son mínimas, el modelo de objetos no está definido por el lenguaje con lo cual los modelos de *Navigator* e *Internet Explorer* son distintos y mutuamente incompatibles entre sí.

A continuación se muestra un enfoque que se está imponiendo actualmente entre todos los desarrolladores[8] puesto que permite evitar estos problemas de incompatibilidad entre los dos modelos de objetos; consiste en definir una serie de variables que han de emplearse para acceder a las diversas propiedades de las capas y que, en función del navegador, toman unos valores u otros.

```
if(is.ns4) {
  doc = "document";
  sty = "";
  htm = ".document"
} else if(is.ie4) {
  doc = "document.all";
  sty = ".style";
  htm = ""
}
```

¿Qué significa lo anterior? La variable **doc** indica la forma de acceder al objeto **document**, la variable **sty** la forma de acceder a los objetos que representan una capa y **htm** la forma de acceder a los elementos *HTML* que contiene dicha capa. Como puede ver el lector, las variables toman valores distintos dependiendo del navegador; eso es debido, como ya se dijo, por las diferencias entre los modelos de objetos de *Navigator* y de *Internet Explorer*. Para que quede más claro se explicará cada uno de ellos comparando el código que habría que usar para programar directamente el acceso en *Navigator* y en *Internet Explorer* con el código a usar gracias al *DOM*.

15.3.1 Acceso a capas

Para acceder a una capa, *Navigator* emplea una ruta de acceso del estilo **document.nombreCapa** mientras *Internet Explorer* lo hace como **document.all.nombreCapa.style**. En cambio, mediante el uso del *DOM* bastaría con hacer

```
accesoCapa = eval(doc + '["nombreCapa"]' + sty);
```

para tener en una variable la ruta de acceso a la capa cuyo atributo **ID** es **nombreCapa**.

15.3.2 Acceso a elementos de la capa

Un caso típico en que se necesita acceder a un elemento de una capa es para modificar una imagen; la forma de acceder en *Navigator* a los objetos de una capa es del estilo **document.nombreCapa.document.nombreObjeto** mientras *Internet Explorer* lo hace de la forma **document.all.nombreCapa.document.nombreObjeto**.

---

[7] En la introducción se dice que *Netscape* introdujo una etiqueta nueva para crear capas, se trata de **<LAYER>**, no aporta nada que no se pueda hacer con *CSS* y no es nada recomendable su uso; por ello se aconseja encarecidamente al lector que emplee *CSS* para las capas de sus páginas.

[8] Se suele denominar *DOM* –Modelo de Objetos del Documento– y existen diversos artículos en *Internet* sobre el tema.

La forma de hacerlo con un *DOM* es la siguiente

```
accesoObjetosCapa = eval(doc + '["nombreCapa"]' + '.document');
```

Dicha variable almacena la ruta de acceso para los elementos contenidos en esa capa, con lo cual el código

```
accesoObjetosCapa.nombreObjeto
```

accede al objeto pero de una forma mucho más sencilla que usando directamente los modelos de objetos de *Navigator* o *Internet Explorer*.

15.3.3 Acceso al código *HTML* de una capa

El acceso al código *HTML* de una capa se hace de la forma siguiente mediante el *DOM*:

```
accesoCodigoCapa = eval (doc + '["nombreCapa"]' + htm);
```

variable que permitirá modificar el código *HTML* de dicha capa[9].

15.4 Creando y manejando capas

Como ha visto el lector en el punto anterior, para facilitar el trabajo con las capas de su página *web* se hace necesario crear una serie de variables globales que almacenen la ruta de acceso a la misma (accesos distintos en función del uso que se va a hacer, manipular la capa, manipular un elemento contenido en la capa o manipular el código de la capa). Por tanto, una vez determinada la naturaleza del navegador debería ejecutarse una función que realizase esa tarea.

Respecto a la forma de crear las capas ya se dijo que se empleaba *CSS*, puede hacerse como se explicó en el capítulo 10, “Introducción a las hojas de estilo”, o emplear el atributo **STYLE**; dicho atributo toma como valor código *CSS* que no es más que una hoja de estilo escrita en línea dentro del elemento *HTML* a que se va a aplicar.

Así, los dos fragmentos de código que se muestran a continuación son equivalentes.

```
<STYLE TYPE=”text/css”>
  #CargandoImagenes {
    position: absolute;
    z-index: 11;
    visibility: visible;
    left: 140px;
    top: 145px;
    width: 200px;
</STYLE>

  ...

<DIV ID="CargandoImagenes">
  ...
</DIV>
```

Este código crea una capa en el documento *HTML* denominada **CargandoImagenes** con una serie de propiedades que se definen en una hoja de estilo incrustada y definida en el mismo documento (también podría hacerse con una hoja de estilo externa).

[9] En teoría, porque en la práctica tanto el árbol del documento como el código *JavaScript* pueden ser lo suficientemente complejos como para que no funcione correctamente. Se le recomienda al lector estudiar detenidamente el código *JavaScript* y *HTML* de la demostración que acompaña a este curso para la visualización de la barra de progreso, puesto que es un ejemplo de cómo sortear estos *bugs* en *Navigator* e *Internet Explorer*.

```
<DIV ID="CargandoImagenes" STYLE="position: absolute; z-index: 11; visibility:
visible; left: 140px; top: 145px; width:200px;">
  ...
</DIV>
```

El código anterior, en cambio, lo hace todo simultaneamente, crea el elemento *HTML* y especifica sus propiedades mediante *CSS* en línea mediante el atributo **STYLE**.

La forma de hacerlo depende del lector, hay quien prefiere hacerlo en línea y quien prefiere definir primero todas las hojas de estilo; en caso de definir todas las hojas de estilo se recomienda entonces tenerlas externas y enlazarlas desde el código *HTML*.

El lector ya puede en estos momentos crear las capas del documento mediante *HTML* y *CSS* y, mediante el uso de *JavaScript* y del *DOM*, acceder a todas sus propiedades (en el modelo de objetos se denominan igual que en *CSS*) y modificarlas de forma dinámica.

Con todo lo anterior el lector ya puede hacer su primera incursión en el terreno del *HTML* dinámico.

15.5 Precarga de imágenes

La precarga de imágenes es un concepto que precedió a *HTML* dinámico puesto que ya se podía hacer con *Navigator* 3.0. Este navegador fue el primero en permitir modificar el aspecto de una página una vez cargada. Así, se podía hacer referencia desde *JavaScript* a una imagen del documento y sustituirla por una imagen distinta; el problema era que dicha imagen debía ser cargada para visualizarla y entonces el efecto se estropeaba por el retardo. Para solucionarlo se introdujo el concepto de precarga que consiste en crear desde *JavaScript* objetos **Image** a los que se les asigna la imagen que deben almacenar y, después, a la hora de hacer los cambios de imagen, utilizar dichos objetos para que los cambios sean fluidos.

La precarga se programa en una función que es disparada por el evento **onLoad** de la etiqueta **<BODY>**. En la actualidad se utiliza muchísimo más, sobre todo para realizar animaciones; cada fotograma de la animación es una imagen, se precargan todos los fotogramas y se van utilizando según sea necesario para mostrar la animación.

El código para hacer la precarga en su forma más simple es el siguiente:

```
varImagen = new Image();
varImagen.src = “fichero_imagen”;
```

El objeto **Image** puede recibir como argumentos las dimensiones de la imagen, aunque en realidad no es necesario especificarlos puesto que tanto *Navigator* como *Internet Explorer* parecen ignorarlas.

En la precarga puede aprovecharse el evento **onLoad** de las imágenes para comprobar que todas las imágenes se han cargado antes de comenzar ninguna animación.

15.6 Animación

En *HTML* dinámico la animación puede hacerse de dos formas, animando las capas o bien animando las imágenes; en el primer caso hay un movimiento real y en el segundo caso un movimiento aparente debido al cambio de fotogramas.

15.6.1 Animación de capas

Es muy sencilla, se basa en el empleo de las propiedades **top** y **left** (o **bottom** y **right**) de la capa, a continuación se muestra el código que mueve dos capas de la *demo* para simular un movimiento de cámara.

```
capaFondo = eval(doc + '["FondoEstrellado"]' + sty);
capaPlaneta = eval(doc + '["PoloPlanetario"]' + sty);

...

// Codigo para simular el movimiento de camara que nos muestra el planeta
//
function animaPlaneta () {
  var y_pos_p = parseInt (capaPlaneta.top);
  var y_pos_f = parseInt (capaFondo.top);
  if (y_pos_p>0) {
    if(is.ie4) {
      capaPlaneta.top=y_pos_p-2;
      capaFondo.top=y_pos_f-2;
    } else {
      capaPlaneta.top=y_pos_p-4;
      capaFondo.top=y_pos_f-4;
    }
    setTimeout[10]("animaPlaneta()",1);
  } else {
    capaDemo.visibility="visible";
    capaDiseno.visibility="visible";
    capaDani.visibility="visible";
    capaAnno.visibility="visible";
    setTimeout("animaMosca()",200);
  }
}
```

En negrita se ha señalado el código que anima la capa que contiene la imagen del planeta para *Internet Explorer*, como se puede ver en primer lugar se determina la posición de la capa y si todavía no ha llegado al punto deseado se modifica. El resto del código sirve para mover otras capas (y moverlas a velocidades distintas en otros navegadores) así como para enlazar con otras animaciones.

15.6.2 Animación de imágenes

La animación de imágenes es si cabe aún más sencilla, su dificultad reside no en la programación de la animación sino en la creación de los fotogramas de la misma. A continuación se muestra el código que permite animar el logotipo que se muestra en la *demo* del curso.

```
capaImgLogotipo = eval(doc + '["LogoStarWars"]' + '.document');

...

setTimeout("animaLogotipo(1)",3000);

...

// Codigo para animar el logotipo de Star Wars
//
function animaLogotipo (frame) {
  eval("capaImgLogotipo.imgStarWars.src=starwars"+ frame +".src");
  if (frame<17) {
    var siguiente=frame+1;
    eval("setTimeout('animaLogotipo(" + siguiente + ")',200)");
  } else {
    setTimeout("capaLogotipo.visibility='hidden'",200);
    setTimeout("capaPrologo.visibility='visible'",500);
    setTimeout("animaPrologo(1)",1300);
  }
}
```

[10] Es muy habitual el uso de temporizadores, para ello se emplea la función **setTimeout** que es muy sencilla, simplemente recibe el código *JavaScript* a ejecutar y el tiempo en milisegundos que debe esperar para ejecutarlo.

Como se puede ver la función recibe como argumento el número de fotograma a visualizar, lo visualiza y si la animación no ha finalizado se invoca de nuevo la propia función con el fotograma siguiente mediante un temporizador que marcará la velocidad de la misma.

El lector se habrá dado cuenta del uso que se hace en muchos puntos del código de una función denominada **`eval`**; dicha función es extremadamente útil puesto que con una sola línea permite hacer cosas muy difíciles de lograr de otra forma. Para explicarla se utilizará el código anterior: para hacer la animación de los fotogramas se emplean unas variables de la forma **`starwars1`**, **`starwars2`**, ..., **`starwars17`** que contienen los fotogramas de la animación, el contenido de esas variables debe asignarse a la imagen de la capa de forma similar a esta:

```
capaImgLogotipo.imgStarWars.src=starwars14.src;
```

Esa línea asignaría a la imagen el fotograma 14, sin embargo para hacer una animación no se puede mostrar siempre el mismo fotograma sino fotogramas sucesivos; si se prueba a hacer lo siguiente:

```
capaImgLogotipo.imgStarWars.src=starwarsframe.src;
```

Pensando que van a ir incrementándose los fotogramas se está cometiendo un error puesto que *JavaScript* interpreta que se quiere asignar la imagen almacenada en la variable **`starwarsframe`**, que no existe, por lo que da un error. Por eso existe **`eval`**, que toma cadenas y variables, evalúa las variables y lo concatena todo para formar código *JavaScript* que se trata de evaluar (léase ejecutar); así si se cambia ligeramente la línea anterior se tiene:

```
eval ( “capaImgLogotipo.imgStarWars.src=starwars” + frame + “.src” );
```

¿Qué hace? Supóngase que **`frame`** tiene el valor 5, entonces **`eval`** toma la cadena **`capaImgLogotipo.imgStarWars.src=starwars`**, evalúa **`frame`** a 5, lo concatena con la cadena anterior y con la cadena **`.src`** obteniendo **`capaImgLogotipo.imgStarWars.src=starwars5.src`**; trata de evaluarlo y obtiene como resultado un cambio en el fotograma.

Con esto puede darse por finalizada la parte dedicada a *HTML* dinámico y el curso pero, antes de terminar, se mostrará un código muy sencillo que cubrirá todo lo explicado en esta tercera y última parte de tal forma que el lector tendrá un punto de partida completo pero fácilmente accesible y un código más avanzado (el de la *demo*) para poder practicar antes de realizar sus propios trabajos con páginas *web* dinámicas.

15.7 Ejemplo sencillo de página web dinámica

A continuación se muestra el código *HTML* y *JavaScript* que implementa un ejemplo sencillo de *HTML* dinámico con los siguientes aspectos:

- Detección de navegador.
- Código multinavegador.
- Precarga de imágenes
- Animación de imágenes.
- Animación de capas.
- Modificación de elementos en una capa.

Ejemplo 15-1 (Código *HTML*)

```
1. <HTML>
2.   <HEAD>
3.     <TITLE>Ejemplo sencillo de DHTML</TITLE>
4.   </HEAD>

5.   <SCRIPT LANGUAGE="JavaScript1.2" SRC="15-1.js"></SCRIPT>

6.   <BODY onLoad="preCarga();" BGCOLOR=BLACK>
7.     <!-- Capa para los numeros -->
8.     <DIV ID="cuentaAtras" STYLE="position: absolute; z-index:0; visibility: hidden;
    left: 0px; top: 0px;">
```

```
9.      <IMG NAME="imgNumero" SRC="00.gif">
10. </DIV>
11.     <!-- Capas para las explicaciones -->
12.     <DIV ID="Explicacion1" STYLE="position: absolute; z-index:0; visibility:
    hidden; left: 0px; top: 0px; color:white; margin: 20px; width:300px; text-
    align:center">
13.     <P>La animaci&oacute;n anterior se realiz&oacute; mediante sustituci&oacute;n
    de im&aacute;genes.</P>
14.     <A HREF="javascript:animaCapaArriba1()" onClick="this.blur()"
    onMouseOver="dentroBoton(1)" onMouseOut="fueraBoton(1)"><IMG NAME="imgMas1"
    SRC="mas-positivo.gif" ALT="Siguiente demostraci&oacute;n" BORDER=0></A>
15.     </DIV>

16.     <DIV ID="Explicacion2" STYLE="position: absolute; z-index:0; visibility:
    hidden; left: -320px; top: 0px; color:white; margin: 20px; width:300px; text-
    align:center">
17.     <P>Y estas mediante desplazamiento de capas.</P>
18.     <A HREF="javascript:explicacionBoton()" onClick="this.blur()"
    onMouseOver="dentroBoton(2)" onMouseOut="fueraBoton(2)"><IMG NAME="imgMas2"
    SRC="mas-positivo.gif" ALT="Siguiente demostraci&oacute;n" BORDER=0></A>
19.     </DIV>

20.     <DIV ID="Explicacion3" STYLE="position: absolute; z-index:0; visibility:
    hidden; left: 0px; top: -150px; color:white; margin: 20px; width:300px; text-
    align:center">
21.     <P>El cambio de la imagen en el bot&oacute;n se hace mediante eventos.</P>
22.     <A HREF="javascript:fin()" onClick="this.blur()" onMouseOver="dentroBoton(3)"
    onMouseOut="fueraBoton(3)"><IMG NAME="imgMas3" SRC="mas-positivo.gif"
    ALT="Siguiente demostraci&oacute;n" BORDER=0></A>
23.     </DIV>

24.     <!-- Capa para el final -->
25.     <DIV ID="Fin" STYLE="position: absolute; z-index:0; visibility: hidden; left:
    0px; top: 0px; color:white; margin: 20px; width:300px; text-align:center">
26.     <IMG SRC="fin.gif" ALT="Se acab&oacute;">
27.     </DIV>
28.   </BODY>
29. </HTML>
```

En el ejemplo anterior se han numerado las líneas para poder señalar con mayor claridad los conceptos que en él se muestran; se representa en negrita el código más representativo, pues la mayor parte del mismo es muy parecido.

En la línea 5 se indica que se cargará un fichero *JavaScript* externo y que el lenguaje es *JavaScript 1.2*, de esta manera un navegador que no pueda ejecutar el código ni siquiera lo intentará evitándose así el riesgo de que un usuario con un navegador inapropiado sólo consiga ver multitud de mensajes de error.

En la línea 6 se indica al navegador que debe ejecutar la función **`preCarga()`** en el momento en que se cargue el documento *HTML*.

En la línea 8 se crea una capa en *HTML* escribiendo las propiedades *CSS* en línea mediante el atributo **`STYLE`**.

En la línea 9 se da un nombre, mediante el atributo **`NAME`**, a la imagen para poder acceder a ella desde *JavaScript*.

En la línea 14 se tiene una imagen que actúa como un enlace, la imagen está nombrada para ser accesible desde *JavaScript* y el enlace apunta a código *JavaScript* que se ejecutará al pulsarlo e indica una serie de manejadores de eventos para manejar la entrada y salida del ratón. El código **`this.blur()`** está destinado a eliminar esa caja punteada con que *Internet Explorer* rodea los enlaces una vez pulsados.

Ejemplo 15-2 (Código *JavaScript*)

```
1. // Funcion para determinar que navegador se esta utilizando
2. //
3. function Is () {
4.   var agent = navigator.userAgent.toLowerCase();
5.   this.major = parseInt(navigator.appVersion);
6.   this.minor = parseFloat(navigator.appVersion);
7.   this.ns  = ((agent.indexOf('mozilla')!=-1) && ((agent.indexOf('spoofer')==-1) &&
   (agent.indexOf('compatible') == -1)));
8.   this.ns2 = (this.ns && (this.major == 2));
9.   this.ns3 = (this.ns && (this.major == 3));
10.  this.ns4b = (this.ns && (this.minor < 4.04));
11.  this.ns4 = (this.ns && (this.major >= 4));
12.  this.ie   = (agent.indexOf("msie") != -1);
13.  this.ie3  = (this.ie && (this.major == 2));
14.  this.ie4  = (this.ie && (this.major >= 4));
15.  this.op3 = (agent.indexOf("opera") != -1);
16.  this.win   = (agent.indexOf("win")!=-1);
17.  this.mac   = (agent.indexOf("mac")!=-1);
18.  this.unix  = (agent.indexOf("x11")!=-1);
19.}

20.var is = new Is ();

21.// Se habilitan una serie de variables que se utilizaran para manejar
22.// el documento y las capas de forma independiente del navegador
23.//
24.if (is.ns4) {
25.  doc = "document";
26.  sty = "";
27.  htm = ".document"
28.} else if(is.ie4) {
29.  doc = "document.all";
30.  sty = ".style";
31.  htm = ""
32.}

33.// Funcion para la precarga de imagenes
34.//
35.function preCarga () {
36.  // Se cargan las imagenes desde 00.gif hasta 09.gif
37.  //
38.  for (i=0;i<=9;i++) {
39.    eval("num" + i + "=new Image()");
40.    eval("num" + i + ".onload=(is.ns4b)?testCarga():testCarga");
41.    eval("num" + i + ".src='0" + i + ".gif'");
42.  }

43.  // Se cargan las imagenes desde 00.gif hasta 09.gif
44.  //
45.  for (i=10;i<=12;i++) {
46.    eval("num" + i + "=new Image()");
47.    eval("num" + i + ".onload=(is.ns4b)?testCarga():testCarga");
48.    eval("num" + i + ".src='" + i + ".gif'");
49.  }

50.  // Se cargan las imagenes en positivo y negativo del boton
51.  // de avance
52.  //
53.  masPositivo=new Image();
54.  masPositivo.onload=(is.ns4b)?testCarga():testCarga;
55.  masPositivo.src="mas-positivo.gif"
```

```
  masNegativo=new Image();
  masNegativo.onload=(is.ns4b)?testCarga():testCarga;
  masNegativo.src="mas-negativo.gif"
}

totalImagenes=0;

// Funcion que controla el proceso de carga
//
function testCarga() {
  totalImagenes++;

  // Si se han cargado todas las imagenes iniciar la demo
  //
  if (totalImagenes==15) {
    estableceCapas();
    capaNumeros.visibility="visible";
    animaNumeros(1);
  }
}

// Funcion que hace la animacion de los numeros
//
function animaNumeros(frame) {
  var siguiente;
  eval("capaImgNumeros.imgNumero.src=num"+ frame +".src");
  if (frame<8) {
    // La animacion para el 5
    //
    siguiente=frame+1;
    eval("setTimeout('animaNumeros(" + siguiente + ")',110)");
  } else if (frame<12) {
    // La animacion para el resto de numeros
    //
    siguiente=frame+1;
    eval("setTimeout('animaNumeros(" + siguiente + ")',200)");
  } else {
    // Se ocultan los numeros y se muestra la explicacion nº 1
    //
    setTimeout("capaNumeros.visibility='hidden'",100);
    setTimeout("capaExplicacion1.visibility='visible'",100);
  }
}

  // Funcion que oculta la explicacion nº 1
  //
  function animaCapaArriba1 () {
    var y_pos = parseInt (capaExplicacion1.top);
    if (y_pos>-150) {
      capaExplicacion1.top=y_pos-4;
      setTimeout("animaCapaArriba1()",1);
    } else {
      // Se muestra la explicacion nº 2
      //
      capaExplicacion2.visibility="visible";
      animaCapaLado1();
     }
   }
```

```
109.  // Funcion que anima la explicacion nº 2
110.  //
111.  function animaCapaLado1 () {
112.    var x_pos = parseInt (capaExplicacion2.left);
113.    if (x_pos<0) {
114.      capaExplicacion2.left=x_pos+4;
115.      setTimeout("animaCapaLado1()",1);
116.    }
117.  }

118.  // Funcion que inicia la explicacion del boton
119.  //
120.  function explicacionBoton() {
121.    setTimeout("animaCapaLado2()",1);
122.  }

123.  // Funcion que oculta la explicacion nº 2
124.  //
125.  function animaCapaLado2 () {
126.    var x_pos = parseInt (capaExplicacion2.left);
127.    if (x_pos>-320) {
128.      capaExplicacion2.left=x_pos-4;
129.      setTimeout("animaCapaLado2()",1);
130.    } else {
131.      // Se inicia la explicacion nº 3
132.      //
133.      capaExplicacion3.visibility="visible";
134.      animaCapaArriba2();
135.    }
136.  }

137.  // Funcion que anima la explicacion nº 3
138.  //
139.  function animaCapaArriba2 () {
140.    var y_pos = parseInt (capaExplicacion3.top);
141.    if (y_pos<0) {
142.      capaExplicacion3.top=y_pos+4;
143.      setTimeout("animaCapaArriba2()",1);
144.    }
145.  }

146.  // Funcion que oculta la ultima explicacion y muestra el final
147.  //
148.  function fin() {
149.    capaExplicacion3.visibility="hidden";
150.    capaFin.visibility="visible";
151.  }


152.  // Funcion que crea los accesos a las capas y sus elementos
153.  //
154.  function estableceCapas() {
155.    // Se crea el acceso a las capas
156.    //
157.    capaNumeros = eval(doc + '["cuentaAtras"]' + sty);
158.    capaExplicacion1 = eval(doc + '["Explicacion1"]' + sty);
159.    capaExplicacion2 = eval(doc + '["Explicacion2"]' + sty);
160.    capaExplicacion3 = eval(doc + '["Explicacion3"]' + sty);
161.    capaFin = eval(doc + '["Fin"]' + sty);

162.    // Se crea el acceso a los elementos de las capas
163.    //
164.    capaImgNumeros = eval(doc + '["cuentaAtras"]' + '.document');
165.    capaImgExplicacion1 = eval(doc + '["Explicacion1"]' + '.document');
166.    capaImgExplicacion2 = eval(doc + '["Explicacion2"]' + '.document');
167.    capaImgExplicacion3 = eval(doc + '["Explicacion3"]' + '.document');
168.  }
```

```
169.   // Funcion para el manejador onMouseOver
170.   //
171.   function dentroBoton(capa) {
172.     eval ("capaImgExplicacion" + capa + ".imgMas" + capa +
   ".src=masNegativo.src");
173.   }

174.   // Funcion para el manejador onMouseOut
175.   //
176.   function fueraBoton(capa) {
177.     eval ("capaImgExplicacion" + capa + ".imgMas" + capa +
   ".src=masPositivo.src");
178.   }
```

En la línea 35 comienza la función **`preCarga`**, se ha señalado el código que se encarga de cargar una serie de imágenes llamadas **`00.gif`**, **`01.gif`**, ..., **`09.gif`**; para mostrar el uso que hace de la función **`eval`**.

En la línea 60 se pone a cero la variable que empleará la función **`testCarga`** para controlar el proceso de carga de imágenes.

En la línea 63 comienza la función **`testCarga`** que va incrementando la variable **`totalImagenes`** hasta llegar a 15 momento en que inicia la demostración propiamente dicha.

En la línea 75 hay una animación mediante sustitución de imágenes; obsérvese la forma en que emplea **`eval`** para ir accediendo en cada iteración a una imagen distinta y como utiliza una variable de acceso a elementos de capa para modificar su contenido.

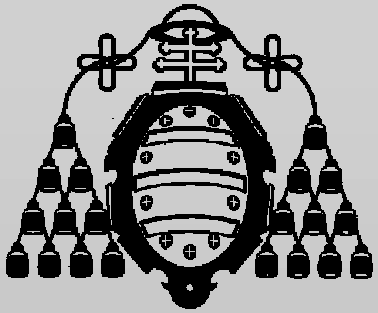
En la línea 97 hay una típica animación por desplazamiento de capas.

En la línea 154 está la función que crea las variables de acceso a las capas y a los elementos de las mismas mediante el uso del *DOM*.

Y en la línea 171 se encuentra la función que ejecutará el manejador del evento **`onMouseOver`** y que no hace nada más que cambiar una imagen dentro de una capa; obsérvese el uso de **`eval`** para poder utilizar la misma función con imágenes distintas en capas diferentes.

15.8 Conclusión

Con el ejemplo anterior concluye el capítulo y también el curso; espero que este documento le haya sido útil (y siga siéndolo en el futuro) pero espero, sobre todo, que haya disfrutado con él y con el curso y que le sirva de aliciente para seguir profundizando más en un campo tan interesante como el desarrollo *web*.



Diseño gráfico de páginas web

HTML 4.0, hojas de estilo y uso de JavaScript en HTML dinámico

Universidad de Oviedo
Área de Expresión Gráfica en la Ingeniería

Abril-Mayo de 2000
E.T.S.I.I.I.G. Campus de Gijón

Daniel Gayo Avello

Universidad de Oviedo
Área de Expresión Gráfica en la Ingeniería
Abril-Mayo de 2000
E.T.S.I.I.I.G. Campus de Gijón

EJERCICIO

1

Diseño gráfico de páginas *web*

TITULO: Introducción al *HTML*.

TEMA(S): Capítulo 2: Estructura y jerarquía de un documento. Utilización de estilos. Capítulo 3: Creación de Listas.

OBJETIVOS: Introducir al alumno en el uso de conceptos básicos del lenguaje, tales como la jerarquía, el empleo de encabezamientos, párrafos, saltos de línea, estilos físicos y lógicos y el empleo de listas.

ENUNCIADO: Escribir el documento *HTML* que de como resultado la siguiente visualización.

Los *Primogénitos*

Llamadlos los *Primogénitos*. Si bien no eran ni remotamente humanos, eran de carne y hueso, y cuando miraban hacia las profundidades del espacio experimentaban admiración respetuosa, asombro..., y soledad. En cuanto poseyeron el poder, se lanzaron a buscar compañía entre las estrellas.

En sus exploraciones encontraron muchas formas de vida y contemplaron el desarrollo de la evolución en un millar de mundos. Fueron testigos de la frecuencia con que los primero destellos de inteligencia alumbraban y morían en la noche cósmica.

Y como en toda la galaxia no habían encontrado nada más precioso que la *Mente*, alentaron su aparición por doquier. Se convirtieron en labradores de un campo de estrellas. Sembraron y, a veces, cosecharon.

Y, en ocasiones, con desapasionamiento, tuvieron que escardar.

Te quiero igual

te quiero pero te llevaste la flor
y me dejaste el florero
te quiero me dejaste la ceniza
y te llevaste el cenicero
te quiero pero te llevaste marzo
y te rendiste en febrero
primero te quiero igual

1. qwert
    - 1234
        1. qwert
        2. yuiop
    - 5678
2. yuiop
3. qwert

Universidad de Oviedo
Área de Expresión Gráfica en la Ingeniería
Abril-Mayo de 2000
E.T.S.I.I.I.G. Campus de Gijón

EJERCICIO

2

Diseño gráfico de páginas *web*

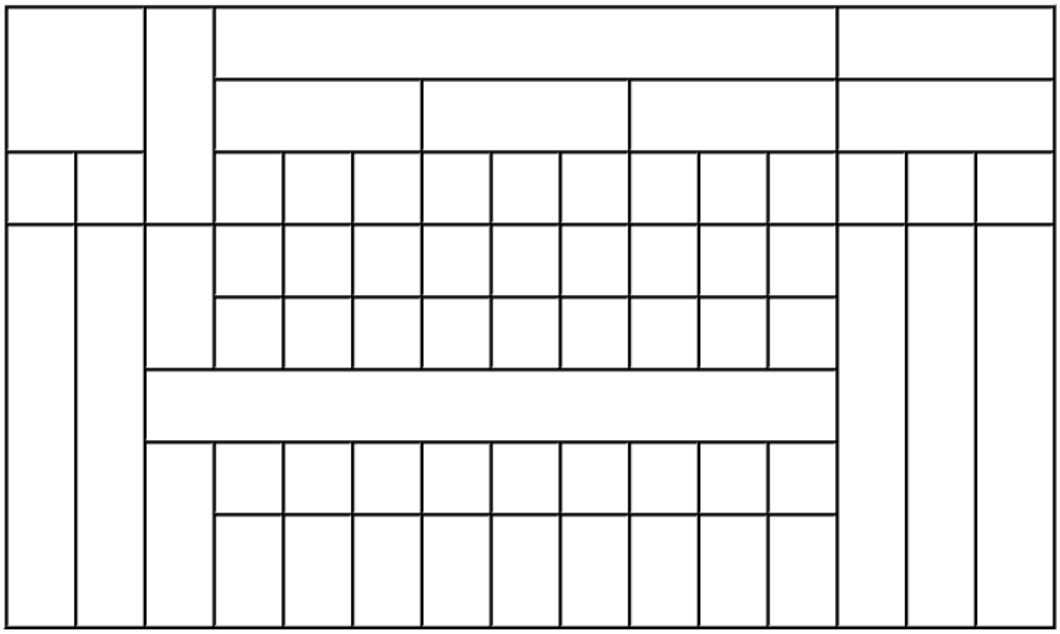
TITULO: Tablas y enlaces.

TEMA(S): Capítulo 4: Creación de tablas. Capítulo 5: Uso de enlaces.

OBJETIVOS: Desarrollo de tablas complejas; creación de enlaces internos y externos; enlaces a distintos tipos de recursos en *Internet*.

ENUNCIADO: Escribir el documento *HTML* que genere una tabla similar a la de la figura y que contenga, además, los siguientes enlaces:

- Ir al final de la página.
- Ir a la página *web* de la Universidad de Oviedo.
- Enviar correo electrónico a un usuario de *Internet*.



Universidad de Oviedo
Área de Expresión Gráfica en la Ingeniería
Abril-Mayo de 2000
E.T.S.I.I.I.G. Campus de Gijón

EJERCICIO

3

Diseño gráfico de páginas *web*

TITULO: Empleo de *frames*.

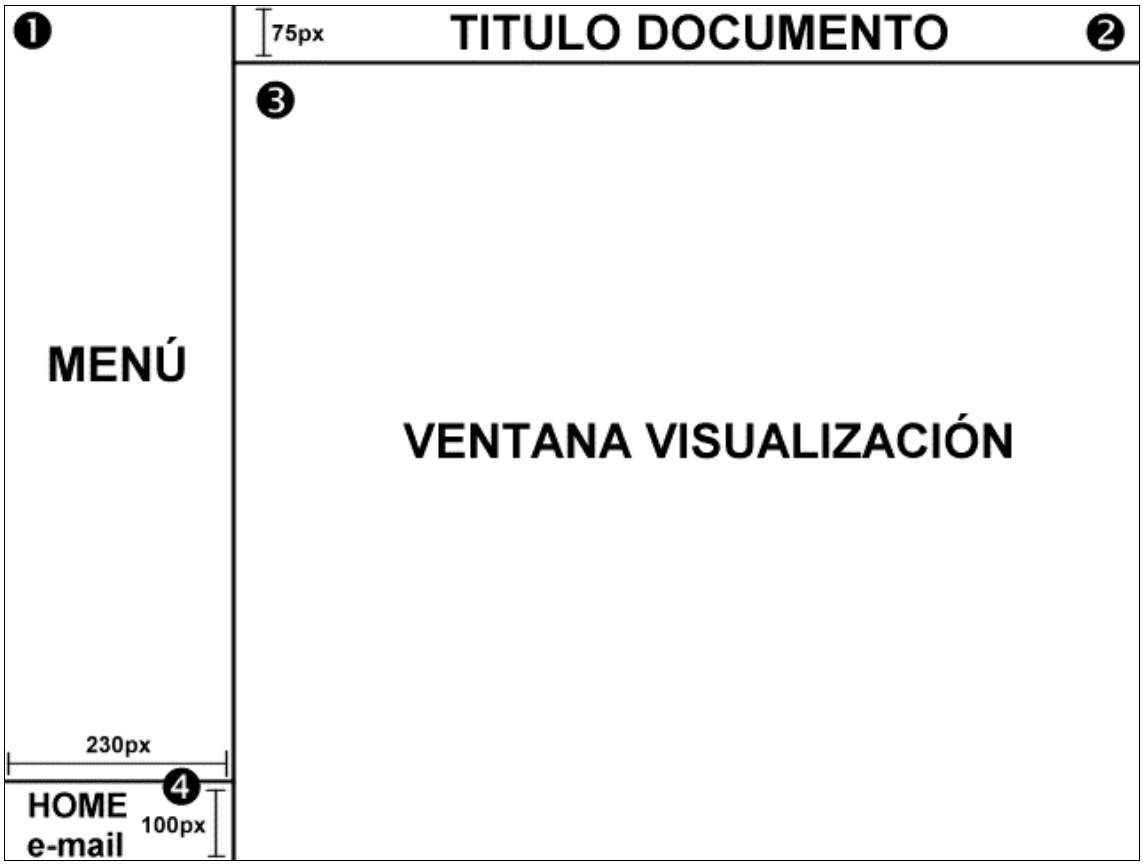
TEMA(S): Capítulo 6: Creación de *frames.*

OBJETIVOS: Desarrollo de documentos complejos mediante el empleo de *frames*.

ENUNCIADO: Escribir el código *HTML* necesario para obtener un documento de *frames* semejante al que se muestra en la figura; siendo el contenido de cada *frame* el que se especifica a continuación:

1. Un menú de navegación con enlaces a diversas páginas *web* que **deben** cargarse en el *frame* número 3. La anchura de dicho *frame* será de 230 pixels.
2. Un título para el documento de *frames.* La altura de dicho *frame* será de 75 pixels.
3. Una ventana de visualización.
4. Una zona con enlaces que siempre deben estar visibles: uno para enviar correo y otro para la página elegida como inicial. Debe tener una altura de 100 pixels y una anchura de 230 pixels.

Ninguno de los *frames* debe poder cambiar de tamaño; los *frames* números 2 y 4 no deben tener barras de *scroll* y los *frames* 1 y 3 en función del documento.



Universidad de Oviedo
Área de Expresión Gráfica en la Ingeniería
Abril-Mayo de 2000
E.T.S.I.I.I.G. Campus de Gijón

EJERCICIO

4

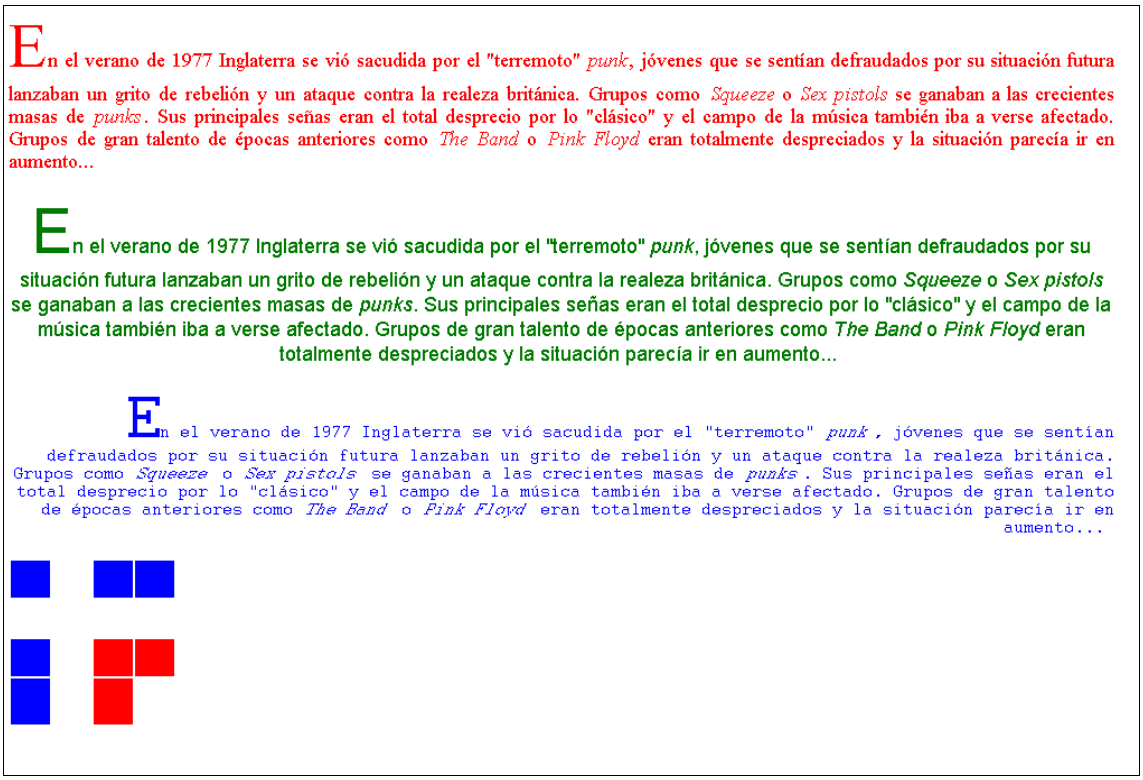
Diseño gráfico de páginas *web*

TITULO: Uso del color.

TEMA(S): Capítulo 7: Colores, alineación y fuentes de texto.

OBJETIVOS: Manejo avanzado de texto (color, alineación y fuentes). Utilización del color en tablas.

ENUNCIADO: Escribir el código *HTML* necesario para obtener un documento semejante al que se muestra en la figura.



Universidad de Oviedo
Área de Expresión Gráfica en la Ingeniería
Abril-Mayo de 2000
E.T.S.I.I.I.G. Campus de Gijón

EJERCICIO

5

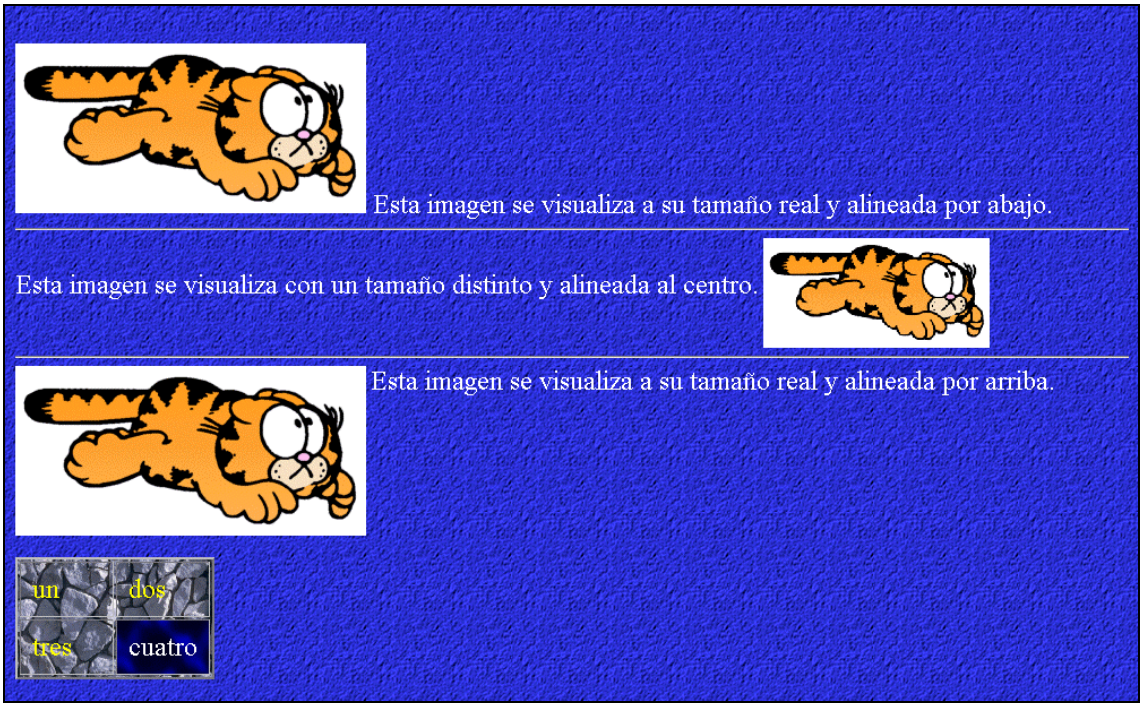
Diseño gráfico de páginas *web*

TITULO: Imágenes.

TEMA(S): Capítulo 8: Inclusión de imágenes, utilización de imágenes como enlaces, uso de imágenes como fondos de página y tabla.

OBJETIVOS: Aplicar lo aprendido sobre imágenes (inclusión, empleo como enlaces y fondos).

ENUNCIADO: Escribir el código *HTML* necesario para obtener un documento semejante al que se muestra en la figura; las imágenes incrustadas deben actuar, además, como enlaces a otros documentos.



Universidad de Oviedo
Área de Expresión Gráfica en la Ingeniería
Abril-Mayo de 2000
E.T.S.I.I.I.G. Campus de Gijón

EJERCICIO

6

Diseño gráfico de páginas *web*

TITULO: *HTML*. Conceptos avanzados.

TEMA(S): Capítulo 9: Objetos incrustados, indicación de la versión de *HTML* empleada, utilización de metadatos.

OBJETIVOS: Comprender mejor el empleo de metainformación, así como la forma de incrustar objetos en documentos *HTML*.

ENUNCIADO: Escribir el código *HTML* para un documento *HTML 4.0* "completo", es decir, con indicación del *DTD* empleado y empleo de metainformación (idioma, descripción del documento, palabras clave y autor); debe incluirse además un objeto (un mundo virtual o un vídeo).

Universidad de Oviedo
Área de Expresión Gráfica en la Ingeniería
Abril-Mayo de 2000
E.T.S.I.I.I.G. Campus de Gijón

EJERCICIO

7

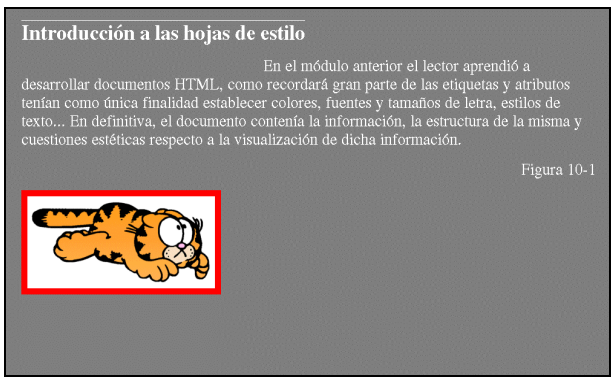
Diseño gráfico de páginas *web*

TITULO: Introducción a las hojas de estilo.

TEMA(S): Capítulo 10: Razón de ser y utilidad de las hojas de estilo, un ejemplo sencillo, inclusión de hojas de estilo.

OBJETIVOS: Aproximarse a las hojas de estilo mediante la experimentación de algunas de sus características; conociendo, además, las distintas formas de asociar estilos a los documentos *HTML*.

ENUNCIADO: Basándose en el ejemplo 10-1 y modificando únicamente el código *CSS* obtener un documento semejante al que se muestra en la figura de la derecha (la figura de la izquierda es la correspondiente al citado ejemplo).



Universidad de Oviedo
Área de Expresión Gráfica en la Ingeniería
Abril-Mayo de 2000
E.T.S.I.I.I.G. Campus de Gijón

EJERCICIO

8

Diseño gráfico de páginas *web*

TITULO: Colores y texto con *CSS*.

TEMA(S): Capítulo 11: Colores y fondos, fuentes de letra, características del texto.

OBJETIVOS: Emplear el color y manipular las características del texto mediante hojas de estilo.

ENUNCIADO: Escribir el código *HTML* necesario para obtener un documento semejante al que se muestra en la figura.

Universidad de Oviedo
Área de Expresión Gráfica en la Ingeniería
Abril-Mayo de 2000
E.T.S.I.I.I.G. Campus de Gijón

EJERCICIO

9

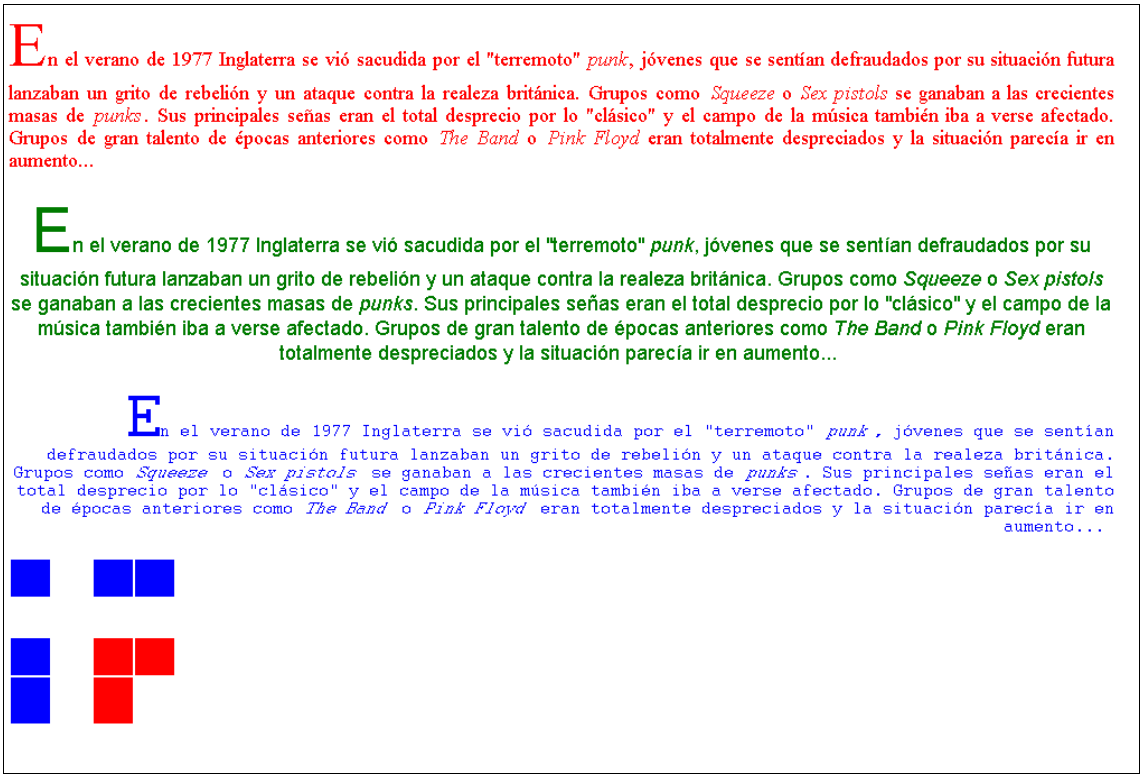
Diseño gráfico de páginas *web*

TITULO: Hojas de estilo y *HTML 4.0* estricto.

TEMA(S): Capítulo 13: Uso de *HTML 4.0* estricto.

OBJETIVOS: Manejo avanzado de texto (color, alineación y fuentes). Utilización del color en tablas.

ENUNCIADO: Escribir el código *HTML* y *CSS* necesario para obtener un documento semejante al que se muestra en la figura; el alumno debería basarse en la resolución del ejercicio 4 eliminando todas aquellas etiquetas correspondientes a *HTML 4.0* transitorio y proporcionando el código *CSS* necesario para conseguir el mismo efecto.



Universidad de Oviedo
Área de Expresión Gráfica en la Ingeniería
Abril-Mayo de 2000
E.T.S.I.I.I.G. Campus de Gijón

EJERCICIO

10

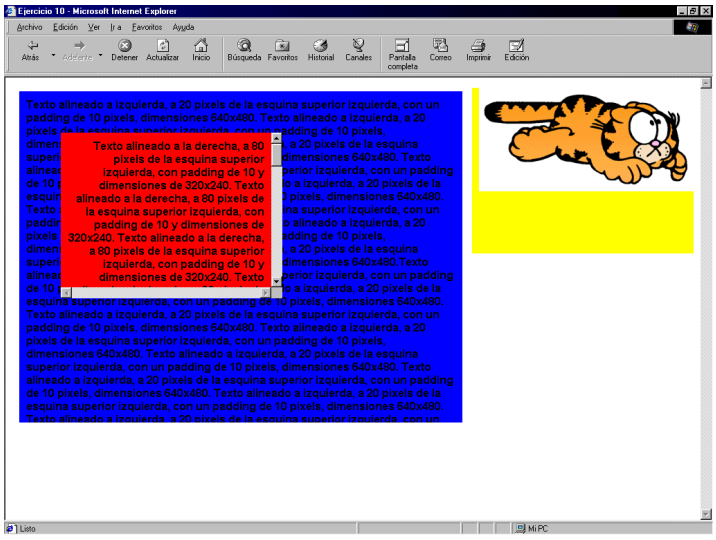
Diseño gráfico de páginas *web*

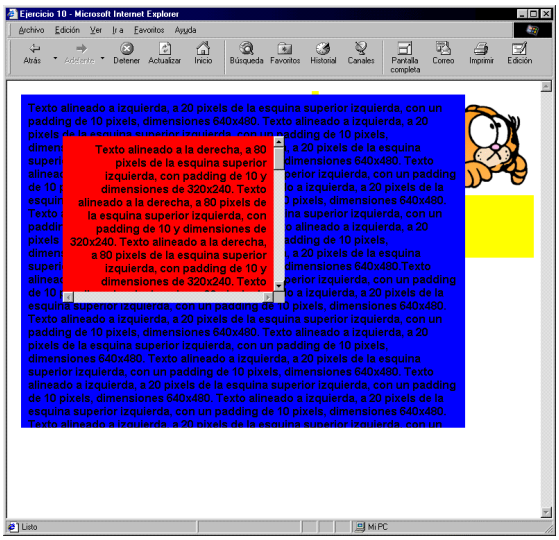
TITULO: Manejo de capas.

TEMA(S): Capítulo 12: El modelo de cajas, posicionamiento, uso de capas, controlando la visualización.

OBJETIVOS: Manejar conceptos relacionados con capas *CSS*: márgenes, posicionamiento, visualización...

ENUNCIADO: Escribir el código *HTML* y *CSS* necesario para obtener un documento con un comportamiento semejante al que se muestra en las figuras.





Universidad de Oviedo
Área de Expresión Gráfica en la Ingeniería
Abril-Mayo de 2000
E.T.S.I.I.I.G. Campus de Gijón

EJERCICIO

11

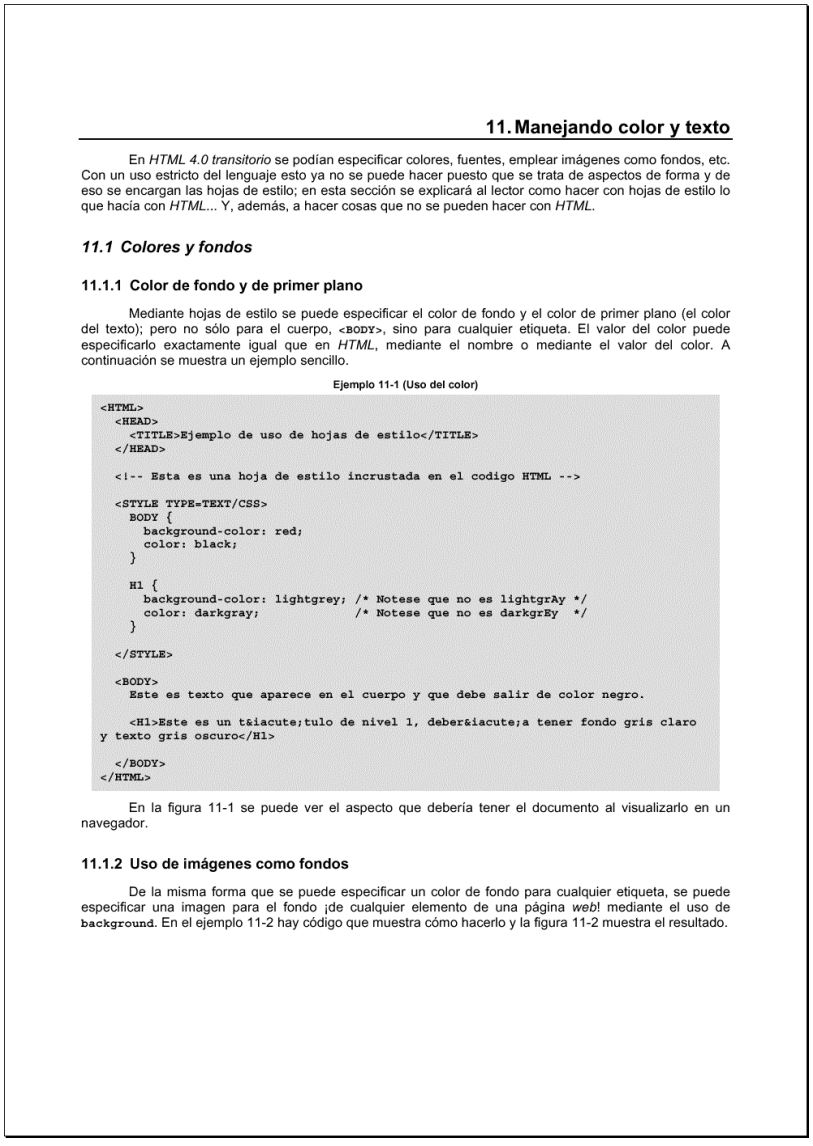
Diseño gráfico de páginas *web*

TITULO: *CSS*. Conceptos avanzados.

TEMA(S): Capítulo 13: Utilización de clases, herencia, pseudoclases.

OBJETIVOS: Emplear clases y herencia para formateo de textos; utilización de pseudoclases de enlaces y dinámicas.

ENUNCIADO: Escribir el código *HTML* y *CSS* necesario para obtener una página *web* lo más parecida posible al documento de la figura; añadir algunos enlaces con comportamiento dinámico.



11. Manejando color y texto

En *HTML 4.0 transitorio* se podían especificar colores, fuentes, emplear imágenes como fondos, etc. Con un uso estricto del lenguaje esto ya no se puede hacer puesto que se trata de aspectos de forma y de eso se encargan las hojas de estilo; en esta sección se explicará al lector como hacer con hojas de estilo lo que hacía con *HTML*... Y, además, a hacer cosas que no se pueden hacer con *HTML*.

11.1 Colores y fondos

11.1.1 Color de fondo y de primer plano

Mediante hojas de estilo se puede especificar el color de fondo y el color de primer plano (el color del texto); pero no sólo para el cuerpo, `<BODY>`, sino para cualquier etiqueta. El valor del color puede especificarlo exactamente igual que en *HTML*, mediante el nombre o mediante el valor del color. A continuación se muestra un ejemplo sencillo.

Ejemplo 11-1 (Uso del color)

```
<HTML>
  <HEAD>
    <TITLE>Ejemplo de uso de hojas de estilo</TITLE>
  </HEAD>

  <!-- Esta es una hoja de estilo incrustada en el codigo HTML -->

  <STYLE TYPE=TEXT/CSS>
    BODY {
      background-color: red;
      color: black;
    }

    H1 {
      background-color: lightgrey; /* Notese que no es lightgrAy */
      color: darkgray;             /* Notese que no es darkgrEy  */
    }

  </STYLE>

  <BODY>
    Este es texto que aparece en el cuerpo y que debe salir de color negro.

    <H1>Este es un t&iacute;tulo de nivel 1, deber&iacute;a tener fondo gris claro
y texto gris oscuro</H1>

  </BODY>
</HTML>
```

En la figura 11-1 se puede ver el aspecto que debería tener el documento al visualizarlo en un navegador.

11.1.2 Uso de imágenes como fondos

De la misma forma que se puede especificar un color de fondo para cualquier etiqueta, se puede especificar una imagen para el fondo ¡de cualquier elemento de una página *web*! mediante el uso de `background`. En el ejemplo 11-2 hay código que muestra cómo hacerlo y la figura 11-2 muestra el resultado.

Universidad de Oviedo
Área de Expresión Gráfica en la Ingeniería
Abril-Mayo de 2000
E.T.S.I.I.I.G. Campus de Gijón

EJERCICIO

12

Diseño gráfico de páginas *web*

TITULO: *JavaScript* (y I).

TEMA(S): Capítulo 14: Incrustando código *JavaScript* en un documento *HTML*, valores, variables y literales, expresiones y operadores.

OBJETIVOS: Aproximarse al lenguaje *JavaScript* practicando conceptos tales como variables, operadores y expresiones.

ENUNCIADO: Escribir el código *JavaScript* (nada de *HTML*) para obtener una página *web* con un contenido lo más parecido posible al que se muestra en la figura; para ello se habrán de utilizar una serie de variables.

NOTA: Para obtener resultados visibles en el navegador emplear la función **`document.write()`**; por ejemplo: **`document.write("<p>Hola</p>")`**, envía dicho código *HTML* al navegador para su visualización.

Hola mundo!

3 + 2 = 5
3 - 2 = 1
3 * 2 = 6
3 / 2 = 1.5

3 += 2 es 5
5 -= 2 es 3
3 *= 2 es 6
6 /= 2 es 3

!true = false
!false = true

true && true = true
true && false = false
false && true = false
false && false = false

true || true = true
true || false = true
false || true = true
false || false = false

Universidad de Oviedo
Área de Expresión Gráfica en la Ingeniería
Abril-Mayo de 2000
E.T.S.I.I.I.G. Campus de Gijón

EJERCICIO

13

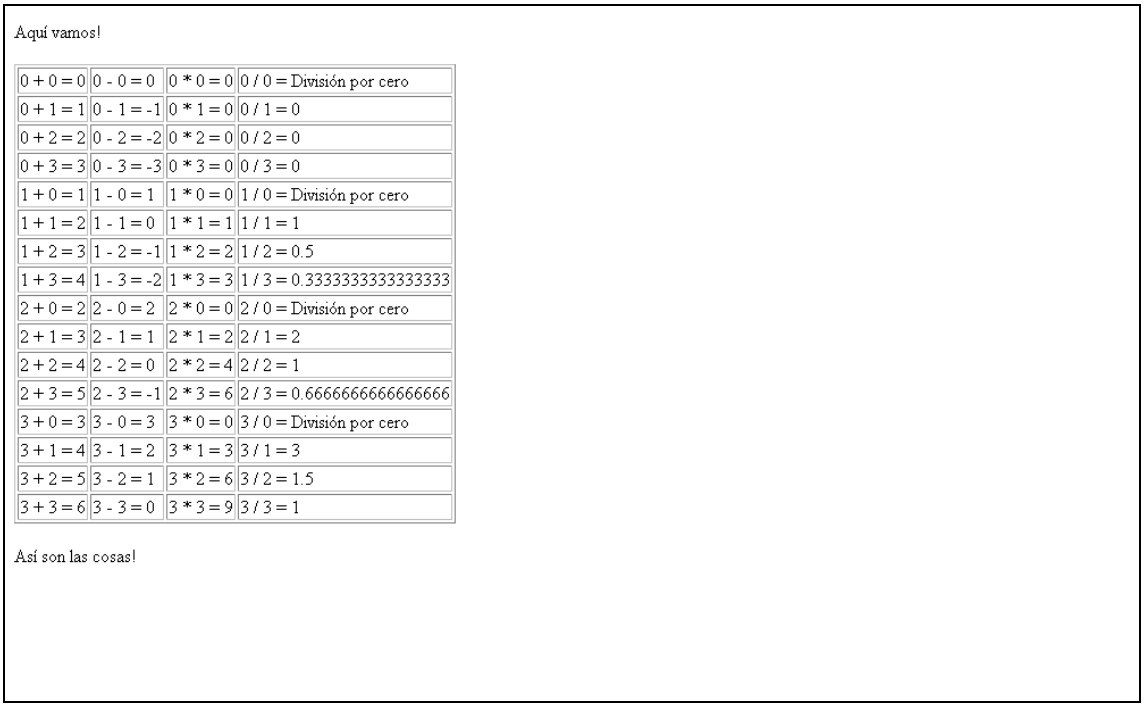
Diseño gráfico de páginas *web*

TITULO: *JavaScript* (y II).

TEMA(S): Capítulo 14: Sentencias, funciones.

OBJETIVOS: Utilización de funciones y estructuras de control.

ENUNCIADO: Escribir el código *JavaScript* (nada de *HTML*) para obtener una página *web* con un contenido lo más parecido al que se muestra en la figura; para ello se habrán de utilizar únicamente dos funciones y algunas estructuras de control.



Aquí vamos!

0 + 0 = 0	0 - 0 = 0	0 * 0 = 0	0 / 0 = División por cero
0 + 1 = 1	0 - 1 = -1	0 * 1 = 0	0 / 1 = 0
0 + 2 = 2	0 - 2 = -2	0 * 2 = 0	0 / 2 = 0
0 + 3 = 3	0 - 3 = -3	0 * 3 = 0	0 / 3 = 0
1 + 0 = 1	1 - 0 = 1	1 * 0 = 0	1 / 0 = División por cero
1 + 1 = 2	1 - 1 = 0	1 * 1 = 1	1 / 1 = 1
1 + 2 = 3	1 - 2 = -1	1 * 2 = 2	1 / 2 = 0.5
1 + 3 = 4	1 - 3 = -2	1 * 3 = 3	1 / 3 = 0.3333333333333333
2 + 0 = 2	2 - 0 = 2	2 * 0 = 0	2 / 0 = División por cero
2 + 1 = 3	2 - 1 = 1	2 * 1 = 2	2 / 1 = 2
2 + 2 = 4	2 - 2 = 0	2 * 2 = 4	2 / 2 = 1
2 + 3 = 5	2 - 3 = -1	2 * 3 = 6	2 / 3 = 0.6666666666666666
3 + 0 = 3	3 - 0 = 3	3 * 0 = 0	3 / 0 = División por cero
3 + 1 = 4	3 - 1 = 2	3 * 1 = 3	3 / 1 = 3
3 + 2 = 5	3 - 2 = 1	3 * 2 = 6	3 / 2 = 1.5
3 + 3 = 6	3 - 3 = 0	3 * 3 = 9	3 / 3 = 1

Así son las cosas!

Universidad de Oviedo
Área de Expresión Gráfica en la Ingeniería
Abril-Mayo de 2000
E.T.S.I.I.I.G. Campus de Gijón

EJERCICIO

14

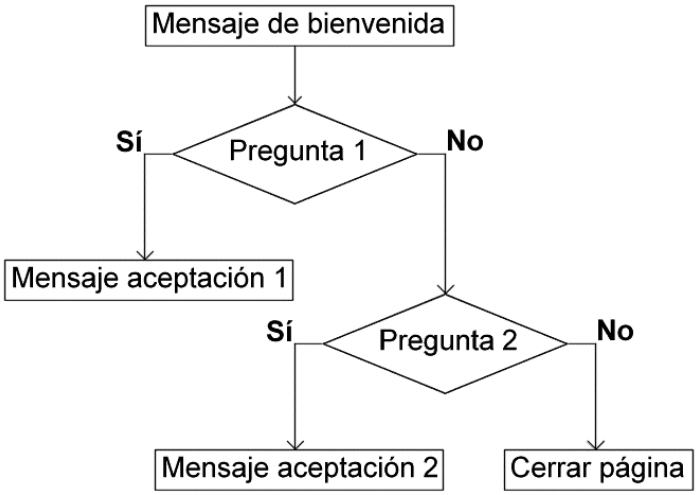
Diseño gráfico de páginas *web*

TITULO: *JavaScript* en el *web*.

TEMA(S): Capítulo 14: objetos predefinidos, empleo de eventos.

OBJETIVOS: Emplear de forma adecuada los principales objetos y manejadores de eventos disponibles en un navegador *web*.

ENUNCIADO: Escribir el código *HTML* y *JavaScript* necesario para obtener una página que responda ante los eventos de carga, "descarga" y sobre los enlaces; el comportamiento para la carga se muestra en la figura. Las acciones para todos los eventos se mostrarán como mensajes para el usuario.



Universidad de Oviedo
Área de Expresión Gráfica en la Ingeniería
Abril-Mayo de 2000
E.T.S.I.I.I.G. Campus de Gijón

EJERCICIO

15

Diseño gráfico de páginas *web*

TITULO: *HTML* dinámico (y I).

TEMA(S): Capítulo 15: detección del navegador y características de la plataforma, código genérico multinavegador, creando y manejando capas.

OBJETIVOS: Desarrollar la estructura de una página *web* dinámica mediante *HTML*, *CSS* y *JavaScript*.

ENUNCIADO: Aplicar todo lo visto hasta el momento para construir una página *web* con tres capas. La primera de ellas tendrá dos botones gráficos, el primero servirá para lanzar una animación de capas mientras el segundo para iniciar una animación de imágenes; esta primera capa será visible mientras las otras dos serán, inicialmente, invisibles. La capa que se empleará para animación de capas podrá tener cualquier contenido y poseerá al menos un botón gráfico para poder cerrarla y volver a la capa inicial. La capa de la animación de imágenes poseerá también un botón gráfico análogo y otra imagen para su animación.

Universidad de Oviedo
Área de Expresión Gráfica en la Ingeniería
Abril-Mayo de 2000
E.T.S.I.I.I.G. Campus de Gijón

EJERCICIO

16

Diseño gráfico de páginas *web*

TITULO: *HTML* dinámico (y II).

TEMA(S): Capítulo 15: precarga de imágenes y animación.

OBJETIVOS: Implementar el código *JavaScript* necesario para que la página escrita en el ejercicio anterior realice la precarga de imágenes, animación de capas y de imágenes.

ENUNCIADO: Partiendo del ejercicio anterior implementar el código *JavaScript* que permita realizar la precarga de las imágenes, la animación de una de las capas, una animación de imágenes en otra capa y la gestión de la demostración.