

HTML. Manual de Referencia

Esta es una versión preliminar del manual en un solo documento, está previsto añadir algunos ejemplos, definir mejor la estructura y corregir los posibles errores.

El documento ha sido realizado por *Sergio Talens Oliag* a partir del capítulo doce del libro sobre Internet que estoy preparando junto a *José Hernández Orallo*, de próxima aparición en la Editorial **Paraninfo**.

Encontrará la última versión del manual en el URL: <http://www.ivia.es/htmlref/>.

Si tiene algún comentario mande un mensaje a sto@ivia.es.

Última modificación: 24/05/1996

Contenidos

- [HTML 2.0](#)
 - [Generalidades](#)
 - [Componentes Léxicos](#)
 - [Caracteres](#)
 - [Marcas](#)
 - [Nombres](#)
 - [Atributos](#)

- [Comentarios](#)
- [Identificación del Nivel HTML de un Documento](#)
- [Estructura de los Documentos](#)
 - [Cabecera](#)
 - [Cuerpo](#)
 - [Encabezados](#)
 - [Definición de Bloques](#)
 - [Listas](#)
 - [Marcado Lógico de Frases](#)
 - [Marcado Tipográfico de Frases](#)
 - [Marcado de Anclaje de Hiperenlaces](#)
 - [Imágenes](#)
- [Juegos de Caracteres de los Documentos](#)
- [Hiperenlaces \(Hyperlinks\)](#)
 - [Activación de Hiperenlaces](#)
 - [Presentación de las Imágenes](#)
 - [Mapas](#)
 - [Identificadores de Fragmentos](#)
 - [Preguntas e Índices](#)
- [Formularios \(Forms\)](#)
 - [Elementos de un Formulario](#)
 - [Envío de Formularios](#)
- [Extensiones del HTML](#)
 - [Propuestas del Borrador del HTML 3.0](#)
 - [Distinción de los Documentos](#)
 - [Nuevos Elementos de la Cabecera <HEAD>](#)
 - [Nuevos Elementos del Cuerpo <BODY>](#)
 - [Nuevos Atributos para los Saltos de Línea
](#)
 - [Marcado Lógico y Tipográfico de Frases](#)
 - [Tablas](#)
 - [División de Bloques <DIV>](#)

- [Extensiones de Netscape y Microsoft](#)
 - [Atributos para Elementos ya Existentes](#)
 - [Nuevos Elementos](#)
 - [Enlaces relacionados con el HTML](#)
 - [Información sobre el estándar](#)
 - [Manuales y guías de estilo \(inglés\)](#)
 - [Manuales y guías de estilo \(castellano\)](#)
 - [Documentación sobre Microsoft y Netscape](#)
-

HTML 2.0

El **HTML** no es más que una aplicación del **SGML** (*Standard Generalized Markup Language*), un sistema para definir tipos de documentos estructurados y lenguajes de marcas para representar esos mismos documentos. El término **HTML** se suele referir a ambas cosas, tanto al tipo de documento como al lenguaje de marcas.

En realidad aún no existe un estándar del **HTML** en Internet, ya que existen tres revisiones o niveles de estandarización que aún no han sido aceptadas: la versión 1.0, que en realidad no existe como estándar ya que nunca se hizo una especificación formal, la 2.0, que es un estándar de hecho, aunque todavía está en la etapa de propuesta de estándar documentada en el RFC-1866 de noviembre de 1995. El siguiente nivel, el HTML 3.0, está en una etapa experimental, aunque muchos de los visores aceptan algunas de las extensiones propuestas.

En este apartado comentaremos la especificación del HTML 2.0 tal y como se describe en el RFC-1866. En el punto siguiente hablaremos de algunas de las propuestas de ampliación para el HTML 3.0 y algunas extensiones soportadas por algunos de los visores.

Generalidades

Antes de comenzar, comentaremos algo de la terminología necesaria para el resto de la descripción.

La definición del **HTML** en **SGML** incluida en el estándar resulta demasiado compleja para tratarla aquí, ya que es una descripción formal basada

en la teoría de lenguajes. Nos limitaremos a una aproximación informal al lenguaje, aunque siguiendo el esquema empleado en el estándar.

Dividiremos la descripción del lenguaje en varias partes:

- *Descripción de los componentes léxicos del HTML.* Donde se indica el formato de los caracteres, marcas, nombres, atributos y comentarios.
- *Estructura de los documentos.* Hablaremos de las marcas empleadas para definir las partes del documento, las estructuras de bloque y lista, las marcas de formato de párrafos y algunas marcas especiales que no entran en las categorías anteriores.
- *Caracteres, palabras y párrafos.* Juegos de caracteres aceptados en el HTML y el marcado alternativo para que el texto sea sólo ASCII de 7 bits, además de describir cómo se tratan las palabras y los párrafos.
- *Soporte de hipertexto.* Además de las marcas de formato necesitamos marcas para definir relaciones entre distintos documentos (e incluso entre partes de un mismo documento). Para ello, el **HTML** define una serie de marcas que denominaremos *hiperenlaces*. Comentaremos su sintaxis y funcionamiento en detalle.
- *Formularios.* Nuevos en el HTML 2.0, permiten la interacción del usuario y el servidor mediante la definición de plantillas de formulario, que el usuario completa y envía al servidor para su proceso. Según el propósito del formulario el cliente recibirá algún tipo de respuesta.

[\[contenidos\]](#)[\[sección\]](#)

Componentes Léxicos

Comenzamos nuestra descripción del **HTML** definiendo los componentes léxicos del lenguaje, es decir, las distintas entidades o elementos que pueden emplearse en los documentos.

Caracteres

Cada documento escrito en **HTML** puede emplear un juego de caracteres distinto, como veremos más adelante. De cualquier modo, todas las marcas se pueden escribir usando el **ISO-646**, el mismo juego de caracteres aceptado por los lectores de correo.

Cualquier cadena de caracteres imprimibles que no represente un marcado se representa literalmente, aunque los espacios y tabuladores se reducen a un solo carácter cuando no están dentro de un bloque preformateado.

Para reducir los documentos a ASCII de 7 bits y representar los caracteres empleados para marcar el texto se definen dos mecanismos de referencia:

- *Referencia por nombre*: el carácter se representa con un & seguido del nombre del carácter y un punto y coma, por ejemplo & ; será el carácter & y < ; el carácter < .
- *Referencia numérica*: en lugar de dar la referencia por nombre se escribe # seguido del número de carácter en el código de caracteres seleccionado, por ejemplo & ; será el carácter & y el < ; el carácter < .

Aunque en algunos casos se puede omitir el punto y coma final, es recomendable ponerlo para evitar errores. De igual forma, el carácter & se representa a sí mismo si no va seguido de # o de una letra, aunque siempre es preferible usar el código por la misma razón que antes.

Marcas

Las marcas delimitan elementos de un documento como cabeceras, párrafos, etc. La mayoría de marcas constan de una marca inicial, que da el nombre y atributos del elemento, seguida del contenido y una marca final.

Las marcas iniciales se escriben entre los símbolos "<" y ">" (menor y mayor) y las finales entre "</" y ">" (menor barra y mayor). Por ejemplo, <H1>Contenido</H1>, indica que "Contenido" es una cabecera de nivel uno.

Algunos elementos sólo tienen una marca inicial (por ejemplo la marca <HR> que representa una línea horizontal) y otros, aunque disponen de ambas, se suelen expresar sólo con la inicial.

El contenido de un elemento es una secuencia de cadenas de caracteres y puede incluir elementos anidados, excepto en el caso de los anclajes, que no pueden incluir otros elementos (pero si estar incluidos en otros).

Nombres

Los nombres consisten en una letra seguida de letras, dígitos, puntos o guiones. Los ejemplos H1 y HR anteriores son ejemplos de nombres. La longitud de un nombre esta limitada a 72 caracteres en la definición del HTML. Los nombres de elementos y atributos no distinguen entre mayúsculas y minúsculas, pero los nombres de entidades (la representación alternativa de los caracteres) sí.

En las marcas, el nombre del elemento debe comenzar inmediatamente después del < .

Atributos

Cuando una marca inicial admite atributos, éstos se escriben a continuación del nombre del elemento. Generalmente los atributos tienen la forma nombre, signo igual, valor del atributo aunque en algunos casos basta con el nombre del atributo. Se pueden poner espacios en blanco antes y después del signo igual.

El valor de un atributo puede ser:

- Una cadena de caracteres entre comillas (simples o dobles) que no contenga el símbolo de fin de marca ">".
- Un nombre como los definidos en el apartado anterior.

La longitud del valor de un atributo no puede superar los 1024 caracteres.

Comentarios

Para incluir comentarios en HTML se emplea la declaración de comentarios. Una declaración de comentarios comienza con <!, le siguen uno o varios comentarios y termina con >. Cada comentario comienza con -- e incluye todo el texto hasta la siguiente aparición de -- . Dentro de una declaración de comentarios, se pueden poner espacios en blanco después de cada uno de ellos, pero no antes del primero. Toda la declaración del comentario se ignora.

Identificación del Nivel HTML de un Documento

Para identificar un documento como HTML que sigue el estándar 2.0, cada documento debe comenzar con la siguiente declaración:

```
<!DOCTYPE HTML PUBLIC "-//IT&Eacute;F//DTD HTML 2.0//EN">
```

Existen más identificadores que especifican otros detalles, por ejemplo si el documento contiene formularios. Este mecanismo también puede ser empleado por los visores para reconocer otros tipos de documentos.

[\[contenidos\]](#)[\[sección\]](#)

Estructura de los Documentos

Los documentos en formato **HTML** son un conjunto de elementos anidados. En el nivel más alto nos encontramos el elemento **HTML** (marca inicial <HTML> y final </HTML>) que consta de dos partes: cabecera y cuerpo.

La cabecera se emplea para proporcionar información acerca del documento, mientras que el cuerpo contiene el texto de la página, es decir, la información que se va a presentar al usuario. En el cuerpo se pueden incluir todo tipo de elementos y marcas.

No es mala idea diseñarse una *plantilla* para desarrollar siempre a partir de ella. Además del esqueleto de la página (marcas de texto HTML, cabecera y cuerpo) podemos incluir en ella otras informaciones que siempre queramos que aparezcan (tanto comentarios como elementos del cuerpo o la cabecera). Prácticamente todos los editores específicos permiten el uso de plantillas y macros para introducir información útil, como por ejemplo la fecha de la última modificación. La siguiente página puede ser un buen punto de partida:

```
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<!-- Comentario sobre el autor -->
<HTML>
<HEAD>
<TITLE></TITLE>
<!-- Datos adicionales -->
</HEAD>
<BODY>
<!-- Datos cabecera pagina -->
<HR>
<HR>
<!-- Datos pie de pagina -->
```

```
</BODY>  
</HTML>
```

Los datos adicionales dependerán del autor. La cabecera y pie pueden ser útiles para dar a todas las páginas el mismo aspecto, aunque dependerán de que la persona que escribe las páginas lo considere necesario.

Entraremos ahora en la descripción de los elementos que pueden aparecer en la cabecera y el cuerpo.

Cabecera

La cabecera es una colección de información acerca del documento. Las marcas de principio y fin son `<HEAD>` y `</HEAD>`.

La cabecera puede contener los siguientes datos, sin importar el orden en que aparezcan:

- *Título* (TITLE). Indica el nombre del documento, los visores lo emplean generalmente como etiqueta de la ventana. Este campo es el único obligatorio en la cabecera.
- *Dirección de base* (BASE). Proporciona una dirección de base para interpretar los enlaces relativos cuando el documento se lee fuera de su contexto (por ejemplo cuando está guardado en un disco).
- *El documento es un índice* (ISINDEX). Si se pone la palabra clave ISINDEX, el cliente interpretará que la página es un índice y permitirá al usuario introducir palabras clave para buscarlas.
- *Enlaces relacionados* (LINK). En la cabecera podemos incluir varios enlaces relacionados con el documento como versiones anteriores, dirección del autor, etc.
- *Metainformación* (META). Este campo sirve para proporcionar información sobre el documento que no pueda ser expresada en los campos anteriores. La información se escribe usando pares nombre/valor que pueden ser empleados para dos propósitos:
 1. Proporcionar datos al servidor de HTTP para que genere campos de cabecera (como por ejemplo la fecha de caducidad de un documento que se actualiza periódicamente) o,
 2. Para que un usuario incluya información adicional sobre el documento, como palabras clave o el nombre del autor.

Para la primera función se emplean atributos del tipo HTTP-EQUIV y para la segunda los del tipo NAME. En ambos casos, el valor se asigna en el campo CONTENT. Un ejemplo del primer caso sería:

```
<META HTTP-EQUIV = "Expires" CONTENT="Dec 1996" >
```

Y del segundo:

```
<META NAME="Autor" CONTENT="Plo+Serg" >
```

El atributo NAME se refiere a nombres elegidos al azar por el usuario, mientras que HTTP-EQUIV significa que el valor tiene una cabecera equivalente real en el protocolo HTTP.

- *Siguiente Identificador* (NEXTID). En la actualidad este campo ya no se usa, lo empleaban los editores de HTML para asignar nombres a documentos de forma automática.

Cuerpo

Como ya hemos dicho, el cuerpo contiene el texto de la página que se va a presentar al usuario. Las marcas de principio y fin del cuerpo son <BODY> y </BODY> respectivamente.

A continuación describiremos los elementos que pueden aparecer en el cuerpo, clasificados por categorías.

Encabezados

Los encabezados se emplean para dividir los documentos en secciones, o más concretamente para marcar los títulos de esas secciones. Las marcas son del tipo <H#>Título</H#>, donde # puede ser un número cualquiera entre 1 y 6.

Aunque el estándar no lo especifica, es recomendable usar los niveles en orden, es decir, después de un encabezado de nivel uno deberemos usar encabezados de nivel dos para las subsecciones y no saltar directamente al tres o al cuatro, por ejemplo.

La representación de los encabezados depende del visor, generalmente se representan en negrita y van cambiando de tamaño y espacios antes y

después, de más a menos, en función del nivel (el nivel uno es el mayor).

Ejemplo:

```
<H1>Don Quijote de la Mancha</H1>
<H2>Capítulo 1</H2>
<P>En un lugar de la Mancha de cuyo nombre no quiero acordarme ...
```

Definición de Bloques

Para definir y separar bloques de texto se emplea una serie de marcas que definen párrafos, texto preformateado o bloques con un significado especial como direcciones o citas. También, y aunque no son propiamente para definir bloques, hablamos en este punto de dos marcas *especiales*, una para representar saltos de línea y otra que inserta una línea horizontal. Ambas permiten dividir el texto, por lo que las hemos incluido aquí.

Las marcas de bloque son:

- `<P>` para separar párrafos. Dado que para el HTML todo el texto es continuo, necesitamos algún mecanismo para indicar el principio y fin de un párrafo. Las marcas inicial y final son `<P>` y `</P>`, aunque generalmente sólo se emplea la inicial, terminando el mismo cuando encontramos cualquier elemento que cause un salto de línea.
- `<PRE>` para texto preformateado. Esta marca se emplea para texto escrito en tipo de letra de caja fija (mono-espaciada) y dentro del bloque marcado los saltos de línea sí son respetados. Dentro de estos elementos se pueden emplear anclajes y marcado tipográfico, pero no elementos que definan formato de párrafo (como marcas de párrafo, encabezados, etc.).
El elemento acepta el atributo opcional `WIDTH`, que indica el máximo número de caracteres por línea para que el visor ajuste el tamaño y tabulación del texto.
Además de la marca `PRE` existen dos elementos muy similares a él: `EXP` (ejemplo) y `LISTING` (listado), que no permiten ningún tipo de elemento anidado. Dado que con `PRE` podemos obtener el mismo resultado, no es recomendable el uso de estas marcas.
- `<ADDRESS>`, empleada para indicar que un texto representa una dirección o una firma. Generalmente se representa en cursiva y puede estar tabulado.
- `<BLOCKQUOTE>`, que indica que un texto es una cita de otra fuente. Se suele representar con tabulaciones a izquierda y derecha y en cursiva. En sistemas que no permiten representar cursivas se puede emplear algún tipo de símbolo al principio de las líneas, de manera similar a lo que se hace en las réplicas o citas (*quote*) del correo electrónico.
- `
`. Este elemento sólo tiene marca inicial y se usa para que el visor termine la línea en el punto en el que encuentre el salto.

- `<HR>`. Al igual que la anterior, sólo consta de una marca inicial. Se emplea para presentar una línea horizontal en el texto, ya sea usando caracteres o un gráfico, dependiendo del visor.
-

Listas

En realidad, también son marcas que permiten definir bloques, pero con características especiales. Las listas se emplean para presentar de forma ordenada una serie de líneas.

En función de su carácter lógico se distinguen los siguientes tipos de lista:

- Lista desordenada, `` (*Unordered List*).
- Lista ordenada, `` (*Ordered List*).
- Directorio, `<DIR>` (*Directory*).
- Menú, `<MENU>` (*Menu*).
- Lista de definición, `<DL>` (*Definition List*).

Exceptuando las listas de definición, el marcado de las líneas es igual en todos los casos: poniendo `` para marcar el principio de cada una, la línea termina cuando aparece un nuevo símbolo `` o se cierra la lista.

Para las listas de definición se emplean las marcas `<DT>` (*Definition Term*) y `<DD>` (*Definition Data*) para cada término y su correspondiente definición. Se pueden poner varios términos antes de una definición (marcas `<DT>`), pero no dos definiciones para un solo término.

Un ejemplo sería:

```
Esto es una lista desordenada:
```

```
<UL>  
<LI>Primer elemento  
<LI>Segundo elemento  
</UL>
```

```
Esto es una definici&oacute;n:
```

```
<DL>  
<DT>Perro  
<DD>Animal al que el hombre tiene la fea costumbre de morder  
</DL>
```

Marcado Lógico de Frases

Existen multitud de marcas para indicar que una palabra o frase tiene una connotación especial.

Los elementos son:

- `<CITE>`, indica que el texto es una cita (por ejemplo de un título). Generalmente se representa en cursiva.
- `<CODE>`, lo marcado es un ejemplo de código dentro del texto. Se representa usando un tipo de letra de caja fija.
- ``, denota énfasis. Generalmente se representa con letra cursiva.
- `<KBD>`, indica que el texto se introduce desde el teclado. Se representa usando un tipo de letra de caja fija.
- `<SAMP>`, ejemplo, es decir, una secuencia de caracteres literales. Se representa usando un tipo de letra de caja fija.
- ``, denota énfasis fuerte. Generalmente en negrita.
- `<VAR>`, lugar de una variable, por ejemplo la que se le pasa a un programa. Se representa usando un tipo de letra de caja fija.

Marcado Tipográfico de Frases

Se usan para indicar explícitamente el formato tipográfico de una palabra o frase.

Los formatos y sus marcas correspondientes son:

- Negrita, ``
- Cursiva, `<I></I>`
- Texto de teletipo (tipo de letra de caja fija) `<TT></TT>`

Aunque no están en el estándar, algunos visores pueden soportar otras marcas de formato tipográfico como `<STRIKE>` (texto tachado) o `<U>` (subrayado).

Marcado de Anclaje de Hiperenlaces

Un caso especial de marcado es el representado por el elemento `<A>`, que se emplea para indicar que un texto hace referencia a otro, es decir, está anclado mediante un hiperenlace.

Para el marcado se emplean la marca inicial con atributos, el texto a anclar y la marca final. Los atributos pueden ser:

- **HREF**. El valor es el URI (*Uniform Resource Identifier*) del anclaje principal de un hiperenlace. El URI se puede referir a otro documento HTML, a un anclaje del mismo documento o a cualquier otro tipo de recurso (**FTP**, **Gopher**, etc.).
- **NAME**. Nombra un anclaje para poder ser usado como anclaje principal de un hiperenlace, es decir, el punto nombrado puede ser referenciado desde otro anclaje que emplee el atributo **HREF**. Por ejemplo, si en el documento `test.html` incluimos un anclaje `Índice`, dentro del mismo lo podremos referenciar mediante `Volver al Índice`. Si queremos referirnos a ese punto dentro del documento desde otro documento que se encuentra en el mismo directorio, la referencia será `Índice del test`.
- **TITLE**. Sugiere un título para el recurso destino. Este atributo es sólo informativo y puede ser usado para que se visualice al colocarnos sobre el enlace o para nombrar recursos que no incluyen un título, como gráficos.
- **REL**. relaciones descritas por el hiperenlace. El valor es una lista de nombres de relaciones separadas por blancos.
- **REV**. Igual que **REL** pero en dirección contraria (si *A* se relaciona con *B* por *X*, un enlace de *A* a *B* con **REL**="X" expresa lo mismo que un enlace de *B* a *A* con **REV**="X").
- **URN**. Especifica un identificador para el primer anclaje del hiperenlace. La sintaxis de los URN (*Uniform Resource Namer*) `a_n` no está especificada.
- **METHODS**. Especifica métodos a usar para acceder al destino. Se escriben como una lista de palabras separadas por espacios y dependen del tipo de URI. Al igual que el elemento **TITLE**, son sólo orientativos para el visor.

Imágenes

Para incluir imágenes en documentos HTML se emplea la marca . Esta marca puede tener los siguientes atributos:

- SRC. Indica la fuente de la imagen, en realidad se trata de un enlace con el documento que contiene la imagen. En la práctica se suelen emplear solo imágenes en formatos de mapa de bits como gif o jpeg.
- ALT. Indica un nombre para referirnos a la imagen si ésta no se representa. Generalmente lo emplean los visores de solo texto o los visores gráficos cuando el usuario selecciona no cargar automáticamente las imágenes. Es el nombre que aparece habitualmente cuando interrumpimos la carga de una página o ésta se corta dejando imágenes sin traer.
- ALIGN. Alineación de la imagen respecto al texto, puede tomar los valores: TOP (arriba), MIDDLE (en medio) o BOTTOM (abajo). En general, si no se especifica, los visores asumen BOTTOM.
- ISMAP. Indica que la imagen es un mapa (lo veremos más adelante)

El único atributo imprescindible es el SRC (como es lógico, sin la imagen poca utilidad tiene la marca).

Una página con varias referencias a imágenes podría ser la siguiente:

```
<HTML>
<HEAD>
<TITLE>Página de prueba de imágenes</TITLE>
</HEAD>
<BODY>
<IMG SRC="foto.gif" ALT="Foto" ALIGN=MIDDLE>
Este soy yo.
<P>Selecciona lo que quieras de este mapa:
<A HREF="/cgi-bin/imagemap/mapa"><IMG SRC="mapa.gif" ISMAP></A>
</BODY>
</HTML>
```

[\[contenidos\]](#)[\[sección\]](#)

Juegos de Caracteres de los Documentos

Como mínimo todos los visores deben soportar todos los caracteres gráficos del alfabeto latino definido en el ISO Latin-1 (ISO 8859-1), que permiten escribir textos en la mayoría de los idiomas occidentales.

De los caracteres de control, sólo están permitidos tres: tabulador, salto de línea y retorno de carro (códigos 9, 10 y 13 respectivamente).

Como muchos sistemas tienen distintos juegos de caracteres ASCII, se han definido dos mecanismos para representar los caracteres especiales usando solamente el ASCII de 7 bits: el uso de referencias numéricas y una tabla de nombres (mnemotécnicos) para algunos de ellos.

En la siguiente tabla damos los caracteres, su número y nombre:

Entidades HTML 2.0

Caracter	Código	Descripción	Nombre
--	� - 	Sin usar	--
			Tabulador horizontal	--
	
	Salto de línea	--
--	 - 	Sin usar	--
		Retorno de Carro	--
--	 - 	Sin usar	--
	 	Espacio	--
!	!	Exclamación	--
"	"	Dobles comillas	--
#	#	Signo de número	--
\$	$	Dolar	--
%	%	Tanto por ciento	--
&	&	Ampersand	--

'	'	Apóstrofe	--
((Paréntesis izquierdo	--
))	Paréntesis derecho	--
*	*	Asterisco	--
+	+	Signo más	--
,	,	Coma	--
-	-	Guión	--
.	.	Punto (fin de párrafo)	--
/	/	Barra de división	--
0 - 9	0 - 9	Dígitos del al 0-9	--
:	:	Dos puntos	--
;	;	Punto y coma	--
<	<	Menor	--
=	=	Igual	--
>	>	Mayor	--
?	?	Cerrar interrogación	--
@	@	Arroba (en)	--
A - Z	A - Z	Letras A-Z	--
[[Abrir corchete (izquierdo)	--
\	\	Barra de división inversa	--
]]	Cerrar corchete (derecho)	--
^	^	Circunflejo	--
_	_	Subrayado	--
`	`	Acento agudo	--
a - z	a - z	Letras a-z	--
{	{	Abrir llave (derecha)	--

	|	Barra vertical	--
}	}	Cerrar llave (izquierda)	--
~	~	Tilde	--
--	 - Ÿ	Sin usar	--
	 	Espacio sin separación	nbsp *
¡	¡	Cerrar Exclamación	iexcl *
¢	¢	Centavo	cent *
£	£	Libra Esterlina	pound *
¤	¤	Signo de divisa general, <i>General currency sign</i>	curren *
¥	¥	Yen	yen *
	¦	Barra vertical partida	brvbar *
§	§	Sección	sect *
¨	¨	Diéresis	uml *
©	©	Copyright	copy *
ª	ª	Género femenino	ordf *
«	«	Doble menor (abrir comillas francesas o anguladas, <i>angle quotation mark</i>)	laquo *
¬	¬	No (símbolo lógico)	not *
-	­	Guión débil (<i>soft hyphen</i>)	shy *
®	®	Registrado	reg *
ˉ	¯	Macrón	macr *
°	°	Grados	deg *
±	±	Más o menos	plusmn *
²	²	Dos superíndice	sup2 *
³	³	Tres superíndice	sup3 *
´	´	Acento agudo	acute *
μ	µ	Micro	micro *

¶	¶	Fin de párrafo	para *
·	·	Punto medio	middot *
¸	¸	Cedilla	cedil *
¹	¹	Uno superíndice	sup1 *
º	º	Género masculino	ordm *
»	»	Doble mayor (cerrar comillas francesas o anguladas, <i>angle quotation mark</i>)	raquo *
¼	¼	Un cuarto	frac14 *
½	½	Mitad	frac12 *
¾	¾	Tres cuartos	frac34 *
¿	¿	Abrir interrogación	iquest *
À	À	A mayúscula, acento grave	Agrave
Á	Á	A mayúscula, acento agudo	Aacute
Â	Â	A mayúscula, acento circunflejo	Acirc
Ã	Ã	A mayúscula, tilde	Atilde
Ä	Ä	A mayúscula, diéresis	Auml
Å	Å	A mayúscula, anillo	Aring
Æ	Æ	Diptongo AE mayúscula (ligadura)	AElig
Ç	Ç	C cedilla mayúscula	Ccedil
È	È	E mayúscula, acento grave	Egrave
É	É	E mayúscula, acento agudo	Eacute
Ê	Ê	E mayúscula, acento circunflejo	Ecirc
Ë	Ë	E mayúscula, diéresis	Euml
Ì	Ì	I mayúscula, acento grave	Igrave
Í	Í	I mayúscula, acento agudo	Iacute
Î	Î	I mayúscula, acento circunflejo	Icirc
Ï	Ï	I mayúscula, diéresis	Iuml

Ð	Ð	Eth mayúscula, Islandesa Mayúscula	ETH
Ñ	Ñ	Eñe mayúscula	Ntilde
Ò	Ò	O mayúscula, acento grave	Ograve
Ó	Ó	O mayúscula, acento agudo	Oacute
Ô	Ô	O mayúscula, acento circunflejo	Ocirc
Õ	Õ	O mayúscula, tilde	Otilde
Ö	Ö	O mayúscula, diéresis	Ouml
×	×	Signo de multiplicación	times *
Ø	Ø	O barrada mayúscula	Oslash
Û	Ù	U mayúscula, acento grave	Ugrave
Ú	Ú	U mayúscula, acento agudo	Uacute
Û	Û	U mayúscula, acento circunflejo	Ucirc
Ü	Ü	U mayúscula, diéresis	Uuml
Ý	Ý	Y mayúscula, acento agudo	Yacute
Þ	Þ	THORN islandesa mayúscula	THORN
ß	ß	Beta minúscula	szlig
à	à	a minúscula, acento grave	agrave
á	á	a minúscula, acento agudo	aacute
â	â	a minúscula, acento circunflejo	acirc
ã	ã	a minúscula, tilde	atilde
ä	ä	a minúscula, diéresis	auml
å	å	a minúscula, anillo	aring
æ	æ	Diptongo ae minúscula (ligadura)	aelig
ç	ç	c cedilla minúscula	ccedil
è	è	e minúscula, acento grave	egrave
é	é	e minúscula, acento agudo	eacute

ê	ê	e minúscula, acento circunflejo	ecirc
ë	ë	e minúscula, diéresis	euml
ì	ì	i minúscula, acento grave	igrave
í	í	i minúscula, acento agudo	iacute
î	î	i minúscula, acento circunflejo	icirc
ï	ï	i minúscula, diéresis	iuml
ð	ð	eth islandesa minúscula	eth
ñ	ñ	eñe minúscula	ntilde
ò	ò	o minúscula, acento grave	ograve
ó	ó	o minúscula, acento agudo	oacute
ô	ô	o minúscula, acento circunflejo	ocirc
õ	õ	o minúscula, tilde	otilde
ö	ö	o minúscula, diéresis	ouml
÷	÷	Signo de división	divide *
ø	ø	o barrada minúscula	oslash
ù	ù	u minúscula, acento grave	ugrave
ú	ú	u minúscula, acento agudo	uacute
û	û	u minúscula, acento circunflejo	ucirc
ü	ü	u minúscula, diéresis	uuml
ý	ý	y minúscula, acento agudo	yacute
þ	þ	thorn islandesa minúscula	thorn
ÿ	ÿ	y minúscula, diéresis	yuml

Todos los nombres con asterisco (*) son propuestos en el HTML 2.0, pero no están aceptados generalmente.

Hiperenlaces (Hyperlinks)

Como ya hemos comentado, además de elementos para definir la presentación de los documentos, el HTML incluye herramientas para expresar hiperenlaces, es decir, relaciones entre dos anclajes, que se denominan cabeza y cola del hiperenlace. El anclaje de cola es el que "apunta" a la cabeza, es decir, el que empleamos para acceder a ella.

Hasta ahora hemos visto anclajes a partes del propio documento o en otro documento de la misma dirección, pero en general, los anclajes se identifican mediante una dirección mucho más completa. En el caso del HTML esta dirección es un **URI** (*Uniform Resource Identifier*) absoluto, seguido opcionalmente por una almohadilla (#) y una secuencia de caracteres, denominada *identificador de fragmento*.

En la dirección de un anclaje, el URI se refiere a un recurso; este recurso puede ser cualquier tipo de entidad (como páginas HTML) y ser obtenido usando distintos protocolos (HTTP para páginas HTML). El *identificador de fragmento* se referirá a alguna vista o porción del recurso (por ejemplo, una sección dentro de una página HTML).

Los siguientes marcados del HTML indican el anclaje de cola de un hiperenlace (o conjunto de ellos):

- <A> (si usan HREF)
- <LINK>
-
- <INPUT> (si tienen el atributo SRC)
- <ISINDEX>
- <FORM> (con método GET)

Todos ellos hacen referencia a anclados cabeza mediante un **URI**, ya sea absoluto o relativo, con o sin identificador de fragmento.

En el caso de tener un **URI** relativo, el **URI** absoluto se obtiene combinando el **URI** base absoluto del documento con el relativo. El **URI** base es el identificado en el elemento <BASE>, o sino existe el del documento actual.

Una vez calculada la dirección del recurso, el visor debe obtenerlo para presentárselo al usuario. Por ejemplo, si el **URI** base es `http://www/alice/` y el documento contiene la marca ``, el visor emplea el **URI** `http://www/img/logo.gif` para obtener la imagen.

Activación de Hiperenlaces

El visor de HTML permite al usuario "navegar" por el contenido del documento y solicitar la activación de hiperenlaces representados con elementos de tipo `<A>` y, opcionalmente, los de tipo `<LINK>` .

Para activar un enlace, el cliente obtiene una representación del recurso identificado por la dirección del anclaje, si lo que obtenemos es otro documento HTML, la posibilidad de navegar comienza de nuevo con él.

Presentación de las Imágenes

Los hiperenlaces de elementos del tipo `` e `<INPUT>` se suelen obtener a la vez que se procesa el documento, es decir, los enlaces a imágenes se procesan sin necesidad de que el usuario lo solicite, de modo que se pueden mostrar dentro de la representación del documento HTML, en el lugar en el que aparecen referenciados, es decir, donde esté el elemento `` o `<INPUT>` .

Los hiperenlaces de tipo `<LINK>` también pueden ser procesados sin la intervención del usuario; si, por ejemplo, se refieren a hojas de estilo, pueden ser procesados antes o durante el proceso del documento.

Mapas

Cuando en un elemento `` aparece el atributo `ISMAP`, el elemento `` debe estar dentro de un anclaje de tipo `HREF`. Esta estructura representa un conjunto de hiperenlaces.

Por ejemplo:

```
<a href="http://www/cgi-bin/imagemap"></a>
```

Si el usuario selecciona alguno de los enlaces marcando un píxel de la imagen, el visor calcula la dirección del recurso añadiendo al **URI** dado en el

elemento `<A>` un interrogante `?` y las coordenadas `x` e `y` del pixel.

En el ejemplo anterior, si el usuario selecciona la esquina superior izquierda, el URI seleccionado será `http://www/cgi-bin/imap?0,0`.

Identificadores de Fragmentos

En un hipertexto cualquier palabra precedida del carácter `#` es un identificador de fragmento. En particular, una dirección de la forma `#sec` se refiere a un anclaje dentro del mismo documento.

El significado de los identificadores de fragmento depende del tipo de documento. Para documentos del tipo `text/html`, se refiere a un elemento `<A>` con un atributo `NAME` cuyo valor es igual al del identificador de fragmento (sin la almohadilla), como hemos visto en algún ejemplo anterior. Los nombres deben ser exactamente iguales, ya que se distingue entre mayúsculas y minúsculas y los nombres dentro de los documentos deben ser únicos (no podemos nombrar dos secciones con el mismo identificador). El visor indica dónde está el nombre desplazándose hasta el anclaje y/o resaltándolo.

Preguntas e Índices

El elemento `<ISINDEX>` representa un conjunto de hiperenlaces. El usuario puede elegir entre ellos proporcionando palabras clave al visor. El visor compone el **URI** del recurso añadiendo un interrogante `?` y las palabras al **URI** base. los caracteres especiales se sustituyen por secuencias de escape y si hay varias palabras se unen empleando el símbolo `+`. Por ejemplo, si un documento contiene:

```
<BASE HREF="http://www/indice"> <ISINDEX>
```

y el usuario introduce las palabras `niño` y `libro`, entonces el visor accederá al recurso `http://www/indice?niño+libro`.

La forma de introducir los datos depende del tipo de visor, algunos presentan un cuadro especial y otros muestran en la página un mensaje y un recuadro para rellenar.

Los elementos <FORM> que emplean el atributo METHOD=GET también se refieren a conjuntos de atributos, como veremos más adelante.

[\[contenidos\]](#)[\[sección\]](#)

Formularios (Forms)

Entramos ahora en la descripción del elemento más novedoso del HTML 2.0 respecto a las versiones anteriores: los *formularios*.

Un formulario es una plantilla para representar un conjunto de datos, el método de enviarlos y el URI de la acción asociada (referencia al programa que va a realizar el proceso de los datos).

El conjunto de datos resultante después de la edición de los campos por el usuario se emplea para acceder a un servicio de información, en función del método y la acción asociada especificados.

El conjunto de datos es una secuencia de campos con pares nombre/valor. Los nombres se especifican en los atributos NAME de los elementos de entrada del formulario y los valores toman un valor inicial empleando distintos marcados, que luego pueden ser editados por el usuario.

Los formularios pueden mezclarse con elementos de definición de bloques, por ejemplo un elemento <PRE> puede contener un <FORM> y al revés, un <FORM> puede contener listas. Esto permite gran flexibilidad a la hora de diseñar el aspecto de los formularios.

Es importante señalar que la utilidad de los formularios está limitada al uso de las páginas junto con servidores (o al menos con acceso a la red, ya que también se pueden enviar por correo electrónico), ya que las acciones asociadas son programas (generalmente *scripts* de CGI). Estos programas deben funcionar en un servidor (al que se le proporcionan los datos del formulario, para ser procesados), aunque hay algunos navegadores capaces de invocar guiones locales.

Elementos de un Formulario

Dentro de un formulario podemos encontrar los siguientes elementos:

- Declaración del formulario (FORM)
- Campos de entrada (INPUT)
- Campo de selección (SELECT)
- Área de texto (TEXTAREA)

A continuación vamos a describir cada uno de los elementos y sus atributos correspondientes.

Declaración del Formulario (FORM)

La declaración del formulario se pone entre las marcas <FORM> y </FORM>.

En su interior aparecen una secuencia de elementos de entrada (*input elements*), junto con elementos de marcado de estructura del documento.

En la definición del formulario se pueden incluir los siguientes atributos:

- ACTION. Especifica el **URI** de la acción asociada al formulario. Si no se especifica, por defecto se asume que el **URI** es el BASE del documento.
- METHOD. Indica el método de acceso al **URI** de la acción. El conjunto de métodos aplicables es función del esquema del **URI**. Se pueden emplear los métodos GET y POST, que se describirán más adelante.
- ENCTYPE. Especifica el tipo de codificación para el transporte de los pares nombre/valor, excepto en los casos en los que el protocolo no imponga uno. Trataremos este tema en el punto referido a la codificación de los formularios.

Campo de Entrada (INPUT)

El elemento <INPUT> representa un campo de entrada de datos. Los atributos posibles del elemento vienen dados por el valor del atributo TYPE, que determina el tipo de entrada.

Los tipos de entrada son:

- Texto: INPUT TYPE=TEXT.

Valor por defecto del atributo TYPE, indica que la entrada es una sola línea. Necesariamente los elementos de este tipo deben incluir el atributo NAME, que indica el nombre del campo.

Como atributos opcionales puede tomar:

- MAXLENGTH, que limita el máximo número de caracteres que pueden ser introducidos en el campo. Si el valor es mayor que el del atributo SIZE, el visor debe permitir el desplazamiento de la línea. El número de caracteres por defecto es ilimitado.
- SIZE, que especifica la cantidad de espacio reservada para este campo. El valor por defecto depende del visor.
- VALUE, que indica el valor inicial del campo.

Ejemplo:

```
Calle: <input name=calle><br>
Número: <input name=numero><br>
Código postal: <input name=cp size=5 maxlength=5 value="99999"><br>
```

- *Password*: INPUT TYPE=PASSWORD.

Es un campo de texto como el anterior, pero el valor no se ve al escribirlo.

Ejemplo:

```
Clave de usuario: <input name=login><br>
Contraseña: <input type=password name=passwd>
```

- Caja de selección: INPUT TYPE=CHECKBOX.

Representa una opción booleana (si o no). Un conjunto de varios elementos de este tipo con el mismo nombre representan un campo de selección múltiple (n de muchos).

Los elementos de este tipo requieren los atributos NAME y VALUE, que indican el nombre del elemento o grupo de elementos y la parte del valor del campo aportada por el elemento, respectivamente.

Opcionalmente podemos incluir el atributo CHECKED, que indica que el estado inicial es seleccionado.

Ejemplo:

```

Qu&eacute; bebidas le gustan:


```

- Botón: INPUT TYPE=RADIO.

Representa una opción booleana (si o no). Un conjunto de varios elementos de este tipo con el mismo nombre representan un campo de selección múltiple, 1 de muchos.

Los elementos de este tipo requieren, al igual que en el caso anterior, los atributos NAME y VALUE.

Opcionalmente podemos incluir el atributo CHECKED, que indica que el estado inicial es seleccionado. En cualquier momento sólo uno de los botones de un conjunto está marcado. Si ninguno de los elementos <INPUT> de un conjunto de botones de tipo *radio* especifica CHECKED, el visor debe marcar el primero de ellos inicialmente.

Ejemplo:

```

Qu&eacute; bebida prefiere:


```

- Píxel de una imagen: INPUT TYPE=IMAGE.

Especifica una imagen para que la muestre el visor y permite la entrada de dos campos, las coordenadas x e y de un píxel seleccionado de la misma. Los nombres de los campos son iguales al del campo, añadiendo al final .x e .y respectivamente. Este tipo implica también TYPE=SUBMIT, es decir, cuando un seleccionamos un píxel, se envía todo el formulario.

Los atributos NAME y SRC son necesarios y el campo ALIGN es opcional (al igual que en el elemento).

Representa una opción de entrada (generalmente mediante un botón) que le indica al cliente que debe reiniciar los valores de sus campos a los que tenían inicialmente. El atributo `VALUE`, si existe, indica la etiqueta a emplear para la entrada (botón).

Ejemplo:

```
Si se ha equivocado, pulse para volver a comenzar: <input type=reset>
```

Campo de Selección (SELECT)

El elemento `<SELECT>` se emplea para reducir el campo a una lista de valores.

Estos valores se presentan empleando elementos de tipo `<OPTION>`. Los atributos del elemento son:

- `MULTIPLE`. Indica que el valor puede incluir más de una opción.
- `NAME`. Especifica el nombre del campo.
- `SIZE`. Determina el número de ítems visibles. Si se indica tamaño uno, se suelen presentar como menús desplegables, mientras que si el tamaño es mayor se suelen presentar como lista con barra de desplazamiento.

Por ejemplo:

```
<SELECT NAME="bebida">  
<OPTION selected>Agua  
<OPTION>Cerveza  
<OPTION VALUE=refresco>Refresco Gaseoso  
<OPTION>Vino  
<OPTION>Zumoz  
</SELECT>
```

El elemento `<OPTION>` sólo puede aparecer dentro de un elemento `<SELECT>` y representa una posible elección. Puede tomar los siguientes atributos:

- `SELECTED`. Indica que esta opción está seleccionada inicialmente. Si ninguna opción tiene este atributo, el visor presenta la primera

seleccionada.

- VALUE. Indica el valor a retornar si se selecciona la opción. Si no se incluye el atributo, se emplea el contenido del elemento.

Área de Texto (TEXTAREA)

El elemento <TEXTAREA> representa un campo de texto de múltiples líneas. Los atributos posibles son:

- COLS. El número de columnas visibles del área de texto, en caracteres.
- NAME. Nombre del campo.
- ROWS. El número de líneas visibles del área de texto, en caracteres.

Por ejemplo:

```
<TEXTAREA NAME="direccion" ROWS=6 COLS=64>
Magallanes, 25 - 28015 MADRID
</TEXTAREA>
```

El contenido del elemento es el valor inicial del campo. La especificación de filas y columnas sólo se refiere a la dimensión del área visible, pero los programas cliente pueden permitir sobrepasar los límites mediante barras de desplazamiento. Generalmente se emplea un tipo de letra de caja fija para mostrar los contenidos del campo.

Envío de Formularios

Un visor de HTML comienza el proceso de un formulario presentando el documento con los campos en su estado inicial. Según el tipo de campo, el usuario puede modificar sus valores (seleccionando un campo, rellenando con texto, etc). Cuando ha terminado, puede enviarlo empleando un botón de envío o una selección de píxel en una imagen. En ese momento el visor analiza las entradas en función del método, acción y tipo de codificación y lo envía.

En caso de que el formulario sólo tenga un campo de entrada de texto de una línea, el visor debe aceptar una pulsación de la tecla de retorno de carro en ese campo como una petición de envío del formulario.

Tipo de Codificación de Formularios

La codificación por defecto de todos los formularios es, según el esquema **MIME**, `application/x-www-form-urlencoded`. Un conjunto de datos de formulario se representa en este caso del siguiente modo:

1. Los nombres de campos y los valores son preprocesados: los espacios son reemplazados por el símbolo `+`, y los caracteres son sustituidos como en los **URL**, es decir, los caracteres no alfanuméricos se representan con un signo de tanto por cien y dos dígitos hexadecimales que indican el código **ASCII** del carácter (`%HH`). Los saltos de línea (empleados en campos de múltiples líneas), se representan con pares `CRLF` (sustituidos por `%0D%0A`).
2. Los campos se listan en el orden en el que aparecen en el documento, con los nombres separados del valor por el símbolo `=` y los pares separados entre sí por el símbolo `&`. Los campos con valores nulos pueden ser omitidos, en particular, los campos no seleccionados en entradas booleanas no deben aparecer en los datos, pero los campos ocultos que tengan el atributo `VALUE` sí.

Formularios de Consulta: **METHOD=GET**

El método de consulta depende de los efectos que el formulario tenga en el estado del resto del mundo, es decir, si el envío va a producir cambios en cualquier documento o programa que no sea nuestro visor.

Si el proceso del formulario es idempotente (no produce cambios), el método debe ser `GET`. Un ejemplo de este tipo de formularios son las consultas a bases de datos, que no tienen efectos laterales visibles.

Para procesar un formulario cuyo **URL** de acción es un **URL** de tipo **HTTP** y el método es `GET`, el visor genera un **URI** que comienza con el de la acción al que se le añade un interrogante (`?`) y el conjunto de datos codificado con el formato `application/x-www-form-urlencoded` visto en el punto anterior. Para acceder a la consulta el visor accede al **URI** de la misma manera que lo hace con los que aparecen en los anclajes.

De todos modos, en algunos casos, la codificación de los datos puede generar un **URI** extremadamente largo, lo que puede provocar un funcionamiento erróneo con algunos servidores de **HTTP** antiguos. Por esta razón, algunos formularios que no tienen efectos laterales, se escriben usando el método `POST`.

Formularios con Efectos Laterales: **METHOD=POST**

Para formularios con efectos laterales (como uno que modifique una base de datos) se emplea el método `POST`.

Para procesar un formulario cuyo **URL** de acción es de tipo **HTTP** y el método es `POST`, el visor gestiona una transacción de tipo `POST` del

protocolo HTTP, usando el **URI** de la acción y el cuerpo de un mensaje de tipo `application/x-www-form-urlencoded` como antes. El visor debe presentar la respuesta del HTTP POST de la misma forma que la respuesta obtenida con el método GET.

Ejemplo de Envío de Formularios

Llegados a este punto, se hace necesario mostrar un pequeño ejemplo para clarificar lo anterior. Si tenemos el siguiente documento:

```
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<title>Ejemplo de envío de formularios HTML</title>
<H1>Cuestionario de Personal</H1>
<P>Por favor, rellene el siguiente cuestionario:
<FORM METHOD="POST" ACTION="/cgi-bin/post-query">
<P>Nombre: <INPUT NAME="nombre" size="48">
<P>Hombre <INPUT NAME="genero" TYPE=RADIO VALUE="hombre">
<P>Mujer <INPUT NAME="genero" TYPE=RADIO VALUE="mujer">
<P>Número de miembros de la familia: <INPUT NAME="familia" TYPE=text>
<P>Idiomas que conoce:
<UL>
<LI>Francés <INPUT NAME="idioma" TYPE=checkbox VALUE="Frances">
<LI>Inglés <INPUT NAME="idioma" TYPE=checkbox VALUE="Ingles">
<LI>Otros
<TEXTAREA NAME="otros" cols=48 rows=4></textarea>
</UL>
<P> Pulse aquí para enviar los datos <INPUT TYPE=SUBMIT>
<P> Puede volver a comenzar en cualquier momento pulsando aquí <INPUT TYPE=RESET>
</FORM>
```

El estado inicial de los datos del formulario es:

```
nombre " "
genero "hombre"
familia " "
otros " "
```

Hay que señalar que la entrada de tipo RADIO tiene valor inicial, mientras que el de tipo CHECKBOX no.

El usuario rellena los campos y solicita el envío. Supongamos que los valores son:

```
nombre "Alicia Lindell"  
genero "mujer"  
familia "4"  
idioma "ingles"  
otros "catalan\neuskera\ngallego"
```

Entonces el visor gestiona una transacción HTTP POST usando el **URI** /cgi-bin/post-query. El cuerpo del mensaje será la siguiente línea:

```
nombre=Alicia+Lindell&genero=mujer&familia=4&idioma=ingles&otros=catalan%0D%0Aeuskera%0D%0Agallego
```

Si quiere probar el ejemplo pinche [aquí](#)

[\[contenidos\]](#)[\[sección\]](#)

Extensiones del HTML

Visores como el **Netscape** anuncian en su publicidad que son compatibles con el HTML 3.0. Esto es en realidad una simple estrategia publicitaria, ya que, como hemos mencionado, tal estándar ni siquiera existe. De hecho el único visor que realmente incorpora todas las propuestas es el **Arena**, ya que esta siendo desarrollado para probar la viabilidad de las mismas.

Veremos en este punto dos tipos de propuestas:

1. Las que aparecen en la primera versión del borrador (*draft*) del HTML-3.0 (soportadas por el visor **Arena**). Dentro de éstas encontramos algunas que soportan la mayoría de visores actuales.
2. Las extensiones de algunos visores (fundamentalmente los de **Microsoft** y **Netscape**) que, de momento, no se han incluido en el estándar.

En ambos casos las extensiones se han incorporado de dos maneras:

1. Incluyendo nuevos atributos a elementos ya existentes, y
2. Añadiendo elementos totalmente nuevos, con atributos y funciones propias.

Tanto unas como otras son ignoradas por los visores que no las soportan, aunque algunas de ellas son de un primer nivel (es decir, contienen información del documento, no sólo de formato), lo que hace que las páginas no tengan ningún sentido en estos últimos.

Por otro lado, las extensiones introducidas por los visores van a cuestionar mucho la utilidad del estándar en un futuro ya que, en muchos casos, no serán incluidas en la definición oficial del HTML, ya sea por reemplazar su utilidad mediante otros sistemas o no ser adecuadas dentro del modelo definido.

Propuestas del Borrador del HTML 3.0

Básicamente el borrador (a partir de unas extensiones denominadas en un principio HTML+, en parte incluidas en el HTML 2.0) amplía el estándar mediante las dos técnicas mencionadas anteriormente: nuevos atributos para elementos existentes y elementos nuevos.

El objetivo es, por un lado, conseguir un mayor control sobre el aspecto de los documentos y por otro, definir mecanismos para representar entidades no contempladas anteriormente.

Entre otras cosas se incluyen elementos para representar:

- *Tablas*. Se pueden presentar todo tipo de informaciones en forma tabular.
- *Figuras*. Mejora del elemento imagen, incluyen soporte para la gestión de mapas por parte del cliente y permite controlar el flujo de texto alrededor de las imágenes.
- *Ecuaciones*. Evita la necesidad de usar imágenes para la representación de fórmulas matemáticas.
- *Banners*. Se emplean para incluir una zona estática para logotipos, avisos o controles de navegación y búsqueda en las páginas.
- *Notas*. Tanto dentro del texto como a pie de página.
- *Hojas de estilos*. Incluye soporte para relacionar las páginas con las hojas de estilos, lo que permite un mayor control sobre el aspecto de los documentos.
- *Divisiones*. Permiten agrupar varios bloques de texto, dividiendo las páginas en distintas partes, lo que permite representar su estructura

lógica y definir formatos comunes para cada parte.

Y nuevos atributos para definir mejor el aspecto de las páginas como:

- Soporte para listas personalizadas.
- Tabuladores horizontales.
- Alineación horizontal de cabeceras y párrafos.
- Ampliaciones de los formularios (selecciones gráficas, nuevos campos y un atributo `SCRIPT` para incluir guiones para la gestión del formulario).

De cualquier modo, como ya hemos comentado, aparte del visor **Arena**, la mayoría de los visores no incluyen soporte para muchas de las extensiones, ya sea por haber definido mecanismos alternativos, por no estar suficientemente desarrolladas las propuestas o no ser prioritarias en la política de desarrollo de los visores.

De los *browsers* actuales, el que más características incorpora es el de **Netscape**, que incluso ha añadido atributos a algunas de las entidades propuestas.

Por todo ello, sólo comentaremos aquí los elementos y atributos en uso actualmente, ya que el resto será desechado o reemplazado, además de no ser útil en la actualidad, ya que no hay visores que los soporten.

Distinción de los Documentos

Para que los visores no se confundan se propone el empleo del tipo **MIME** `text/html; version=3.0` y la extensión `.html3` o `.ht3` para los documentos. Con este método, los *browsers* que no soportan el HTML 3.0 no intentan analizar los documentos y generalmente nos ofrecen la posibilidad de seleccionar una aplicación auxiliar o guardar el documento.

Por otro lado, se pueden desarrollar *scripts* para convertir el HTML 3.0 en 2.0, de modo que las páginas se puedan ver con visores nuevos y antiguos sin necesidad de escribir páginas distintas, basta que los servidores identifiquen el cliente y devuelvan la versión en HTML 2.0 o 3.0.

Nuevos Elementos de la Cabecera <HEAD>

El único elemento nuevo es el `STYLE`, que hace referencia a la hoja de estilos a emplear para visualizar el documento.

Por ejemplo:

```
<HEAD>
<STYLE HREF=" ../estilos/estilo1.css" >
...
</HEAD>
```

Indica el **URL** de la hoja de estilos que se debe cargar y usar. La extensión CSS se refiere a uno de los métodos en desarrollo para representar las hojas de estilos. Hace poco la **W3 Organization** ha llegado a un acuerdo con muchas de las empresas que desarrollan productos para la **WWW** para definir el estándar de las hojas de estilos, refinando la propuesta del formato *CSS (Cascading Style Sheets)*. De cualquier modo, en el momento de escribir estas páginas, todavía no se ha incorporado el soporte en los visores de uso generalizado.

Nuevos Elementos del Cuerpo `<BODY>`

Quizá son los que se han popularizado más rápidamente por impactar de forma más importante en el aspecto de las páginas.

Atributos para el Elemento `BODY`

La primera extensión notable es el atributo `BACKGROUND` dentro de la declaración del cuerpo. Esto permite especificar una imagen como fondo para las páginas. Ya existen múltiples visores que incorporan este atributo.

Alineación Horizontal de Encabezados y Párrafos

El HTML 3.0 propone un atributo `ALIGN` para encabezados y párrafos, que permite al autor especificar que tipo de alineación desea para el texto.

Tanto encabezados como párrafos permiten los valores `ALIGN="left"`, `ALIGN="center"` y `ALIGN="right"` para alinear a la izquierda, centrar o alinear a la derecha el texto respectivamente. Muchos visores soportan las alineaciones izquierda y centrado, mientras que la alineación derecha es soportada por muy pocos.

Para los párrafos el atributo `ALIGN` también puede tomar el valor `ALIGN="justify"` para presentar las líneas justificadas.

Otra característica añadida a los párrafos es el atributo `ID = "nombre"`, que permite marcar el párrafo para ser referenciado mediante un URL (en realidad viene a reemplazar las marcas del tipo ``). Esta característica es soportada por unos pocos visualizadores, como el `Emacs-w3`.

Nuevos Atributos para los Saltos de Línea `
`

Dado que en el HTML 3.0 el texto puede fluir alrededor de las imágenes, es necesario algún mecanismo para indicar en qué punto se debe cortar el texto para que continúe en la línea posterior a la imagen.

Esto se controla con el atributo `CLEAR` que puede tomar los valores `"left"`, `"right"` y `"all"`. `CLEAR="left"` provoca que la siguiente línea empiece tan pronto como el margen izquierdo esté libre, mientras que `CLEAR="right"` hace lo mismo pero respecto al margen derecho. `CLEAR="all"` no comienza una línea hasta que los dos márgenes están libres.

Marcado Lógico y Tipográfico de Frases

Para ampliar un poco las capacidades del control del aspecto de las letras, el HTML 3 incluye nuevas marcas tipográficas y lógicas. Las marcas son:

- U. Subrayado.
- BIG. Texto grande (respecto al tamaño normal).
- SMALL. Texto pequeño (respecto al tamaño normal).
- SUB. Subíndice.
- SUP. Superíndice.

Actualmente sólo unos pocos visualizadores las soportan todas.

Tablas

Las tablas del HTML están contenidas en elementos de tipo `<TABLE>`. El elemento define el rango de la tabla y sus propiedades. En la definición del borrador las tablas sólo tienen un atributo, `BORDER`, que indica que las tablas deben ser dibujadas con un borde alrededor y entre cada una de las celdas de la tabla. Si no se incluye el atributo, las tablas se dibujan sin borde.

Dentro de las tablas sólo se pueden incluir dos elementos, los de tipo `CAPTION` y `TR`.

`CAPTION` define una etiqueta para la tabla y sólo puede tomar el atributo `ALIGN`, para indicar la posición de la etiqueta respecto a la tabla. Los valores posibles son "top", "bottom", "left" y "right" (*arriba, abajo, izquierda y derecha*).

`TR` define una fila que contiene celdas del tipo `TD` (*Datos*) o `TH` (*Cabeceras*). El número de elementos `TD` o `TH` determina el número de columnas de la tabla y el de elementos `TR` el de filas.

En general, los elementos `TD` se usan para datos, mientras que los `TH` se emplean para las cabeceras de filas o columnas. Ni `TD` ni `TH` necesitan marcas finales.

`TR` puede tomar los atributos `ALIGN` y `VALIGN`, que determinan la alineación horizontal y vertical de las celdas respectivamente. `ALIGN` puede tomar los valores "left", "center" o "right" (*izquierda, centro o derecha*) y `VALIGN` los valores "top", "middle" o "bottom" (*arriba, en medio o abajo*). Los valores por defecto son `ALIGN="left"` y `VALIGN="middle"`.

Las celdas individuales definidas por `TD` y `TH` también pueden tomar atributos de alineación, que prevalecen sobre los dados por `TR`. Los valores por defecto para las celdas `TD` son `ALIGN="left"` y `VALIGN="middle"`, y `ALIGN="center"` y `VALIGN="middle"` para las `TH`.

Además de los de alineación, las celdas `TD` y `TH` pueden tomar los atributos `COLSPAN` y `ROWSPAN`, que permiten que una celda ocupe el espacio de varias, expandiéndose hacia la derecha (siguiente columna) o hacia abajo (siguiente fila). `COLSPAN` indica cuántas columnas (contando desde la derecha) son ocupadas por la celda y `ROWSPAN` indica cuántas filas (hacia abajo) se expande la celda.

Aunque no lo hemos dicho antes, cuando se crea una fila `TR` hay que asegurarse de que el número de celdas coincide con el de columnas (determinado por la primera fila). En ese cálculo deben incluirse las celdas que ocupan varias columnas (`COLSPAN`) o vienen de otras filas (`ROWSPAN`).

Por último, es interesante indicar que las tablas pueden contener tablas, es decir, cada celda de una tabla puede contener su propia tabla.

División de Bloques <DIV>

Este elemento permite agrupar varios bloques en uno solo. La ventaja es que el elemento DIV puede incluir el atributo ALIGN y todos los bloques (P, BLOCKQUOTE, etc.) dentro de la división heredarán la alineación especificada. Además, DIV también puede tomar el atributo CLASS, que permite especificar el significado semántico del bloque. El programa **Netscape Navigator 2.x** soporta el elemento DIV.

[\[contenidos\]](#)[\[sección\]](#)

Extensiones de Netscape y Microsoft

Quizás por la rápida comercialización de la Red y por el deseo de los diseñadores y usuarios de la telaraña de tener un mayor control sobre el aspecto de las páginas, los dos navegadores más populares **Netscape Navigator** y **Microsoft Explorer** han introducido extensiones propias al HTML, aunque, como hemos visto, han incorporado muchas de las propuestas del HTML 3.0 a sus visores.

En este caso comentaremos las extensiones separando entre los elementos nuevos y los atributos añadidos a los ya existentes.

Atributos para Elementos ya Existentes

Prácticamente todos los comentarios son propios del visor **Netscape**. Separaremos aquí entre las extensiones para elementos de la cabecera y para elementos del cuerpo.

Atributos para Elementos de la Cabecera (HEAD)

Dentro de los elementos de la cabecera se han incluido atributos para los elementos <ISINDEX> y <META> .

En el primero se ha incluido el atributo PROMPT, que permite al autor indicar que mensaje debe aparecer en la página antes del campo de entrada del índice. Si no se emplea el atributo el mensaje por defecto es:

```
This is a searchable index. Enter search keywords:
```

La extensión del segundo está relacionada con la actualización dinámica de los documentos, mediante el empleo del atributo HTTP-EQUIV con el valor "Refresh".

Por ejemplo, la siguiente cabecera hace que después de 19 segundos se acceda al **URL** especificado:

```
<META HTTP-EQUIV="Refresh" CONTENT="19; URL=http://www/19.html">
```

Mientras que una cabecera como:

```
<META HTTP-EQUIV="Refresh" CONTENT="10">
```

hace que el visor espere 10 segundos y vuelva a acceder al documento presentado actualmente.

Atributos para el Elemento BODY

Netscape ha introducido nuevos atributos para el elemento BODY, principalmente para indicar los colores del texto y el fondo de los documentos (además de soportar el uso del BACKGROUND del HTML 3.0):

- BGCOLOR="#rrggbb". Pone como color de fondo el correspondiente al valor **RGB** dado. **RR GG** y **BB** son valores hexadecimales para los niveles de Rojo, Verde y Azul, con valores entre 0 y 255 (es decir, de 00 a FF). El color "#000000" es negro, y el "#FFFFFF" es blanco. Si además de un color se incluye una imagen de fondo, el color BGCOLOR es el que se encuentra debajo de la misma (si la imagen es transparente, es el que se ve de fondo).
- TEXT="#rrggbb". Especifica que el texto debe representarse en el color **RGB** dado.
- LINK="#rrggbb". Establece el color de los anclajes de hiperenlaces.
- VLINK="#rrggbb". Establece el color de los anclajes de hiperenlaces que se han visitado recientemente (están en la memoria caché).

Es muy probable que todos ellos sean incorporados al estándar.

Atributos para la Línea Horizontal (<HR>)

Se han añadido cuatro atributos para permitir ajustar el aspecto de la línea horizontal. Los atributos son SIZE, WIDTH, ALIGN y NOSHADE:

- `<HR SIZE=núm>`. El atributo SIZE permite indicar qué grosor debe tener la línea.
- `<HR WIDTH=núm|tanto por cien>`. Por defecto la línea horizontal es tan ancha como la página. Con el atributo WIDTH se puede especificar el ancho exacto en píxeles o el tamaño relativo (en tanto por cien, por ejemplo WIDTH=80%) respecto al ancho de la página.
- `<HR ALIGN=left|right|center>`. Dado que, con el atributo WIDTH, las líneas no tienen porque coincidir con el ancho de la página, se hace necesario un mecanismo para especificar su alineación horizontal, en este caso con el atributo ALIGN, que puede tomar los valores izquierda, derecha y centro.
- `<HR NOSHADE>`. Este atributo especifica que la barra debe ser sólida, es decir, no debe tener efectos de sombra.

Hay que señalar que prácticamente todos estos atributos sólo tienen sentido para visores gráficos, e incluso algunos sólo en el **Netscape**, como el NOSHADE, ya que asumen una presentación con sombra por defecto de la línea que no tiene por que ser igual en todos los visores.

Atributos para las Listas

Básicamente se trata de atributos para controlar el aspecto de las marcas de cada línea de la lista y los números en las listas ordenadas.

Para la lista desordenada (``), se emplean por defecto marcas circulares en cada línea, que van cambiando conforme las listas se van anidando. En **Netscape** pasan de un disco sólido a un círculo o un cuadrado. El nuevo atributo TYPE permite especificar que tipo de símbolo queremos emplear en nuestras líneas independientemente del nivel de anidamiento: TYPE=disc, TYPE=circle o TYPE=square (*disco, círculo o cuadrado*).

Las listas ordenadas (``) siempre comienzan en 1 y van subiendo progresivamente. Se han añadido a este elemento dos atributos: TYPE y START. El primero permite indicar que letras se deben emplear para cada línea: letras mayúsculas (TYPE=A), letras minúsculas (TYPE=a), números romanos en mayúscula (TYPE=I), números romanos en minúscula (TYPE=i) o números (TYPE=1).

El atributo START permite especificar el número del primer ítem de la lista, para cuando queramos que comiencen en un valor distinto al uno. El orden se da siempre en número, y se presenta según el tipo especificado. Por ejemplo START=5 se mostraría como 'E', 'e', 'V', 'v', o '5' según el tipo.

Para dar aún mayor flexibilidad en las listas se han añadido también atributos al elemento ``. Por un lado, se ha añadido el atributo TYPE, que puede tomar los mismos valores que toma en la lista en la que se encuentra la línea. Cuando se especifica, cambia el tipo de lista para ese ítem y los siguientes.

Además, si la línea pertenece a una lista ordenada también se puede emplear el atributo `VALUE`, de manera que se puede modificar el número de cuenta para ese ítem y los siguientes.

Atributos para las Imágenes ()

Probablemente uno de los elementos con mayor número de cambios es la marca `IMG`. En realidad esto se debe a que se han incorporado muchas de las posibilidades del elemento `FIG` del borrador al elemento `IMG`, sin incorporar soporte para el primero.

En primer lugar se ha extendido el número de valores posibles de la alineación de las imágenes. Los valores posibles son: `left`, `right`, `top`, `texttop`, `middle`, `absmiddle`, `baseline`, `bottom` y `absbottom`.

Los dos primeros valores, `"left"` y `"right"` (izquierda y derecha), tienen características especiales, ya que se emplean para que las imágenes sean "flotantes".

Así, una imagen incluida con `` se colocará en el primer hueco disponible a partir del margen izquierdo, hacia abajo, y el texto subsiguiente se colocará a la derecha de la imagen. En el caso del alineamiento a la derecha (`ALIGN=right`), la imagen se coloca a la derecha y el texto a la izquierda.

La posibilidad de imágenes flotantes ha hecho que **Netscape** haya implementado el soporte para el atributo `CLEAR` en los saltos de línea (`
`), con el mismo funcionamiento que en la propuesta del estándar descrita anteriormente.

El resto de opciones son simplemente variaciones sobre las tres originales (`top`, `middle` y `bottom`), necesarias para determinar claramente las posiciones del texto respecto a las imágenes. Los valores y las alineaciones asociadas son:

- `ALIGN=top` alinea la imagen con el elemento más alto de la línea.
- `ALIGN=texttop` hace lo mismo que el elemento anterior pero sólo contemplando el texto más alto, sin considerar cualquier otro elemento. Generalmente esta marca tiene el mismo efecto que `TOP`, pero no siempre.
- `ALIGN=middle` alinea la línea base del texto con la mitad de la imagen.
- `ALIGN=absmiddle` alinea el punto medio vertical de la línea de texto con la mitad de la imagen.
- `ALIGN=bottom` alinea el inferior de la imagen con la línea base del texto.
- `ALIGN=baseline` es exactamente idéntico al anterior, simplemente existe por que los diseñadores de **Netscape** son más listos que nadie y enmiendan la plana al estándar añadiendo nuevos valores sólo por que les parece más adecuado el nuevo nombre.
- `ALIGN=absbottom` alinea el inferior de la imagen con el inferior de la línea.

En realidad, los nuevos valores sólo tienen sentido en el **Netscape Navigator**, ya que no todos los visores implementan igual las alineaciones de imágenes y textos, de hecho han incluido nuevos valores para no estropear las páginas ya hechas, pero en realidad hubiera bastado con modificar la gestión de los tres valores estándar en el visor. Además de los nuevos valores para ALIGN, **Netscape** incorpora varios atributos nuevos:

- ``. Los atributos WIDTH y HEIGHT (*anchura* y *altura*) se incluyen para aumentar la velocidad de visualización de las páginas, ya que si se especifican, el visor puede reservar el espacio antes de obtener la imagen, continuando con el resto del texto antes de traerla.
- ``. El atributo BORDER permite especificar el ancho del borde de las imágenes. Si se pone BORDER=0 las imágenes no van recuadradas.
- ``. VSPACE indica el espacio a reservar por encima y por debajo una imagen, mientras que HSPACE se refiere al espacio a reservar a derecha e izquierda de la misma. Esto es especialmente útil para las imágenes flotantes, ya que evita que el texto se quede demasiado pegado a las mismas.

Pinche [aquí](#) para ver un ejemplo de las alineaciones.

Extensiones para los Anclajes (TARGET)

En el HTML 2.0 se incluye el atributo TITLE para los anclajes, que permite nombrar los recursos antes de obtenerlos. Ese nombre se puede emplear para las ventanas en las que se presentan recursos que no tienen nombre.

En el **Netscape Navigator** se ha introducido algo similar (aunque no exactamente igual), el atributo TARGET, que nos da el nombre de la ventana del *Navegador* a emplear, de modo que cuando se pincha en el enlace, el documento aparece en una ventana que tiene ese nombre. Si la ventana no existe, se abre una nueva y se le asigna el nombre dado por TARGET. Generalmente el nombre no se ve (se emplea el del recurso obtenido), pero otros anclajes pueden hacer referencia a esa ventana y, al seleccionarlos, el visor los muestra en ella.

La sintaxis es:

```
<A HREF="url.html" TARGET="nom_ventana">Pinche aquí; para abrir otra ventana</A>
```

Además del atributo para los anclajes, se ha creado una marca BASE que permite indicar un nombre por defecto para cada enlace de un documento que no tiene el atributo TARGET. El formato es:

```
<BASE TARGET="ventana por defecto">
```

Los nombres de las ventanas deben comenzar por un carácter alfanumérico, si no son ignorados. De todos modos, existe una serie de nombres especiales que empiezan con el carácter subrayado (`_`):

- `TARGET= "_blank "`. Con este valor, el enlace siempre se cargará en una nueva ventana sin nombre.
- `TARGET= "_self "`. Con este valor el enlace se cargará en la misma ventana en la que se encuentra. Esto es útil para anular el efecto de una asignación global con `<BASE TARGET= " " >`.
- `TARGET= "_parent "`. Este valor hace que el enlace se cargue en el `FRAMESET` inmediatamente superior al documento actual. Veremos `FRAMESET` al hablar del elemento `FRAME`. Si el documento no tiene nada por encima el efecto es el mismo que con `"_self "`.
- `TARGET= "_top "`. Este valor hace que el enlace se cargue en el cuerpo de la ventana. Se comporta como `"_self "` si el documento ocupa toda la página, pero resulta muy útil para salir de un bloque de `FRAMES` anidadas.

La utilidad de este atributo está en que podemos hacer que el cliente abra distintas ventanas para cada enlace, sin dejar de tener nuestra página disponible; además de sus usos en el nuevo elemento `FRAME`, que comentaremos más adelante.

Extensiones para las Tablas

Respecto a las tablas definidas en el HTML 3.0, **Netscape** ha incluido algunos atributos, relacionados con el control del tamaño de las tablas y los bordes:

- `BORDER` puede tomar un valor, con lo que podemos especificar el ancho en píxeles del borde externo de la tabla.
- Dos nuevos atributos `CELLPADDING` y `CELLSPACING` también toman valores numéricos. `CELLPADDING` define el espacio en píxeles entre el contenido de las celdas y sus bordes, mientras `CELLSPACING` define el espacio entre celdas (el ancho de los bordes).
- Por último, el atributo `WIDTH` indica el ancho de la tabla. El tamaño se puede expresar con un valor absoluto en píxeles o como un porcentaje del ancho de la página (p. ej. `WIDTH=80%`).

Extensiones de los Formularios

Encontramos dos extensiones:

- El atributo `WRAP` en el elemento `TEXTAREA`, que permite controlar la manera de gestionar el flujo del texto dentro de las áreas de entrada de texto en formularios. `WRAP` puede tomar los valores `OFF`, `VIRTUAL` y `PHYSICAL`; en el primer caso las líneas se envían tal y como las introduce el usuario, en el segundo se parten para ajustarse a la caja, pero se envían como una sola línea sin caracteres de salto y en la última se hace lo mismo que en la segunda, pero sí que se envían los saltos.

- El atributo ENCTYPE dentro de la declaración de los formularios, que permite enviar ficheros a los servidores de HTTP, de modo que se pueden escribir formularios que soliciten al usuario el envío de un fichero.

Un ejemplo de este tipo de formularios sería:

```
<FORM ENCTYPE="multipart/form-data" ACTION="_URL_" METHOD=POST>
Enviar este archivo:
<INPUT NAME="userfile" TYPE="file">
<INPUT TYPE="submit" VALUE="Enviar">
</FORM>
```

Nuevos Elementos

Los nuevos elementos definidos por **Netscape** y **Microsoft** son los siguientes:

- NOBR/WBR. Permite indicar cómo se cortan las líneas en el visor.
- CENTER. Permite centrar bloques.
- FONT. Cambia tamaño y/o propiedades de los tipos de letra.
- BASEFONT. Indica el tipo de letra por defecto.
- EMBED. Permite la inclusión de cualquier tipo de objetos en una página HTML (sólo **Netscape Navigator 2.0** y posteriores).
- APPLET. *Java Applets* (sólo **Netscape Navigator 2.0** y posteriores).
- MARQUEE. Marquesina de texto (sólo **Microsoft Explorer 2.0** y posteriores).
- MAP. Mapa de selección dirigido por el cliente.
- FRAME. Divide en vistas (marcos) los documentos (**Netscape Navigator 2.0** y posteriores).
- SCRIPT. Programa de script del documento (**Netscape Navigator 2.0** y posteriores).

NOBR/WBR

El elemento NOBR viene de *NO BReak (Sin cortes)*. Esto quiere decir que el texto situado entre las marcas <NOBR> y </NOBR> no puede ser

representados con saltos de línea entre ellos. Aunque en algunos casos esta marca es necesaria, es recomendable controlar el uso de este elemento, ya que una línea larga dentro de un elemento NOBR puede tener un aspecto realmente extraño.

El elemento WBR viene de *Word Break (Partición de palabra)*. Este elemento es para el caso en el que tenemos una sección NOBR y sabemos en qué punto exacto queremos que se corte. El elemento sólo es informativo, es decir, no provoca el salto de línea (para eso está la marca BR), sólo le dice al visor que ese sería un buen lugar por donde cortar si hace falta.

CENTER

Todas las líneas de texto entre las marcas <CENTER> y </CENTER> se centran respecto a los márgenes izquierdo y derecho actuales.

El uso de la nueva marca en lugar de aprovechar el atributo ALIGN de los párrafos (<P align="center">) está motivado por que el uso de este último hace que muchos visores existentes fallen, además de ser mucho menos general y no soportar todos los casos en los que sería deseable el centrado. Esta marca está soportada por la mayoría de navegadores de última generación.

FONT

Con el nuevo elemento FONT se puede cambiar el tamaño de los tipos, usando la marca Texto. Los tamaños van de 1 a 7. Los valores dados en el atributo SIZE pueden tener un signo + o - delante, indicando un incremento o disminución del tamaño respecto al tamaño base de la página. El tamaño base por defecto es 3, aunque se puede cambiar con el elemento BASEFONT .

Además de soportar el elemento FONT, el visor de **Microsoft** soporta el atributo FACE para especificar el tipo de letra a emplear. Por ejemplo FACE="arial" indica que el tipo a emplear es el arial. Esta característica sólo es útil en **Windows**, ya que los nombres de los tipos se toman del Gestor de Tipos de ese sistema, por lo que es necesario saber los nombres y tener los tipos instalados para que esto funcione. Por tanto, en máquinas bajo el **MacOS** o alguna versión de **UNIX** no es posible sacar partido de esta facilidad.

Por último, con la aparición del **Netscape 2.0** se ha incluido un nuevo atributo al elemento FONT, COLOR, que permite especificar el valor RGB (de la misma manera que en el cuerpo de la página) empleado para mostrar un bloque de texto.

BASEFONT

Especifica el tamaño base de los tipos (atributo `SIZE`) para los cambios de tamaño relativos. Generalmente se pone al principio del cuerpo de la página. Por defecto se asume `<BASEFONT SIZE=3>`.

Hay que indicar que si ponemos `<BASEFONT SIZE=7>`, entonces `` no funcionará, ya que 7 es el máximo tamaño de los tipos. Lo mismo sucede con los decrementos de tamaño para un tamaño base de 1.

EMBED

El elemento `EMBED` permite la inserción de cualquier tipo de objetos directamente en una página HTML. Estos objetos son soportados por módulos específicos de los visores (*Netscape Plug-ins*). `EMBED` puede tomar todo tipo de atributos. A nivel general se han definido los siguientes:

- `SRC`. **URL** del objeto. Este atributo es obligatorio.
- `HEIGHT`. Altura del objeto.
- `WIDTH`. Anchura del objeto.

La imagen del objeto se escalará para encajar en el alto y ancho especificados.

APPLET

Netscape Navigator 2.0 y el **HOT JAVA Browser** soportan la inclusión de *JAVA Applets* (programas escritos en **Java**, que pueden ser incluidos y ejecutados en un documento HTML). Esta inclusión se realiza mediante el elemento `APPLET`.

Veamos un ejemplo:

```
<APPLET CODE="Blink.class" WIDTH=300 HEIGHT=100>
<PARAM name=lbl value="Este es un texto que se desplaza ... ">
<PARAM name=speed value="4">
</APPLET>
```

Aquí, CODE da el nombre de la aplicación a ejecutar, WIDTH y HEIGHT el espacio en píxeles que necesita y los elementos PARAM dentro de APPLET, los parámetros a pasar al programa. Es probable que en un futuro se reemplace el elemento APPLET por otro más genérico, que podría ser el EMBED comentado antes.

MARQUEE

Este elemento es soportado únicamente por el programa **Microsoft Internet Explorer 2.0** y se emplea para crear una marquesina de texto que se desplaza.

Por ejemplo:

```
<MARQUEE ALIGN="top">Texto que se desplaza ...</MARQUEE>
```

crea una marquesina con el texto desplazándose a través del marco. La utilidad de este elemento es relativa, ya que con la inclusión de los *JAVA Applets*, se puede obtener el mismo resultado con un programa que, además, puede hacer muchas otras cosas.

MAP

Una de las cosas que ha popularizado el uso del Web es el empleo de *Mapas de Selección (Image-maps)*. El uso más común es el de permitir a los usuarios acceder a documentos pinchando en distintas áreas de una imagen.

Pese a ser tan popular, la implementación actual tiene varias limitaciones, como ya hemos apuntado anteriormente:

1. Sólo funciona mediante el protocolo HTTP, haciéndolo inútil para leer documentos locales (en el disco duro, disquete o **CD-ROM**) o

- accedidos mediante otros protocolos.
2. Se hace necesaria una transacción con el servidor sólo para devolver un enlace, lo que puede requerir mucho tiempo si accedemos a uno distante.
 3. A diferencia de lo que sucede con los enlaces normales, no hay ningún medio para el programa cliente de dar información visual al usuario de a dónde va a saltar si pincha en una porción de la imagen antes de que lo haga.
 4. Por último, la implementación de los mapas de selección es dependiente del servidor, por lo que la portabilidad de los documentos es relativa.

Una posible solución sería la implementación del elemento `FIG` del borrador del HTML 3.0, pero no se ha considerado apropiada por varias razones:

1. El soporte completo del elemento `FIG` requiere un proceso adicional considerable por parte del visor.
2. El mapa no puede ser empleado en los visores que no soportan el elemento `FIG`.
3. Precisa que la descripción del mapa esté especificada cuando aparece el mapa, lo que no es apropiado en algunas aplicaciones.

La extensión propuesta resuelve todos estos problemas; por un lado se añade un nuevo elemento para describir los mapas (`MAP`) y por otro se añade un atributo al elemento `IMG` para indicar que se debe usar la descripción para gestionar el mapa (`USEMAP`).

Las regiones de cada imagen se describen usando el elemento `MAP`. Este elemento describe cada región de la imagen e indica a dónde apunta. El formato básico del elemento es:

```
<MAP NAME="nombre_mapa">
<AREA [SHAPE="figura"] COORDS="x,y,..."
[ HREF="referencia" | NOHREF]>
</MAP>
```

El atributo `NAME` indica el nombre del mapa, para poder referenciarlo desde los elementos `IMG` (es por lo tanto necesario).

En el elemento `AREA` se indica el tipo de figura (`SHAPE`), las coordenadas de la misma (`COORDS`) y la referencia a emplear cuando se selecciona un punto del área (`HREF` o `NOHREF`).

El tipo de figura puede ser `SHAPE="RECT"`, `SHAPE="POLY"`, `SHAPE="CIRCLE"` o `SHAPE="DEFAULT"` (rectángulo, polígono, círculo o por defecto). Si se omite el tipo de figura, se asume `RECT`.

El atributo `COORDS` da las coordenadas de la figura en píxeles y sus valores dependen del tipo de figura:

- Para las regiones rectangulares las coordenadas se dan de la forma "izquierda, arriba, derecha, abajo". La región definida incluye la esquina inferior derecha. Por ejemplo, para especificar el área total de una imagen de 100x100 píxeles las coordenadas serían "0,0,99,99").
- Para los polígonos se especifica una lista de puntos ("x1, y1, x2, y2, ..."). El visor cierra el polígono automáticamente.
- Los círculos se definen con un punto central y un radio (un total de tres valores, coordenadas x e y, y el valor del radio).

Por último se incluye el atributo `HREF` o `NOHREF`, el primero indicará a donde ir si se pincha en ese área y el segundo que no se debe hacer nada si se pincha en ese área. Hay que indicar que los anclajes relativos se expandirán tomando como base el **URL** de la descripción del mapa (si hay una marca `BASE` en el documento que contiene la descripción, será ese **URL** el empleado, no el del documento desde el que se referencia).

Se puede especificar un número arbitrario de atributos `AREA`. Si dos de las zonas intersectan, la que aparece en primer lugar en el mapa toma precedencia en la zona en la que se superponen.

El atributo `USEMAP` indica que la imagen es un mapa gestionado por el cliente, aunque puede ser usado junto al atributo `ISMALP`, de manera que un visor que no soporte `USEMAP` accederá al mapa del servidor.

El valor del atributo indica el mapa a emplear con la imagen, en un formato similar al del atributo `HREF` en los anclajes. Así, una referencia a un mapa que comience con una almohadilla se encontrará en el mismo documento que la referencia.

Veamos un ejemplo completo:

```
<HTML>
<HEAD>
<TITLE>Ejemplos de mapas de selecci&oacute;n</TITLE>
</HEAD>
<BODY>
<!-- Mapa para una imagen de 160x60 -->

<MAP NAME="colores">
<AREA SHAPE="POLY" COORDS="10,49,29,10,49,49" HREF="rojo.html">
<AREA SHAPE="RECT" COORDS="60,10,99,49" HREF="verde.html">
<AREA SHAPE="CIRCLE" COORDS="130,30,20" HREF="azul.html">
<AREA SHAPE="RECT" COORDS="0,0,159,59" HREF="negro.html">
<!-- La última área hace que todo lo que no estaba marcado por las anteriores sea negro -->

</MAP>
<H1>Ejemplos de mapas de Selecci&oacute;n</H1>
<P>S&oacute;lo podr&aacute; seleccionar en esta barra si su visor soporta mapas sensibles
controlados por &eacute;l:</P>
<IMG SRC="colores.gif" USEMAP="#colores">
<P>Este mapa funcionar&aacute; independientemente del tipo de visor:</P>
<A HREF="/cgi-bin/imagemap/colores">
<IMG SRC="colores.gif" USEMAP="#colores" ISMAP>
</A>
<P>Pinchando aqu&iacute; llegar&aacute; a una p&aacute;gina con el mismo contenido de la
p&aacute;gina en formato texto (siempre y cuando su visor no soporte mapas sensibles de
usuario):</P>
<A HREF="colores.html">
<IMG SRC="colores.gif" USEMAP="#colores">
</A>
</BODY>
</HTML>
```

El ejemplo es autoexplicativo, puede verlo pinchando [aquí](#).

Para terminar diremos que este modelo de mapas de selección basado en los clientes se justifica por varias razones:

- La sintaxis da flexibilidad al autor para diseñar páginas utilizables en visores que no soporten el mecanismo, ya que los elementos MAP y AREA serán ignorados y el si el documento está en un servidor, éste puede proporcionar el mismo servicio con ISMAP. Por otro lado, si no se usa el servidor, el autor puede elegir entre no mostrar la imagen como un anclaje o enlazarla con otra página que puede proporcionar una lista equivalente de opciones en modo texto.
 - La necesidad de mecanismos no basados en el HTTP para el uso de mapas de selección también se incrementará al aparecer cada vez más archivos en HTML en discos flexibles y **CD-ROM**. Esto puede ser fundamental también para el método alocativo de acceso, es decir, aquél en el que la información se trae una sola vez y se puede acceder repetidamente a ella en local.
-

FRAMES

Las vistas (*frames*) permiten dividir las páginas HTML en varias regiones con barras de desplazamiento, lo que permite presentar la información de manera muy flexible.

Cada vista o región tiene distintas características:

- Se le puede asignar un URL, de modo que puede cargar información independientemente de otras vistas de la página.
- Puede asignársele un nombre (NAME), permitiendo que sean referenciadas por otros URL.
- Puede redimensionarse dinámicamente si el usuario cambia el tamaño de la ventana (aunque el redimensionamiento puede deshabilitarse, asegurando un tamaño constante de las vistas).

Estas propiedades ofrecen nuevas posibilidades:

- Los elementos que el usuario debe ver siempre, como barras de control, *copyrights* o títulos gráficos pueden colocarse en vistas individuales estáticas. Mientras el usuario navega por el servidor en las vistas dinámicas, los contenidos de la vista estática permanecen fijos, independientemente de que otras vistas sean redibujadas.
- Los índices de contenidos son más funcionales. Una vista puede contener una página con enlaces que, al seleccionarse, muestren los resultados en una vista contigua.
- El diseño de vistas paralelas permite enviar consultas desde una de ellas y ver los resultados en la otra, teniendo pregunta y respuesta visibles en la misma página.

La sintaxis de las vistas es muy similar a la de las tablas, y están diseñadas para ser procesadas rápidamente por los visores.

Los nuevos elementos definidos son:

- `FRAMESET`, que define un conjunto de vistas,
- `FRAME`, que define las características de una vista concreta, y
- `NOFRAMES`, que permite incluir información para visores que no disponen de soporte para múltiples vistas.

Comentaremos a continuación cada uno de ellos.

El primer elemento, `<FRAMESET>`, es el principal contenedor para una vista. Toma dos atributos `ROWS` y `COLS` (filas y columnas). Un documento con vistas no tiene cuerpo (`BODY`) y ninguna de las marcas que normalmente se colocarían en él puede aparecer antes de la marca `<FRAMESET>` o esta última será ignorada.

La marca inicial `FRAMESET` tiene su correspondiente marca de cierre `</FRAMESET>`, y dentro de ellas sólo se pueden tener otras marcas de `FRAMESET` anidadas, marcas `FRAME` o la marca `NOFRAMES`.

Los valores de los atributos (`ROWS` y `COLS`) necesitan explicación; tanto uno como otro toman como valor una lista de valores separados por comas. Estos valores pueden ser: valores absolutos en píxeles, porcentajes entre 1 y 100 (tantos por cien), o valores de escala relativos.

En el caso del atributo `ROWS`, el número de filas está implícito en el número de elementos de la lista. Dado que el tamaño total de todas las filas debe coincidir con la altura de la ventana, el alto de las filas debe ser normalizado. Si no se incluye el atributo `ROWS`, se asume una sola fila de la misma altura que la ventana. El atributo `COLS` se comporta de manera similar.

Una vez definidas las filas y columnas, la asociación de elementos se hace en función de la forma de declararlas, por ejemplo si tenemos 4 filas y 2 columnas, tendremos un total de 8 valores, donde los primeros cuatro se asignarán a las vistas 1, 2, 3 y 4 de la primera columna, mientras los cuatro restantes corresponderán a las mismas vistas de la segunda columna.

Comentemos con algo más de detalle la sintaxis de la lista de valores:

- `valor`. Se asume que un valor numérico simple es un tamaño fijo en píxeles. éste es el tipo de valor más crítico, ya que el tamaño de la ventana del cliente variará mucho entre unos y otros. Si se usan valores fijos, será necesario mezclarlos con uno o más valores relativos, ya que en otro caso el visor del usuario probablemente ignorará los valores dados para asegurarse que las proporciones totales de las vistas toman el 100% del ancho y alto de la ventana del usuario.

- `valor%`. Este valor indica un porcentaje simple entre 1 y 100. Si el total de porcentajes supera 100, todos los porcentajes se escalan hacia abajo. Si el total es menor que 100, y existen vistas de tamaño relativo, el espacio sobrante se les dará a ellas. Si no hay vistas de tamaño relativo, todos los porcentajes se escalarán hacia arriba para llegar a un total del 100%.
- `valor*`. El valor de este campo es opcional, un sólo asterisco implica una vista de "tamaño relativo", lo que se interpreta como una petición de darle a la vista todo el espacio que quede libre. Si hay varias vistas de tamaño relativo, el espacio libre se divide entre ellas. Si hay un valor delante del asterisco, la vista que lo tenga toma más espacio relativo, por ejemplo "2*,*" daría 2/3 del espacio a la primera vista y un tercio a la segunda.

Veamos algunos ejemplos (sólo hemos empleado filas, pero se haría lo mismo para poner sólo columnas o para definir filas y columnas):

La siguiente declaración implica una página con tres vistas, la primera y la segunda más pequeñas que la central:

```
<FRAMESET ROWS="20%,60%,20%">
```

esta otra implica tres filas con las dos de los extremos de tamaño fijo y la central ocupa el espacio restante (variará según el tamaño de la ventana):

```
<FRAMESET ROWS="100,* ,100">
```

La marca `FRAMESET` puede estar incluida en otras marcas `FRAMESET`. En ese caso, la subvista completa se coloca en el espacio que hubiera sido empleado para vista si en lugar de una marca `FRAMESET` hubiéramos puesto una marca `FRAME`.

La marca `<FRAME>` define una vista dentro de un conjunto de ellas. La marca `FRAME` no contiene nada, por lo que no tiene marca de cierre. Puede tener hasta seis atributos: `SRC`, `NAME`, `MARGINWIDTH`, `MARGINHEIGHT`, `SCROLLING`, y `NORESIZE`. Veamos qué indica cada uno de ellos:

- `SRC="url"`. El atributo `SRC` toma como valor el URL del documento que se debe mostrar en esa vista en particular. Si no se incluye, se muestra un espacio en blanco del tamaño que debería haber tenido la vista.
- `NAME="nom_vista"`. El atributo `NAME` se emplea para asignarle un nombre a una vista, de manera que pueda ser referenciada por enlaces en otros documentos (generalmente desde otras vistas en el mismo documento). El atributo es opcional; por defecto las ventanas no tienen nombre. Los nombres deben comenzar con caracteres alfanuméricos y pueden tener marcados sus contenidos con el nuevo atributo `TARGET`.
- `MARGINWIDTH="valor"`. Este atributo se emplea cuando se quiere controlar el ancho de los márgenes izquierdo y derecho de una vista. Si se especifica un valor, será en píxeles. Los márgenes no pueden tener un tamaño menor que uno (los objetos dentro de la vista no pueden tocar los bordes) y no pueden tener un tamaño que no deje sitio para los contenidos del documento. Este atributo es opcional; por defecto es el visor el que decide el tamaño apropiado.

- `MARGINHEIGHT="valor"`. Es igual que el anterior, pero se refiere a los márgenes superior e inferior.
- `SCROLLING="yes | no | auto"`. El atributo `SCROLLING` se emplea para indicar si la vista debe tener barras de desplazamiento o no. Si ponemos `yes` tendremos siempre barras en esa vista, si ponemos `no`, nunca usaremos barras, mientras que `auto` hace que el visor decida cuando son necesarias y las coloque donde sea cuando hagan falta. Este atributo es opcional; el valor por defecto es `auto`.
- `NORESIZE`. Este atributo no tiene valores, es simplemente un indicador que dice que la vista no puede ser redimensionada por el usuario. Para redimensionar los usuarios seleccionan un borde de la vista y lo desplazan a una nueva posición. Si una vista adyacente a un borde no se puede redimensionar, todo ese borde no se podrá mover, lo que condicionará el redimensionado de otras vistas. El atributo es opcional, por defecto todas las vistas son redimensionables.

Un visor que no soportara vistas no mostraría nada de un documento con cuerpo `<FRAME>`, para solucionarlo existe el último elemento mencionado, `<NOFRAMES>`, que se emplea para incluir una página alternativa para esos visores. Un visor que sí soporte la marca `<FRAME>` ignoraría todas las marcas y datos entre `<NOFRAMES>` y `</NOFRAMES>`.

Veamos un ejemplo:

```
<HTML>
<HEAD> <TITLE>La ventana Indiscreta</TITLE> </HEAD>
<FRAMESET ROWS="100, *, 100">
<NOFRAMES>
<BODY>
Su visor no tiene vistas, pinche <A HREF="nfindex.html">aqu&iacute;</A> para ver un &iacute;ndice
de contenidos.
</BODY>
</NOFRAMES>
<FRAME SRC="menu.html">
<FRAMESET COLS="30%, 70%">
<FRAME NAME="indice">
<FRAME NAME="contenido">
</FRAMESET>
<FRAME SRC="copyright.html">
</FRAMESET>
</HTML>
```

SCRIPT

Introducida por **Netscape**, permite incluir el código de programas (*scripts*) directamente en el documento HTML. Sólo funciona en las versiones del **Netscape Navigator 2.0** y superiores.

La sintaxis de la inclusión de los scripts en los documentos es:

```
<SCRIPT>
Instrucciones en JavaScript
...
</SCRIPT>
```

El atributo opcional LANGUAGE especifica el lenguaje de programación empleado para escribir el guión (pudiendo ser empleado en un futuro para incluir guiones en otros lenguajes de automatización como el *AppleScript*, *PERL* o *VisualBASIC*):

```
<SCRIPT LANGUAGE="JavaScript">
Instrucciones en JavaScript ...
</SCRIPT>
```

La marca `<SCRIPT>`, y su cierre, `</SCRIPT>`, pueden contener cualquier número de sentencias **JavaScript** en un documento. El **JavaScript** distingue entre mayúsculas y minúsculas.

Una de las peculiaridades de esta marca es que su contenido no es ignorado por los visores que no lo soportan, por lo que se ha incorporado un mecanismo de ocultación del código: los guiones o *scripts* se pueden colocar dentro de comentarios:

```
<SCRIPT LANGUAGE="JavaScript">
<!-- Comienza la ocultación del guión.
Instrucciones en JavaScript ...
// Termina aquí la ocultación. -->
</SCRIPT>
```

Aunque no vamos a describir aquí el **JavaScript**, hay que indicar un par de cosas sobre cómo se analizan los guiones y dónde se deben colocar en las páginas:

- *Definición y llamada de funciones.* Los guiones colocados entre marcas SCRIPT se evalúan después de cargar toda la página. Las funciones se almacenan, pero no se ejecutan hasta que no se produce algún evento en la página. Es importante marcar la diferencia entre la definición de la función y la llamada a la misma: la definición simplemente le asigna un nombre y especifica qué hacer cuando es llamada, mientras que la llamada es la que realmente ejecuta el código empleando los parámetros indicados.
- *Situación de los guiones.* Generalmente, se deben definir las funciones para una página en la cabecera de la misma (HEAD). Dado que la cabecera es lo primero que se carga, esta práctica garantiza que las funciones se han cargado antes de que el usuario tenga opción de hacer cualquier cosa que pueda llamar a una función.

Puede encontrar más información sobre el **JavaScript** en el **URL**:

<http://home.netscape.com/eng/mozilla/Gold/handbook/javascript/>.

[\[contenidos\]](#)[\[sección\]](#)

Enlaces relacionados con el HTML

Información sobre el estándar

- [Información sobre el HTML en la W3 Organization](#)
- [Información sobre el HTML del HTML Working Group \(IETF\)](#).
- [Propuesta de estándar para el HTML 2.0 \(RFC 1866, copia local\)](#)
- [Borrador del HTML 3.0.](#)

Manuales y guías de estilo (inglés)

- [A Beginner's Guide to HTML \(Marc Andreessen\)](#)
- [Introduction to HTML documentation \(Ian Graham, U of Toronto\)](#)
- [How to write HTML files \(Peter Flynn, UCC Ireland\)](#)

- [HTML Reference Manual \(Sandia National Laboratories\)](#)
- [HTML Quick Reference \(Michael Grobe, U of Kansas\)](#)
- [Composing Good HTML \(James "Eric" Tilton, Willamette U\)](#)
- [Style Guide for Online Hypertext \(Tim Berners-Lee, CERN\)](#)
- [Style Guide for Online Hypertext \(Alan Richmond, NASA GSFC\)](#)

Manuales y guías de estilo (castellano)

- [Manual Práctico de HTML \(Álvaro Martínez Echevarría, U. Politécnica de Madrid\)](#)
- [Curso de HTML \(Pedro J. Casanova Pelaez, U. Jaen\)](#)

Documentación sobre Microsoft y Netscape

- [Extensiones del HTML 2.0 de Netscape](#)
- [Extensiones del HTML 3.0 de Netscape](#)

[\[contenidos\]](#)[\[sección\]](#)
