

# Introducción a HTML, XHTML y CSS

Jesús Arias Fisteus  
Norberto Fernández García

## 1. Introducción a HTML

Para poder publicar la información y distribuirla globalmente, se necesita un lenguaje universalmente entendible por las máquinas. Este lenguaje es en el *World Wide Web* HTML (*HyperText Markup Language*).

HTML permite:

- Publicar documentos.
- Acceder a documentos cargándolos directamente en el navegador o, simplemente, haciendo click en un hipere enlace existente en otro documento.
- Rellenar formularios con información que es enviada al servidor para que la procese y realice cierta tarea con ella.
- Incluir objetos multimedia (vídeos, imágenes, sonidos, animaciones, etc) en los documentos.

HTML fue desarrollado inicialmente por *Tim Berners-Lee* en el CERN (Centro Europeo de Investigación Nuclear). La aparición del primer navegador, *Mosaic*, desarrollado en el NCSA (Centro Nacional de Aplicaciones de Supercomputación en EE.UU.) contribuyó a la rápida popularización de esta nueva tecnología, y esa popularidad motivó un rápido desarrollo de la misma. Así, en noviembre de 1995 apareció el primer estándar HTML, llamado HTML 2.0, desarrollado bajo el auspicio del IETF y recogido en la RFC 1866. La siguiente versión de la especificación sería ya HTML 3.2 (las anteriores como HTML 3.0 no fueron sino meros borradores), aparecida en enero de 1997. Finalmente, en diciembre de ese mismo año apareció la última versión, llamada HTML 4, que fue revisada en abril de 1998 y ligeramente modificada en diciembre de 1999, dando lugar a la versión HTML 4.01.

Los principales aportes de la versión 4 de la especificación fueron:

- Aparición de los mecanismos para asociar información de estilo.
- *Scripts*, marcos y objetos.
- Mejoras en los formularios.
- Mejoras en aspectos de internacionalización y accesibilidad.

## 2. Componentes básicos de un documento HTML

### 2.1. Elementos

Un documento HTML está formado por una jerarquía de elementos y texto. Cada elemento puede contener, a su vez, otros elementos y texto. Un elemento está delimitado por marcas (*tags*), que pueden ser de tres tipos:

- Marcas de inicio de elemento: contienen el nombre del elemento y sus atributos. Por ejemplo: `<table width="300">`.
- Marcas de fin de elemento: contienen sólo el nombre del elemento. Por ejemplo: `</table>`.
- Marcas de elemento vacío: útil cuando un elemento no tiene contenido. Por ejemplo: `<br />`. Es equivalente a `<br></br>`. Estas marcas también pueden contener declaración de atributos.

El contenido de un elemento es la porción del documento que se encuentra entre su marca de inicio y su marca de fin. El tipo de contenido de cada elemento está especificado, y debería ser respetado. Por ejemplo, el elemento *html* debería contener, en este orden, un elemento *head* y un elemento *body*. Existen dos tipos básicos de elementos, por su forma de ser representados: elementos de bloque (*block*) y elementos en línea (*inline*). Los primeros siempre son representados comenzando una nueva línea (por ejemplo: párrafos, tablas, listas, etc.) Los segundos se representan siempre en la misma línea que el resto del texto o elementos a representar, sin provocar un salto de línea, salvo que sea necesario por falta de espacio en la línea anterior (por ejemplo: imágenes, elementos para especificar fuentes, cursiva, negrilla, etc.)

### 2.2. Atributos

Los elementos, además de contenido, pueden tener atributos. Un atributo está formado por un nombre y un valor. Los atributos de un elemento se especifican en su marca de inicio (o de elemento vacío), indicando, en este orden, su nombre, el símbolo = y su valor, que suele incluirse entre comillas dobles o simples. Por ejemplo:

```

```

En caso de que sea necesario especificar un tipo de comillas dentro del valor de un atributo, se puede utilizar el otro tipo para englobarlo, sin dar lugar a confusiones. Por ejemplo:

```

```

El nombre y tipo de los atributos que puede tener un determinado elemento, así como su significado, se encuentran especificados, y deberían ser respetados.

### 2.3. Referencias a entidades

Las referencias a entidades permiten incluir caracteres que no pueden aparecer directamente en el documento, normalmente debido a alguna de estas razones: es un carácter *reservado* (<, >, &), el editor no es capaz de representarlo, o el sistema de codificación del

documento no lo permite (por ejemplo, en ASCII de 7 bits no es posible introducir tildes o la letra ñ).

En la siguiente tabla se recoge la forma de referenciar a las entidades más habituales:

Referencia	Carácter
&amp;	&
&lt;	<
&gt;	>
&aacute;	á
&eacute;	é
&iacute;	í
&oacute;	ó
&uacute;	ú
&ntilde;	ñ
&iquest;	¿

Para escribir en castellano es, sin embargo, más cómodo declarar el sistema de codificación de caracteres *ISO Latin 1* (ISO-8859-1), que permite utilizar todos los caracteres de los lenguajes de Europa occidental, sin necesidad de utilizar entidades. Para ello, se puede introducir en *head* el siguiente elemento:

```
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
```

## 2.4. Texto

El texto de un documento HTML se introduce como contenido de aquellos elementos que lo permitan. Los caracteres de fin de línea son tratados como espacios en blanco. Por otra parte, dos o más espacios en blanco consecutivos son tratados como un único espacio en blanco. Los navegadores se encargan de dividir el texto en líneas dependiendo del ancho de la ventana de representación, tamaño de letra, etc.

Si se desea forzar un cambio de línea, puede recurrirse al elemento *br* (marca `<br />`), pero sólo en situaciones justificadas. En la mayoría de las ocasiones, basta con dividir el texto en párrafos (es decir, englobar cada párrafo dentro de un elemento *p*). Por ejemplo:

```
<p>
El texto de un documento HTML se introduce como contenido
de aquellos elementos que lo permitan. Los caracteres de fin
de línea son tratados como espacios en blanco.
</p>
```

```
<p>
Por otra parte, dos o más espacios en blanco
consecutivos son tratados como un único espacio en blanco.
</p>
```

## 2.5. Comentarios

Los comentarios permiten incluir cualquier texto en el documento que los navegadores deben ignorar. Así, por ejemplo, el creador del documento puede dejar indicaciones útiles

para la siguiente vez que tenga que modificarlo, ocultar temporalmente texto o elementos, etc. Se especifican entre “<!--” y “-->”. Por ejemplo:

```
<!-- un comentario en una línea -->
<!-- comentario que ocupa más de una
línea -->
```

## 3. Elementos más utilizados

### 3.1. Elementos estructurales básicos

Todo el documento HTML está contenido dentro del elemento *html*. Por otra parte, los documentos HTML contienen dos partes claramente diferenciadas: la cabecera (*head*) y el cuerpo (*body*). La cabecera contiene declaraciones globales para todo el documento (título, sistema de codificación de caracteres, hojas de estilo, meta-datos, etc.) El cuerpo contiene la parte directamente representable del documento (texto, imágenes, tablas, etc.) Cada una de estas partes está contenida dentro de un elemento. La estructura básica es, por tanto:

```
<html>
  <head>
    ...
  </head>
  <body>
    ...
  </body>
</html>
```

### 3.2. Títulos

Normalmente, un documento HTML debe tener un título, que se suele mostrar en la barra superior de la ventana del navegador. Éste se especifica mediante el elemento *title*, dentro de la cabecera:

```
<html>
  <head>
    <title>Hola Mundo!</title>
  </head>
  <body>
    ...
  </body>
</html>
```

Por otra parte, es muy frecuente la necesidad de estructurar un documento en secciones (capítulos, apartados, etc), y que cada una de éstas tenga un título. En HTML se pueden utilizar los elementos *h1*, *h2*, . . . , *h6*. Estos elementos, de tipo bloque, aceptan como contenido el título de la sección. Cuanto menor es el número del nombre del elemento, de más alto nivel será la sección, como se puede ver en el siguiente ejemplo:

```

...
<body>
  <h1>Noticias de última hora</h1>
  <h2>Nacional</h2>
  <h3>Palacio asegura no haber recibido ninguna...</h3>
  <p>
    Ayer, Marruecos acusó a España en la prensa...
  </p>

  <h3>...</h3>
  <p>
    El ministro de...
  </p>

  <h2>Internacional</h2>
  <h3>...</h3>
  <p>
    Ayer se celebró en...
  </p>
</body>

```

### 3.3. Párrafos

El elemento *p* es un elemento de bloque que permite estructurar el texto en párrafos. Puede contener texto y cualquier elemento *en línea*.

### 3.4. Imágenes

Las imágenes se insertan mediante el elemento en línea *img*. La mayoría de los navegadores son capaces de representar imágenes GIF, JPEG y PNG. Este elemento no puede tener contenido, y en él se especifica mediante atributos la siguiente información:

- Fuente de la imagen (*src*): URL, absoluta o relativa, de la imagen (atributo obligatorio).
- Texto alternativo (*alt*): texto descriptivo de la imagen, pensado para navegadores incapaces de representar imágenes y para navegadores que representen el documento mediante sonido (atributo obligatorio).
- Tamaño de la imagen (*width* y *height*): ambos son atributos opcionales que permiten al navegador conocer el tamaño de las imágenes sin necesidad de descargarlas. Su utilización facilita a los navegadores el ir representando la página mientras descarga las imágenes, permitiendo así al usuario ver la página mientras se realiza la descarga.

A continuación se muestran dos ejemplos de utilización:

```




```

### 3.5. Enlaces

Los enlaces a otras páginas se pueden realizar mediante el elemento en línea *a*. Su contenido es el texto (o imágenes) que el usuario puede pinchar para activar el enlace. El destino del enlace se especifica mediante el atributo *href*. El siguiente ejemplo muestra un enlace a otra página dentro de un párrafo:

```
<p>
  La Escuela Politécnica de la <a href="http://www.it.uc3m.es">Universidad
  Carlos III de Madrid</a> está situada en Leganés.
</p>
```

Al igual que en el caso de las imágenes, el destino puede ser codificado de forma absoluta o relativa. También se puede asociar un enlace a una imagen, como se refleja en el siguiente ejemplo:

```
<a href="http://www.it.uc3m.es">
  
</a>
```

Por último indicar que también es posible enlazar una parte de un documento HTML desde el propio documento o desde otro documento externo. Para ello no hay más que utilizar un elemento *a* pero acompañado de un atributo *name* (o *id* en XHTML) en lugar de *href*. Este elemento *a* actuará así como destino del enlace, seleccionando un bloque de documento. Para acceder a ese bloque se usará también el elemento *a* indicando en su *href* el nombre del destino precedido del símbolo *#*. Por ejemplo:

```
<h1>Este es el inicio del documento</h1>
<a name="inicio" />

<p>
  Contenido del documento
</p>

<a href="http://www.foo.org/index.html#parte1">
  Nos lleva a la parte1 de un documento externo
</a>

<h1>Este es el fin del documento</h1>
<a href="#inicio">
  Nos lleva al inicio del documento
</a>
```

Esto puede resultar útil para movernos rápidamente por el interior del documento cuando éste tiene un tamaño considerable. Asimismo puede ayudarnos para construir un índice, con distintas secciones a las que se puede acceder directamente, dentro del documento HTML.

### 3.6. Listas

Las listas permiten estructurar información en *puntos*, comenzando cada punto en una nueva línea. Existen tres elementos, todos de tipo bloque, para especificar listas, uno para cada uno de los tipos de lista permitidos: listas ordenadas, listas no ordenadas y listas de definiciones.

Las listas ordenadas (elemento *ol*) asignan automáticamente un número a cada *item* de la lista. El contenido de este elemento es uno o más *items de lista* (elemento *li*).

Las listas no ordenadas (elemento *ul*) son como las anteriores, pero no asignan números a los *items*.

Las listas de definiciones (elemento *dl*) tienen como contenido elementos *dt* y *dd*. El primero permite especificar el término a definir, y el segundo, la propia definición.

A continuación se muestran ejemplos de listas, y cómo se verían en un navegador:

<pre>&lt;p&gt; Lista no ordenada: &lt;/p&gt; &lt;ul&gt;   &lt;li&gt;Listas ordenadas.&lt;/li&gt;   &lt;li&gt;Listas no ordenadas.&lt;/li&gt;   &lt;li&gt;Listas de definiciones.&lt;/li&gt; &lt;/ul&gt; &lt;p&gt; Lista ordenada: &lt;/p&gt; &lt;ol&gt;   &lt;li&gt;Listas ordenadas.&lt;/li&gt;   &lt;li&gt;Listas no ordenadas.&lt;/li&gt;   &lt;li&gt;Listas de definiciones.&lt;/li&gt; &lt;/ol&gt; &lt;p&gt; Lista de definiciones: &lt;/p&gt; &lt;dl&gt;   &lt;dt&gt;HTML&lt;/dt&gt;   &lt;dd&gt;HiperText Markup Language&lt;/dd&gt;   &lt;dt&gt;XML&lt;/dt&gt;   &lt;dd&gt;eXtensible Markup Language&lt;/dd&gt; &lt;/dl&gt;</pre>	<p>Lista no ordenada:</p> <ul style="list-style-type: none"><li>▪ Listas ordenadas.</li><li>▪ Listas no ordenadas.</li><li>▪ Listas de definiciones.</li></ul> <p>Lista ordenada:</p> <ol style="list-style-type: none"><li>1. Listas ordenadas.</li><li>2. Listas no ordenadas.</li><li>3. Listas de definiciones.</li></ol> <p>Lista de definiciones:</p> <p>HTML HiperText Markup Language XML eXtensible Markup Language</p>
--	--

### 3.7. Estilo del texto

Los elementos para controlar el estilo del texto se dividen en dos categorías: estilo físico y estilo lógico.

Los elementos de estilo físico regulan el formato de los caracteres de su contenido explícitamente (*b*: negrilla; *i*: cursiva; *u*: subrayado; *tt*: texto mecanografiado; *font*: tamaño y color de la fuente).

Los elementos de estilo lógico aportan información semántica acerca de su contenido, lo cual puede afectar, indirectamente, a su estilo de presentación. Se recogen en la siguiente tabla:

cite	Cita
code	Código fuente
dfn	Definido
em	Enfatizado
kdb	Palabra clave
samp	Ejemplo
strong	Resalta
var	Variable

### 3.8. Tablas

Las tablas resultan muy útiles para estructurar la información en filas y en columnas. Existen bastantes elementos relacionados con tablas, pero aquí sólo se explican los más relevantes: *table*, *th* (cabecera), *tr* (fila) y *td* (celda).

En HTML, una tabla se representa como un conjunto de filas. A su vez, cada fila contiene una o más celdas. En las misma tabla, todas las filas deben tener el mismo número de celdas, para poder representar las columnas correctamente alineadas.

Por defecto, cada celda se corresponde con una columna. El atributo *colspan* de *td* permite variar esta correspondencia, estableciendo su valor como el número de columnas por las que se debe expandir una celda.

A continuación se presenta un ejemplo, con su aspecto aproximado:

<pre>&lt;table border="1"&gt;   &lt;tr&gt;     &lt;th&gt;izda.&lt;/th&gt;     &lt;th&gt;centro&lt;/th&gt;     &lt;th&gt;drcha.&lt;/th&gt;   &lt;/tr&gt;   &lt;tr&gt;     &lt;td&gt;1 izda.&lt;/td&gt;     &lt;td&gt;1 centro&lt;/td&gt;     &lt;td&gt;1 drcha.&lt;/td&gt;   &lt;/tr&gt;   &lt;tr&gt;     &lt;td&gt;3 izda.&lt;/td&gt;     &lt;td colspan="2"&gt;3 drcha.&lt;/td&gt;   &lt;/tr&gt; &lt;/table&gt;</pre>	<table border="1"> <thead> <tr> <th>izda.</th> <th>centro</th> <th>drcha.</th> </tr> </thead> <tbody> <tr> <td>1 izda.</td> <td>1 centro</td> <td>1 drcha.</td> </tr> <tr> <td>3 izda.</td> <td colspan="2">3 drcha.</td> </tr> </tbody> </table>	izda.	centro	drcha.	1 izda.	1 centro	1 drcha.	3 izda.	3 drcha.	
izda.	centro	drcha.								
1 izda.	1 centro	1 drcha.								
3 izda.	3 drcha.									

### 3.9. Marcos (*Frames*)

Los marcos permiten la división del espacio de visualización del navegador en subventanas independientes. En cada una de esas subventanas se puede presentar un documento HTML diferente.

A continuación se presenta un ejemplo con su aspecto aproximado:

```

<html>
<head>
  <title>Ejemplo simple de marcos</title>
</head>
<frameset cols="20%, 80%">
  <frameset rows="100, 200">
    <frame src="contenidos1.html">
    <frame src="contenidos2.gif">
  </frameset>
  <frameset rows="100, 200">
    <frame src="contenidos3.html">
    <frame src="contenidos4.gif">
  </frameset>
<noframes>
  <p>Este documento contiene
  <ul>
    <li><a href="contenidos1.html">Un documento</a>
    <li>
    <li><a href="contenidos3.html">Otro documento</a>
    <li>
  </ul>
</noframes>
</frameset>
</html>

```



Como se puede ver, tres son los principales elementos que intervienen en el proceso de creación de marcos:

**frameset** utilizado para determinar la disposición de los marcos en la ventana de visualización. Observar que sustituye al elemento *body* como raíz del cuerpo del documento.

**frame** utilizado para definir los contenidos y apariencia de un marco concreto.

**noframes** utilizado para especificar el contenido alternativo a presentar por el navegador en caso de que éste no soporte el uso de marcos.

### 3.10. Formularios

Los formularios son elementos de bloque que pueden contener *controles*, mediante los cuales el usuario puede interactuar con la página, normalmente para enviar datos al servidor *web*.

#### 3.10.1. Controles

Cada control tiene un nombre, especificado mediante el atributo *name*. Además, cada control tiene asociado un valor inicial (salvo excepciones, especificado en el atributo *value*) y un valor actual (introducido por el usuario o por un *script*).

Cuando el formulario es enviado, se envía el nombre de cada uno de sus controles, y su valor actual.

### 3.10.2. Tipos de controles

En un formulario se pueden utilizar distintos tipos de controles:

- *Botón*: se pueden especificar tanto con el elemento *input* (*type="button"*) como con el elemento *button*. Puede ser de tres tipos: *submit* (envía el formulario), *reset* (establece el valor inicial en todos los controles del formulario) y *push* (utilizado para cualquier otra cosa, en combinación con *scripts*).
- *Checkbox*: permite especificar valores *on/off*. Se especifican mediante el elemento *input* (*type="checkbox"*). Pueden utilizarse, en el mismo formulario, varios con el mismo nombre. Si se especifica el atributo *checked="checked"*, se inicializa a *on*.
- *Botón radio*: también permite especificar valores *on/off*, pero de tal forma que sólo uno de los que comparten el mismo nombre puede estar en *on* simultáneamente. Se especifican mediante el elemento *input* (*type="radio"*). Si se especifica el atributo *checked="checked"*, se inicializa a *on*.
- *Menú*: permite al usuario seleccionar una opción entre varias. Se crean mediante los elementos *select*, *optgroup* y *option*.
- *Texto*: permite que el usuario introduzca una o varias líneas de texto. Se especifica mediante el elemento *input* (*type="text"*), cuando se desea sólo una línea, o mediante el elemento *textarea* cuando éste puede introducir cualquier número de líneas.
- *Fichero*: permite al usuario seleccionar un fichero del sistema local de ficheros. Mediante este control se pueden enviar ficheros al servidor. Se especifica mediante el elemento *input* (*type="file"*).
- *Valor oculto*: no se muestra al usuario. Su valor será siempre el valor inicial, salvo que algún *script* lo modifique. Permite que los *scripts* puedan pasar valores al servidor, y como ayuda para el control de sesiones.

### 3.10.3. Envío de un formulario

Cuando el usuario pincha en un botón de tipo *submit*, se produce el envío del formulario. Existen dos métodos de envío, que se especifican mediante el atributo *method* del elemento *form*:

- *GET*: utiliza el método GET de HTTP. Los parámetros se codifican concatenados en el URI especificado en el atributo *action* del elemento *form*. No se debe utilizar este método para operaciones no idempotentes.
- *POST*: utiliza el método POST de HTTP. Los parámetros se codifican en el cuerpo del mensaje HTTP.

Los parámetros a los cuales se hace mención son pares que contienen el nombre y valor actual correspondientes a todos los controles *con éxito* (en general, son aquellos que no se encuentren deshabilitados, y los botones radio y *checkboxes* que se encuentren en *on*).

Por otra parte, hay dos formas de codificar los parámetros a enviar al servidor, seleccionables mediante el atributo *enctype* del elemento *form*:

- *application/x-www-form-urlencoded*: se sustituyen los espacios en blanco por “+”, y todos los valores no alfanuméricos y caracteres reservados por “%” seguido de su valor ASCII (dos dígitos en hexadecimal). Los nombres se separan de los valores por un carácter “=”, y cada par nombre–valor se separa del resto por un carácter “&”. Por ejemplo: nombre=Juan+P%C3%A9rez&edad=29. Éste es tipo de codificación por defecto, y el único permitido con el método GET.
- *multipart/form-data*: más eficiente para el envío de grandes cantidades de datos, así como datos binarios. Debe ser utilizado, por ejemplo, para el envío de ficheros al servidor. El cuerpo del mensaje HTTP se separa en varias partes, cada una de las cuáles envía los datos asociados a un control.

### 3.10.4. Ejemplo

A continuación se muestra el código de ejemplo de un formulario.

```
<form action="http://acme.es/submit" method="post"
  encoding="multipart/form-data" >
<h3>Información Personal</h3>
  <p>
    Apellidos: <INPUT name="personal_apellidos" type="text">
    Nombre: <INPUT name="personal_nombre" type="text">
    Dirección: <INPUT name="personal_direccion" type="text">
  </p>

<h3>Historia Médica</h3>
  <p>
    <input name="historia_enfermedades"
      type="checkbox" value="Sarampión" /> Sarampión
    <input name="historia_enfermedades"
      type="checkbox" value="Varicela" /> Varicela
  </p>

<h3>Medicación actual</h3>
  <p>
    Está tomando usted medicación actualmente:
    <input name="medicacion_ahora" type="radio" value="Si">Sí
    <input name="medicacion_ahora" type="radio" value="No">No
  </p>
  <p>
    Si la respuesta es afirmativa, indique el nombre de los
    medicamentos:
  </p>
  <p>
    <textarea name="medicación actual:" rows="20" cols="50">
    </textarea>
  </p>
  <p><input type="submit" value="Enviar" /></p>
</form>
```

## 4. Introducción a XHTML

XHTML es una redefinición de HTML 4 en XML. A pesar de detener distinto nombre, conserva la mayoría de los elementos y atributos de HTML. En 4.1 comentaremos en detalle las diferencias existentes entre HTML 4 y XHTML, pero adelantamos que las más destacables están en la ausencia de elementos y atributos relacionados con estilo (fuentes, colores, etc.), que ya en HTML 4 están desaconsejados, y en la existencia de unas normas más estrictas en la colocación de etiquetas de elementos.

En cuanto a los motivos por los que se vió la necesidad de redefinir HTML 4 como aplicación XML, podemos citar los siguientes:

- En XHTML se refuerza la separación contenido/presentación, eliminando de la especificación aquellos elementos y atributos relacionados con el estilo. Esto tiene la ventaja de facilitar el cambio de la información de presentación para adaptarla a las características del dispositivos de salida concreto (asistente digital, teléfono móvil, ordenador, televisor, etc)
- Dado que los documentos XHTML son un tipo de documentos XML, todas las herramientas disponibles para trabajar con XML se pueden emplear también con XHTML.
- En XHTML existen reglas estrictas acerca del formato que debe tener un documento. Estas reglas se refieren por un lado a la buena formación del documento (establecen por ejemplo que todo elemento debe cerrarse adecuadamente) y por otro a la validez del mismo (indicando por ejemplo que dentro de un elemento `<html>` sólo puede haber un `<body>`). Todo documento XHTML, para ser tal, debe cumplir estas reglas, y el hecho de que las cumpla va a facilitar su procesado automatizado.
- Modularización: en 4.2 veremos que existen distintas versiones de XHTML, y que, a partir de la versión 1.1, se inicia un proceso de modularización de la especificación. Este proceso permitirá que los dispositivos de capacidades reducidas implementen únicamente ciertas partes de la especificación, y que los desarrolladores web puedan adaptar mejor sus documentos a los dispositivos en los que se van a visualizar. Asimismo la modularización permitirá la extensión de XHTML sin rehacer de nuevo la especificación, bastará con añadir nuevos módulos aprovechando al máximo la capacidad de extensibilidad de XML.

### 4.1. Diferencias con HTML 4

Como ya indicamos, XHTML no es más que una redefinición de HTML 4 como aplicación XML. Como consecuencia de este hecho la semántica de los elementos y atributos de XHTML es exactamente la misma que en HTML 4. Existen sin embargo pequeñas diferencias entre los dos lenguajes de hipertexto, las principales de las cuales son:

- Los nombres de elementos y atributos deben escribirse en minúscula en los documentos XHTML, mientras que en HTML 4 era posible escribirlos en minúscula o mayúscula.
- Los valores de los atributos deben escribirse entre comillas (“ o ’) en XHTML. No son válidas construcciones del tipo `<font color=red>`.

- Todos los elementos tienen marca de inicio o finalización (o elemento vacío). Por ejemplo: `<br />`, `<p></p>`. No son válidas construcciones del tipo `<br>` o `<ul> <li>A <li>B </ul>`.
- La anidación de elementos debe ser correcta. No son válidas construcciones del tipo `<i><b></i></b>`
- Los elementos y atributos utilizados para especificar preferencias en cuanto al estilo de la información (como por ejemplo `<font>` o el atributo `bgcolor`) desaparecen. Las características de presentación se establecen ahora mediante el uso de lenguajes específicos como CSS.
- El atributo *id* sustituye al atributo *name* en: `a`, `applet`, `frame`, `iframe`, `img` y `map`.
- Es obligatorio añadir al comienzo del documento XHTML una declaración *DOCTYPE* que referencia al DTD (Definición de Tipo de Documento) donde se indican las reglas de construcción del documento (como por ejemplo qué elementos y atributos son válidos y dónde puede aparecer cada uno). Ese DTD será específico de cada versión de XHTML.

## 4.2. Versiones de XHTML

Actualmente, existen varias versiones de XHTML:

**XHTML 1.0** [2] el más parecido a HTML 4, con tres variantes:

**Transitional** permite el uso de las capacidades de presentación de HTML y está pensado para trabajar con navegadores con soporte de CSS limitado.

**Strict** no se permite el uso de los elementos y atributos relacionados con aspectos de presentación. Pensado para ser usado con hojas de estilo CSS.

**Frameset** permite el uso de marcos (frames) para dividir la ventana del navegador.

**XHTML 1.1** [3] nueva versión, que parte de XHTML 1.0 *strict*. No permite el uso de elementos y atributos relacionados con el estilo (pero sí, obviamente, hojas de estilo, como CSS). Se introduce el concepto de modularización.

**XHTML Basic** [4] versión simplificada de XHTML 1.1 pensada para dispositivos de capacidad limitada de procesamiento, como televisores, teléfonos, móviles, PDAs, etc.

**XHTML 2.0** [5] se encuentra actualmente, y desde hace unos cuantos años, en proceso de estandarización (*Working Draft*) por parte del W3C (*World Wide Web Consortium*). Retoma la línea iniciada con XHTML 1.1 de modularizar XHTML, añadiendo nuevos módulos como *XML Events* y *XForms*. Estos módulos nacen con el objetivo de minimizar el uso de *scripts* dentro de documentos XHTML mediante la inclusión, como parte del propio lenguaje, de los medios necesarios para implementar las funcionalidades más importantes que requerían el uso de dichos *scripts* (eventos y formularios).

### 4.3. Ejemplo

A continuación se muestra un ejemplo sencillo de documento XHTML 1.0 Transicional:

```
<?xml version="1.0" encoding="iso-8859-1"?>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="es">

  <head>
    <title>Mi primera página XHTML</title>
  </head>

  <body>
    <h1>Hola Mundo!</h1>
  </body>

</html>
```

## 5. Hojas de estilo CSS

En HTML existen elementos y atributos que permiten especificar el estilo de los documentos (por ejemplo, los tipos y tamaños de fuente, colores de texto, colores de fondo, ancho de elementos, etc). El uso de estas características de estilo de HTML presenta sin embargo una serie de problemas:

- La información de estilo va dentro del propio documento, con lo que se dificulta la homogeneización de las características presentacionales de un conjunto de páginas de un mismo sitio web. Así por ejemplo si tenemos un grupo de documentos con la misma información de estilo y queremos realizar un cambio por pequeño que sea, debemos modificar todos y cada uno de esos documentos.
- Para adaptar la presentación del documento a las características de los distintos dispositivos en los que se puede presentar no queda otro remedio que hacer distintas versiones del mismo documento.

Para evitar estas dificultades es necesario separar el contenido del documento, su información, de la apariencia con la que esta información se presenta al usuario, consiguiendo así que cambios en una de las partes no afecten a la otra. Por eso la evolución de la recomendación HTML va por la línea de separar estos aspectos dejando la representación de la información a HTML/XHTML e introduciendo mecanismos que permitan representar la información de estilo. Uno de esos mecanismos es CSS (*Cascading Style Sheets*) [7].

En 5.1 veremos como asociar la información de estilo CSS a documentos HTML/XHTML, pero podemos adelantar que el mecanismo más usado es el de almacenar la información presentacional en un documento denominado *hoja de estilo* que se enlaza desde la página HTML/XHTML. De este modo, cuando queremos presentar de la misma manera un conjunto de documentos HTML/XHTML no tenemos más que asociarles a todos la misma hoja de estilo, y realizando cambios en esa hoja modificamos el aspecto de todos los

documentos. Asimismo, para presentar un documento en dispositivos distintos, basta con asociar a ese documento hojas de estilo distintas o incluso una única hoja de estilo en la que la información a considerar estará condicionada al *medio* o dispositivo destino<sup>1</sup>.

## 5.1. Declaración de hojas de estilo

El lenguaje HTML es independiente del lenguaje utilizado para especificar las hojas de estilo. Por ello, en la cabecera del documento HTML, es recomendable declararlo. En el caso de CSS la declaración necesaria es la siguiente:

```
<meta http-equiv="Content-Style-Type" content="text/css">
```

Por otra parte, las hojas de estilo a utilizar para un documento pueden ser incluidas en el propio documento (con el elemento *style* en la cabecera), en un documento externo (elemento *link* en la cabecera) o directamente en un elemento (atributo *style*). A continuación se muestran algunos ejemplos:

```
<head>
  ...
  <link href="special.css" rel="stylesheet" type="text/css">
</head>
```

```
<head>
  ...
  <style type="text/css">
    h1 {border-width: 1; border: solid; text-align: center;}
  </style>
</head>
```

```
<p style="font-size: 12pt; color: fuchsia">
Aren't style sheets wonderful?
</p>
```

## 5.2. Colores y tamaños

Aquí se presentan los tipos de datos más habituales de las propiedades de CSS: colores y tamaños.

### 5.2.1. Colores

Hay dos formas de especificar un color: por sus componentes RGB (rojo, verde y azul), o, para los colores predefinidos, por su nombre.

Las componentes RGB se dan indicando para cada uno de estos colores (rojo, verde, azul) un valor entre 0 y 255. Por ejemplo, `rgb(64,128,255)` especifica un color con 64 de rojo, 128 de azul y 255 de verde.

Los colores predefinidos en CSS son los siguientes: *aqua*, *black*, *blue*, *fuchsia*, *gray*, *green*, *lime*, *maroon*, *navy*, *olive*, *purple*, *red*, *silver*, *teal*, *white* y *yellow*.

---

<sup>1</sup>Esto sólo es posible a partir de la versión 2 de la especificación CSS, denominada CSS2

### 5.2.2. Tamaños

Los tamaños permiten especificar altura y anchura de elementos, tamaños de márgenes, tamaños de fuentes, etc. Pueden ser especificados de forma relativa a otra propiedad (*em*, tamaño de fuente; *ex*, valor *x-height* de la fuente; *px*, tamaño de pixel) o de forma absoluta (*in*, pulgadas; *cm*, centímetros; *mm*, milímetros; *pt*, puntos, o 1/72 pulgadas; *pc*, picas, o 12 puntos).

También se pueden especificar longitudes como porcentajes. En el siguiente ejemplo se establece un margen izquierdo del 10% del ancho del párrafo:

```
p { margin-left: 10% }
```

### 5.3. Asignación de propiedades a elementos

CSS define un conjunto de propiedades y sus posibles valores. Cada regla de estilo está compuesta por dos campos: un *selector* y un *bloque de declaraciones*. El selector establece a qué elementos se les aplica la información de estilo. Las declaraciones especifican propiedades (y sus valores) para los elementos seleccionados. Por ejemplo, la siguiente regla de estilo CSS asigna un color de texto, un tamaño de fuente y texto en *negrilla* a todos los elementos *p* del documento:

```
p { color: rgb(0,0,128);  
    font-size: 14pt;  
    font-weight: bold }
```

Como se ve en él, primero se especifica el selector, y a continuación las declaraciones entre llaves, separadas entre sí por “;”. El nombre y el valor de cada propiedad se separan por “:”. Los tamaños de fuente se suelen especificar en *puntos* (*pt* en CSS). Los colores pueden ser especificados utilizando las coordenadas RGB, o el nombre del color, para aquellos predefinidos.

En la siguiente tabla se muestran los patrones más importantes que se pueden utilizar como selectores en documentos CSS.

Patrón	Elementos seleccionados
*	Cualquier elemento.
E	Cualquier elemento E.
E,F	Cualquier elemento E o F.
E F	Cualquier elemento F descendiente de E.
E > F	Cualquier elemento F hijo de E.
E:first-child	Cualquier elemento que sea el primer hijo de un elemento E.
E:link	Cualquier elemento E tal que sea origen de un enlace cuyo destino no haya sido visitado.
E:visited	Ídem, con destino visitado.
E + F	Cualquier elemento F inmediatamente precedido por un elemento E.
E[foo]	Cualquier elemento E en el que esté establecido el atributo <i>foo</i> .
E[foo=warning]	Cualquier elemento E cuyo atributo <i>foo</i> tenga el valor <i>warning</i> .
E[foo ≠warning]	Lo contrario que el anterior.
E.warning	Cualquier elemento E cuyo atributo <i>class</i> tenga el valor <i>warning</i> .

## 5.4. Reglas de *cascada*

Cabe preguntarse qué pasaría si definimos dos veces en distintas partes de una hoja de estilo, o en dos hojas de estilo diferentes que se aplican al mismo documento HTML/XHTML, la misma regla con valores contradictorios. Eso ocurriría por ejemplo si nuestra hoja de estilo, además de la declaración anterior (ver 5.3), contuviese la siguiente:

```
p.clase1 { color: rgb(128,128,128); }
```

Dado que los elementos que son seleccionados por *p.clase1* también lo son por *p*. ¿Qué color tomarían esos elementos?. Aquí es donde entra en juego el mecanismo de *cascada* que da nombre a esta especificación. Dicho mecanismo establece las siguientes reglas para la resolución de conflictos:

- Encontrar todas las declaraciones que incluyan al elemento y propiedad en cuestión.
- Ordenar las declaraciones según el origen. Si esas declaraciones proceden de información de estilo especificada por el diseñador web tendrán mayor importancia que si proceden de información de estilo establecida por el usuario o tomada por el navegador por defecto. Esta primera clasificación no suele ayudar mucho, porque en general toda la información de estilo es proporcionada por el diseñador, bien mediante hojas de estilo externas (enlazadas al documento utilizando el elemento *link*) bien incluyendo la información de estilo en el propio documento (mediante elementos/atributos *style*).
- Si aún persisten los conflictos, se ordenan las declaraciones más prioritarias (las procedentes del diseñador web) por la especificidad del selector. Este es el caso en que nos encontramos, ya que el selector *p.clase1*, al ser más específico que el *p*, tiene más prioridad.

- De seguir existiendo contradicciones, el último aspecto a considerar debe ser el orden. Las declaraciones que aparecen más tarde serán consideradas prioritarias.

## 5.5. Ejemplos

A continuación se muestran ejemplos sencillos de utilización de propiedades de CSS básicas:

```
p { color: red }
p { background-color: rgb(128,129,255) }
p { background-image: url("http://www.foo.com/image.png" }
p { background: url("http://www.foo.com/image.png" }
p { background: black }

p { text-indent: 3em }           // em = tamaño de fuente
p { text-align: center }        // left, center, right, justify
p { text-decoration: underline } // underline, overline,
                                // line-through, blink, none
p { text-transform: capitalize } // capitalize, uppercase, lowercase,
                                // none

p { font-style: italic }        // normal, italic, oblique
p { font-weight: bold }         // normal, bold, bolder, lighter,
                                // 100, 200, ..., 900
p { font-size: large }         // xx-small, x-small, small,
                                // medium, large, x-large,
                                // xx-large, o un tamaño
                                // relativo o absoluto.

p { margin-left: 10% }         // 10% del ancho
p { margin-top: 10px }         // 10 pixels
p { margin-bottom: 1cm }       // 1 cm
p { margin-right: 5em }        // 5 veces el tamaño de fuente

p { padding-left: 10% }        // otros: padding-right, padding-top,
                                // padding-left, padding-bottom
```

## 5.6. Agrupación de elementos

En muchas ocasiones resulta interesante aplicar las mismas propiedades a un grupo de elementos o texto. Se pueden agrupar utilizando los elementos *div* y *span*. La diferencia entre ambos es que el primero es de tipo bloque (y, por tanto, fuerza un salto de línea antes y después), mientras que el segundo es de tipo *en línea* y no fuerza a dicho salto. Se muestra, a continuación, un ejemplo que hace uso de ambos:

```
<head>
...
<style type="text/css">
  div.resumen { text-align: justify }
  span.inicio { font-weight: bold }
```

```

</style>
</head>
<body>
  <div class="resumen">
    <p><span class="inicio">La Bolsa española</span> acumula
      nueve jornadas de caídas consecutivas, la peor estadística
      desde la Guerra del Golfo. </p>
    <p><span class="inicio">El Ibex </span> llegó al triste
      récord de cinco mínimos anuales seguidos y restó un
      3,39% para situarse en los 5.390 puntos. </p>
  </div>
</body>

```

## 5.7. Modelo de representación de CSS

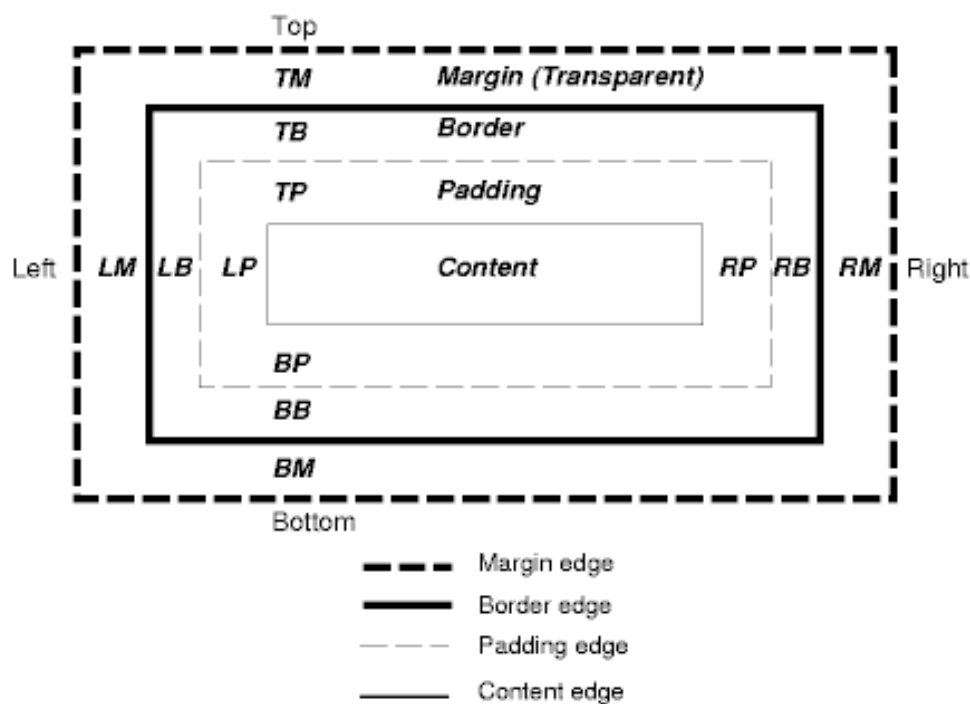


Figura 1: Modelo de caja CSS

Cada elemento presente en el documento HTML/XHTML al que se le asocia la hoja de estilo CSS, va a dar lugar, a la hora de visualizar la información en pantalla, a cero o más cajas, cada una de las cuales presenta la estructura representada en la figura 1.

Como se puede ver el contenido de la caja está rodeado de tres capas: un relleno (*padding*), un borde (*border*) y un margen (*margin*). Las propiedades recogidas en las recomendaciones CSS1 y CSS2 permiten dar formato (tamaño, color, estilo) a cada una de esas capas así como también al contenido que rodean.

## A. Referencia de elementos y atributos HTML 4

A Define un “ancla”: origen o destino de un enlace. Atributos:

**href** Especifica la localización (URL) del recurso al que apunta el enlace.

**name** Le pone nombre al elemento para que pueda servir de destino a otro enlace.

**ADDRESS** Proporciona al lector información de contacto, típicamente sobre el desarrollador de la página web.

**B** Texto en negrita (*bold*).

**BLOCKQUOTE** Citas textuales.

**cite** URI (*Uniform Resource Identifier*) que identifica el origen de la cita.

**BODY** Cuerpo del documento HTML.

**alink** Color de los enlaces cuando son seleccionados.

**background** URL de una imagen de fondo.

**bgcolor** Color de fondo. Un color en HTML es un valor RGB hexadecimal (#RRGGBB) o bien una de las siguientes cadenas de texto: *black, silver, gray, white, maroon, red, purple, fuchsia, green, lime, olive, yellow, navy, blue, teal, aqua*.

**link** Color de los enlaces no visitados.

**text** Color de texto.

**vlink** Color de los enlaces visitados.

**BR** Causa la inserción de un fin de línea.

**CENTER** Su contenido se representará centrado en la ventana de visualización.

**DD** Descripción de un elemento definido.

**DIV** Usado para estructurar documentos. Los elementos que conformen su contenido estarán agrupados y se presentarán como un bloque. Diseñado para ser usado en conjunción con hojas de estilo.

**DL** Indica el comienzo de una lista de definiciones de términos.

**DT** Término a ser definido en una lista de definición.

**FONT** Establecer el color y tamaño de fuente.

**color** Color de fuente.

**size** Tamaño de fuente. Número entero entre 1 y 7 o bien +/- y un número para indicar un incremento/decremento respecto al valor actual. En este segundo caso el valor final absoluto deberá estar igualmente comprendido entre 1 y 7. Así por ejemplo si el valor actual de tamaño de fuente es 5 y el atributo *size* vale +3, el valor absoluto final será 7 y no 8.

**FORM** Formulario.

**action** URL con la ubicación del programa que se encargará de procesar la información del formulario.

**method** Especifica el método HTTP que se utilizará para enviar al servidor los datos del formulario. Puede ser *get* o *post*.

**FRAME** Marco.

**scrolling** Para indicar si queremos o no que el navegador proporcione barras de desplazamiento lateral para el marco. Valores posibles *yes*, *no*, *auto*. El valor por defecto es *auto* que deja decidir al navegador.

**src** URL con la localización del documento que se va a presentar inicialmente como contenido del marco.

**name** Asocia un nombre al marco para poder crear enlaces con destino a dicho marco desde otras partes del documento.

**FRAMESET** Utilizado para estructurar la ventana de visualización del navegador dividiéndola en marcos.

**cols** Define el número de subespacios verticales.

**rows** Define el número de subespacios horizontales.

El valor de cualquiera de estos dos atributos es una lista de longitudes separadas por comas. Típicamente las longitudes se expresan en *pixeles* o como porcentajes del tamaño del espacio de visualización.

**H1-H6** Suelen contener títulos que se utilizan para destacar comienzos de secciones o apartados dentro del documento.

**HEAD** Cabecera del documento HTML.

**HR** Permite dibujar una regla horizontal en el interior del documento. Típicamente empleado para separar dos partes diferentes de un mismo documento.

**align** Alineamiento de la regla. Valores posibles: *left*, *center*, *right*. Por defecto el valor adoptado es *center*.

**width** Tamaño horizontal de la regla. El valor por defecto es 100% (se ocupa a lo ancho la ventana de presentación).

**HTML** Raíz de todo documento HTML.

**I** Texto en cursiva.

**IMG** Permite insertar una imagen en un documento.

**alt** Texto alternativo a presentar al usuario si hay alguna clase de problema al cargar la imagen (por ejemplo, que no se encuentra en la localización indicada con *src*).

**height** Altura de imagen.

**src** URL con la localización de la imagen a ser insertada.

**width** Anchura de imagen.

**INPUT** Elemento utilizado para que el usuario pueda introducir información en un formulario (control).

**checked** Cuando el control es de tipo *radio* o *checkbox* y este atributo aparece y toma el valor *checked*, el control estará por defecto inicialmente activo.

**maxlength** Cuando el control es de tipo *text* o *password* este atributo indica la longitud máxima de la cadena de texto a introducir.

**name** Asigna un nombre al control. Este nombre se envía al servidor para que pueda reconocer los controles existentes.

**size** Indica el ancho inicial del control en *pixeles*. Cuando el control es de tipo *text* o *password* el número se refiere al ancho en caracteres. Si *maxlength* es mayor que *size*, el navegador debería proporcionar barras de desplazamiento para poder visualizar todo el control.

**src** Cuando el control es de tipo *image* este atributo indica cual es la localización de esa imagen.

**type** Tipo del control. Los valores aceptados son: *text*, *password*, *checkbox*, *radio*, *submit*, *reset*, *file*, *hidden*, *image* y *button*.

**value** Valor inicial del control. Es opcional salvo en controles de tipo *radio* y *checkbox*.

**LI** Elemento de una lista.

**LINK** Representa un enlace a un elemento externo, por ejemplo, a una hoja de estilo CSS. Puede aparecer únicamente en el *head* del documento.

**href** URL con la localización del recurso enlazado.

**rel** Describe de manera textual la relación existente entre el recurso enlazado y la página desde la que se enlaza. Así por ejemplo cuando *link* se utiliza para asociar hojas de estilo a los documentos, *rel* toma el valor *stylesheet*.

**type** Representa el tipo MIME (*Multipurpose Internet Mail Extensions*) de los datos que se almacenan en el recurso enlazado.

**META** Identifica propiedades de un documento (autor, fecha, etc.) y valores asociados a esas propiedades.

**content** Valor de la propiedad.

**name** Nombre de la propiedad.

**http-equiv** Puede ser utilizado en lugar de *name* y provoca que el servidor HTTP a la hora de elaborar la respuesta con el documento HTML, introduzca esos valores en la cabecera del protocolo.

**NOFRAMES** Establece el contenido alternativo que debe ser presentado al usuario en caso de que el navegador no soporte el empleo de marcos.

**NOSCRIPT** Indica el contenido alternativo que debe ser presentado al usuario en caso de que el navegador no soporte el empleo de *scripts* codificados en un lenguaje concreto (como por ejemplo *Javascript*).

**OBJECT** Permite incluir objetos en un documento. Un objeto no es más que un conjunto de datos que el navegador no sabe interpretar si no es mediante la utilización de una aplicación externa o *plug-in*.

**data** URL con la localización del objeto que debe ser incluido en el documento.

**type** Tipo MIME de los datos referenciados por el atributo *data*.

**OL** Lista ordenada.

**start** Establece el número del primer elemento de la lista (por si no queremos que empiece en 1).

**type** Estilo de representación de cada nodo de la lista. Los valores posibles son: *1* para números arábigos, *a* para letras minúsculas, *A* para letras mayúsculas, *i* para números romanos en minúsculas, *I* para números romanos en mayúsculas.

**OPTGROUP** Permite agrupar las opciones de un menú de manera lógica.

**OPTION** Cada una de las opciones de un menú.

**selected** Cuando este atributo aparece y toma el valor *selected*, la opción aparecerá por defecto inicialmente seleccionada.

**P** Representa un párrafo.

**align** Alineamiento del párrafo. Valores posibles: *left*, *center*, *right* y *justify*. El valor por defecto es *right*.

**PRE** Permite incluir texto preformateado. El navegador debe representar los contenidos de este elemento tal cual, sin cambiar por ejemplo el número de espacios en blanco o sustituir las entidades por sus valores.

**SCRIPT** Permite incluir trozos de código escritos en un lenguaje de *script* en el interior del documento HTML. Típicamente se utiliza para asociar código *Javascript* a los documentos. Puede aparecer tanto en el *head* como en el *body* de una página HTML.

**src** URL del código del *script* si éste es externo. Si el código es interno se almacena en el contenido del propio elemento.

**type** Tipo MIME asociado al lenguaje de *script*. Por ejemplo en *Javascript* es *text/javascript*.

**SELECT** Control de tipo menú.

**multiple** Cuando está presente y toma el valor *multiple* permite que se puedan seleccionar varios elementos en el menú a la vez.

**name** Nombre del control.

**SPAN** Usado para estructurar documentos. Los elementos que conformen su contenido estarán agrupados y se presentarán como un elemento *inline*. Diseñado para ser usado en conjunción con hojas de estilo.

**STYLE** Puede aparecer únicamente en el *head* de un documento. Su contenido consiste en declaraciones de estilo, típicamente en lenguaje CSS.

**type** Tipo MIME asociado al lenguaje en el que están escritas las reglas de estilo del elemento. Para CSS es *text/css*.

**SUB** Permite añadir un subíndice.

**SUP** Permite añadir un superíndice.

**TABLE** Utilizado para la definición de tablas.

**align** Alineamiento de la tabla en el documento. Los valores posibles son *left*, *center* y *right*. El valor por defecto es *center*.

**bgcolor** Color de fondo de las celdas de la tabla.

**border** Tamaño del borde de la tabla en *pixeles*.

**cellpadding** Espaciado entre el borde de una celda y su contenido. Las unidades más empleadas son *pixeles* o % del espacio total disponible.

**cellspacing** Espaciado que separa los bordes de la tabla de las celdas, así como las celdas entre si. Las unidades más empleadas son *pixeles* o % del espacio total disponible.

**TD** Define una celda que contiene datos en una tabla.

**align** Alineamiento del contenido de la celda respecto a los límites de la misma. Los valores más importantes son *left*, *center*, *right* y *justify*. El valor por defecto es *center*.

**bgcolor** Color de fondo de la celda.

**rowspan** Número de filas por las que se expande la celda. El valor 0 quiere decir que ocupa todas las filas hasta la última.

**colspan** Número de columnas por las que se expande la celda. El valor 0 quiere decir que ocupa todas las columnas hasta la última.

**TH** Define una celda que contiene información de cabecera (título de una fila o columna) dentro de una tabla.

**align** Alineamiento del contenido de la celda respecto a los límites de la misma. Los valores más importantes son *left*, *center*, *right* y *justify*. El valor por defecto es *center*.

**bgcolor** Color de fondo de la celda.

**rowspan** Número de filas por las que se expande la celda. El valor 0 quiere decir que ocupa todas las filas hasta la última.

**colspan** Número de columnas por las que se expande la celda. El valor 0 quiere decir que ocupa todas las columnas hasta la última.

**TR** Contenedor de celdas de una misma fila de una tabla.

**align** Alineamiento del contenido de las celdas de la fila respecto a los límites de las mismas. Los valores más importantes son *left*, *center*, *right* y *justify*. El valor por defecto es *center*.

**bgcolor** Color de fondo de las celdas de la fila.

**U** Texto subrayado.

**UL** Lista desordenada.

**type** Estilo de representación de cada nodo de la lista. Los valores posibles son: *circle* para circunferencias, *disc* para circunferencias rellenas, *square* para cuadrados.

Además de los atributos incluidos en esta sección, existen otros tres que por su importancia y amplio alcance merecen ser tratados de manera especial. Estos son:

**class** Se aplica a todos los elementos HTML excepto a *BASE*, *BASEFONT*, *HEAD*, *HTML*, *META*, *PARAM*, *SCRIPT*, *STYLE*, *TITLE*. Permite asignar uno o varios nombres de clase a un elemento en un documento. Varios elementos distintos pueden pertenecer a la misma clase. Esto puede ser usado para agrupar los elementos y asociarles la misma información de estilo a todos.

**id** Se aplica a todos los elementos HTML excepto a *BASE*, *HEAD*, *HTML*, *META*, *SCRIPT*, *STYLE*, *TITLE*. Permite asignar un identificador único a los elementos de un documento. Elementos distintos no pueden tener el mismo identificador. Esto puede ser usado para seleccionar un elemento concreto dentro del documento y asociarle información de estilo específica sin tener que usar el atributo *style*.

**style** Se aplica a todos los elementos HTML excepto a *BASE*, *BASEFONT*, *HEAD*, *HTML*, *META*, *PARAM*, *SCRIPT*, *STYLE*, *TITLE*. Permite asociar información de estilo específica (típicamente CSS) a un elemento concreto.

## B. Referencia de propiedades básicas CSS

**background-color** Color de fondo de los elementos seleccionados.

- Valores más usados: [Color]<sup>2</sup> o *transparent*.

**background-image** Imagen de fondo de los elementos seleccionados.

- Valores más usados: una URL con la localización de la imagen en formato *url(localización)*.

**color** Color del texto de los elementos seleccionados.

- Valores más usados: [Color].

**text-align** Establece cómo se alinea el texto de los elementos seleccionados.

- Valores más usados: *left*, *right*, *center*, *justify*.

**text-decoration** Decoración del texto de los elementos seleccionados.

- Valores más usados: *underline* (subrayado), *overline* (superrayado), *line-through* (con una línea tachándolo horizontalmente) y *blink* (parpadeando).

**text-indent** Indentación de la primera línea de texto de un bloque.

- Valores más usados: [Tamaño]<sup>3</sup>

---

<sup>2</sup>[Color] representa un color cualquiera especificado utilizando los mecanismos expuestos en 5.2.1

<sup>3</sup>[Tamaño] se corresponde con lo definido en 5.2.2: un valor absoluto o relativo o un porcentaje.

**text-transform** Transformación aplicada al texto de los elementos seleccionados.

- Valores más usados: *uppercase* (poner en mayúscula), *lowercase* (poner en minúscula) y *capitalize* (poner la primera letra en mayúscula).

**font** Permite establecer el tamaño, tipo y estilo (entre otras propiedades) de la fuente que será empleada para representar el texto de los elementos seleccionados.

- Valores más usados: su valor será la concatenación de los valores concretos de tamaño, tipo y estilo (suponiendo que sólo se establezcan estas tres características de la fuente). Así por ejemplo valores válidos de esta propiedad serían: `{font: bold italic large Courier;}` o `{font: 12pt sans-serif;}`.

**font-family** Especifica una lista de posibles familias de fuentes que pueden ser empleadas para representar el texto de los elementos seleccionados.

- Valores más usados: pueden ser nombres concretos de familias de fuentes (como *Courier*, *Arial*, etc) o nombres genéricos de familias de fuentes (como *serif*, *sans-serif*, *cursive*, etc.). La lista estará compuesta por una serie de estos valores separados por comas. En caso de incluir nombres concretos y genéricos, los concretos deben especificarse con anterioridad, puesto que los navegadores elegirán como familia la primera de la lista que puedan utilizar.

**font-size** Permite especificar el tamaño de la fuente utilizada para representar el contenido de los elementos seleccionados.

- Valores más usados: [Tamaño], *xx-small*, *x-small*, *small*, *medium*, *large*, *x-large*, *xx-large* cuyo valor dependerá de la implementación del navegador, siendo la única restricción que los valores están ordenados de manera creciente (es decir, que el tamaño asociado a *xx-small* sea menor que el de *x-small*, éste a su vez menor que el de *small* y así sucesivamente).

**font-style** Estilo de la fuente que se utilizará para representar el texto de los elementos seleccionados.

- Valores más usados: *normal*, *italic* (cursiva) y *oblique* (cursiva inversa, doblada hacia el otro sentido).

**font-weight** Especifica el “peso” de la fuente, si esta es más clara o más oscura.

- Valores más usados: *normal*, *bold*, *bolder*, *lighter*, *100*, *200*,..., *900*.

**display** Opciones de visualización de los elementos.

- Valores más usados: *inline* y *block* cuyo significado se puede deducir leyendo la sección 2.1.

**border** Mecanismo abreviado de definir el ancho, estilo y color de los cuatro bordes de la caja que encierra a un elemento a la hora de representarlo.

- Valores más usados: Se concatenan los valores de ancho, estilo y color. Así un posible ejemplo de valor válido de esta propiedad sería: `{border: thick solid red;}`

**border-bottom-width** Ancho del borde inferior de la caja que encierra a un elemento a la hora de representarlo.

- Valores más usados: Longitud (no porcentaje) o uno de los siguientes valores: *thin* (delgado), *medium*, *thick* (grueso).

**border-bottom-style** Estilo del borde inferior de la caja que encierra a un elemento a la hora de representarlo.

- Valores más usados:

**dotted** El borde es una serie de puntos.

**dashed** El borde está formado por segmentos cortos de recta.

**solid** Borde sólido.

**double** El borde está compuesto por dos líneas y un espacio entre ellas.

**groove** El borde parece que está introducido en la ventana.

**ridge** El borde parece que está saliendo de la ventana del navegador.

**inset** El borde hace parecer que la caja está introducida en la ventana.

**outset** El borde hace parecer que la caja que encierra está saliendo de la ventana del navegador.

**border-bottom-color** Color del borde inferior de la caja que encierra a un elemento a la hora de representarlo.

- Valores más usados: [Color].

**border-top-width** Ancho del borde superior de la caja que encierra a un elemento a la hora de representarlo.

- Valores más usados: [ver *border-bottom-width*]

**border-top-style** Estilo del borde superior de la caja que encierra a un elemento a la hora de representarlo.

- Valores más usados: [ver *border-bottom-style*]

**border-top-color** Color del borde superior de la caja que encierra a un elemento a la hora de representarlo.

- Valores más usados: [ver *border-bottom-color*].

**border-left-width** Ancho del borde izquierdo de la caja que encierra a un elemento a la hora de representarlo.

- Valores más usados: [ver *border-bottom-width*]

**border-left-style** Estilo del borde izquierdo de la caja que encierra a un elemento a la hora de representarlo.

- Valores más usados: [ver *border-bottom-style*]

**border-left-color** Color del borde izquierdo de la caja que encierra a un elemento a la hora de representarlo.

- Valores más usados: [ver *border-bottom-color*]

**border-right-width** Ancho del borde derecho de la caja que encierra a un elemento a la hora de representarlo.

- Valores más usados: [ver *border-bottom-width*]

**border-right-style** Estilo del borde derecho de la caja que encierra a un elemento a la hora de representarlo.

- Valores más usados: [ver *border-bottom-style*]

**border-right-color** Color del borde derecho de la caja que encierra a un elemento a la hora de representarlo.

- Valores más usados: [ver *border-bottom-color*]

**margin** Permite especificar el tamaño de los márgenes de la caja que encierra al elemento. Es una notación abreviada de *margin-top*, *margin-bottom*, *margin-left* y *margin-right*.

- Valores más usados: Concatenación de 1 a 4 [Tamaños]. Si sólo aparece uno, ese se aplica a todos los márgenes, si aparecen 2, el primero se aplica a los márgenes superior e inferior y el segundo a los otros, si aparecen 3, el primero se aplica al margen superior, el segundo a los márgenes derecho e izquierdo y el tercero al margen inferior. Por último, si aparecen los cuatro tamaños, se van aplicando a los márgenes en el siguiente orden: *top*, *right*, *bottom* y *left*.

**margin-bottom** Tamaño del margen inferior de la caja que encierra al elemento.

- Valores más usados: [Tamaño]

**margin-top** Tamaño del margen superior de la caja que encierra al elemento.

- Valores más usados: [Tamaño]

**margin-left** Tamaño del margen izquierdo de la caja que encierra al elemento.

- Valores más usados: [Tamaño]

**margin-right** Tamaño del margen derecho de la caja que encierra al elemento.

- Valores más usados: [Tamaño]

**padding** Tamaño del relleno de la caja que encierra al elemento.

- Valores más usados: [ver *margin*]

**padding-bottom** Tamaño del relleno inferior de la caja que encierra al elemento.

- Valores más usados: [Tamaño]

**padding-top** Tamaño del relleno superior de la caja que encierra al elemento.

- Valores más usados: [Tamaño]

**padding-left** Tamaño del relleno izquierdo de la caja que encierra al elemento.

- Valores más usados: [Tamaño]

**padding-right** Tamaño del relleno derecho de la caja que encierra al elemento.

- Valores más usados: [Tamaño]

**list-style-type** Permite especificar el tipo de marcador que se utilizará para señalar los elementos de una lista.

- Valores más usados:

**decimal** Número decimales empezando en 1.

**decimal-leading-zero** 01, 02, 03, ..., 99

**lower-roman** i, ii, iii, ...

**upper-roman** I, II, III, ...

**lower-latin o lower-alpha** a,b,c, ...

**upper-latin o upper-alpha** A, B, C, ...

**lower-greek** Letras minúsculas del alfabeto griego clásico.

**disc, circle o square** Con el mismo significado que el del atributo *type* del elemento *ul* de HTML (ver A).

## Referencias

- [1] “HTML 4.01 Specification”  
<http://www.w3.org/TR/html4>
- [2] “XHTML 1.0 The Extensible HyperText Markup Language (Second Edition)”  
<http://www.w3.org/TR/xhtml1>
- [3] “XHTML 1.1 - Module-based XHTML”  
<http://www.w3.org/TR/xhtml11>
- [4] “XHTML Basic”  
<http://www.w3.org/TR/xhtml-basic>
- [5] “XHTML 2.0 (Working Draft, July 26 2006)”  
<http://www.w3.org/TR/xhtml2>
- [6] “Cascading Style Sheets, level 1”  
<http://www.w3.org/TR/REC-CSS1>
- [7] “Cascading Style Sheets, level 2. CSS2 Specification”  
<http://www.w3.org/TR/REC-CSS2>