
El texto central de HTML

PID_00253482

Ben Buchanan
Jenifer Hanen
Mark Norman Francis
Christian Heilmann

Tiempo mínimo previsto de lectura y comprensión: 9 horas



Universitat
Oberta
de Catalunya

Índice

1. Etiquetar contenido textual en HTML	7
1.1. El espacio: la última frontera	7
1.2. Elementos de bloque	8
1.2.1. Títulos de sección de página	8
1.2.2. Párrafos genéricos	9
1.2.3. Citar otras fuentes	9
1.2.4. Texto preformateado	10
1.3. Elementos en línea	10
1.3.1. Citas breves	11
1.3.2. Énfasis	11
1.3.3. Texto en cursiva	12
1.4. Elementos presentacionales: no utilizar nunca	13
Resumen	14
2. Listas HTML	15
2.1. Los tres tipos de listas	15
2.1.1. Listas no ordenadas	15
2.1.2. Etiquetado de listas no ordenadas	16
2.1.3. Listas ordenadas	16
2.1.4. Etiquetado de listas ordenadas	17
2.1.5. Comenzar listas ordenadas con un número diferente de 1	18
2.1.6. Listas de definiciones	19
2.2. Elegir entre tipos de listas	21
2.3. La diferencia entre las listas y el texto en HTML	21
2.4. Anidar listas	22
2.5. Ejemplo paso a paso	23
2.5.1. Etiquetado de la página principal	23
2.5.2. Añadir un poco de estilo	25
2.5.3. La página de la receta	26
2.5.4. Etiquetado de la página de la receta	27
2.5.5. Estilos de la página de la receta	28
Resumen	29
Preguntas de repaso	29
Lecturas complementarias	30
3. Imágenes en HTML	31
3.1. Una imagen dice más que mil palabras, ¿o no?	31
3.2. Diferentes tipos de imágenes en la web: imágenes de contenido y de fondo	32
3.3. El elemento <code>img</code> y sus atributos	32
3.3.1. Ofrecer una alternativa de texto con el atributo <code>alt</code>	33

3.3.2. Añadir información pertinente con el atributo <code>title</code> ...	34
3.3.3. Visualización más rápida de las imágenes definiendo las dimensiones con <code>width</code> y <code>height</code>	35
3.4. Las imágenes en línea	36
3.5. Imágenes de fondo con CSS	36
3.5.1. Aplicación de fondos con CSS	37
Resumen	38
Preguntas de repaso	39
4. Enlaces HTML: ¡construyamos una web!	40
4.1. ¿Qué son los enlaces?	40
4.2. La anatomía de un enlace ancla	40
4.3. ¿Enlace o destino? Los atributos <code>id</code> y <code>href</code>	41
4.4. No permitáis ninguna ambigüedad en vuestros enlaces	44
4.4.1. Ofrecer información adicional con un atributo <code>title</code> ...	44
4.4.2. Enlaces a recursos no HTML: no obliguéis a los visitantes a hacer conjeturas	46
4.4.3. Enlaces externos y enlaces internos	47
4.5. Marcos y ventanas emergentes: ¡ni hablar!	47
4.6. Ventajas de los enlaces de salida y de los enlaces de entrada	48
4.7. Descripción de los enlaces	49
4.8. Estilos de los enlaces	50
Resumen	50
Preguntas de repaso	51
5. Tablas HTML	52
5.1. La tabla más básica	52
5.2. Añadir otras funciones	54
5.3. Seguir estructurando la tabla	55
5.4. El CSS nos ayuda: una tabla con un aspecto mejor	57
Resumen	59
Preguntas de repaso	60
Lecturas complementarias	60
6. Formularios HTML: conceptos básicos	61
6.1. Paso uno: el código básico	61
6.2. Paso dos: añadir estructura y comportamiento	64
6.3. Paso tres: añadir semántica, estilo y un poco más de estructura	68
Resumen	70
Preguntas de repaso	70
Lecturas complementarias	71
7. Elementos semánticos menos conocidos	72
7.1. Destacar la información de contacto	72
7.2. Lenguajes y código de programación	73

7.3. Mostrar la interacción con ordenadores	74
7.4. Variables	75
7.5. Citas	75
7.6. Abreviaturas	76
7.7. Instancias definidoras	76
7.8. Superíndice y subíndice	78
7.9. Saltos de línea	78
7.10. Líneas horizontales	79
7.11. Cambios en los documentos (inserciones y supresiones)	80
Resumen	81
8. Contenedores genéricos: los elementos div y span	82
8.1. Semánticamente neutros	82
8.2. En línea y de bloque	82
8.3. Más explicaciones sobre la agrupación de contenido	83
8.4. Información adicional	86
8.5. Enganches para JavaScript, y también para CSS	87
8.6. “div-itis”	87
8.7. Semántica inadecuada	87
8.7.1. Párrafos “genéricos”	87
Resumen	88
Preguntas de repaso	88
9. Creación de múltiples páginas con menús de navegación	89
9.1. Las herramientas de vuestro menú HTML:	
enlaces, anclas y listas	89
9.2. La necesidad de flexibilidad	90
9.3. Diferentes tipos de menús	91
9.3.1. Navegación por la página (tabla de contenidos)	91
9.3.2. Navegación por el sitio	92
9.3.3. Menús contextuales	93
9.3.4. Mapas del sitio	94
9.3.5. Paginación	95
9.4. Cuando las listas no son suficientes: mapas de imagen y formularios	95
9.4.1. Definir zonas sensibles con mapas de imagen	96
9.4.2. Ahorrar espacio de pantalla y evitar la sobrecarga de enlaces con formularios	97
9.5. Dónde poner el menú, y ofrecer opciones para saltárselo	98
Resumen	99
Preguntas de repaso	100
10. Validar el HTML	101
10.1. Errores	101
10.2. ¿Qué es la validación?	102
10.3. ¿Por qué validar?	102

10.4.Los diferentes navegadores interpretan el HTML no válido	
de una manera diferente	103
10.4.1.El “quirks mode”	104
10.5.Cómo validar las páginas	104
10.5.1.El validador de HTML del W3C	106
Resumen	108
Preguntas de repaso	108
Otras herramientas que podéis utilizar	108

1. Etiquetar contenido textual en HTML

Mark Norman Francis

En este apartado explicaremos los conceptos básicos del uso de HTML para describir el significado del contenido dentro del cuerpo de vuestro documento.

Veremos elementos estructurales generales como los títulos y los párrafos y la incrustación de citas y código. Después también veremos el contenido en línea, como por ejemplo las citas cortas y el énfasis, y acabaremos con un análisis rápido del contenido presentacional anticuado.

Nota

Después de cada ejemplo de código hay un enlace “Ved ejemplos reales” que os llevará a la visualización del resultado de este código fuente en un archivo diferente; de esta manera, podréis ver ejemplos reales de cómo se ve realmente este código fuente en el navegador, aparte del código en sí.

1.1. El espacio: la última frontera

Una cuestión muy importante que queremos tratar antes de empezar a hablar sobre el texto es la del espacio, y concretamente la del **espacio entre palabras**. Cuando se escribe el HTML, el documento fuente contendrá lo que se conoce como “espacios en blanco”, que son los caracteres del archivo que sirven para separar el texto. El carácter de espacio más habitual es el que se obtiene al pulsar la barra espaciadora, pero hay otros, como el tabulador y la marca entre dos líneas independientes de un documento (conocido como retorno o línea nueva).

En el HTML, cuando un carácter de éstos aparece varias veces seguidas, se considera (casi) siempre como un único carácter de espacio.

Por ejemplo, un navegador interpretaría lo siguiente:

```
<h3>In    the  
beginning</h3>
```

Ved ejemplos reales en: “Whitespace example”

http://mosaic.uoc.edu/ac/le/operafiles/HTMLtext_examples.html#whitespace

como equivalente a:

```
<h3>In the beginning</h3>
```

El único caso en el que esto no es así es el elemento `pre`, del cual hablaremos con detalle más adelante en este mismo apartado.

Esto puede provocar algunas confusiones para los autores que escriben por primera vez un documento en HTML y que intentan introducir espacios adicionales dentro del texto para conseguir el sangrado deseado, dejar más espacio

después de un punto entre frases o añadir más espacio vertical entre párrafos. El diseño visual de los documentos no se debe definir en el código fuente HTML, sino que se debe hacer mediante las hojas de estilos, de las que hablaremos más adelante en esta serie de apartados.

1.2. Elementos de bloque

En este subapartado explicaremos la sintaxis y el uso de los elementos de bloque más frecuentes* utilizados para formatear texto.

* <http://mosaic.uoc.edu/ac/le/ca/m3/ud1/index.html#blockinline>

1.2.1. Títulos de sección de página

Una vez que hayáis dividido la página en secciones lógicas, cada una de estas secciones debería ir introducida por un título adecuado. De ello se habla con más detalle en el apartado “Qué necesita una buena página web”.

Podéis ver el apartado 2 del módulo “Conceptos de diseño web”.

El HTML define seis niveles de títulos: h1, h2, h3, h4, h5 y h6 (desde el más importante hasta el menos importante). Hablando en general, h1 sería el título principal de toda la página que introduciría el tema. h2 se utilizaría para dividir la página en secciones, h3 en subsecciones y así sucesivamente.

Es muy importante utilizar los niveles de títulos para describir el documento en referencia a las secciones, subsecciones, subsubsecciones, etc., ya que esto hace que el documento sea más comprensible para los lectores de pantalla* y para los procesos automatizados (como los robots de indexación de Google).

* <http://www.accessibilitytips.com/2008/03/10/avoid-skipping-header-levels/>

Un buen ejemplo de una estructura de títulos, utilizando este documento como plantilla, podría ser el siguiente:

```
<h1>Etiquetar contenido textual en HTML</h1>
<h2>Introducción</h2>
<h2>El espacio: la última frontera</h2>
<h2>Elementos de bloque</h2>
<h3>Títulos de sección de página</h3>
<h3>Párrafos genéricos</h3>
<h3>Citar otras fuentes</h3>
<h3>Texto preformateado</h3>
<h2>Elementos en línea</h2>
```

Ved ejemplos reales en: “Headings example”

http://mosaic.uoc.edu/ac/le/operafiles/HTMLtext_examples.html#headings

1.2.2. Párrafos genéricos

El párrafo es el componente básico de la mayoría de los documentos. En HTML, un párrafo se representa con el elemento `p`, que no tiene ningún atributo especial.

Por ejemplo:

```
<p>This is a very short paragraph. It only has two sentences.</p>
```

Ved ejemplos reales en: “Paragraph example”

http://mosaic.uoc.edu/ac/le/operafiles/HTMLtext_examples.html#paragraph

En muchos artículos y libros, un párrafo puede contener sólo una frase. Aunque el significado (en cuanto a la prosa escrita) de la palabra *párrafo* está bastante claro, en la web hay áreas de texto mucho más cortas que a menudo aparecen entre elementos de párrafos por el simple hecho de que el autor cree que es más “semántico” que utilizar un elemento `div` (hablaremos de ello en otro apartado titulado “Contenedores genéricos”).

Ejemplo

Un párrafo es un conjunto de una o más frases, igual que en los periódicos y libros. En la web es una buena idea utilizar el elemento de párrafo para estos conjuntos de frases y no para cualquier parte aleatoria de texto de la página. Si son sólo unas cuantas palabras y no una frase propiamente dicha, entonces quizá sería mejor no etiquetarlo como un párrafo.

1.2.3. Citar otras fuentes

A menudo, los artículos, los apuntes de bloques y los documentos de referencia citan total o parcialmente algún otro documento. En el HTML, estas citas se marcan con el elemento `blockquote` para citas largas, como frases enteras, párrafos, listas o similares.

Si la cita es de otra página web, podéis indicarlo utilizando el atributo `cite`.

Como en el ejemplo siguiente:

```
<p>HTML 4.01 is the only version of HTML that you should use when creating a new web page, as, according to the specification:</p>
<blockquote cite="http://www.w3.org/TR/html401/">
<p>This document obsoletes previous versions of HTML 4.0, although W3C will continue to make those specifications and their DTDs available at the W3C Web site.</p>
</blockquote>
```

Ved ejemplos reales en: “Quoting other sources example”

http://mosaic.uoc.edu/ac/le/operafiles/HTMLtext_examples.html#cite

El atributo `cite` no es necesario en el caso de que la cita se tome de una novela, una revista o de alguna otra forma de contenido fuera de línea.

1.2.4. Texto preformateado

Cualquier texto en el que el formato y el espacio en blanco (podéis ver más arriba) sean importantes se debería etiquetar con el elemento `pre`.

En la mayoría de los navegadores web, el texto marcado como preformateado se mostrará al usuario tal como aparezca en el código fuente, algunas veces utilizando un tipo de letra de ancho fijo, lo cual da al texto el aspecto de haber sido escrito con una máquina de escribir. Este uso de tipos de letras de ancho fijo es uno de los primeros recursos que utilizaron los programadores para el texto preformateado.

En éste podéis ver un fragmento de código escrito en el lenguaje de programación Perl:

```
1  <pre><code class="language-perl">
2  # read in the named file in its entirety
3  sub slurp {
4      my $filename = shift;
5      my $file      = new FileHandle $filename;

6      if ( defined $file ) {
7          local $/;
8          return <$file>;
9      }
10     return undef;
11 };
12 </code></pre>
```

Ved ejemplos reales en: "Preformatted example"

http://mosaic.uoc.edu/ac/le/operafiles/HTMLtext_examples.html#preformatted

El uso de `code` que se hace en este ejemplo se explicará en el apartado sobre los elementos semánticos menos conocidos, que encontraremos más adelante en este mismo módulo.

Podéis ver el apartado 7 de este módulo.

1.3. Elementos en línea

En este subapartado explicaremos la sintaxis y el uso de los elementos en línea más frecuentes utilizados para formatear texto.

Podéis ver el apartado 1 del módulo "Fundamentos de HTML".

1.3.1. Citas breves

Las citas breves que se utilizan en una frase o párrafo normal se ponen en el elemento `q`. Igual que el elemento `blockquote`, éste puede contener un atributo `cite`, que indica la página de Internet en la que se puede encontrar la cita.

Una cita breve aparece normalmente entre comillas. Según la especificación HTML*, estas citas las debe insertar el agente usuario para poder imbricarlas correctamente y que sepan el idioma que se utiliza en el documento. Se puede utilizar CSS para controlar las comillas utilizadas; este aspecto se tratará en el apartado sobre estilos de texto.

* <http://www.w3.org/TR/html401/struct/text.html#h-9.2.2.1>

Podéis ver el apartado 3 del módulo "Conceptos básicos de CSS".

Un ejemplo de `q` en acción:

```
<p>This did not end well for me. Oh well,  
<q lang="fr">c'est la vie</q> as the French say.</p>
```

Ved ejemplos reales en: "Inline quote example"

http://mosaic.uoc.edu/ac/le/operafiles/HTMLtext_examples.html#q

1.3.2. Énfasis

El HTML contiene dos métodos para indicar que el texto de su interior debe aparecer enfatizado delante del usuario, como por ejemplo los mensajes de error, las advertencias o las notas.

En los navegadores visuales esto significa normalmente la aplicación de un color o un tipo de letra diferente o la visualización del texto en negrita o cursiva. Para los usuarios de lectores de pantalla, el resultado puede ser una voz diferente o algún otro efecto acústico. Para presentar texto con énfasis hay que utilizar el elemento `em`.

Como en el ejemplo siguiente:

```
<p><em>Please note:</em> the kettle is to be unplugged  
at night.</p>
```

Ved ejemplos reales en: "Emphasis example"

http://mosaic.uoc.edu/ac/le/operafiles/HTMLtext_examples.html#em

Si hay que enfatizar toda una frase pero una parte concreta de ésta se debe enfatizar de una manera aún más importante, se pueden anidar elementos `em` para indicar un énfasis más fuerte de lo normal, como en el ejemplo siguiente:

```
<p><em>Please note: the kettle <em>must</em> be  
unplugged every evening, otherwise it will explode -  
<em>killing us all</em></em>.</p>
```

Ved ejemplos reales en: “Emphasis and strong example”

http://mosaic.uoc.edu/ac/le/operafiles/HTMLtext_examples.html#emstrong

1.3.3. Texto en cursiva

Originalmente el elemento `i` era puramente presentacional: significaba exclusivamente que el texto que rodeaba debía mostrarse en letra cursiva. Pero la especificación de HTML5 dice:

“El elemento `i` representa un texto en una voz o un modo alternativo, o que de alguna manera queda fuera de la prosa normal.”

La especificación también dice explícitamente que los autores deben considerar si es posible aplicar otros elementos que, como `em`, que se ajusten mejor semánticamente a aquello que se quiere expresar.

Un uso habitual de `i` es marcar términos en otros idiomas. En ese caso usaremos el atributo `lang` para especificar dicho idioma:

Como dicen los franceses, `<i lang="fr">c'est la vie</i>...`

También se aconseja emplear el elemento `class` para indicar el uso específico que se hace de `i` (y así poder presentar mediante el CSS usos diferentes de maneras distintas). Hay que recalcar que, si bien la presentación por defecto de `i` es la letra cursiva, con el CSS podremos indicar cualquier otra presentación (por ejemplo, con fondo amarillo, como si hubiésemos marcado el texto con un rotulador).

De la misma forma que `i`, `b` era inicialmente un elemento presentacional que indicaba el uso de negrita. Pero la especificación de HTML dice:

El elemento `b` representa una secuencia de texto a la que se quiere dirigir la atención por motivos utilitarios sin darle ninguna importancia extra y sin implicar una voz o modo alternativos, como, por ejemplo, palabras clave en el resumen de un documento, nombres de productos en una crítica, palabras sobre las que se puede actuar en un juego de texto, o la entradilla de un artículo.

Como en el caso de `i`, es conveniente usar clases para indentificar por qué se usa el elemento, y aplicar los estilos correspondientes.

La especificación es dura en cuanto al uso de `b`:

...debería usarse como último recurso cuando ningún otro elemento sea más adecuado. En particular, los encabezamientos deberían usar los elementos `h1` a `h6`, el énfasis debería usar el elemento `em`, la importancia debería denotarse con `strong`, y el texto marcado o destacado debería usar el elemento `mark`.

1.4. Elementos presentacionales: no utilizar nunca

La especificación de HTML incluye varios elementos que se describen de manera general como “presentacionales”, ya que sólo especifican qué aspecto debe tener el texto que contienen, pero no qué significan.

La especificación ha etiquetado algunos de estos elementos como desaprobados. Eso significa que han sido sustituidos por un método más nuevo que permite conseguir los mismos resultados.

Aquí los describiremos brevemente, pero hay que tener en cuenta que esta explicación tiene casi exclusivamente un interés puramente histórico; estos elementos no se deben utilizar nunca en ninguna página web moderna. El efecto de todos estos elementos se debe conseguir de alguna otra manera, que se describirá en otros dos apartados: “Estilos de texto con el CSS” y “Elementos semánticos menos conocidos”.

Podéis ver el apartado 3 del módulo “Conceptos básicos de CSS” y el apartado 7 de este módulo.

- `font face=“...” size=“...”`. El navegador mostrará el texto en estos elementos utilizando un tipo de letra diferente del tipo por defecto; los tipos de letra, sin embargo, se deben definir con CSS.
- `strike`. El texto aparece rayado con una línea que lo atraviesa; si es sólo un efecto presentacional, este efecto se debería conseguir con CSS. Además, si el texto se quiere marcar como suprimido o no deseado, se debería etiquetar con el elemento `del` que se describe más adelante.
- `tt`. El texto se presenta en un tipo de letra de “teletipo” o de ancho fijo; este efecto se debería conseguir con el CSS o con algún elemento semántico más apropiado, como por ejemplo `pre`, tal como se ha visto anteriormente.
- `big`. Se ajusta el tamaño del texto; se debería conseguir con el CSS.

Resumen

En este apartado hemos hablado de algunos de los elementos más comunes que se utilizan para etiquetar contenido textual. En el siguiente apartado hablaremos de otro tipo de contenido: las listas.

2. Listas HTML

Ben Buchanan

Las listas se utilizan para agrupar datos relacionados para que queden claramente asociadas entre ellas y sean fáciles de leer. En el desarrollo de las webs modernas, las listas son elementos muy habituales que se utilizan tanto para la navegación, como para el contenido general.

Las listas son muy adecuadas desde un punto de vista estructural, ya que ayudan a crear un documento bien estructurado, más accesible y fácil de mantener. También son útiles por una razón puramente práctica: aportan elementos adicionales a los que adjuntar estilos CSS y disponer así de una gran variedad de estilos (más adelante en esta asignatura ya hablaremos del CSS; para saber cuáles son los apartados sobre el CSS que hay disponibles, podéis consultar el índice de contenidos).

En este apartado explicaremos los diferentes tipos de listas disponibles en HTML, cuándo y cómo se deben utilizar y cómo aplicar algunos estilos básicos.

2.1. Los tres tipos de listas

En el HTML hay tres tipos de listas:

- **lista no ordenada:** se utiliza para agrupar elementos relacionados sin ningún orden en concreto,
- **lista ordenada:** se utiliza para agrupar elementos relacionados en un orden concreto,
- **lista de definiciones:** se utiliza para mostrar parejas de nombres y valores, como por ejemplo términos y sus definiciones u horas y actos.

Cada una de estas listas tiene una finalidad y un significado específicos y no son intercambiables.

2.1.1. Listas no ordenadas

Las listas no ordenadas, o listas de picos, se utilizan cuando tenemos una serie de elementos que se pueden colocar en cualquier orden.

La lista de la compra

Un ejemplo de esto podría ser una lista de la compra:

- leche
- pan

- mantequilla
- café

Todos estos elementos forman parte de una lista, pero se pueden poner en cualquier orden sin que la lista pierda su sentido:

- pan
- café
- leche
- mantequilla

Se puede utilizar CSS para cambiar el pico a uno de los distintos estilos por defecto, puede usarse una imagen propia o incluso se pueden hacer listas sin picos; más adelante, en este mismo apartado, veremos cómo hacerlo y en otros apartados sobre el CSS lo explicaremos con más detalle. Si lo encontráis un poco confuso, no es necesario que os preocupéis por el CSS ahora; ya lo aclararemos todo más adelante.

2.1.2. Etiquetado de listas no ordenadas

Las listas no ordenadas utilizan las etiquetas `` alrededor de múltiples grupos de ``.

```
<ul>
  <li>pan</li>
  <li>café</li>
  <li>leche</li>
  <li>mantequilla</li>
</ul>
```

2.1.3. Listas ordenadas

Las listas ordenadas, o listas numeradas, se utilizan cuando tenemos una serie de elementos que se deben colocar en un orden concreto.

Un ejemplo de esto podrían ser las instrucciones de cocina, que se deben seguir por orden para que la receta salga bien:

1. Tened todos los ingredientes a mano
2. Mezclad todos los ingredientes
3. Poned la mezcla en una bandeja para el horno
4. Dejadlo cocer durante una hora
5. Sacadlo del horno
6. Dejadlo reposar durante diez minutos
7. Servidlo

Si los elementos de la lista se desordenan, la información no tendrá ningún sentido:

1. Tened todos los ingredientes a mano
2. Dejadlo cocer durante una hora
3. Sacadlo del horno
4. Servidlo
5. Poned la mezcla en una bandeja para el horno
6. Dejadlo reposar durante diez minutos
7. Mezclad todos los ingredientes

Las listas ordenadas se pueden mostrar con varios sistemas de numeración o alfabéticos; es decir, letras o números. El sistema por defecto de la mayoría de los navegadores son los números decimales, pero hay más opciones:

- Letras
 - Letras ASCII en minúsculas (a, b, c...)
 - Letras ASCII en mayúsculas (A, B, C...)
 - Letras griegas clásicas en minúsculas: (? , ? , ?...)
- Números
 - Números decimales (1, 2, 3...)
 - Números decimales con ceros iniciales (01, 02, 03...)
 - Números romanos en minúsculas (i, ii, iii...)
 - Números romanos en mayúsculas (I, II, III ...)
 - Numeración georgiana tradicional (an, ban, gan...)
 - Numeración armenia tradicional (mek, yerku, yerek...)

Aquí también podéis utilizar el CSS para cambiar el estilo de vuestras listas.

2.1.4. Etiquetado de listas ordenadas

Las listas ordenadas utilizan las etiquetas `` alrededor de múltiples grupos de ``.

```
<ol>
  <li>Tened todos los ingredientes a mano</li>
  <li>Mezclad todos los ingredientes</li>
  <li>Poned la mezcla en una bandeja para el horno</li>
  <li>Dejadlo cocer durante una hora</li>
  <li>Sacadlo del horno</li>
  <li>Dejadlo reposar durante diez minutos</li>
  <li>Servidlo</li>
</ol>
```

2.1.5. Comenzar listas ordenadas con un número diferente de 1

Es posible hacer que una lista ordenada empiece con un número diferente del 1 (o i, I, etc.). Esto se consigue con el atributo `start`, que define un valor numérico incluso aunque utilicéis el CSS para cambiar los contadores de la lista y hacer que sean caracteres alfabéticos o números romanos con la propiedad `list-style-type`. Esto puede ser útil si tenéis una única lista de puntos pero queréis romperla con una nota u otro tipo de información relacionada.

Lo podríamos hacer, por ejemplo, con la lista anterior:

```
<ol>
  <li>Tened todos los ingredientes a mano</li>
  <li>Mezclad todos los ingredientes</li>
  <li>Poned la mezcla en una bandeja para el horno</li>
</ol>
<p class="note">Antes de poner la mezcla en la bandeja,
precalentad el horno a 180 grados centígrados para
tenerlo a punto para el siguiente paso</p>
<ol start="4">
  <li>Dejadlo cocer durante una hora</li>
  <li>Sacadlo del horno</li>
  <li>Dejadlo reposar durante diez minutos</li>
  <li>Servidlo</li>
</ol>
```

Con ello se obtiene el resultado siguiente:

1. Tened todos los ingredientes a mano
2. Mezclad todos los ingredientes
3. Poned la mezcla en una bandeja para el horno

Antes de poner la mezcla en la bandeja, precalentad el horno a 180 grados centígrados para tenerlo a punto para el siguiente paso

4. Dejadlo cocer durante una hora
5. Sacadlo del horno
6. Dejadlo reposar durante diez minutos
7. Servidlo

El atributo `start` es un buen ejemplo de cómo cambian las cosas entre especificaciones de HTML: era válido en HTML3, fue desaprobado en la última versión de las especificaciones de HTML4, y ha sido añadido de nuevo en HTML5; afortunadamente, pues resulta útil.

2.1.6. Listas de definiciones

Las listas de definiciones asocian elementos concretos y sus definiciones en un formato de lista. Por ejemplo, si queréis dar una definición de los elementos de vuestra lista de la compra, podéis hacerlo de la siguiente manera:

```
leche
    Producto lácteo líquido de color blanco
pan
    Alimento cocinado en el horno a base de harina
mantequilla
    Producto lácteo sólido de color amarillo
café
    Semillas de los frutos de algunos árboles del café
```

Cada definiciones y término es un grupo de definición (o grupo nombre-valor). Podéis tener tantos grupos de definición como queráis, pero debe haber como mínimo un término y una definición en cada uno de los grupos. No es posible tener un término sin ninguna definición o una definición sin ningún término.

Podéis asociar más de un término a una única definición, o viceversa. Por ejemplo, el término *café* puede tener varios significados, que podéis mostrar uno después del otro:

```
café
    bebida hecha a partir de granos de café tostados y molidos
    una taza de café
    una reunión social en la que se consume café
    un color marrón tirando a oscuro
```

También es posible tener más de un término para la misma definición. Esto se utiliza para mostrar variaciones de un término, todas ellas con el mismo significado:

```
soda
gaseosa
bebida gaseosa
cola
    Bebida dulce con gas
```

Las listas de definiciones son diferentes de los otros tipos de listas, ya que utilizan términos y descripciones en lugar de elementos de lista.

Así pues, las listas de definiciones utilizan las etiquetas `<dl>`/`</dl>` alrededor de grupos de `<dt>`/`</dt>` y `<dd>`/`</dd>`: Se debe aparejar como mínimo un `<dt>`/`</dt>` con como mínimo un `<dd>`/`</dd>`; el `<dt>`/`</dt>` debe ir siempre primero.

Una lista de definiciones muy sencilla con términos únicos y definiciones únicas tendría el siguiente aspecto:

```
<dl>
  <dt>Término</dt>
  <dd>Definición del término</dd>
  <dt>Término</dt>
  <dd>Definición del término</dd>
  <dt>Término</dt>
  <dd>Definición del término</dd>
</dl>
```

Y se representa de la manera siguiente:

```
Término
  Definición del término
Término
  Definición del término
Término
  Definición del término
```

En este ejemplo hemos asociado más de un término a una definición, y viceversa:

```
1  <dl>
2    <dt>Término</dt>
3    <dd>Definición del término</dd>
4    <dt>Término</dt>
5    <dt>Término</dt>
6    <dd>Definición aplicable a los dos términos
      anteriores</dd>
7    <dt>Término que puede tener las dos definiciones
      siguientes</dt>
8    <dd>Una definición del término</dd>
9    <dd>Otra definición del término</dd>
10  </dl>
```

Y se representaría de la manera siguiente:

```
Término
  Definición del término
Término
Término
  Definición aplicable a los dos términos anteriores
Término que puede tener las dos definiciones siguientes
  Una definición del término
  Otra definición del término
```

En general, no es demasiado habitual asociar múltiples términos a una única definición, pero está bien saber que es posible por si alguna vez hay que hacerlo.

La especificación de HTML5 indica que las listas de definición se pueden usar, además de para términos y sus definiciones, para nombres de metadatos y sus valores, para preguntas o respuestas y para otros grupos cualesquiera de nombres y valores.

2.2. Elegir entre tipos de listas

A la hora de decidir el tipo de lista que utilizaréis, normalmente podréis tomar la decisión correcta respondiendo a dos preguntas muy sencillas:

1. ¿Estoy definiendo términos (o asociando otras parejas nombre/valor)?
 - a. Si la respuesta es afirmativa, utilizad una lista de definiciones.
 - b. Si la respuesta es negativa, no utilizéis una lista de definiciones y pasad a la pregunta siguiente.
2. ¿El orden de los elementos de la lista es importante?
 - a. Si la respuesta es afirmativa, utilizad una lista ordenada.
 - b. Si la respuesta es negativa, utilizad una lista no ordenada.

2.3. La diferencia entre las listas y el texto en HTML

Es posible que os estéis preguntando cuál es la diferencia entre una lista HTML y el texto con picos o números escritos manualmente. El uso de una lista HTML tiene unas cuantas ventajas:

- Si tenéis que cambiar el orden de los elementos de una lista ordenada, sólo será necesario que los mováis en el código HTML. Si hubierais escrito los números a mano, debéis revisarlo a pesar de cambiar el número de cada uno de los elementos para corregir el orden, lo cual sería muy pesado.

- El uso de una lista HTML permite aplicar un estilo a la lista. Si os limitáis a utilizar un texto normal, os será mucho más difícil aplicar un estilo a los elementos individuales de una manera adecuada.
- Si utilizáis una lista HTML, el contenido tendrá la estructura semántica adecuada y no será sólo un efecto visual “tipo lista”. Esto tiene unas ventajas importantes; por ejemplo, los lectores de pantalla pueden indicar a los usuarios con alguna deficiencia visual que están leyendo una lista y no un lío confuso de texto y números.

O, para decirlo de otra manera: **el texto y las listas no son lo mismo**. El uso de texto en lugar de una lista hará que tengáis más trabajo y puede provocar problemas en los lectores del documento. Por lo tanto, si vuestro documento necesita una lista, deberíais utilizar la lista HTML correcta.

2.4. Anidar listas

Un elemento de una lista puede contener otra lista; eso se conoce como “anidar” una lista. Puede ser útil para elementos como tablas de contenido:

1. Capítulo uno
 1. Sección uno
 2. Sección dos
 3. Sección tres
2. Capítulo dos
3. Capítulo tres

La clave para anidar listas es recordar que la lista anidada debe estar relacionada con un elemento concreto de la lista. Para reflejarlo en el código, la lista anidada debe estar contenida en este elemento de la lista.

El código de la lista anterior es el siguiente:

```
<ol>
  <li>Capítulo uno
    <ol>
      <li>Sección uno</li>
      <li>Sección dos</li>
      <li>Sección tres</li>
    </ol>
  </li>
  <li>Capítulo dos</li>
  <li>Capítulo tres</li>
</ol>
```

Observad que la lista anidada empieza después de `` y del texto del elemento de la lista que la contiene (“Capítulo uno”), y que acaba antes de `` del elemento de la lista que la contiene. Las listas anidadas son normalmente la base para los menús de navegación de sitios web, ya que son una buena manera de definir la estructura del sitio.

En teoría, se pueden anidar tantas listas como se quiera, aunque en la práctica una anidación excesiva puede llegar a crear una situación muy confusa. En el caso de listas muy largas, puede ser mejor dividir el contenido en varias listas con títulos o incluso dividirla en varias páginas.

2.5. Ejemplo paso a paso

A continuación, veremos un ejemplo paso a paso en el que incluiremos todo lo que hemos explicado hasta ahora. Imaginemos la siguiente situación:

Estáis creando un pequeño sitio web para la Escuela de Cocina HTML. En la página principal debe aparecer una lista de recetas clasificadas por categorías que nos deben llevar a las páginas de las recetas. Cada una de las páginas de recetas incluye una lista con los ingredientes necesarios, notas sobre estos ingredientes y el método de preparación. Las tres categorías son “Pasteles” (que incluye recetas para “Bizcocho”, “Pastel de chocolate” y “Pastel de manzana”); “Galletas” (que incluye recetas para “Galletas de avena y coco”, “Gotas de mermelada” y “Momentos dulces”); y “Panecillos rápidos” (que incluye recetas para “Pan de soda” y “Panecillos de leche”). El cliente no tiene ninguna preferencia por el orden en el que aparezcan las categorías y las recetas; sólo quiere estar seguro de que la gente sabrá qué elementos son categorías y cuáles son recetas.

Ahora iremos pasando por todo el proceso de creación de este sitio. En este subapartado explicaremos la creación del etiquetado y mostraremos la manera de añadir algunos estilos a las listas. Los estilos los explicaremos de una manera más superficial, ya que más adelante en esta serie encontraréis un apartado dedicado a los estilos de las listas y de los enlaces.

2.5.1. Etiquetado de la página principal

1) Cread una página HTML de la manera adecuada que incluya el `doctype` y los elementos `html`, `head` y `body` y guardadla como *stepbystep-main.html*. Añadid el título principal (`h1`) “Escuela de cocina HTML” y un subtítulo (`h2`) “Recetas”:

```
<h1>Escuela de cocina HTML</h1>
<h2>Recetas</h2>
```

2) Debéis presentar tres categorías de recetas y el orden no es importante; en este caso, lo más adecuado es una lista desordenada y, por lo tanto, deberéis añadir lo siguiente a vuestra página:

```
<h2>Recetas</h2>
<ul>
  <li>Pasteles</li>
  <li>Galletas</li>
  <li>Panes rápidos</li>
</ul>
```

El sangrado de los elementos `li`, aunque no es necesario, hace que el código sea más legible.

3) Ahora deberéis añadir las recetas como subelementos; por ejemplo, “Bizcocho”, “Pastel de chocolate” y “Pastel de manzana” deben ir a la categoría “Pasteles”. Para hacerlo, debéis crear una lista anidada en cada uno de los elementos. Como el orden no es importante, aquí también deberemos utilizar listas no ordenadas. Con el fin de facilitar las cosas en este tutorial, enlazaremos todas las recetas a una única página de receta (más adelante se explican los enlaces HTML con más detalle. Si queréis información sobre los enlaces, podéis ir a la sección correspondiente):

Podéis ver el apartado 5 de este módulo.

```
<h2>Recetas</h2>
<ul>
  <li>Pasteles
    <ul>
      <li><a href="stepbystep-recipe.html">Bizcochos</a></li>
      <li><a href="stepbystep-recipe.html">Pastel de chocolate</a></li>
      <li><a href="stepbystep-recipe.html">Pastel de manzana</a></li>
    </ul>
  </li>
  <li>Galletas
    <ul>
      <li><a href="stepbystep-recipe.html">Galletas de avena y coco</a></li>
      <li><a href="stepbystep-recipe.html">Gotas de mermelada</a></li>
      <li><a href="stepbystep-recipe.html">Momentos dulces</a></li>
    </ul>
  </li>
  <li>Panes/panes rápidos
    <ul>
      <li><a href="stepbystep-recipe.html">Pan de soda</a></li>
      <li><a href="stepbystep-recipe.html">Panecillos de leche</a></li>
    </ul>
  </li>
</ul>
```


2.5.2. Añadir un poco de estilo

Debemos volver a señalar que no debéis preocuparos excesivamente del CSS en este apartado, si no lo entendéis,. Aquí sólo veremos el CSS de una manera muy superficial, y ya hablaremos de él con mucho más detalle más adelante en esta misma asignatura.

Al cliente le gusta esta organización, pero quiere que las categorías se indiquen con unas flechas pequeñas en lugar de picos. También quiere que las categorías queden alineadas a la izquierda de la página. Para hacerlo, debéis especificar una imagen en lugar de un pico y que entonces defináis los ajustes de margen/separación.

1) Para evitar conflictos con otras listas del sitio, deberéis añadir una clase a la lista para poder crear selectores contextuales específicos en vuestra hoja de estilos. La clase “lista de recetas” podría ser adecuada:

```
<h2>Recetas</h2>

<ul class="lista-de-recetas">
```

2) Ahora debéis crear una hoja de estilo y añadir unas cuantas reglas, pero en primer lugar debéis añadir etiquetas `style` de apertura y de cierre en el `head` de vuestro documento.

3) Ahora eliminaréis el espaciado de la lista. Por defecto, algunos navegadores utilizan `margin` y otros `padding` para separar los elementos y, por lo tanto, deberéis ajustar los dos a cero; debéis añadir lo siguiente entre las etiquetas `style`:

```
ul.lista-de-recetas {
  margin-left: 0;
  padding-left: 0;
}
```

4) A continuación, debéis crear una imagen de pico personalizada; si queréis, podéis utilizar la nuestra (podéis ver la figura 1).

Figura 1



Imagen de pico personalizada

5) Ahora deberéis eliminar los picos de los elementos de la lista, definir el pico como imagen de fondo para éstos y añadir un poco de separación a fin de que

el texto no quede sobre la imagen de fondo. Podéis hacerlo añadiendo el siguiente CSS justo antes de la etiqueta `style` de cierre:

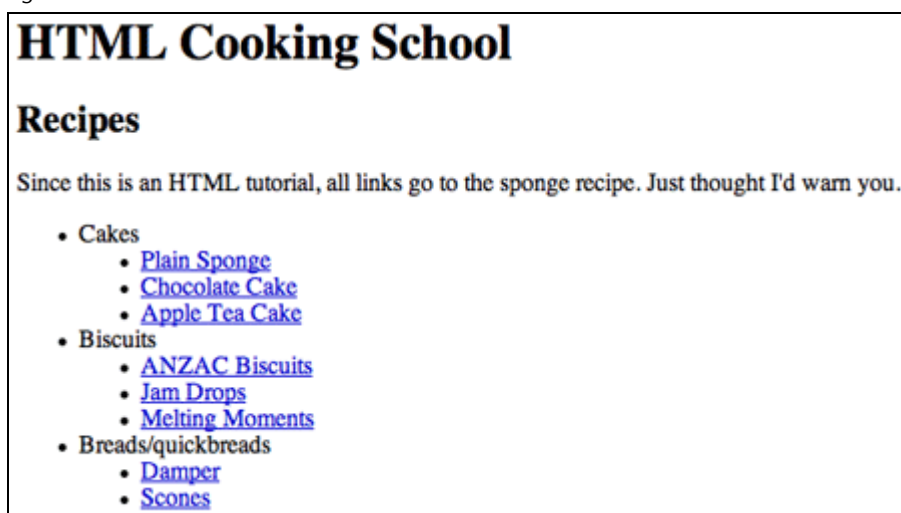
```
ul.lista-de-recetas li {  
    list-style-type: none;  
    background: #fff url("example-bullet.gif") 0 0.4em no-repeat;  
    padding-left: 10px;  
}
```

6) Finalmente, deberéis volver a poner los picos en los elementos de la lista anidada y definir un fondo de color blanco (la segunda norma es más específica, por lo que anulará la norma de la imagen de fondo). Recordad que la lista anidada heredará la primera norma del CSS, por lo que deberéis “deshacer” todos los ajustes de la lista principal. Añadid el siguiente CSS justo antes de la etiqueta `style` de cierre:

```
ul.lista-de-recetas li li {  
    list-style-type: disc;  
    background: #fff;  
}
```

El resultado final debería ser algo similar a lo que aparece en la figura 2:

Figura 2



La página principal acabada con imágenes de picos personalizadas

También podéis ver más ejemplos en: “HTML Cooking School”

<http://mosaic.uoc.edu/ac/le/operafiles/U3/stepbystep-main.html>

2.5.3. La página de la receta

En este ejemplo sólo crearemos la página de la receta para el bizcocho, pero, si queréis, podéis crear las otras por vuestra cuenta utilizando esta página

como plantilla. El cliente os ha suministrado la receta del bizcocho en un archivo de texto que tiene el siguiente aspecto:

```
Bizcocho
Ingredientes
3 huevos
100 gr de azúcar glas
85 gr de harina con levadura
Notas sobre los ingredientes:
  Azúcar glas: azúcar blanco granulado muy fino.
  Harina con levadura: una mezcla ya preparada de harina y agentes gasificantes
    (normalmente sal y levadura artificial).
Preparación
1. Precalentad el horno a 190 °C.
2. Engrasad un molde redondo de 20 cm.
3. En un bol de tamaño medio, mezclad los huevos y el azúcar glas hasta obtener
  una mezcla esponjosa.
4. Incorporad la harina.
5. Poned esta mezcla en el molde preparado.
6. Horneadlo durante 20 minutos en el horno precalentado, o hasta que la parte
  superior del bizcocho recupere la forma al apretarla ligeramente.
7. Dejadlo enfriar dentro del molde sobre una rejilla metálica.
```

2.5.4. Etiquetado de la página de la receta

1) Cread otro documento HTML de la manera adecuada y guardadlo como *stepbystep-recipe.html*. Añadidle los títulos siguientes:

```
<h1>Bizcocho</h1>
<h2>Ingredientes</h2>
<h3>Notas sobre los ingredientes</h3>
<h2>Preparación</h2>
```

2) La lista de ingredientes tiene varios elementos, pero el orden no es importante. Así pues, hay que utilizar una lista desordenada. Añadid lo siguiente a body de vuestro HTML:

```
<h2>Ingredientes</h2>
<ul>
<li>3 huevos</li>
<li>100 gr de azúcar glas</li>
<li>85 gr de harina con levadura</li>
</ul>
```

3) Las notas sobre los ingredientes son para definir adecuadamente qué son algunos de los ingredientes. Debéis asociar el ingrediente (el término) con su definición. Para ello, hay que utilizar una lista de definiciones. Añadid lo siguiente a vuestro HTML, bajo la lista desordenada del paso anterior:

```
<h3>Notas sobre los ingredientes</h3>
<dl>
<dt>Azúcar glas</dt>
<dd>Azúcar blanco granulado muy fino</dd>
<dt>Harina con levadura</dt>
<dd>Una mezcla ya preparada de harina y agentes
gasificantes (normalmente sal y azúcar artificial).</dd>
</dl>
```

4) La preparación debe seguir, como es obvio, el orden correcto, por lo que deberemos utilizar una lista ordenada; añadid lo siguiente a vuestro HTML, bajo la lista de definiciones:

```
<h2>Preparación</h2>
<ol>
<li>Precalentad el horno a 190 °C.</li>
<li>Engrasad un molde redondo de 20 cm.</li>
<li>En un bol de tamaño medio, mezclad los huevos y el
azúcar glas hasta obtener una mezcla esponjosa.</li>
<li>Incorporad la harina</li>
<li>Poned esta mezcla en el molde preparado.</li>
<li>Horneadlo durante 20 minutos en el horno
precalentado, o hasta que la parte superior del bizcocho
recupere la forma al apretarla ligeramente.</li>
<li>Dejadlo enfriar dentro del molde sobre una reja
metálica.</li>
</ol>
```

2.5.5. Estilos de la página de la receta

El cliente está muy contento con los resultados, pero quiere que las definiciones aparezcan en negrita para hacerlas más legibles. Añadid lo siguiente al head de vuestro HTML:

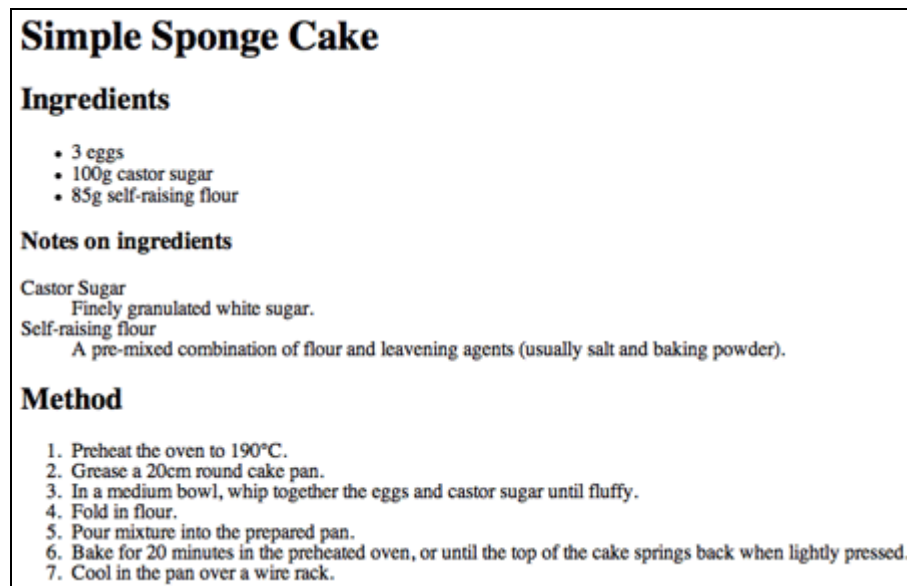
```
<style>
dt {

    font-weight: bold;

}
</style>
```

La página debería tener un aspecto similar al de la figura 3.

Figura 3



La página de la receta acabada con los términos de las definiciones en negrita

También podéis ver la página de ejemplo en: “Simple Sponge Cake”

<http://mosaic.uoc.edu/ac/le/operafiles/U3/stepbystep-recipe.html>

¡Ya habéis acabado!

Resumen

Llegados a este punto, ya deberíais entender perfectamente los tres tipos de listas diferentes de HTML. Con este ejemplo paso a paso habéis creado las tres y habéis aprendido a anidar listas dentro de elementos de listas.

Ahora que ya sabéis cómo utilizar adecuadamente las listas HTML, muy probablemente veréis que las utilizáis con mucha frecuencia. Hay mucho contenido en la web que se debería haber puesto en una lista, pero que simplemente se ha colocado en un elemento genérico con unas cuantas etiquetas de saltos de línea. Es una práctica de vagos y que genera más problemas de los que soluciona; ¡no lo hagáis nunca! Debéis crear siempre listas semánticamente correctas para ayudar a la gente a leer vuestros sitios web. A la hora de ir haciendo el mantenimiento de los sitios web es mejor para todo el mundo y no sólo para quien los ha creado.

Preguntas de repaso

Preguntas a las que deberíais poder responder:

1. ¿Cuáles son los tres tipos de listas de HTML?
2. ¿Cuándo utilizaríais cada uno de los tipos de listas? ¿Cómo elegís uno u otro?

3. ¿Cómo se hace para anidar listas?
4. ¿Por qué se debe utilizar CSS y no HTML para aplicar estilos a las listas?

Lecturas complementarias

A List Apart: Taming Lists

<http://www.alistapart.com/articles/taminglists/>

W3C CSS2: list-style-type definition

<http://www.w3.org/TR/REC-CSS2/generate.html#lists>

3. Imágenes en HTML

Christian Heilmann

En este apartado hablaremos de una de las cosas que hacen que el diseño web sea algo bonito: las imágenes. Una vez lo hayáis leído sabréis cómo añadir imágenes a vuestros documentos web de manera accesible (para que la gente con discapacidades visuales también pueda utilizar la información de vuestro sitio) y cómo y cuándo utilizar imágenes en línea para aportar información o imágenes de fondo en el diseño de la página.

Nota

Podéis descargaros los archivos de ejemplo que se utilizan en este apartado desde el archivo "code.zip"; a lo largo de este apartado iremos haciendo referencias a estos archivos.

* <http://mosaic.uoc.edu/ac/le/operafiles/U3/img/code.zip>

3.1. Una imagen dice más que mil palabras, ¿o no?

Resulta muy tentador utilizar un montón de imágenes en los sitios web. Las imágenes son una manera muy buena de dar un aire concreto a la página y las ilustraciones resultan muy útiles para comunicar información compleja de una manera más fácil a la gente que la visita.

El inconveniente de las imágenes en la web es que no todo el que navega las puede ver. Si retrocedemos hasta los días en los que los navegadores empezaron a aceptar las imágenes, veremos que muchos de los visitantes tenían las imágenes desactivadas para así ahorrar tráfico y poder navegar más rápidamente; las conexiones eran normalmente muy lentas y se debía pagar mucho dinero por cada minuto que se pasaba en línea. Aunque actualmente las cosas son muy diferentes, todavía no estamos ni mucho menos totalmente fuera de peligro.

- Las personas que utilizan dispositivos móviles pueden seguir teniendo las imágenes desactivadas a causa de las reducidas dimensiones de las pantallas y del coste de la descarga de datos.
- Vuestro sitio web puede ser visitado por personas ciegas o con alguna discapacidad visual que les impida ver correctamente las imágenes.
- También puede recibir visitas de personas de otras culturas que no entiendan los iconos que habéis utilizado.
- Los motores de búsqueda sólo indexan texto y (todavía) no analizan las imágenes, lo que significa que la información almacenada en imágenes no se podrá encontrar ni indexar.

Por lo tanto, es muy importante elegir las imágenes con prudencia y utilizarlas sólo cuando sea adecuado. Y aún es más importante que os aseguréis de que ofrecéis siempre alguna alternativa para aquellas personas que no puedan ver vuestras imágenes. De momento, veamos las tecnologías que tenemos disponibles para añadir imágenes a un documento HTML.

En el apartado "Navegación de la web y menús" de este mismo tutorial encontraréis más información sobre los problemas que generan los iconos y las imágenes mal utilizadas.

3.2. Diferentes tipos de imágenes en la web: imágenes de contenido y de fondo

Hay dos maneras básicas de añadir imágenes a un documento: imágenes de contenido utilizando el elemento `img` e imágenes de fondo aplicadas a elementos utilizando CSS. El uso de uno o de otro dependerá de qué queráis hacer:

- 1) Si la imagen es crucial para el contenido del documento, como por ejemplo una fotografía del autor o un gráfico que presenta datos, lo deberíais añadir como un elemento `img` con el texto alternativo adecuado.
- 2) Si la imagen tiene sólo una finalidad estética, deberíais utilizar las imágenes de fondo de CSS. La razón es que estas imágenes no deben tener ningún texto alternativo (¿de qué serviría “ángulo redondeado de color verde que parpadea” para una persona ciega?) y que hay muchas más opciones en CSS que en HTML para trabajar con estilos de imágenes.

3.3. El elemento `img` y sus atributos

Con el elemento `img` es muy fácil añadir una imagen a un documento HTML.

El documento `inlineimageexample.html` del archivo zip muestra la fotografía `balconyview.jpg` en un navegador (siempre que tengáis la imagen en la misma carpeta que el archivo HTML).

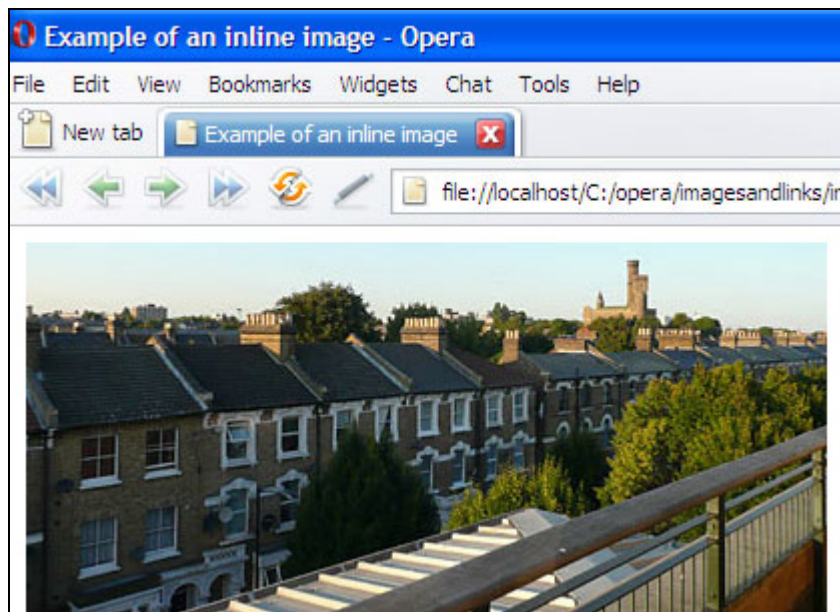
```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
  <title>Example of an inline image</title>
</head>
<body>

</body>
</html>
```

Archivo fuente: "inlineimageexample.html"
<http://mosaic.uoc.edu/ac/le/operafiles/U3/img/inlineimageexample.html>

Si ejecutáis este código en un navegador, veréis lo que sale en la figura 1.

Figura 1



La imagen tal como se ve en un navegador.

3.3.1. Ofrecer una alternativa de texto con el atributo `alt`

La imagen se ve bien, pero es un código HTML no válido porque el elemento `img` necesita un atributo `alt`. Este atributo contiene el texto que aparece si, por alguna razón, la imagen no está disponible. La imagen puede no estar disponible porque no se puede encontrar o no cargar o porque el agente de usuario (normalmente un navegador) no acepta imágenes. Además, las personas con alguna discapacidad visual utilizan tecnologías de asistencia que les leen las páginas web. Estas tecnologías leen en voz alta el contenido del atributo `alt` de los elementos `img` a los usuarios. Por lo tanto, es muy importante escribir un buen texto alternativo para describir el contenido de la imagen y ponerlo dentro del atributo `alt`.

En Internet encontraréis muchos textos que hablan sobre las “etiquetas `alt`”. Esto es objetivamente un error, ya que no hay ninguna etiqueta (o elemento) que tenga este nombre, sino que es un atributo del elemento `img` que es extremadamente importante tanto para la accesibilidad, como para la optimización en motores de búsqueda.

Para que la imagen sea comprensible para todo el mundo, es necesario que añadáis un texto alternativo adecuado; en este caso, por ejemplo, “View from my balcony, showing a row of houses, trees and a castle” (vista desde mi balcón, desde donde se ve una hilera de casas, árboles y un castillo).

```
<!DOCTYPE html>
<html>
<head>
  <title>Example of an inline image</title>
</head>
```

```
<body>

</body>
</html>
```

Archivo fuente: "inlineimageexamplealt.html"

<http://mosaic.uoc.edu/ac/le/operafiles/U3/img/inlineimageexamplealt.html>

El atributo `alt` contiene el texto que aparece cuando la imagen no está disponible. La información del atributo `alt` no debería aparecer cuando la imagen se carga correctamente; algunas versiones antiguas de Internet Explorer lo hacen mal y muestran este texto como si fuera un indicador de función al poner el puntero del ratón sobre la imagen durante unos momentos. Esto es un error, ya que hace que mucha gente utilice el atributo `alt` para añadir información adicional sobre la imagen. Si queréis añadir información adicional, debéis utilizar el atributo `title`, que explicaremos en el subapartado siguiente.

3.3.2. Añadir información pertinente con el atributo `title`

La mayoría de los navegadores muestran el valor del atributo `title` de un elemento `img` como si fuera un indicador de función al poner el puntero del ratón encima (podéis ver la figura 5). Esto puede ayudar al visitante a saber más cosas sobre la imagen, pero no os podéis fiar de que todos los visitantes tendrán un ratón.

El atributo `title` puede ser muy útil, pero no es una manera segura de ofrecer información crucial. En cambio, sí que es una buena manera, por ejemplo, para escribir las sensaciones que provoca la imagen o el significado que tiene dentro del contexto.

```
1  <!DOCTYPE html>
2  <html>
3  <head>
4      <title>Example of an inline image with alternative text and title</title>
5  </head>
6  <body>
7  
```

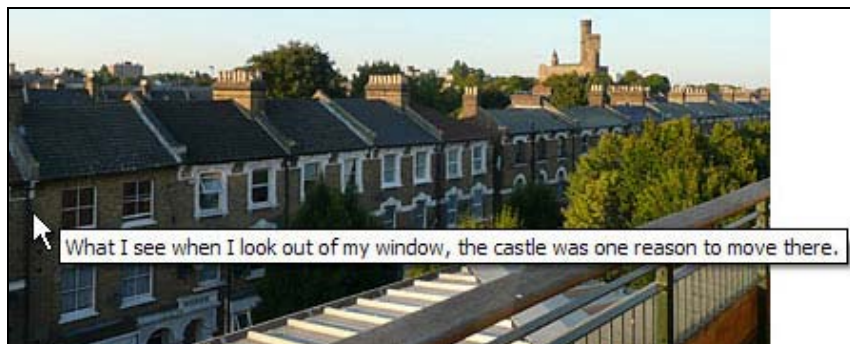
```
8 </body>
9 </html>
```

Archivo fuente: "inlineimagewithtitle.html"

<http://mosaic.uoc.edu/ac/le/operafiles/U3/img/inlineimagewithtitle.html>

Si cargáis este código en vuestro navegador, veréis lo que hay en la figura 2.

Figura 2



En muchos navegadores, los atributos `title` aparecen como si fueran indicadores de función.

3.3.3. Visualización más rápida de las imágenes definiendo las dimensiones con `width` y `height`

Cuando el agente de usuario encuentra un elemento `img` en el HTML, empieza a cargar la imagen señalada por el atributo `src`. Por defecto, éste no sabe cuáles son las dimensiones de la imagen, por lo que se limitará a mostrar todo el texto apilado y desplazará el resto del documento cuando finalmente se carguen y aparezcan las imágenes. Esto puede provocar que la carga de la página sea más lenta y genere confusión entre las personas que la visitan. A fin de que eso no ocurra, podéis indicar al agente de usuario que asigne el espacio adecuado a las imágenes antes de que éstas se carguen especificándole las dimensiones de la imagen con los atributos `width` y `height`.

```
<!DOCTYPE html>
<html>
<head>
  <title>Example of an inline image with dimensions</title>
</head>
<body>

</body>
</html>
```

Archivo fuente: "inlineimagewithdimensions.html"

<http://mosaic.uoc.edu/ac/le/operafiles/U3/img/inlineimagewithdimensions.html>

De esta manera, aparecerá un indicador de posición de la imagen hasta que ésta se cargue y ocupe su lugar, con lo que se evitará el desagradable desplazamiento de la página. Con estos atributos también se pueden modificar las dimensiones de la imagen (dividid por dos los valores de los atributos del ejemplo anterior, guardadlo y volved a cargar la página), pero no es una buena idea porque la calidad de esta modificación de las dimensiones no es buena en todos los navegadores. Algo que está especialmente desaconsejado es el hecho de redimensionar las imágenes para convertirlas en miniaturas, ya que la idea de las miniaturas no es sólo tener una imagen con unas dimensiones más pequeñas, sino también un archivo que ocupe menos. No hay nadie que quiera cargar una fotografía de 300 KB para acabar teniendo una imagen pequeña que podría ser de sólo 5 KB.

3.4. Las imágenes en línea

Hay muchos atributos que podéis utilizar para las imágenes, pero la mayoría están desaprobados porque definen la maquetación y la alineación de la imagen. Si el objetivo es éste, no se debe utilizar el HTML, sino el CSS, que se inventó precisamente para ello. Aquí sólo diremos que es importante recordar que las imágenes son, por defecto, elementos en línea. Esto significa que pueden aparecer entre palabras dentro del texto sin crear líneas nuevas. Eso es muy útil para añadir iconos pequeños dentro del texto, pero puede ser muy molesto cuando intentéis crear maquetaciones utilizando imágenes y texto. Con CSS podréis hacer que las imágenes no queden en línea por defecto y hacer que aparezcan como si fueran elementos de bloque (elementos que aparecen en una línea nueva al añadirlos a un documento).

3.5. Imágenes de fondo con CSS

Lo que está muy claro es que el diseño de webs se volvió mucho más divertido cuando los navegadores empezaron a aceptar CSS. En lugar de manipular el HTML utilizando celdas de tablas para definir la posición de los diferentes componentes de la página, espacios no separables () para mantener el espaciado y GIF separadores (imágenes GIF transparentes de 1 x 1 píxeles redimensionadas para crear márgenes), ahora podemos utilizar la separación, los márgenes, las dimensiones y el posicionamiento de CSS y dejar que el HTML se dedique exclusivamente a definir la estructura del contenido.

Con CSS también podéis utilizar las imágenes de fondo de una manera muy versátil: podéis ponerlas detrás del texto o en torno a éste de la manera que deseéis, y también podéis repetir las imágenes utilizando patrones regulares para crear fondos. Aquí sólo hablaremos muy brevemente de las del CSS para

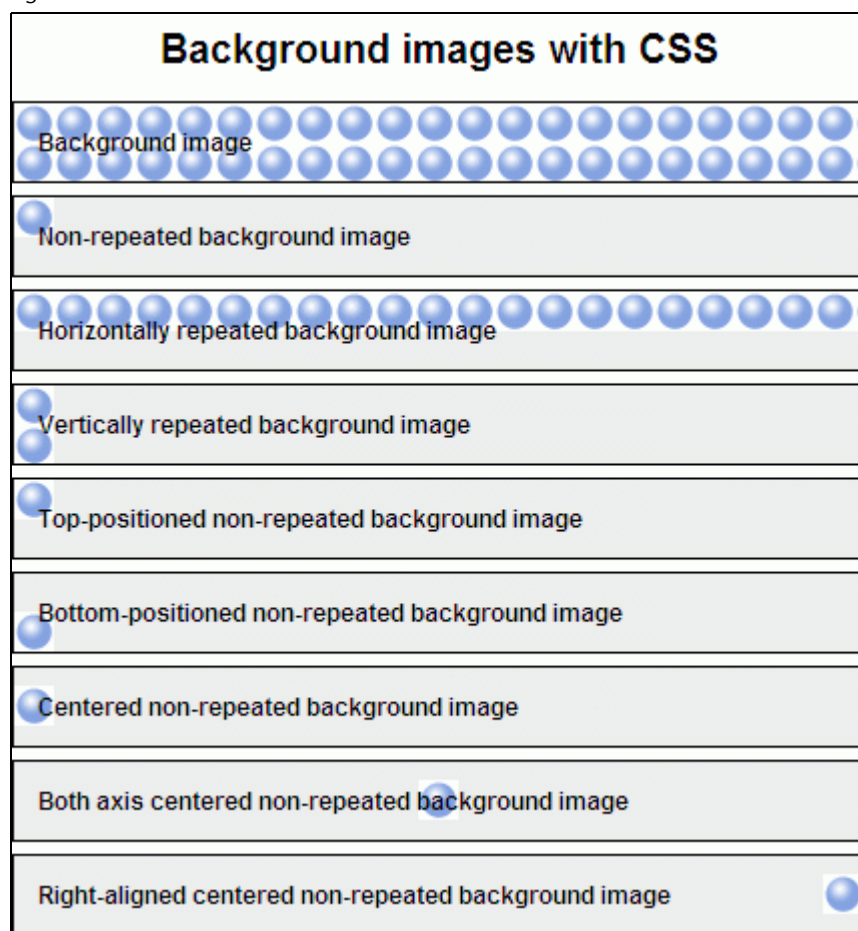
imágenes porque en otro apartado ya hablaremos con mucho más detalle del CSS para imágenes de fondo.

3.5.1. Aplicación de fondos con CSS

El uso de CSS para aplicar imágenes como fondo es muy sencillo. Antes de mirar el código CSS siguiente, cargad el archivo de ejemplo “imagesandcss.html” en vuestro navegador o consultad la figura 4 para haceros una idea de todo lo que se puede hacer con las imágenes de fondo del CSS.

* <http://mosaic.uoc.edu/ac/le/operafiles/U3/img/imagesandcss.html>

Figura 3



Imágenes de fondo con CSS

Los diferentes cuadros son en realidad elementos de título h2 con una cierta separación y bordes aplicados con CSS para así tener espacio suficiente para mostrar la imagen de fondo. Si comprobáis el archivo HTML, veréis que cada uno de los elementos h2 tiene un id único, de manera que cada uno puede tener aplicada una norma CSS diferente. El CSS para el primer ejemplo es el siguiente:

```
background-image:url(ball.gif);
background-color:#eee;
```

La imagen se añade con el selector `background-image` y se le da una URL entre paréntesis para especificar la imagen que hay que incluir. Como alternativa en caso de que la imagen no esté disponible, también deberíais definir un color de fondo con el selector `background-color` y un valor de color (en hexadecimal, por nombre o en RGB). En este caso, hemos elegido un color gris claro.

Por defecto, las imágenes de fondo se repetirán tanto horizontal como verticalmente para llenar todo el espacio del elemento. Pero se puede definir una repetición diferente con el selector `background-repeat`:

- No repetir la imagen: `background-repeat:no-repeat;`
- Repetir la imagen sólo horizontalmente: `background-repeat:repeat-x;`
- Repetir la imagen sólo verticalmente: `background-repeat:repeat-y;`

Por defecto, la imagen de fondo (si no se repite) se colocará en el ángulo superior izquierdo del elemento. También podéis utilizar `background-position` para desplazar la imagen de fondo. Los valores más sencillos que se pueden elegir son `top`, `center` y `bottom` para la alineación vertical y `left`, `center` y `right` para la alineación horizontal. Por ejemplo, para colocar la imagen en la parte inferior derecha deberíais utilizar `background-position:bottom-right;`, mientras que para centrar la imagen verticalmente y alinearla horizontalmente a la derecha utilizaríais `background-position:center-right;`.

Controlando la repetición y la posición de las imágenes de fondo y usando imágenes ingeniosas podréis crear muchos efectos impresionantes que no eran posibles antes del CSS; y si ponéis todas las definiciones del fondo en un archivo CSS independiente, lo tendréis muy fácil para cambiar el aspecto de todo el sitio web cambiando sólo algunas líneas del código. Esto se explicará más adelante en el apartado 30.

! Podéis ver el apartado 4 del módulo "Conceptos básicos de CSS".

Resumen

Esto es todo lo que debéis saber para empezar a añadir imágenes a vuestro HTML. Hay muchos más trucos que utilizan imágenes y CSS, pero de momento podéis empezar a trabajar con lo que habéis aprendido aquí concentrándoos en las mejores prácticas para la aplicación de imágenes. Hemos hablado de:

1. El elemento `img` y sus atributos básicos:
 - a. `src` para la ubicación del archivo de la imagen,
 - b. `alt` para el texto que debe estar disponible cuando la imagen no se carga o no se puede ver,
 - c. `title` para la información adicional interesante (pero no esencial),
 - d. `width` y `height` para indicar al navegador las dimensiones de la imagen y, por lo tanto, el espacio que le debe asignar.
2. Los conceptos básicos de las imágenes de fondo del CSS:

- a. cuándo utilizar fondos (básicamente, cuando no es necesario que la imagen tenga un texto alternativo porque tiene sólo una finalidad estética para la maquetación),
- b. cómo colocar y repetir las imágenes de fondo en el CSS.

Preguntas de repaso

1. ¿Por qué es importante añadir un buen texto a una imagen en un atributo `alt`? ¿Y es necesario?
2. Si tenéis una imagen de 1.280×786 Píxeles y queréis presentar una miniatura de 40×30 píxeles, ¿podéis hacerlo con HTML? ¿Y es aconsejable hacerlo?
3. Buscad información sobre el atributo `longdesc`, que era válido en HTML4 y no lo es en HTML5. ¿Qué hace y cómo lo muestran los navegadores?
4. ¿Qué hacen los atributos `valign` y `align` y por qué no los hemos explicado aquí?
5. ¿Dónde se colocan por defecto las imágenes de fondo del CSS en un elemento y cuál es el patrón de repetición por defecto?

4. Enlaces HTML: ¡construyamos una web!

Christian Heilmann

En este apartado encontraréis información sobre uno de los inventos más revolucionarios de toda la historia de Internet: los enlaces. Los enlaces permiten que el lector de un documento los siga para ir hacia otro documento y pueda pasar de un servidor a otro sin necesidad de desconectarse y volver a conectarse cada vez. Desde que se inventaron han pasado muchas cosas, pero hay una que no ha cambiado nada: los enlaces son una parte muy importante de la experiencia en Internet, y según cómo los utilicéis podéis poner las cosas fáciles o difíciles a las personas que visiten vuestra web.

Cuando hayáis leído este apartado, sabréis cómo crear enlaces que sean fáciles de entender y que funcionen en cualquier entorno. Además, también aprenderéis cómo los enlaces afectan a vuestra popularidad en los motores de búsqueda y recibiréis algunos consejos sobre su descripción.

Nota

Como ya es habitual, este apartado va acompañado de un archivo "links_code.zip"* que contiene varios archivos a los que iremos haciendo referencia.

* http://mosaic.uoc.edu/ac/le/operafiles/links_code.zip

4.1. ¿Qué son los enlaces?

Los enlaces son partes de un sitio web (normalmente creado con HTML, pero no siempre) que apuntan hacia otros recursos, como por ejemplo otros documentos HTML, archivos de texto, PDF, etc. Hay enlaces que el navegador debe seguir automáticamente y que se crean con elementos `link` (ya habéis encontrado algunos en apartados anteriores, que se utilizaban para importar archivos CSS a un documento HTML) y también hay enlaces que el usuario puede activar opcionalmente. Éstos se conocen como **anclas** y podéis añadirlos al documento con el elemento `a`.

4.2. La anatomía de un enlace ancla

Podéis convertir cualquier elemento insertado en el documento en un enlace ancla añadiendo un elemento `a` delante y detrás.

Por ejemplo, en el siguiente documento HTML el texto *Yahoo Developer Network* se convierte en un enlace.

```
<!DOCTYPE html>
<html>
<head>
<meta charset=utf-8">
```



```
<title>Link Example</title>
<link rel="stylesheet" href="styles.css">
</head>
<body>
  <h1>A link to the YDN</h1>
  <p><a href="http://developer.yahoo.com">Yahoo Developer Network</a></p>
</body>
</html>
```

Archivo fuente: "linkexample.html"

<http://mosaic.uoc.edu/ac/le/operafiles/linkexample.html>

Las personas que visiten la web y activen este enlace (al hacer clic con el ratón o al activarlo con el teclado o, en algunos casos, con la voz) dejarán el lugar actual y pasarán a la YDN. El enlace en sí experimenta otros cambios, que ya veremos cuando hablemos de los estilos de los enlaces.

El ancla tiene varios atributos que podéis añadir:

- `href`: el recurso hacia el que apunta (ya sea un archivo externo o un ID de ancla).
- `id`: el ID de ancla si el ancla es un destino y no un enlace.
- `title`: información adicional sobre el recurso externo.

Ahora veremos en primer lugar los atributos más importantes y después hablaremos sobre lo que podéis hacer a fin de que todo sea más comprensible para las personas que visiten vuestra web.

4.3. ¿Enlace o destino? Los atributos `id` y `href`

Un elemento `a` puede tener varias funciones según los atributos que defináis. El atributo más habitual que utilizaréis es `href`, que define el recurso al que apunta el enlace. Este atributo puede contener diferentes valores:

- Una URL en la misma carpeta (`help.html`), relativa a la carpeta actual (por ejemplo `../help/help.html`; los 2 puntos significan "sube un nivel dentro de la jerarquía de carpetas del sitio") o absoluta en la raíz del servidor. Por ejemplo `/help/help.html`; una barra inclinada al principio de la dirección significa que ésta empieza en la raíz del ordenador en el que se encuentra la página.
- Una URL en un servidor totalmente diferente. Por ejemplo `http://wait-till-i.com` o `ftp://ftp.opera.com/` o `http://developer.yahoo.com/yui`.
- Un identificador de fragmento o un nombre de ancla precedido de una almohadilla. Éste apunta a un destino dentro del mismo documento. Por ejemplo `#menu`.

- Una combinación de URL e identificadores de fragmentos: podéis enlazar directamente con una sección de un documento diferente haciendo que el atributo `href` apunte hacia una URL seguida de un identificador de fragmento. Por ejemplo, “`http://developer.yahoo.com/yui/#cheatsheets`”.

Cualquiera de estos valores lo convertirá en un enlace, ya que apuntará hacia otro lugar. Por otra parte, un atributo `id` lo convertirá en un ancla dentro de la página; algo a lo que apunta otro enlace. Esto puede ser un poco confuso, ya que ambos utilizan el elemento de ancla (`a`). Para poder recordarlo más fácilmente, podéis verlo de la manera siguiente: un atributo `id` convierte un enlace en un ancla y lo podéis utilizar para enlazar con secciones específicas del documento.

El siguiente bloque de código HTML contiene ejemplos de todos los tipos diferentes de enlaces:

```
10 <!DOCTYPE html>
11 <html>
12 <head>
13 <meta charset=utf-8">
14 <title>Different Link Example</title>
15 <link rel="stylesheet" href="linkexamplestyles.css">
16 </head>
17 <body>
18   <h1>Different Link examples</h1>
19   <h2>Example of in-page navigation with fragment identifiers, links and
      anchors</h2>
20   <div id="nav">
21     <ul id="toc">
22       <li><a href="#sec1">Section One</a></li>
23       <li><a href="#sec2">Section Two</a></li>
24       <li><a href="#sec3">Section Three</a></li>
25       <li><a href="#sec4">Section Four</a></li>
26       <li><a href="#sec5">Section Five</a></li>
27     </ul>
28   </div>
29   <div id="content">
30     <div>
31       <h2><a id="sec1">Section #1</a></h2>
32       <p><a href="#toc">Back to menu</a></p>
33     </div>
34     <div>
35       <h2><a id="sec2">Section #2</a></h2>
36       <p><a href="#toc">Back to menu</a></p>
37     </div>
```

```
38     <div>
39         <h2><a id="sec3">Section #3</a></h2>
40         <p><a href="#toc">Back to menu</a></p>
41     </div>
42     <div>
43         <h2><a id="sec4">Section #4</a></h2>
44         <p><a href="#toc">Back to menu</a></p>
45     </div>
46     <div>
47         <h2><a id="sec5">Section #5</a></h2>
48         <p><a href="#toc">Back to menu</a></p>
49     </div>
50 </div>
51 <h2>Some other link examples</h2>
52 <ul>
53     <li><a href="http://developer.yahoo.com">Yahoo Developer Network</a></li>
54     <li><a href="http://dev.opera.com/articles/view/the-freelancing-business-
55         part-1-pricing/#marketing">Tips on marketing yourself</a></li>
56     <li><a href="ftp://get.opera.com/pub/opera/win/">Download different Opera
57         versions</a></li>
58     <li><a href="http://farm1.static.flickr.com/56/
59         188791635_0b8bdd808d.jpg?v=0">Photo of my book</a></li>
60 </ul>
61 </body>
62 </html>
```

Archivo fuente: "linkexample.html"

<http://mosaic.uoc.edu/ac/le/operafiles/linkexamples.html>

Abrid este archivo en el navegador y experimentad. Veréis que si activáis cualquiera de los enlaces de la primera lista, pasaréis a la sección correspondiente del documento. Esto es así porque están conectados por el mismo identificador de fragmento; el primer enlace de la lista, por ejemplo, tiene un atributo `href` de `#sec1`, que coincide con el valor ID del enlace que hay en el interior del primer elemento `h2` del contenido. Esto es todo lo que necesitáis para conectar dos elementos de ancla en un documento; utilizad el mismo valor precedido por una almohadilla si lo enlazáis en un atributo `href`. También puede que os hayáis dado cuenta de que la URL de la barra de ubicación del navegador ha cambiado y que ahora muestra un identificador de segmento al final, lo cual significa que los visitantes pueden añadir esta sección a las direcciones de interés o enviar el enlace por correo electrónico a otras personas para indicarles exactamente dónde deben ir.

No obstante, si activáis cualquiera de los enlaces "Back to menu" (volver al menú) ocurrirá lo mismo. ¿Cómo puede ser? Los identificadores de fragmento pueden ser cualquier elemento con un ID.

Para recapitular:

- Los enlaces de ancla pueden tener un identificador de fragmento como valor del atributo `ref`; este identificador de fragmento debe empezar con una almohadilla (#).
- Al activarlo, este enlace llevará hacia cualquier elemento HTML con un `id` de este valor. Estos ID de cada página deben ser únicos.
- Los ID siguen algunas convenciones con respecto a los nombres. Las más importantes es que deben empezar con un carácter alfanumérico y que no pueden contener espacios.

Todo esto cubre el menú y las diferentes secciones del ejemplo, pero ¿qué sucede con los otros enlaces? Si los probáis, veréis que apuntan hacia destinos diferentes: uno va a otro sitio, otro muestra una fotografía y el tercero muestra una sección concreta de otra página web (que se encuentra pasando a un ID específico). Si todo ha funcionado bien, ¡perfecto! Pero ¿qué sucede si no habéis entendido, o el navegador no ha entendido, algunos de estos recursos?

4.4. No permitáis ninguna ambigüedad en vuestros enlaces

Lo más importante que debéis recordar sobre los enlaces es que son una parte básica para vuestra relación con las personas que os visiten. Estas personas se fían de que cuando les ofrezcáis un enlace podrán seguirlo y acceder a información de calidad y pertinente. Si vuestros enlaces no funcionan porque el recurso al que apuntan no está disponible o se encuentra en un formato que el visitante no puede leer, habréis traicionado su confianza y habréis perdido credibilidad. No debéis permitir nunca que esto suceda.

4.4.1. Ofrecer información adicional con un atributo `title`

Igual que con cualquier otro elemento HTML, podéis añadir un atributo `title` a un elemento `a` para añadir información adicional. Los navegadores mostrarán lo que se conoce como un indicador de función cuando los visitantes pasen el cursor del ratón sobre el enlace. Este indicador de función les indica la naturaleza de este enlace.

Por ejemplo, podéis escribir una pequeña introducción al contenido y la ubicación del documento enlazado:

```

<!DOCTYPE html>
<html>
<head>
<meta charset=utf-8">
<title>Adding extra information with a title attribute</title>
<link rel="stylesheet" href="linkexamplestyles.css">
</head>
<body>
  <h1>Adding extra information with a title attribute</h1>
  <ul>
    <li>Find more information on the <a title="The Yahoo Developer Network is the
      main hub for all the developer tools Yahoo offers, including the Yahoo User
      Interface Library (YUI) and the Design Patterns repository"
      href="http://developer.yahoo.com">Yahoo Developer Network</a>.</li>
  </ul>
</body>
</html>

```

Archivo fuente: "titleexample.html"

<http://mosaic.uoc.edu/ac/le/operafiles/titleexample.html>

Figura 1

Recent Major Volcanic Eruptions in the Pacific Northwest			
Volcano Name	Location	Last Major Eruption	Type of Eruption
Mt. Lassen	California	1914-17	Explosive Eruption
Mt. Hood	Oregon	1790s	Pyroclastic flows and Mudflows
Mt. St. Helens	Washington	1980	Explosive Eruption
Compiled in 2008 by Ms Jen			

Añadir un atributo de título para mostrar la información como un indicador de función cuando los visitantes pasen por encima el enlace.

Pero no podéis esperar que los visitantes tengan la suficiente paciencia y coordinación mano-ojo para poder fiaros de este recurso para dar información crucial. Es muy probable que los usuarios con alguna discapacidad visual, que no pueden ver la página, no puedan acceder a esta información. Aunque los lectores de pantalla incluyen la opción de leer en voz alta los atributos `title` para el usuario final, esta opción está desactivada por defecto; por este motivo, no deberíais utilizar nunca el atributo `title` para dar información crucial sobre el enlace. Esa información crucial podría ser:

- Enlazar con recursos no HTML, como archivos PDF, imágenes, vídeos, archivos de sonido o descargas.
- Hacer que salgamos del sitio actual y que pasemos a otro servidor (enlaces externos versus enlaces internos).
- Enlazar con un documento que se abrirá en un marco diferente o en una ventana emergente.

4.4.2. Enlaces a recursos no HTML: no obliguéis a los visitantes a hacer conjeturas

Puede ser muy molesto hacer clic en un enlace y que el navegador no sepa qué hacer con aquello a lo que apunta el. No obstante, no es nada extraño ver sitios web que enlazan con imágenes, documentos PDF y vídeos sin avisar a los visitantes para que estén preparados. Los vídeos son una causa muy habitual de problemas de los navegadores. Además, también es posible que el recurso tenga un volumen muy grande (un PDF de 20 MB, por ejemplo), lo que significa que quizá los visitantes preferirían descargarlo en lugar de abrirlo dentro del navegador y añadirlo así a un consumo de memoria que ya es considerable; o quizá incluso decidirían no acceder a él.

Uno de los factores principales del éxito de un producto web es el hecho de no obligar a las personas que lo visitan a hacer conjeturas sobre qué pasará cuando hagan algo; siempre se debe explicar claramente qué efectos tendrán sus acciones. En el caso de los enlaces, todo lo que debéis hacer para evitar muchas frustraciones es explicar bien a los visitantes qué es el recurso al que lleva el enlace.

Aquí encontraréis algunos ejemplos:

```
1  <!DOCTYPE html>
2  <html>
3  <head>
4  <meta charset=utf-8">
5  <title>Linking non-HTML resources</title>
6  <link rel="stylesheet" href="linkexamplestyles.css">
7  </head>
8  <body>
9    <h1>Linking non-HTML resources</h1>
10   <ul>
11     <li>Find more information on the <a href="http://developer.yahoo.com">Yahoo
        Developer Network site (external)</a></li>
12     <li>Download the <a href="http://www.wait-till-i.com/stuff/JavaScript-
        DOM- Cheatsheet.pdf">Dom Cheatsheet (PDF, 85KB)</a></li>
13     <li>Pick and <a href="ftp://get.opera.com/pub/opera/win/">download
        different Opera versions from their FTP (external)</a></li>
14     <li>Check out a <a href="http://farm1.static.flickr.com/56/
        188791635_0b8bdd808d.jpg?v=0">Photo of my book (JPG, 200KB)</a></li>
15   </ul>
16 </body>
17 </html>
```

Archivo fuente: "linkingnonhtml.html"

<http://mosaic.uoc.edu/ac/le/operafiles/linkingnonhtml.html>

Si ofrecéis toda esta información sobre el archivo enlazado y el espacio que ocupa, los visitantes podrán decidir qué harán y no deberéis suponer que tienen el navegador configurado de una manera concreta o algunos programas instalados. Si lo combináis con unos estilos inteligentes, incluso podréis conseguir que tenga un buen aspecto y que todo sea muy intuitivo. Si os queréis asegurar del todo, también podéis ofrecer una sección de ayuda que explique qué son los diferentes formatos de archivo y dónde se pueden conseguir los programas necesarios para abrirlos.

4.4.3. Enlaces externos y enlaces internos

Una de las cosas que más temen los responsables de los negocios con respecto a los sitios web de sus compañías es que la gente que los visite salga de ellos prematuramente. Ésta es a menudo la razón para no ofrecer nunca enlaces hacia terceros (a menos que éstos terceros paguen dinero por el privilegio del redireccionamiento del tráfico hacia ellos). Más adelante ya volveremos a hablar de este criterio tan equivocado; de momento, hablaremos de lo que hace la gente para evitar que los visitantes abandonen su sitio y las consecuencias que tienen estas medidas para el éxito del sitio web.

4.5. Marcos y ventanas emergentes: ¡ni hablar!

El temor a perder visitantes que van hacia otros sitios web, combinado con el deseo de seguir enlazando con estos otros sitios, nos ha traído algunos inventos en el desarrollo de las webs que se han convertido durante años en un verdadero problema para la usabilidad de los sitios: los marcos y las ventanas emergentes.

El uso de marcos HTML significa que estáis separando la página que aparece en el navegador en diferentes documentos. La ventaja es que el documento sigue siendo aparentemente el mismo incluso al cargar partes de éste, ya sea desde el servidor propio o desde servidores de terceros. Pero aquí es donde acaba su utilidad; los marcos son una experiencia terrible para el usuario y, de hecho, perniciosos:

- Los motores de búsqueda no pueden indexar nunca toda la página, sino que en los resultados de la búsqueda sólo muestran partes que no tienen ningún sentido fuera de contexto.
- Los visitantes no pueden añadir la página a la lista de direcciones de interés; la próxima vez que abran la página desde esta lista verán el estado inicial del conjunto de marcos y no la página tal como la dejaron la última vez.
- Los visitantes que dependan de tecnologías de asistencia lo tienen muy difícil para navegar por conjuntos de marcos.

- Es posible que los sitios web de terceros no quieran aparecer dentro de un conjunto de marcos y que utilicen *scripts* “rompedores de marcos”, que sustituirán a los conjuntos de marcos para la URL real cuando intentéis incrustarlos. Esto se hace, por ejemplo, para evitar que los delincuentes consigan con engaños que los usuarios de Internet introduzcan información como el número de la tarjeta de crédito en un sitio web que aparentemente parece de un banco (lo que se conoce como *phishing*).

Los enlaces dentro de un conjunto de marcos utilizan el atributo `target` del ancla para dirigirse al marco correcto. Cada uno de los marcos de un conjunto de marcos tiene un nombre concreto y al activar el enlace se abre el documento definido en el atributo `href` de este marco. Si el conjunto de marcos no está disponible (por ejemplo, cuando un visitante ha encontrado el documento con los enlaces por medio de un motor de búsqueda), cada uno de los enlaces se abre en una nueva instancia del navegador.

Abrir una nueva instancia del navegador es otra de las maneras habituales de enlazar con sitios web de terceros, ya sea con una ventana emergente de *scripts* o con un atributo `target` con el valor `_blank`. El hecho de que todos los navegadores modernos tengan un bloqueador de ventanas emergentes ya es toda una indicación de si es seguro o no fiarse de esta técnica... ¡No lo es en absoluto!

Claro y conciso: **no utilizéis el atributo `target` para crear enlaces a no ser que sepáis exactamente qué estáis haciendo**. De todas maneras, ésta ya es una idea anticuada; actualmente la mayoría de los navegadores tienen interfaces con pestañas, con lo cual los usuarios pueden abrir sitios de terceros en el fondo para volver a ellos más tarde y mientras tanto quedarse en vuestro sitio web. En algunas circunstancias, es posible que queráis indicar la diferencia entre los enlaces externos e internos, pero debéis dejar siempre que sea el visitante quien decida qué hará con ellos.

4.6. Ventajas de los enlaces de salida y de los enlaces de entrada

Hay algunas buenas razones para enlazar con sitios de terceros incluso cuando éstos son de la competencia.

- Os da credibilidad ante vuestros visitantes; estáis tan seguros de la calidad de vuestro contenido, que no os asustáis ante la competencia.
- Es una oportunidad para ofrecer un servicio completo; os podéis vincular al contenido y a artículos o incluso a productos de otros sitios web que vosotros no ofrecéis pero que pueden ser de interés para los visitantes que quieren ampliar más en el tema en cuestión.

- Podéis demostrar que tenéis razón mencionando un artículo anterior de un tercero y ofreciendo una solución mejor o diferente y hacer referencia a este artículo mediante un enlace.

La utilidad de los enlaces de entrada (los enlaces que apuntan desde el sitio web de un tercero hacia el vuestro) genera mucho menos debate. Cuantos más sitios web válidos y de alta calidad se enlacen con el vuestro con las palabras clave pertinentes, más arriba apareceréis en los motores de búsqueda como Google. Pero antes de que eso suceda, debéis demostrar que no tenéis miedo de enlazaros con otros sitios.

Las palabras clave pertinentes nos llevan hacia otra parte muy importante de la creación de unos buenos enlaces: cómo describirlos.

4.7. Descripción de los enlaces

Ya hemos hablado un poco de ellos en el subapartado sobre los enlaces hacia recursos no HTML, pero siempre es bueno recordar que los enlaces no sólo son parte del texto de la página, sino que también son unos elementos interactivos del documento.

Algunas tecnologías de asistencia ofrecen una lista de los enlaces en vez de todo el documento con el fin de permitir a los usuarios que puedan navegar rápidamente por él, lo que significa que debéis comprobar que los textos de vuestros enlaces tengan sentido fuera del contexto en el que se encuentran. Podéis comprobarlo muy fácilmente en Opera abriendo cualquier sitio web y seleccionando “Tools” > “Links” (herramientas > enlaces) en el menú o presionando “Ctrl” + “Mayúsculas” + “L”. Se abrirá una pestaña nueva en la que podréis ver todos los enlaces del documento y hacia dónde apuntan.

Esto significa que debéis comprobar no sólo que todos los textos de los enlaces tengan sentido, sino también que no haya varios enlaces con la misma descripción pero que apunten hacia recursos diferentes. El error clásico que se comete es el de los enlaces “haced clic aquí” que se describen, por ejemplo, como “Para descargar la última versión de nuestra herramienta, haced clic aquí”. Es mucho mejor utilizar un texto de enlace que explique hacia dónde apunta; en este caso, “Podéis descargar y probar la última versión de nuestra herramienta”.

Esto mismo también se aplica a los enlaces “más”. Los encontraréis en los sitios de noticias, en los que veréis un titular y un texto resumido y un enlace “más” o “noticia completa” que se debe seguir. La solución a este problema es utilizar una imagen “más” enlazada y darle un texto alternativo único o añadir un espacio dentro del enlace después de “más” y ocultarlo con el CSS. Explicaremos todos estos trucos más detalladamente en el apartado sobre los menús y la navegación, que encontraréis más adelante en esta asignatura.



Podéis ver el apartado 9 de este módulo.

4.8. Estilos de los enlaces

Todavía no ha llegado el momento de hablar del CSS, pero en este punto es útil tener en cuenta que el aspecto de los enlaces es muy importante y que hay varios y diferentes estados de enlaces que hay que considerar. Los estados de los enlaces (que más adelante se relacionan con los pseudoselectores del CSS; puede sonar complejo, pero no lo es) son:

- `link`: es el estado por defecto; define el aspecto que deben tener los enlaces en una parte concreta del documento. Por defecto, los enlaces no visitados son de color azul.
- `visited`: el estilo de un enlace que ya se ha visitado (y que quizá ya se encuentra en la memoria caché del navegador). Por defecto, los enlaces ya visitados son de color púrpura.
- `hover`: el estilo de un enlace mientras el cursor del ratón se encuentra sobre él.
- `active`: el estilo del enlace mientras está activado; es decir, mientras está en curso la conexión con el otro sitio; también es el estilo del último enlace activado cuando llegáis al documento al ir atrás en el navegador.

Resumen

Esta vez hemos explicado muchas cosas, pero es muy importante recordar el funcionamiento de los enlaces y qué deben hacer. A lo largo de vuestra trayectoria profesional como desarrolladores web, aprenderéis muchos trucos y técnicas para anular estos comportamientos por defecto, y esperamos que os paréis a pensar si lo que queréis hacer es realmente necesario.

Hemos hablado de:

- La anatomía del elemento `a` y sus atributos (no desaprobados).
- La diferencia entre los elementos `a` como enlaces (con un atributo `href`) y como anclas (con un atributo `name`).
- La necesidad de que el nombre de un ancla sea único.
- La necesidad de explicar a los visitantes qué pueden esperar cuando siguen un enlace (el formato del archivo y si es muy grande).
- Cómo añadir información con el atributo de título que se muestra como un indicador de función; y por qué ésta no es una manera segura de ofrecer información crucial.

- La diferencia entre los enlaces externos (que apuntan hacia sitios web de terceros) y los internos (que apuntan hacia documentos del mismo servidor).
- Las prácticas anticuadas, como las ventanas emergentes y los marcos, y por qué hay que evitarlas.
- Las ventajas de enlazarse con otros sitios y que otros sitios se enlacen con el vuestro.
- Cómo describir adecuadamente los enlaces de manera que tengan sentido fuera de contexto, y por qué es necesario.
- Los antecedentes de los estilos básicos de los enlaces.

Con todos estos conocimientos, deberíais ser capaces de escribir documentos HTML que se enlacen adecuadamente, y ya estáis preparados para empezar a pensar en los menús y la navegación del sitio.

Preguntas de repaso

1. ¿Qué problema tiene el enlace siguiente: `nuestro último informe?`
2. ¿Para qué sirve el atributo `target` en los enlaces? ¿Se puede utilizar de alguna manera positiva?
3. Hemos hablado sobre las relaciones de los enlaces y las conexiones entre los enlaces y las anclas. ¿Hay algún atributo para los enlaces que describa también las relaciones entre documentos?
4. ¿Cómo se puede escribir un enlace que envíe a los visitantes hacia un elemento que hay más abajo de la página al hacer clic en él? ¿Qué hay que comprobar de antemano?

5. Tablas HTML

Jenifer Hanen

¡Vaya! ¿Cómo debemos utilizar los estándares de la web para organizar pilas de datos? El simple hecho de pensar en las toneladas de elementos anidados necesarios para conseguir que todos estos datos se distribuyan en filas y cajas ya hace que nuestro cerebro quede totalmente colapsado; pero hay una solución: ¡las tablas pueden salvarnos!

En el diseño web, las tablas son una manera muy buena de organizar datos en un formato tabulado. Es decir, debemos ver las tablas, los gráficos y los demás elementos visuales informativos como una ayuda para ver y procesar una gran cantidad de información de una manera resumida que nos ayuda a comparar y contrastar datos diferentes. Podemos ver estos elementos muy a menudo en los sitios web, ya sea para resumir o comparar resultados de elecciones políticas, estadísticas deportivas, precios, tallas u otros datos.

En la era jurásica de Internet, antes de la popularización del CSS como método para separar la presentación de la estructura del HTML, las tablas se utilizaban para maquetar las páginas web: para crear columnas y cajas y, en general, para distribuir el contenido. Pero ésta no es la manera correcta de hacer las cosas; la maquetación con tablas daba como resultado unas páginas cargadas y confusas con un montón de tablas anidadas; archivos muy grandes, difíciles de mantener y difíciles de modificar. Todavía podréis ver sitios web hechos así en Internet, pero debéis tener muy claro que actualmente las tablas se deben utilizar sólo para lo que fueron diseñadas: los datos tabulares, y el CSS, para controlar la maquetación.

En este apartado explicaremos cómo utilizar adecuadamente las tablas HTML.

5.1. La tabla más básica

Empezaremos con el código HTML semántico necesario para crear una tabla básica; este ejemplo en concreto compara erupciones volcánicas recientes de la región del Pacífico de América del Norte. A mí me gustan los volcanes y cuando era pequeña convencí a mi madre para que me llevara a varios de estos volcanes durante los viajes que hacíamos en verano para visitar a la abuela. Siempre esperaba que alguno de estos volcanes del nordeste del Pacífico* entrara en erupción mientras estábamos de vacaciones, pero fue en vano.

* <http://volcanoes.usgs.gov/Volcanoes/Historical.html>

La primera tabla es la siguiente:

```
<table>
  <tr>
    <td>Volcano Name</td>
    <td>Location</td>
    <td>Last Major Eruption</td>
    <td>Type of Eruption</td>
  </tr>
  <tr>
    <td>Mt. Lassen</td>
    <td>California</td>
    <td>1914-17</td>
    <td>Explosive Eruption</td>
  </tr>
  <tr>
    <td>Mt. Hood</td>
    <td>Oregon</td>
    <td>1790s</td>
    <td>Pyroclastic flows and Mudflows</td>
  </tr>
  <tr>
    <td>Mt. St. Helens</td>
    <td>Washington</td>
    <td>1980</td>
    <td>Explosive Eruption</td>
  </tr>
</table>
```

Este código se representa de la manera siguiente:

Volcano Name	Location	Last Major Eruption	Type of Eruption
Mt. Lassen	California	1914-17	Explosive Eruption
Mt. Hood	Oregon	1790s	Pyroclastic flows and Mudflows
Mt. St. Helens	Washington	1980	Explosive Eruption

Empezaremos descomponiendo el etiquetado HTML utilizado en el código anterior:

- `<table></table>`: El elemento `table` es necesario para indicar al navegador que queréis organizar el contenido de una manera tabular.
- `<tr></tr>`: El elemento `tr` establece una fila de la tabla. Esto permite que el navegador organice todo el contenido que hay entre las etiquetas `<tr>` y `</tr>` de una manera horizontal, todas ellas en una fila.
- `<td></td>`: El elemento `td` define la celda de la tabla o cada uno de los espacios individuales para el contenido dentro de la fila. Utilizad sólo las

celdas de tabla `td` necesarias según los datos que se quieran incluir. No utilizéis celdas `td` vacías para crear espacio en blanco o separación; para crear el espacio en blanco o la separación, es necesario que utilizéis el CSS, ya que no sólo es una buena manera de separar la presentación de la estructura, sino que también hace que la tabla sea más fácil de entender por parte de las personas con discapacidades visuales que utilizan lectores de pantalla para leerles el contenido de la tabla en voz alta.

Tened en cuenta que los elementos básicos se deben anidar de la manera siguiente:

```
<table>
  <tr>
    <td>content</td>
    <td>content</td>
    <td>content</td>
  </tr>
</table>
```

Si los ordenamos de otra manera, el navegador generará un embrollo considerable y mostrará la tabla de una manera muy extraña.

5.2. Añadir otras funciones

Ahora que ya tenemos la tabla básica, podéis añadir algunas funciones de tabla un poco más complejas. En primer lugar, añadiremos un título y unos encabezamientos de columna para mejorar los datos tanto respecto a la semántica, como a la legibilidad por parte de los lectores de pantalla. El etiquetado de la tabla mejorada es el siguiente:

```
<table>
  <caption>Recent Major Volcanic Eruptions in the Pacific Northwest</caption>
  <tr>
    <th>Volcano Name</th>
    <th>Location</th>
    <th>Last Major Eruption</th>
    <th>Type of Eruption</th>
  </tr>
  <tr>
    <td>Mt. Lassen</td>
    <td>California</td>
    <td>1914-17</td>
    <td>Explosive Eruption</td>
  </tr>
```

```

<tr>
  <td>Mt. Hood</td>
  <td>Oregon</td>
  <td>1790s</td>
  <td>Pyroclastic flows and Mudflows</td>
</tr>
<tr>
  <td>Mt. St. Helens</td>
  <td>Washington</td>
  <td>1980</td>
  <td>Explosive Eruption</td>
</tr>
</table>

```

Este código se representa de la manera siguiente:

Recent Major Volcanic Eruptions in the Pacific Northwest

Volcano Name	Location	Last Major Eruption	Type of Eruption
Mt. Lassen	California	1914-17	Explosive Eruption
Mt. Hood	Oregon	1790s	Pyroclastic flows and Mudflows
Mt. St. Helens	Washington	1980	Explosive Eruption

Los elementos nuevos que hemos utilizado son:

- `<caption></caption>`: El elemento `caption` permite añadir un título a los datos de la tabla. La mayoría de los navegadores centran el título y lo reproducen con el mismo ancho que la tabla, a menos que se utilice el CSS para alinear el texto de una manera diferente.
- `<th></th>`: El elemento `th` define el contenido que hay entre estas etiquetas como encabezamiento para la columna correspondiente de la tabla. Esto es útil no sólo para ayudar a describir semánticamente cuál es la función de este contenido, sino que también ayuda a reproducirlo de una manera más precisa en varios navegadores y dispositivos. Este ejemplo es la manera más básica de utilizar el elemento `th`.

5.3. Seguir estructurando la tabla

Como paso final para acabar de estructurar nuestra tabla, definiremos unas secciones de encabezamiento y de cuerpo de la tabla, le añadiremos un pie y definiremos el ámbito de los títulos de fila y columna. También añadiremos un atributo de resumen para describir el contenido de la tabla. El etiquetado definitivo de la tabla es el siguiente:

```

<table summary="a summary of recent major volcanic eruptions in the Pacific Northwest">
  <caption>Recent Major Volcanic Eruptions in the Pacific Northwest</caption>
  <thead>
    <tr>
      <th scope="col">Volcano Name</th>
      <th scope="col">Location</th>
      <th scope="col">Last Major Eruption</th>
      <th scope="col">Type of Eruption</th>
    </tr>
  </thead>
  <tfoot>
    <tr>
      <td colspan="4">Compiled in 2008 by Ms Jen</td>
    </tr>
  </tfoot>
  <tbody>
    <tr>
      <th scope="row">Mt. Lassen</th>
      <td>California</td>
      <td>1914-17</td>
      <td>Explosive Eruption</td>
    </tr>
    <tr>
      <th scope="row">Mt. Hood</th>
      <td>Oregon</td>
      <td>1790s</td>
      <td>Pyroclastic flows and Mudflows</td>
    </tr>
    <tr>
      <th scope="row">Mt. St. Helens</th>
      <td>Washington</td>
      <td>1980</td>
      <td>Explosive Eruption</td>
    </tr>
  </tbody>
</table>

```

Este código de tabla tiene el siguiente aspecto en un navegador:

Recent Major Volcanic Eruptions in the Pacific Northwest			
Volcano Name	Location	Last Major Eruption	Type of Eruption
Mt. Lassen	California	1914-17	Explosive Eruption
Mt. Hood	Oregon	1790s	Pyroclastic flows and Mudflows
Mt. St. Helens	Washington	1980	Explosive Eruption
Compiled in 2008 by Ms Jen			

Los nuevos elementos/atributos son los siguientes:

- Los elementos `thead`, `tbody` y `tfoot`: estos elementos definen respectivamente el título, el cuerpo y el pie de la tabla. A menos que estemos codificando una tabla realmente compleja con muchas columnas y filas de datos, si los utilizamos corremos el riesgo de excedernos. En una tabla compleja, sin embargo, su uso no sólo estructurará el contenido para el codificador, sino también para los navegadores y otros dispositivos.
- Los atributos `colspan` y `rowspan`: el atributo `colspan` crea una celda de tabla que ocupará más de una columna. En el caso del pie anterior, queríamos que una celda de la tabla ocupara todo el ancho y por ello le hemos dicho que debía extenderse a lo largo de cuatro columnas. También podéis añadir un atributo `rowspan` de celda de tabla que permitirá que la celda ocupe *x* filas, por ejemplo `<td rowspan="3">`.
- El atributo `summary`: este atributo se utiliza para definir un resumen del contenido de la tabla para su uso por parte de los lectores de pantalla (observad que este resumen no aparece en la versión de la tabla que se ve en pantalla). Tened en cuenta que en las recomendaciones W3C antiguas, WCAG 1.0 y HTML 4.0, se dice que se puede utilizar el atributo `summary` tal como hemos explicado anteriormente. En los borradores más nuevos de las especificaciones, sin embargo, este atributo `summary` ya no se menciona. El hecho de seguir utilizando el atributo `summary` es una cuestión que está por resolver; de momento, los integrantes de la Web Standards Curriculum hemos llegado a la conclusión de que vale la pena seguir utilizándolo. Al fin y al cabo, no va mal y presenta algunas ventajas para la accesibilidad.
- El atributo `scope`: también es posible que hayáis observado los atributos `scope` de las etiquetas `th`, y el hecho de que hayamos definido los nombres de los volcanes también como títulos de columnas, dentro de las filas de datos. Esto es perfectamente posible, pero de hecho nos estamos apartando del tema. El atributo `scope` se puede utilizar dentro del elemento `th` para indicar a los lectores de pantalla que el contenido de `th` es el título de una columna o una fila. El atributo `scope` sólo se utiliza dentro del elemento `th`.

5.4. El CSS nos ayuda: una tabla con un aspecto mejor

Los elementos y atributos anteriores son todo lo que necesitáis para codificar una buena tabla de datos. Ahora que ya hemos definido la estructura HTML, pasamos a ver un CSS muy sencillo para hacer que la tabla tenga un aspecto todavía mejor:

```
body{
  background: #ffffff;
  margin: 0;
  padding: 20px;
  line-height: 1.4em;
  font-family: tahoma, arial, sans-serif;
  font-size: 62.5%;
}
```

```

table {
  width: 80%;
  margin: 0;
  background: #FFFFFF;
  border: 1px solid #333333;
  border-collapse: collapse;
}

td, th {
  border-bottom: 1px solid #333333;
  padding: 6px 16px;
  text-align: left;
}

th {
  background: #EEEEEE;
}

caption {
  background: #E0E0E0;
  margin: 0;
  border: 1px solid #333333;
  border-bottom: none;
  padding: 6px 16px;
  font-weight: bold;
}

```

Cuando lo aplicamos al etiquetado definitivo de nuestra tabla, el aspecto que tendrá es el que se puede ver en la figura 1. También podéis comprobar la tabla real de ejemplo.

Figura 1

Recent Major Volcanic Eruptions in the Pacific Northwest			
Volcano Name	Location	Last Major Eruption	Type of Eruption
Mt. Lassen	California	1914-17	Explosive Eruption
Mt. Hood	Oregon	1790s	Pyroclastic flows and Mudflows
Mt. St. Helens	Washington	1980	Explosive Eruption
Compiled in 2008 by Ms Jen			

Ahora la tabla es visualmente mucho más atractiva.

Archivo fuente: "table3.html"

<http://mosaic.uoc.edu/ac/le/operafiles/csstable/table3.html>

¡Ooooh...! Mucho mejor, ¿no? Podéis aplicar los estilos que queráis a la tabla, pero el ejemplo anterior es un punto de referencia a partir del cual podéis empezar a trabajar. En otro apartado aprenderéis muchas más cosas sobre los es-

tilos de las tablas; de momento, nos limitaremos a explicar qué hace cada una de las secciones de este CSS:

- `body`: en el CSS anterior hemos añadido propiedades para definir el margen (en este caso a cero), una separación (*padding*) para crear un poco de espacio, un color de fondo (blanco), el tamaño y la familia del tipo de letra, así como la interlínea para hacerlo todo un poco más espacioso. Podéis descargaros el código de este ejemplo en <http://mosaic.uoc.edu/ac/le/operafiles/U5/csstable.zip>; intentad modificar las propiedades en el archivo CSS para ver cómo cambian las cosas.
- `table`: los bordes se han añadido con la declaración de bordes de CSS. Para que todo funcionara correctamente, también hemos tenido que añadir la propiedad `border-collapse` a `collapse`; de esta manera, se reinician los valores de los bordes de la tabla y `border-bottom` es una línea recta continua que ocupa toda la fila en lugar de quedar partida entre cada una de las celdas de la tabla. En este ejemplo hemos definido `width` al 80%. Esto hace que la tabla ocupe el 80% del ancho de la pantalla y el ancho de la tabla cambiará siempre que se hagan cambios en el ancho de pantalla del navegador.
- `th` y `td`: en el CSS anterior hemos definido la alineación del texto a la izquierda, pero también puede ser centrada o incluso podéis dar nombres de clase a los diferentes elementos `th` y `td` y entonces utilizar el CSS para tener un mayor control sobre cada una de las filas o columnas (en el caso de las filas, daréis un nombre de clase a la etiqueta del elemento `tr`). También hemos añadido un poco de separación a `th` y `td` para abrir las filas y conseguir así que la legibilidad fuera mejor. En el caso del selector `th` anterior, hemos definido otro color para diferenciar los títulos del resto de la tabla.
- `caption`: si no definís las propiedades CSS para el selector `caption`, éste no tendrá ningún borde y tendrá el mismo color de fondo que toda la página aunque el etiquetado HTML para el título de la tabla se encuentre en la etiqueta `table`. Así pues, en el ejemplo anterior hemos definido un borde para el título (sin ningún borde inferior, ya que el borde de la tabla ya desempeña esta función), un color de fondo diferente y un tipo en negrita para diferenciar el título de la fila de los títulos de columna.

Resumen

En este apartado hemos explicado todo lo que debéis saber para crear unas tablas de datos en HTML muy efectivas. ¡Eso es todo! Para terminar, os dejaremos con un par de ideas básicas que debéis recordar:

- Es importante que las tablas estén correctamente codificadas para que se puedan leer en todos los navegadores, dispositivos móviles y todo tipo de soportes. El HTML de la tabla debe ser tan reducido como sea posible, y

para definir los estilos se debe utilizar el CSS. Más adelante, en esta misma asignatura, aprenderéis mucho más sobre el CSS.

- Los dispositivos móviles y los usuarios que utilicen lectores de pantalla podrán acceder a las tablas siempre que su código sea limpio y que se utilicen los atributos como `scope` y `summary`, así como el elemento `caption`, para anunciar clara y semánticamente para qué sirven las secciones respectivas. Para la accesibilidad también es importante no utilizar celdas vacías para crear espacio en blanco (para ello debéis utilizar el CSS).

Preguntas de repaso

1. Empezad escribiendo el código de una tabla sencilla con sólo los 3 elementos de tabla principales: `table`, `tr` y `td`. Guardadla y abridla en un navegador.
2. Igual que en el segundo ejemplo anterior, añadid un título, títulos de columna y un pie. ¿Cómo cambia lo que podéis ver en el navegador?
3. ¿Qué podéis hacer para que la tabla sea más accesible para los lectores de pantalla y los dispositivos portátiles?
4. Ahora cread un archivo CSS. Intentad definir un estilo para los bordes, la separación y el espaciado de celda de la tabla sólo con el CSS y sin utilizar ningún atributo del etiquetado HTML. Añadid un color de fondo y definid el estilo de los tipos de letra.

¡Pasadlo bien!

Lecturas complementarias

Recomendación HTML 4 para tablas del W3C

<http://www.w3.org/TR/html401/struct/tables.html>

Recomendación CSS 2 para tablas del W3C

<http://www.w3.org/TR/CSS21/tables.html>

“Bring on the Tables” de Roger Johansson

http://www.456bereastreet.com/archive/200410/bring_on_the_tables/

6. Formularios HTML: conceptos básicos

Jenifer Hanen

Todo el mundo ha visto un formulario. Todo el mundo ha utilizado alguno. Pero ¿habéis codificado alguno?

Un formulario en línea es cualquier área en la que se puede introducir información en una página web; por ejemplo, introduciendo texto o números en un cuadro de texto, marcando una casilla de selección, seleccionando un botón de opción o eligiendo una opción de una lista. Entonces, el formulario se envía al sitio web pulsando el botón de enviar.

En la web encontraréis formularios por todas partes: para introducir nombres de usuario y contraseñas en una pantalla de acceso, para hacer comentarios en blogs, para escribir vuestro perfil en una red social o para especificar la información de facturación en un sitio de compras.

Crear un formulario es muy sencillo, pero ¿qué hay que saber sobre los formularios para que sean acordes con los estándares de la web? De momento, si habéis ido siguiendo el currículo de estándares web de Opera, probablemente ya tendréis muy claro que es muy importante seguir siempre estos estándares. El código necesario para crear un formulario accesible y conforme con los estándares no supone más trabajo que la creación de un formulario chapucero.

Así pues, empezaremos con el formulario más básico y sencillo que podríamos utilizar y lo iremos complicando; en este apartado explicaremos todos los conceptos básicos que hay que conocer para crear unas estructuras de formularios elegantes y accesibles con HTML.

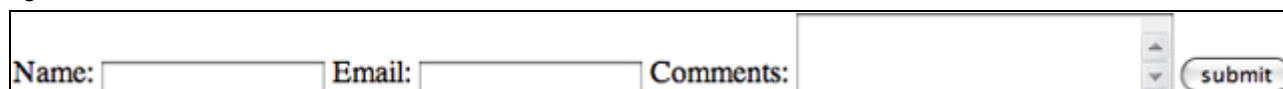
6.1. Paso uno: el código básico

Empezaremos con un formulario de comentario muy muy básico, el tipo de formulario que utilizaríais en un sitio web para permitir que la gente pueda dar su opinión sobre algo como un artículo que hayáis escrito, o para permitir que pueda enviaros un mensaje sin saber vuestra dirección electrónica. El código es el siguiente:

```
<form>
Name: <input type="text" name="name" id="name" value="" />
Email: <input type="text" name="email" id="email" value="" />
Comments: <textarea name="comments" id="comments" cols="25" rows="3"></textarea>
<input type="submit" value="submit" />
</form>
```

Si introducís este código en un documento HTML y abrís este documento en un navegador, lo que veréis es lo que aparece en la figura 1.

Figura 1

The image shows a simple web form. It has three input fields: 'Name:', 'Email:', and 'Comments:'. Each field is followed by a text input box. To the right of the 'Comments' field is a vertical scrollbar. At the bottom right of the form is a button labeled 'submit'.

El primer ejemplo de formulario básico

Archivo fuente: "step-1-form.html"

<http://mosaic.uoc.edu/ac/le/operafiles/U6/step-1-form.html>

Probadlo; introducid este código en vuestro documento HTML de muestra y cargadlo en un navegador, o id a la página del primer ejemplo de formulario básico* para navegar hasta el formulario en otra página. Intentad jugar un poco con los diferentes controles del formulario para ver todo lo que podéis hacer.

* <http://mosaic.uoc.edu/ac/le/operafiles/U6/step-1-form.html>

Si leéis el código, veréis una etiqueta `<form>` de apertura, una etiqueta `</form>` de cierre y algunas otras cosas entre estas dos. El elemento contiene dos campos de texto en los que el lector de la página puede introducir su nombre y su dirección electrónica, y un área de texto que se puede llenar con un comentario para enviarlo al propietario del sitio web.

¿Qué tenemos, pues, aquí?

- `<form></form>`: estas dos etiquetas son esenciales para iniciar y acabar un formulario; sin éstas no tendríamos formulario web. Todos los formularios deben empezar y acabar con las etiquetas `<form>` y `</form>` respectivamente.

La etiqueta `<form>` puede tener algunos atributos, que explicaremos en el paso dos, pero de entrada hay que tener bien claro que no se pueden anidar formularios dentro de formularios.

- `<input>` (si utilizáis un doctype XHTML, debería ser `<input />`): esta etiqueta define el área en la que podréis insertar información. En nuestro ejemplo anterior, las etiquetas `input` definen cuadros de texto en los que los visitantes pueden escribir su nombre y la dirección electrónica.

Todas las etiquetas de entrada deben tener un atributo `type` para definir el tipo de información que recibirán. Los valores posibles para el atributo son `text`, `button`, `checkbox`, `file`, `hidden`, `image`, `password`, `radio`, `reset` o `submit`.

Todas las etiquetas `<input>` (introducción de datos) también deben tener un nombre (excepto en algunos casos especiales en los que el atributo `value` –valor– tiene siempre el mismo valor que el atributo `type` –tipo–, como por ejemplo `type="submit"` o `"reset"`), que debe elegir el codifi-

cador. El atributo `name` (nombre) informa sobre dónde se envían los datos cuando se envía el formulario (sea a una base de datos o un mensaje electrónico que se envía al administrador del sitio a través de un *script* del servidor) y sobre el nombre que tiene la información del cuadro de introducción de datos. Al enviar el formulario, la mayoría de los *scripts* utilizan el atributo `name` para colocar los datos del formulario en una base de datos o en un mensaje electrónico que puede ser leído por una persona.

Por lo tanto, si el elemento `<input>` sirve para que el visitante del sitio pueda introducir su nombre, entonces el atributo `name` sería `name="nombre"`, `name="name"` o `name="first name"`, etc. Si la etiqueta `<input>` es para la dirección electrónica, entonces el atributo `name` sería `name="email"`. Para facilitarlas las cosas, y para facilitarlas a la persona que utilizará el formulario, se recomienda que pongáis un nombre descriptivo al elemento `<input>`.

Por *nombre descriptivo* se entiende un nombre relacionado con su función, tal como se ha explicado antes. Si el objetivo es obtener una dirección electrónica, entonces el nombre debería ser `name="correo-electronico"`. Si debe ser la dirección postal de la persona que visita el sitio, entonces el nombre debería ser `name="street-address"` o `name="direccion-postal"`. Cuanto más esmerado sea el uso de estos nombres, no sólo lo tendréis más fácil para codificar el formulario e ir realizando el mantenimiento, sino que la persona o la base de datos que lo reciba también lo tendrá más fácil para entenderlo. Debéis dedicar todo el tiempo necesario a pensar bien estos nombres.

Todas las etiquetas `<input>` también deben tener un atributo `value`. Su valor se puede dejar en blanco (`value=""`), con lo que indicaréis al script de procesamiento que inserte cualquier cosa que el visitante del sitio escriba en este cuadro. En el caso de una casilla de selección (*checkbox*), un botón de opciones (*radio*), oculto (*hidden*), enviar (*submit*) o cualquier otro atributo, podéis definir el valor a fin de que sea igual a lo que queréis que sea por defecto. En otros casos, como por ejemplo enviar u oculto, debéis definir el valor a fin de que sea igual en la entrada final. Ejemplos: `value="yes"` para sí, `value="submit"` para un botón de envío, `value="reset"` para un botón de reinicialización, `value="http://www.opera.com"` para un redireccionamiento oculto, etc.

El atributo `value`

Algunos ejemplos sobre el uso del atributo `value`:

1) Un atributo de valor en blanco cuyo valor viene determinado por aquello que introduzca el usuario:

- El código dice: `<input type="text" name="first-name" id="first-name" value="" />`
- El usuario introduce: Jenifer
- El valor de `first-name` se envía como "Jenifer" en el momento de enviar el formulario.

2) Un valor predeterminado:

- El código dice: `<input type="checkbox" name="mailing-list" id="mailing-list" value="yes" />`

- El usuario selecciona la casilla porque se quiere unir a la lista de correo del sitio web.
- El valor de `mailing-list` se envía como “yes” en el momento de enviar el formulario.
- Después de los dos elementos `<input>` podéis ver algo un tanto diferente: el elemento `textarea`.

`textarea` es un espacio nuevo y mejorado muy adecuado para introducir texto. No es lo mismo que el antiguo cuadro de texto normal de una única línea que ofrece nuestro amigo `<input>`; el elemento `textarea` permite introducir múltiples líneas e incluso podéis definir cuántas líneas habrá disponibles para introducir texto. Fijaos en los atributos `cols` y `rows`; estos atributos son necesarios para todos los elementos `textarea` y especifican las columnas y filas que tendrá el área de texto. Estos valores se miden en caracteres.

- Y para terminar, pero no por ello menos importante, hay un elemento `<input>` especial con el atributo `value="submit"`. En lugar de mostrar un cuadro de una única línea para introducir texto, el envío del texto introducido mostrará un botón de envío que, al hacer clic en él, envía el formulario al destino que especifique el formulario para enviar los datos (en este momento esto aún no está definido, por tanto el envío del formulario no hará nada).

6.2. Paso dos: añadir estructura y comportamiento

Así pues, habéis hecho clic en el enlace del formulario #1 anterior, lo habéis rellenado y habéis hecho clic en el botón “Enviar”; ¿por qué no ha pasado nada? ¿Y por qué tiene tan mal aspecto y aparece todo en una línea? La respuesta es que todavía no lo habéis estructurado ni habéis definido ningún lugar donde enviar los datos que recoge el formulario.

Volvamos a la mesa de dibujo para crear un formulario nuevo:

```
<form id="contact-form" action="script.php" method="post">
  <input type="hidden" name="redirect" value="http://www.opera.com" />
  <ul>
    <li>
      <label for="name">Name:</label>
      <input type="text" name="name" id="name" value="" />
    </li>
    <li>
      <label for="email">Email:</label>
      <input type="text" name="email" id="email" value="" />
    </li>
    <li>
      <label for="comments">Comments:</label>
      <textarea name="comments" id="comments" cols="25" rows="3"></textarea>
    </li>
  </ul>
```



```
<li>
    <input type="submit" value="submit" />
    <input type="reset" value="reset" />
</li>
</ul>
</form>
```

Cuando lo abríis en un navegador, este formulario tiene el aspecto que se puede ver en la figura 2:

Figura 2

The image shows a web browser window displaying a simple contact form. The form is enclosed in a rectangular border. It contains three labeled input fields: 'Name:', 'Email:', and 'Comments:'. The 'Name' and 'Email' fields are single-line text boxes. The 'Comments' field is a multi-line text area with a vertical scrollbar on the right. Below the input fields are two buttons: 'submit' and 'reset', both with a 3D effect and rounded corners.

El segundo ejemplo de formulario; su aspecto es mejor, pero todavía no es perfecto.

Podéis jugar con este formulario mejorado en: "step-2-form.html"
<http://mosaic.uoc.edu/ac/le/operafiles/U6/step-2-form.html>

Aquí hemos añadido algunas cosas al formulario básico. Veámoslo para saber qué es lo que hemos hecho:

- Dentro de la etiqueta `<form>` hay algunos atributos nuevos. Hemos añadido un atributo `id` no sólo para dar un nombre semánticamente descriptivo al formulario, sino también para ofrecer un ID único para identificarlo de manera que se puedan poner estilos más fácilmente con el CSS o se pueda manipular utilizando JavaScript si es necesario. Sólo podéis tener un `id` de cada por página; en este caso, lo hemos llamado `contact-form` (formulario de contacto).
- ¡Luces, cámara, acción! Cuando habéis pulsado el botón de envío en el primer formulario y no ha sucedido nada, la razón era que no incluía ninguna acción ni método. El atributo `method` (método) especifica cómo se envían los datos al *script* que los procesará. Los dos métodos disponibles son "GET" y "POST". El método "GET" enviará los datos de la URL de la página (siempre veréis URLs del estilo `http://www.example.com/page.php?data1=value1&data2=value2...`; éstos son datos que se transportan con el método "GET"). Si no tenéis una razón concreta para utilizar

"GET", probablemente es mejor no utilizarlo si intentáis enviar información segura, ya que todo el mundo puede ver la información mientras se transmite a través de la URL. "POST" envía los datos por medio del *script* que ofrece el formulario; estos datos se pueden enviar a un mensaje electrónico que se envía al administrador del sitio o a una base de datos que se almacenará y a la que se podrá acceder más adelante, y no en la URL del "GET". "POST" es la opción más segura y normalmente la mejor*.

* <http://www.w3.org/2001/tag/doc/whenToUseGet.html>

Si la seguridad de los datos del formulario es un aspecto que os preocupa especialmente, por ejemplo si enviáis un número de tarjeta de crédito a un sitio de compras, entonces deberíais utilizar el protocolo https con un protocolo de capa de conexión segura (SSL, *secure socket layer*). Básicamente, esto significa que los datos se enviarán a través del protocolo https y no del protocolo http. La próxima vez que paguéis algo en un sitio de compras o que utilicéis la banca en línea, observad las URL; es muy probable que veáis https:// en lugar de http://. La diferencia es que una conexión https es un poco más lenta de transmitir que una http, pero los datos se encriptan de manera que ningún pirata que hurgue por ahí pueda entender nada de los datos mientras están en tránsito. Hablad con vuestro proveedor de servicios web para saber cómo os puede ofrecer una conexión con https y SSL.

- El atributo `action` (acción) especifica el fichero de *script* al que se deben enviar los datos del formulario para su procesamiento. Muchos proveedores de servicios web tienen un *script* genérico de envío de correo electrónico u otros *scripts* de formularios disponibles para su uso (podéis consultar esta información en la documentación de vuestro proveedor de servicios web) que han adaptado para sus servidores. Por otra parte, podéis utilizar un *script* de servidor que vosotros mismos o alguien más haya creado para ofrecer vuestro formulario. La mayoría de las veces se utilizan lenguajes como PHP, Perl o Ruby para crear un *script* que procesará el formulario; por ejemplo, podéis enviar un mensaje electrónico que contenga la información del formulario, o introducir la información del formulario en una base de datos que se almacenará para su uso en el futuro.

El alcance de esta asignatura no nos permite explicaros la manera de redactar un *script* de servidor ni enseñaros a escribir un código de servidor por vuestra cuenta; os deberéis limitar a preguntar a vuestro proveedor de servicios web qué es lo que os ofrece o a haceros amigos de un buen programador.

Aquí os damos unos cuantos recursos para empezar a introducirlos en este campo en caso de que queráis investigar los *scripts* de servidor:

- Perl: <http://www.perl.com/>
- PHP: <http://www.php.net>
- Documentación PHP sobre formularios:
<http://uk3.php.net/manual/en/tutorial.forms.php>
- Python: <http://python.org/>
- Ruby: <http://www.ruby-lang.org>
- Sendmail: <http://www.sendmail.org/>
- ASP.NET: <http://www.asp.net/>

- La segunda línea que hemos añadido a nuestro formulario del paso dos es el campo de entrada “oculto”, que es un redireccionamiento. ¿Qué?

Con el objetivo de separar la estructura del etiquetado por una parte y la presentación y el comportamiento por otro, lo mejor es utilizar el *script* que ofrecerá el formulario también para redirigir al usuario una vez que el formulario ya se haya enviado. Lo que no queréis de ninguna de las maneras es que los usuarios se encuentren mirando la página del formulario sin saber qué deben hacer una vez lo han enviado; estamos muy seguros de que todos estaréis de acuerdo en que es mucho mejor redirigir al usuario a una página de agradecimiento que incluya unos enlaces sobre “qué debéis hacer a continuación” una vez que el formulario se haya enviado correctamente. Esta línea en concreto especifica que, después de enviar el formulario, el usuario pasará a la página inicial de Opera.

- Para mejorar el aspecto del formulario, hemos puesto todos sus elementos en una lista no ordenada para así poder utilizar el etiquetado para alinearlos y entonces utilizar CSS para pulir su aspecto.

Alguien podría decir que no deberíamos utilizar una lista no ordenada para etiquetar un formulario, sino una lista de definiciones. Otros podrían decir que ni siquiera deberíamos utilizar una lista, sino el CSS para aplicar estilos en las etiquetas `<label>` e `<input>`. No entraremos en este debate y dejaremos que hagáis investigaciones por vuestra cuenta sobre este tema y lleguéis a una conclusión propia sobre qué es semánticamente más correcto. Para nuestro ejercicio tan sencillo utilizaremos la lista no ordenada.

- Y lo último del paso dos, pero no por ello menos importante, es que hemos etiquetado los elementos del formulario. Tanto en términos de significado como de accesibilidad del formulario en una gama amplia de dispositivos habilitados para Internet, lo mejor es poner etiquetas a todos los elementos del formulario; comprobad el contenido de los elementos `label`, ya que estas etiquetas están ligadas a los elementos respectivos del formulario mediante los elementos `id` de los elementos `input` y `textarea`, que tienen el mismo valor que los atributos `for` de las etiquetas. Esto es perfecto porque no sólo aporta un indicador visual sobre la finalidad de cada uno de los campos del formulario en pantalla, sino que también da más significado semántico a los campos del formulario. Por ejemplo, una persona con alguna deficiencia visual que utilice la página con un lector de pantalla podrá ver qué etiqueta corresponde a cada uno de los elementos del formulario. Los `id` también se pueden utilizar para aplicar estilos en los campos individuales del formulario con el CSS.

Es muy probable que en este momento os estéis preguntando por qué, además de los atributos `name`, se incluyen atributos `id` como identificadores en los elementos del formulario. La respuesta es que los elementos `input`

sin atributos `name` no se envían al servidor y, por lo tanto, son absolutamente necesarios. Los atributos `id` son necesarios para asociar los elementos del formulario con sus elementos `label` correspondientes. Por lo tanto, deberíais utilizar los dos.

El segundo formulario tiene un aspecto un poco mejor, pero todavía lo podemos seguir mejorando un poco más. Ya ha llegado el momento de añadir otros detalles antes de aplicar estilos.

6.3. Paso tres: añadir semántica, estilo y un poco más de estructura

Ahora terminaremos lo que hemos empezado al principio de este apartado con la versión definitiva de nuestro formulario de ejemplo:

```
1  <form id="contact-form" action="script.php" method="post">
2    <fieldset>
3      <legend>Contact Us:</legend>
4      <ul>
5        <li>
6          <label for="name">Name:</label>
7          <input type="text" name="name" id="name" value="" />
8        </li>
9        <li>
10         <label for="email">Email:</label>
11         <input type="text" name="email" id="email" value="" />
12       </li>
13       <li>
14         <label for="comments">Comments:</label>
15         <textarea name="comments" id="comments" cols="25" rows="3"></textarea>
16       </li>
17       <li>
18         <label for="mailing-list">Would you like to sign up for our mailing list?
19         </label>
20         <input type="checkbox" checked="checked" id="mailing-list" value="Yes,
21         sign me up!" />
22       </li>
23       <li>
24         <input type="submit" value="submit" />
25         <input type="reset" value="reset" />
26       </li>
27     </ul>
28   </fieldset>
29 </form>
```

Cuando se muestra en un navegador, este formulario tiene el aspecto que se puede ver en la figura 3:

Figura 3



El formulario de ejemplo definitivo en todo su esplendor

Podéis ver la versión real de este formulario en un navegador y jugar con él en: “step-3-form”.
<http://mosaic.uoc.edu/ac/le/operafiles/U6/step-3-form.html>

Los dos últimos elementos principales que hemos añadido a este formulario son `fieldset` (conjunto de campos) y `legend` (leyenda). Ninguno de ellos es obligatorio, pero son muy útiles para formularios más complejos y para la presentación.

El elemento `fieldset` os permite organizar el formulario en módulos semánticos. En un formulario más complejo, podríais por ejemplo utilizar diferentes `fieldset` para contener información de la dirección, información de facturación, información sobre las preferencias del cliente, etc. El elemento `legend` permite dar un nombre a cada uno de las secciones `fieldset`.

También hemos aplicado un poco de CSS a este formulario para dar estilo al etiquetado estructural. Esto se aplica al tercer formulario de ejemplo con una hoja de estilos externa; para ver los estilos, id a la página de ejemplo*. Las dos cosas más importantes que queríamos que hiciera el CSS básico eran añadir márgenes para alinear las etiquetas y los cuadros de introducción de datos, y eliminar los picos de la lista desordenada. Éste es el CSS que reside en la hoja de estilos externa:

* <http://mosaic.uoc.edu/ac/le/operafiles/U6/form.css>

```
#contact-form fieldset {width:40%;}
#contact-form li {margin:10px; list-style: none;}
#contact-form input {margin-left:45px; text-align: left;}
#contact-form textarea {margin-left:10px; text-align: left;}
```

¿Qué hace este CSS? La primera línea define el estilo del borde del conjunto de campos para que no ocupe toda la página; también podéis definir que no haya ningún borde con `{border: none;}` en caso de que no queráis ninguno. La segunda línea añade un margen de 10 píxeles a los elementos `li` para

poner un poco de espacio visual entre cada uno de los elementos de la lista. Las líneas tercera y cuarta definen un margen izquierdo en los elementos `input` y `textarea` para que no queden a la misma altura que las etiquetas y se alineen adecuadamente. Si queréis más información sobre los estilos de un formulario, podéis consultar el apartado sobre los estilos de los formularios de este currículo de estándares web o el artículo de Nick Rigby en *A List Apart*, “*Prettier Accessible Forms*”*. También podréis encontrar más información sobre los márgenes y los bordes más adelante en esta asignatura.

* <http://alistapart.com/articles/prettyaccessibleforms>

Resumen

En este apartado hemos explicado los tres pasos más básicos para crear un formulario conforme a los estándares web. El mundo de los formularios incluye muchas más cosas que no hemos explicado aquí y que, en caso de que queráis, deberéis explorar por vuestra cuenta. Existen las teclas de acceso, las casillas de selección y los botones de opción, los botones de envío y reinicialización personalizados y los cuadros de selección.

En el formulario anterior del paso tres hemos añadido una casilla de selección (`checkbox`) con el fin de mostrar que se pueden utilizar los atributos adicionales del elemento `input` para recoger información que queda más allá de la introducción de texto en una única línea o de la introducción de texto en un área de múltiples líneas. Los valores de los atributos de botón `checkbox` y `radio` se pueden utilizar para añadir la posibilidad de hacer preguntas cortas y dar al lector una lista de opciones entre las que podrá elegir (las casillas de selección permiten seleccionar varias opciones, mientras que los botones de opción sólo permiten seleccionar una). Los botones de opción pueden ser muy útiles en un formulario de encuesta.

El elemento `select`, que tampoco se ha tratado en este apartado, se puede utilizar para un menú desplegable con opciones (por ejemplo, una lista de países o de estados/provincias).

Preguntas de repaso

Ya es hora de que escribáis el código de vuestro formulario de contacto.

1. Cread un formulario de contacto sencillo que pida al usuario su nombre, su dirección electrónica y un comentario.
2. Añadid una casilla de selección que pida al lector si se quiere unir a vuestra lista de correo.
3. Utilizad CSS para aplicar estilos al formulario: definid una anchura para el formulario, alinead las etiquetas a la izquierda, poned un color de fondo en la página, etc.

- **Deberes voluntarios:** cuanto más juguéis con los elementos del formulario y con diferentes CSS, mejor veréis todo lo que podéis hacer con un formulario de contacto muy sencillo.
- **Deberes aún más voluntarios:** si queréis adentraros aún más en este tema, buscad un *script* o utilizad uno que os ofrezca vuestro proveedor de servicios web para enviaros el formulario a vosotros mismos. Si el *script* para crear el formulario se complica excesivamente, también podéis intentar sobornar a algún buen programador...

Lecturas complementarias

Cameron Adams, “Accessible, stylish form layout”.

<http://www.themaninblue.com/writing/perspective/2004/03/24/>

Brian Crescimanno, “Sensible Forms: A Form Usability Checklist”.

<http://www.alistapart.com/articles/sensibleforms/>

Simon Willison, “Easier form validation with PHP”.

<http://blog.simonwillison.net/post/58225435446/theholysrail>

Las especificaciones. No cualquier especificación antigua, sino la especificación W3C.

<http://www.w3.org/TR/html401/interact/forms.html>

Si sufrís de insomnio y un vaso de leche caliente o contar ovejas no os solucionan el problema, podéis leer las especificaciones completas de HTML 4.01 o XHTML 1.0 en w3.org. Es mucho más barato y más efectivo que cualquier otro remedio. Que [insertad aquí el nombre de la deidad] bendiga a los ingenieros de todo el mundo.

Los amables integrantes de W3.org han definido las diferencias entre “GET” y “POST” y cuándo se deben utilizar:

<http://www.w3.org/2001/tag/doc/whenToUseGet.html>

Y todo el agradecimiento del mundo para el Sr. Rigby:

<http://alistapart.com/articles/prettyaccessibleforms>

7. Elementos semánticos menos conocidos

Mark Norman Francis

En este apartado presentaremos algunos de los elementos semánticos del HTML más oscuros y menos conocidos y utilizados. Hablaremos del etiquetado de código de programación, interacción con ordenadores, citas y abreviaturas, visualización de los cambios realizados en los documentos y más. Acabaremos comentando algunas de las propuestas de elementos semánticos nuevos que se han hecho en el borrador de HTML5.

Nota

Después de cada ejemplo de código hay un enlace “Archivo fuente”, que, al hacer clic en él, os llevará a la visualización del resultado de este código fuente en un archivo diferente; de esta manera, podréis ver ejemplos reales de cómo aparece realmente este código fuente en el navegador, aparte del código en sí.

7.1. Destacar la información de contacto

El elemento `address` es probablemente el que tiene el nombre más equivocado y peor entendido de HTML. A primera vista, con un nombre como *address* (dirección) parecería que se utiliza para encapsular direcciones, ya sean electrónicas o postales, o cosas similares. Esto sólo es así en parte.

El significado real de `address` es ofrecer información de contacto correspondiente al `article` que lo contiene. En el caso en que no esté contenido en ningún `article`, debe interpretarse como la información de contacto de la página. Esta información puede ser un nombre, una dirección electrónica, una dirección postal o un enlace a otra página con más información de contacto.

Por ejemplo:

```
<address>
  <span>Mark Norman Francis</span>,
  <span class="tel">1-800-555-4865</span>
</address>
```

Archivo fuente: “simple address example”

<http://mosaic.uoc.edu/ac/le/operfiles/U6/lesserknownsemantics.html#address1>

En el ejemplo siguiente, la dirección se encuentra en el párrafo del pie de página y es un enlace hacia otra página del sitio web. La información de contacto ampliada de la página a la que apunta este enlace puede contener muchos más datos, y de esta manera no hay que repetirla una y otra vez en el sitio web.


```
<p class="footer">© Copyright 2008</p>
<address>
  <a href="/contact/">Mark Norman Francis</a>
</address>
```

Archivo fuente: "Advanced address example"

<http://mosaic.uoc.edu/ac/le/operafiles/U6/lesserknownsemantics.html#address2>

(Evidentemente, si el sitio web tuviera más de un autor, se podría utilizar este mismo patrón y definir enlaces hacia páginas de contacto diferentes para los diferentes autores.)

Lo que es **incorrecto** es utilizar el elemento `address` para indicar cualquier otro tipo de direcciones, como lo que podéis ver en la página web de ejemplo incorrecto.

```
<p> Our company address: </p>
  <address>
    Opera Software ASA,
    Waldemar Thranes gate 98,
    NO-0175 OSLO,
    NORWAY
  </address>
```

Archivo fuente: "Incorrect address example"

<http://mosaic.uoc.edu/ac/le/operafiles/U6/lesserknownsemantics.html#address3>

(Evidentemente, si Opera asumiera la responsabilidad colectiva por el artículo, este código sería correcto aunque nosotros, y no todo el personal de Opera, fuéramos los autores reales de esta página en concreto).

Para cualquier dirección general, podéis utilizar lo que se conoce como "microformato" para indicar que un párrafo contiene una dirección. Encontraréis más información sobre los microformatos en otros artículos de la página web de Opera*.

* <http://dev.opera.com/articles/html/>

7.2. Lenguajes y código de programación

El elemento `code` se utiliza para indicar el código informático o los lenguajes de programación, como PHP, JavaScript, CSS, XML y otros. En el caso de muestras cortas dentro de una frase, simplemente debéis rodear el elemento con el elemento `code`, como por ejemplo:

```
<p>It is bad form to use event handlers like  
<code>onclick</code> directly in the HTML.</p>
```

Archivo fuente: “First code example”

<http://mosaic.uoc.edu/ac/le/operafiles/U6/lesserknownsemantics.html#code1>

En caso de muestras de código más grandes que pueden ocupar múltiples líneas, podéis utilizar un bloque preformateado, tal como se explica en el apartado “Etiquetado del contenido de texto en HTML”.

Podéis ver el apartado 1 de este módulo.

Aunque no hay ningún método definido para indicar el lenguaje de programación o el código informático dentro del elemento `code`, podéis utilizar el atributo `class`. Una práctica muy habitual (que se menciona en la especificación de HTML5) es utilizar el prefijo `language-` (lenguaje) y entonces añadirle el nombre del lenguaje.

Así pues, el ejemplo anterior se convertiría en:

```
<p>It is bad form to use event handlers like  
<code class="language-javascript">onclick</code>  
directly in the HTML.</p>
```

Archivo fuente: “Second code example”

<http://mosaic.uoc.edu/ac/le/operafiles/U6/lesserknownsemantics.html#code2>

Algunos lenguajes de programación tienen nombres que no se pueden representar fácilmente en clases, como por ejemplo C#. La manera correcta de escribir este nombre sería `class='language-c\#'`, que puede resultar confuso y se puede escribir mal con mucha facilidad. Aconsejaríamos utilizar una clase consistente sólo en letras y guiones y deletrearla completamente. Así pues, en este caso se podría utilizar `class='language-csharp'`.

7.3. Mostrar la interacción con ordenadores

Los dos elementos `samp` y `kbd` se pueden utilizar para indicar la entrada y la salida de la interacción con un programa informático.

Por ejemplo:

```
<p>One common method of determining if a computer is connected  
to the internet is to use the computer program <code>ping</code>  
to see if a computer likely to be running is reachable.</p>  
  
<pre><samp class="prompt">kittaghy%</samp> <kbd>ping -o google.com</kbd>  
  <samp>PING google.com (64.233.187.99): 56 data bytes  
  64 bytes from 64.233.187.99: icmp_seq=0 ttl=242 time=108.995 ms
```

```
--- google.com ping statistics ---
 1 packets transmitted, 1 packets received, 0% packet loss
round-trip min/avg/max/stddev = 108.995/108.995/108.995/0.000 ms
</samp></pre>
```

Archivo fuente: "Computer interaction example"

<http://mosaic.uoc.edu/ac/le/operafiles/U6/lesserknownsemantics.html#computer>

El elemento `samp` indica un ejemplo de la salida de un programa informático. Tal como se puede ver en este ejemplo, con el atributo `class` se pueden indicar diferentes tipos de salidas. A pesar de ello, no hay ninguna convención que se haya adoptado de una manera general sobre el tipo de clases que se debe utilizar.

El elemento `kbd` indica la entrada por parte del usuario que interactúa con el ordenador. Aunque tradicionalmente estas entradas se hacen desde el teclado (y por ello se utiliza la contracción `kbd`, del inglés *keyboard* –teclado–), también se debe utilizar para indicar otros tipos de entradas, como por ejemplo por voz.

7.4. Variables

El elemento `var` se utiliza para indicar variables en el contenido textual. Esto puede incluir expresiones matemáticas algebraicas o dentro del código de programación.

Por ejemplo:

```
<p>The value of <var>x</var> in
 3<var>x</var>+2=14 is 4.</p>

<pre><code class="language-perl">
my <var>$welcome</var> = "Hello world!";
</code></pre>
```

Archivo fuente: "Variables example"

<http://mosaic.uoc.edu/ac/le/operafiles/U6/lesserknownsemantics.html#variables>

7.5. Citas

El elemento `cite` (cita) representa el título de una obra. `cite` no debe usarse para referenciar nombres de persona.

Por ejemplo:

```
<p>La frase <q>Mi reino por un caballo</q>, como tantas otras, la escribió
Shakespeare en su <cite>Ricardo III</cite>.</p>
```

7.6. Abreviaturas

El elemento `abbr` se utiliza para indicar dónde hay abreviaturas y ofrece un método para desarrollarlas sin interrumpir innecesariamente el flujo del documento.

El texto de la abreviatura se encuentra dentro del elemento `abbr` y la versión completa se sitúa dentro del atributo `title`, es decir:

```
<p>Styling is added to <abbr title="Hypertext Markup
Language">HTML</abbr> documents using <abbr
title="Cascading Style Sheets">CSS</abbr>.</p>
```

Archivo fuente: "Abbreviations example"

<http://mosaic.uoc.edu/ac/le/operafiles/U6/lesserknownsemantics.html#abbreviations>

7.7. Instancias definidoras

Existe cierta confusión sobre el uso adecuado de `dfn`, que se describe en la especificación HTML como "la instancia definidora del término incluido". Este concepto es muy parecido a la idea del elemento `dt` (término de definición) utilizado en las listas de definiciones.

La diferencia es que el término utilizado en `dfn` no debe formar parte de una lista de términos y definiciones y que se puede utilizar como parte del flujo de texto normal, incluso en un estilo de prosa conversacional. Veamos, pues, un ejemplo del uso de `dfn`:

```
<p><dfn>HTML</dfn>: HTML significa "HyperText Markup
Language". Es el lenguaje que se utiliza para describir
el contenido de documentos web.</p>
```

Aparece el término *HTML* seguido inmediatamente de su definición, con lo cual es el lugar ideal para utilizar el elemento `dfn`. Lo deberíais utilizar sólo una vez en una página, en el punto en el que se define el término por primera vez; de todas maneras, los términos se deben definir sólo una vez en una página, por lo que no es demasiado problemático.

Todo eso está muy bien, pero un ejemplo aislado no es demasiado práctico; el uso de `dfn` se recomienda cuando una abreviatura se utiliza más de una vez en una página. Por ejemplo, en el módulo "Fundamentos del HTML" de esta

asignatura, la abreviatura HTML aparecía más de cuarenta veces. Utilizar el código `<abbr title="HyperText Markup Language">HTML</abbr>` todas y cada una de las veces que se utiliza esta abreviatura sería un despilfarro de ancho de banda, sería visualmente muy molesto y para los usuarios de lectores de pantallas sería probablemente muy pesado y aburrido, ya que HTML se expande una y otra vez aunque ya se ha explicado qué significa. En lugar de ello, lo que se podría hacer es insertar el código en el punto en el que se define por primera vez para los lectores:

```
<p><dfn><abbr>HTML</abbr></dfn> ("HyperText Markup Language") is  
a language to describe the contents of web documents.</p>
```

Archivo fuente: "First defining instances example"

<http://mosaic.uoc.edu/ac/le/operafiles/U6/lesserknownsemantics.html#dfn1>

Entonces, cada vez que se utilice HTML se puede etiquetar simplemente como `<abbr>HTML</abbr>`. Así, un agente de usuario podría poner a disposición del usuario algún método para recuperar la instancia definidora de la abreviatura. Desafortunadamente, hoy por hoy no existe ningún agente de usuario que lo haga, ni siquiera los lectores de pantalla. Por lo tanto, sería mejor utilizar también el atributo `title` para ofrecer esta información:

```
<p><dfn><abbr title="HyperText Markup Language">HTML</abbr></dfn> ("HyperText Markup Language") is a language  
to describe the contents of web documents.</p>
```

Archivo fuente: "Second defining instances example"

<http://mosaic.uoc.edu/ac/le/operafiles/U6/lesserknownsemantics.html#dfn2>

Ahora, sin embargo, hemos doblado el término ampliado para HTML, lo cual puede ser un problema para los usuarios de lectores de pantalla. No obstante, el hecho de dejar la ampliación visible provoca que el documento sea menos útil para los usuarios videntes, que en prácticamente todos los casos serán la mayoría.

Creemos que éste es un compromiso aceptable cuando sólo hay una o dos abreviaturas que necesitan una definición (en páginas que exigen un número mayor de definiciones, quizá sería mejor crear una sección de glosario o una página en la que se pueda utilizar el etiquetado más riguroso de las listas de definiciones). Si esto es un tema que os afecta, el código podría ser:

```
<p><abbr title="HyperText Markup Language">HTML</abbr>  
(<dfn>HyperText Markup Language</dfn>) is a language to  
describe the contents of web documents.</p>
```

Archivo fuente: "Third defining instances example"

<http://mosaic.uoc.edu/ac/le/operafiles/U6/lesserknownsemantics.html#dfn3>

No obstante, el agente de usuario debería tener algún método para conectar la definición con todas las apariciones del término definido. Actualmente, no hay ningún navegador que haga algo útil con `dfn`, aunque sigue siendo un vínculo útil para aplicar estilos con CSS. La solución propuesta anteriormente es actualmente la mejor que tenemos.

Éste es un caso muy desgraciado de una especificación que se ha creado sin unas directrices claras sobre cómo se debe utilizar un elemento; probablemente no se basó en ningún uso de este elemento en el mundo real, ya que en caso contrario habría algún método para combinar el término con su descripción o definición completa. La especificación HTML 5 entra en muchos más detalles sobre el uso de `dfn`*, pero sigue siendo sólo un borrador y todavía no se puede utilizar en la web.

* <http://www.w3.org/html/wg/html5/#the-dfn>

7.8. Superíndice y subíndice

Para etiquetar una parte de un texto como superíndice o subíndice (ligadamente por encima o por debajo del resto del texto) hay que utilizar los elementos `sup` y `sub`.

Algunos lenguajes necesitan estos elementos para el uso correcto de las abreviaturas y se pueden utilizar para etiquetar contenido matemático de poca envergadura, ya que así se evita recurrir a MathML (un lenguaje de marcas matemático específico pero bastante complicado creado con el único objetivo de etiquetar fórmulas matemáticas complicadas).

Un ejemplo de los dos tipos:

```
<p>The chemical formula for water is H<sub>2</sub>O, and it is also known as  
hydrogen hydroxide.</p>  
<p>The famous formula for mass-energy equivalence as derived by Albert Einstein  
is E=mc<sup>2</sup> - energy is equal to the mass multiplied by the speed of light  
squared.</p>
```

Archivo fuente: "Superscript and subscript example"

<http://mosaic.uoc.edu/ac/le/operafiles/U6/lesserknownsemantics.html#supsub>

7.9. Saltos de línea

Teniendo en cuenta el modo como HTML define el espacio en blanco, no es posible controlar el lugar al que saltarán las líneas de texto (por ejemplo, etiquetar una dirección postal como un único párrafo pero querer que tenga un aspecto visual en el que cada parte de la dirección aparezca en una línea independiente) simplemente apretando la tecla de retorno mientras se va escribiendo el texto.

Con el elemento `br` es posible introducir un salto de línea en el documento. No obstante, sólo se debería utilizar para forzar saltos de línea en los lugares en los que sea estrictamente necesario, y nunca para aplicar más espaciado vertical entre párrafos o similares en un documento; eso se debería hacer con CSS.

Algunas veces puede ser más fácil utilizar un bloque de texto preformateado en lugar de insertar elementos `br`. O, si queréis que una parte concreta de un texto aparezca en una línea independiente, aunque eso es puramente una cuestión de estilo, podéis poner esta parte en un elemento `span` y definirla a fin de que se muestre como un elemento de bloque.

Así, por ejemplo, podéis escribir la dirección de contacto de Opera que hemos visto antes en este apartado cuando hablábamos sobre el elemento `address` de la siguiente manera:

```
<p>Our company address: </p>
<address>
Opera Software ASA,<br>Waldemar Thranes gate 98,<br>
NO-0175 OSLO,<br>NORWAY
</address>
```

Archivo fuente: "Line breaks example"

<http://mosaic.uoc.edu/ac/le/operafiles/U6/lesserknownsemantics.html#breaks>

Evidentemente, si escribís en XHTML en lugar de HTML, el elemento debería ser autocerrado, así: `
`.

7.10. Líneas horizontales

Las líneas horizontales se crean en HTML con el elemento `hr`. Este elemento inserta una línea en el documento que representa un límite entre secciones diferentes.

Algunos dicen que éste es un efecto inherentemente no semántico y puramente visual y de presentación, pero de hecho hay precedentes en la literatura que justifican la existencia de un elemento así. En un capítulo (que se podría describir como una sección de un libro) en ocasiones aparece una línea horizontal entre escenas que tienen lugar en tiempos y/o lugares diferentes. En la poesía también se utilizan algunas separaciones decorativas para delimitar las diferentes estrofas del poema.

Ninguno de estos usos justificaría la existencia de un elemento de encabezamiento nuevo, que es la manera aceptada de marcar los límites entre secciones de un documento.

El elemento `hr` no tiene ningún atributo y su estilo se deberá definir con el CSS en caso de que el aspecto por defecto no sea satisfactorio.

También, igual que en el salto de línea, si escribís en XHTML y no en HTML deberéis utilizar la forma autocerrada: `<hr />`.

7.11. Cambios en los documentos (inserciones y supresiones)

Si se han introducido cambios en un documento desde la primera vez que estuvo disponible, podéis marcar estos cambios para que las personas que lo vuelvan a leer o los procesos automatizados puedan saber qué se ha cambiado y cuándo.

El texto nuevo (inserciones) se debe poner dentro del elemento `ins`. El texto que se ha eliminado (supresiones) se debe poner dentro del elemento `del`. Si en un punto concreto del documento se ha hecho una supresión y una inserción, la manera correcta de proceder es poner primero el texto suprimido y después el texto insertado.

Estos dos elementos pueden tener dos atributos que aportan más significado a las ediciones.

Si la razón del cambio se explica en la página o en algún otro lugar de la web, deberíais crear un enlace hacia este documento o fragmento del atributo de cita. Lo que estamos diciendo en realidad es que “Este cambio se ha hecho por la razón siguiente”.

También podéis indicar la hora en la que se realizó el cambio con el atributo `datetime`. El valor debería ser una marca horaria conforme al estándar ISO, que normalmente tiene la forma AAAA-MM-DD HH:MM:SS ±HH:MM" (encontraréis más información sobre esto en la Wikipedia*).

* http://en.wikipedia.org/wiki/ISO_8601

Un ejemplo que utiliza los dos atributos:

```
<p>We should only solve problems that actually arise. As
  <cite><del datetime="2008-03-25 18:26:55 Z"
    cite="/changes.html#revision-4">Donald Knuth</del><ins
      datetime="2008-03-25 18:26:55 Z"
      cite="/changes.html#revision-4">C. A. R. Hoare</ins></cite>
  said: <q>premature optimization is the root of all
  evil</q>.</p>
```


Archivo fuente: “Inserting and deleting example”

<http://mosaic.uoc.edu/ac/le/operafiles/U6/lesserknownsemantics.html#insdelexample>

Resumen

En este apartado hemos explicado algunos de los elementos semánticos del HTML menos conocidos y que menos se utilizan. En el apartado siguiente seguiremos examinando el uso correcto de dos elementos semánticos de HTML semánticamente neutros: `div` y `span`.

8. Contenedores genéricos: los elementos `div` y `span`

Mark Norman Francis

En este apartado explicaremos cómo y cuándo utilizar los dos elementos de HTML que no se utilizan para describir el contenido. Los elementos `span` y `div` no confieren en realidad ningún significado al contenido que rodean; de hecho, son un mecanismo genérico que permite crear una estructura y unas agrupaciones personalizadas de elementos en los lugares en los que no hay ningún otro elemento del HTML que resulte realmente adecuado, a los que entonces se pueden aplicar estilos con CSS o manipular con JavaScript. A pesar de que los elementos `div` no añaden ningún significado semántico, se puede considerar que éstos representan una división estructural del etiquetado junto con la clase semántica o el nombre de ID adecuados.

Son “el último recurso” y sólo se deben utilizar cuando no hay ningún otro elemento del HTML que cumpla los requisitos necesarios, ya que no tienen ningún significado para las tecnologías de asistencia, los motores de búsqueda, etc.

8.1. Semánticamente neutros

La mayoría de los elementos de HTML existen para describir el contenido, como por ejemplo las imágenes, las listas o los títulos, o para ayudar a componer el documento, como por ejemplo `head`, `body`, `link`, `meta`, etc. Sin embargo, hay dos elementos que no tienen ningún significado asignado: los elementos `div` y `span`, junto con los atributos `id` y `class`, ofrecen un mecanismo genérico para añadir estructura a los documentos.

Estos dos elementos se pueden considerar como el andamio de HTML. Dan la posibilidad de agrupar el contenido y de añadir información adicional relativa al contenido y vínculos para incorporar estilos e interactividad. Pero no añaden ningún significado semántico nuevo al documento, ni en sí mismos ni de ellos mismos.

8.2. En línea y de bloque

Como ya habéis aprendido, los elementos de bloque son elementos que ayudan a informar sobre la estructura de un documento. El elemento `div`, que es la abreviatura de división, es el contenedor genérico de los bloques. Normalmente se utiliza para rodear otros elementos de bloque con el objetivo de agruparlos. También se puede utilizar para reunir un grupo de elementos insertados y/o texto que no encajan de manera lógica bajo ningún otro elemento de bloque, pero debería ser el último recurso.



Podéis ver el subapartado siguiente, donde encontraréis una explicación más detallada sobre la agrupación del contenido.

El elemento `span` es el contenedor genérico para los elementos en línea. También ayuda a informar sobre la estructura del documento, pero se utiliza para agrupar o rodear otros elementos insertados y/o texto más que elementos de bloque.

En un primer momento, la línea de separación entre estos dos tipos diferentes podría parecer arbitraria. La diferencia que hay que tener en cuenta es el tipo de contenido y el modo como debería aparecer al escribirlo sin ningún estilo. Un elemento `div` se pone alrededor de un grupo de elementos de bloque; por ejemplo, para rodear un título y una lista de enlaces para crear un menú de navegación. Un elemento `span` rodea un grupo de elementos en línea o (la mayoría de las veces) texto plano. La palabra clave es *grupo*; si un elemento `div` rodea sólo un elemento de bloque, o un elemento `span` sólo un elemento en línea, entonces su uso es innecesario.

Por ejemplo, observad el uso que se hace de los elementos `div` y `span` en el siguiente etiquetado, que es muy simple:

```
1 <body>
2   <div id="mainContent">
3     <h1>Título de la página</h1>
4     <p>Éste es el primer párrafo de contenido de mi página de ejemplo.</p>
5     
6     <p>Éste es el segundo párrafo de contenido de mi página de ejemplo. Es muy
      parecido al primero, pero aquí hay una <span id="specialAlert">alerta especial
      que queremos colorear y cuyo tamaño de texto queremos aumentar mediante
      CSS</span>.
      No se trata de un énfasis estándar, sino más bien un estilo, de manera que
      <em> y <strong> no son del todo adecuados.</p>
7   </div>
8 </body>
```

Ahora podéis seleccionar el contenido del interior de los elementos `div` y `span` utilizando sus atributos `id` y aplicarles estilos y posicionamientos especiales con el CSS.

8.3. Más explicaciones sobre la agrupación de contenido

Si observamos el código fuente de muchas páginas de Internet, veremos muchos elementos `div` anidados que incluyen algunas metáforas habituales en las `class` y/o `id` de los elementos; por ejemplo, `header`, `footer`, `content`, `sidebar`, etc.

Vuestros nombres de `class` e `id` deben ser siempre semánticos, lo cual quiere decir que deberían remitir al significado/papel del contenido y no limitarse a

hacer referencia a su aspecto visual. Así pues, por ejemplo, `barraLateral` y `mensajeAlerta` son buenos nombres de `class`, mientras que `columnaIzquierdaRoja` y `textoAzulIntermitente`, no. ¿Qué sucedería si después quisierais cambiar el color de la barra lateral de rojo a azul, o su posición en el sitio web de la izquierda a la derecha? ¿Y si quisierais cambiar vuestros mensajes de alerta de azules intermitentes a verde sin intermitencia?

Estas divisiones aportan predictibilidad a la hora de crear estructuras de página y, seguramente mucho más importante, cuando se vuelve a revisar el HTML más adelante ofrecen pistas sobre la parte de la página en la que os encontráis. Una página bien dividida se documenta prácticamente en sí misma con respecto a sus intenciones y a su contenido.

Con la buena intención de dejarlo todo un poco más claro, ahora veremos una estructura de `div` de un sitio web real, concretamente la página de inicio de `dev.opera.com`. Tened en cuenta que este ejemplo de código no incluye ningún contenido, si dejamos de lado algunos otros elementos que hemos puesto porque son importantes para la estructura del sitio. Básicamente, lo que queremos es únicamente reproducir la estructura real del sitio tal como está definida con los elementos `div`. En el código siguiente, leed con atención los comentarios HTML; los hemos puesto para explicar la estructura del sitio. Mientras vais revisando este código, abrid la página principal de `dev.opera.com` en una nueva pestaña o ventana del navegador para poder ver el aspecto del sitio mientras exploráis su estructura.

```
1 <body>
2 <!-- En primer lugar tenemos un elemento div "wrap", que rodea toda la página
   y permite un control más preciso de todo él en general -->
3 <div id="wrap">
4 <!-- Esta lista desordenada contiene la lista de enlaces de todos los sitios
   diferentes de Opera, que podéis ver en la parte superior de la página -->
5 <ul id="sitenav" class="hidemobile">
6     ...
7 </ul>
8     ...
9 <!-- Éste es el formulario de búsqueda - el cuadro de búsqueda que podéis
   ver en la parte superior derecha de la página -->
10 <form action="/search/" method="get" id="search">
11     <div>
12         ...
13     </div>
14 </form>
15 <!-- Esta lista desordenada contiene el menú de navegación principal del
   sitio - el menú horizontal de pestañas que hay justo debajo del gráfico del
   título principal -->
```

```
13     <ul id="menu">
14         ...
15     </ul>
16     <!-- Este elemento div anidado forma la estructura del cuadro de inicio de
17     sesión, en el que introduciréis el nombre de usuario y la contraseña para
18     acceder al sitio. Sólo lo veréis si aún no habéis iniciado ninguna sesión. -->
19     <div id="loginbox">
20         <div id="login">
21             ...
22         </div>
23     </div>
24     <!-- Esta serie de elementos div anidados es donde está el contenido
25     principal de la página - todos los resúmenes de los artículos que forman
26     el grueso del contenido de la página -->
27     <div id="content2">
28         <div id="main">
29             ...
30             <div class="major">
31                 ...
32             </div>
33             <div class="major">
34                 ...
35             </div>
36             ...
37         </div>
38     </div>
39     <!-- Este elemento div contiene la barra lateral de la página - las
40     categorías de artículos, los últimos comentarios, etc -->
41     <div id="side">
42         ...
43     </div>
44     <!-- Este elemento div contiene el pie de página, que es donde veréis el
45     mensaje de copyright y varios enlaces en la parte inferior de la página. -->
46     <div id="footer">
47         ...
48     </div>
49     <!-- El final de la página - es la etiqueta de cierre para el elemento div
50     "wrap" -->
51 </div>
52 </body>
```

8.4. Información adicional

Algunos contenidos tienen información adicional, que es de utilidad para los agentes de usuario y para otros analizadores, y ésta se debe comunicar por medio de un atributo. Los elementos `span` son a menudo una buena manera de adjuntar esta información al contenido de una página web, tal como veréis a continuación.

Un buen ejemplo de esto es un idioma diferente que aparece en un documento. Por ejemplo:

```
<p><q>Plus ça change, plus c'est la même chose</q> she said.</p>
```

Aunque el idioma del documento principal es el inglés, la cita es en francés. Esto se indicaría mediante el uso del atributo `lang`, de la siguiente manera:

```
<p><q lang='fr'>Plus ça change, plus c'est la même chose</q> she said.</p>
```

En este ejemplo era muy fácil indicar este cambio de idioma, ya que aparecía sólo en una cita, con lo cual el elemento `q` era perfecto para rodear el contenido. Hay algunos casos, sin embargo, en los que no existe ningún elemento semántico adecuado fácilmente disponible, con lo cual deberíais recurrir a `span` o `div`. Por ejemplo:

```
<p>A screen reader will read the French word chat (cat) as chat (to talk informally) unless it is properly marked up.</p>
```

En este ejemplo, la primera aparición de la palabra *chat*, que es una palabra en francés en un documento en inglés, debería tener esta diferencia indicada a fin de que no se pudiera interpretar como la palabra inglesa *chat*. En este caso, la manera correcta de hacerlo es con un elemento `span` en torno a la palabra *chat*, ya que no hay ningún otro elemento HTML adecuado para rodear la palabra francesa (no queremos añadir ningún énfasis a la palabra, no es una cita, no es un código, etc.). Como es una única palabra dentro de una frase, estaremos trabajando con un elemento insertado. Por lo tanto, este ejemplo se debería escribir de la manera siguiente:

```
<p> A screen reader will read the French word <span lang='fr'>chat</span> (cat) as chat (to talk informally) unless it is properly marked up.</p>
```

Ésta es la misma técnica que la utilizada en los microformatos* para etiquetar formatos de datos comunes en páginas web. Podéis encontrar mucha más información sobre los microformatos en algunos de los apartados más avanzados sobre HTML en developer.mozilla.org**.

* <http://microformats.org/about/>
** <https://developer.mozilla.org/en-US/docs/Web/HTML/microformats>

8.5. Enganches para JavaScript, y también para CSS

Antes ya hemos hablado de cómo se pueden utilizar `div` y `span` junto con los atributos `id` y `class` para ofrecer enganches con los que aplicar estilos y posicionamiento CSS en ciertas partes de vuestro contenido. Lo mismo también se puede hacer para aplicar JavaScript a vuestro documento.

Si el JavaScript debe encontrar y manipular un elemento concreto, lo más habitual es aplicarle un `id` y entonces utilizar la función `getElementById` para encontrarlo. En la última parte de esta asignatura aprenderéis muchas más cosas sobre JavaScript.

8.6. “div-itis”

Una de las cosas que hay que tener en cuenta es un efecto que, entre la comunidad de desarrolladores de webs, se conoce normalmente como “div-itis”.

Es muy fácil añadir estilos mediante muchos elementos `div` o `span` anidados, pero es una tentación en la que deberíamos evitar caer siempre que sea posible. En la mayoría de los casos, se pueden añadir estilos o funciones JavaScript a los elementos ya existentes del documento. El uso de un contenedor genérico debería ser el último recurso; es mejor intentar escribir páginas web empezando sólo con los elementos semánticos y añadir contenedores sólo cuando sea necesario.

8.7. Semántica inadecuada

En esta sección exploraremos algunos de los errores más habituales que se deben tener en cuenta a la hora de etiquetar el contenido con HTML y que se deben evitar siempre que sea posible.

8.7.1. Párrafos “genéricos”

Algunas veces es muy tentador utilizar un elemento `p` (párrafo) en torno a cualquier bloque de texto, pero eso no es demasiado correcto.

Tal como ya hemos explicado en el apartado anterior sobre el etiquetado del contenido:

“Si son sólo unas cuantas palabras y no una frase propiamente dicha, entonces quizá sería mejor no etiquetarlo como un párrafo”.

Los elementos correctos para rodear contenido textual inconexo que no tiene ninguna otra relación semántica cubierta por otros elementos del HTML son `div` o `span` (según la posición exacta).

Resumen

Aquí acaban nuestras explicaciones sobre los elementos `span` y `div`; ahora ya deberíais entender mucho mejor cuál es su finalidad y ser capaces de utilizarlos con toda confianza. Los apartados del módulo “Conceptos básicos de CSS” se detendrán mucho más en su uso para crear maquetaciones de página y para otras finalidades.

Preguntas de repaso

1. ¿Cuál es la diferencia entre `div` y `span`?
2. Decid dos de los usos principales de estos elementos en las páginas web.
3. Observad el código fuente de una de las páginas de vuestro sitio web preferido. Mirad qué estructura tiene. ¿Utiliza muchos elementos `div` y `span`? ¿Podéis ver algo malo o inadecuado en la manera de utilizarlos? ¿Cómo se podría mejorar?

9. Creación de múltiples páginas con menús de navegación

Christian Heilmann

En este apartado hablaremos sobre la navegación y menús en sitios. Veréis los diferentes tipos de menús y la manera de crearlos con HTML. También hablaremos sobre la cuestión de la usabilidad y la accesibilidad de los menús. No entraremos aún en la aplicación de estilos en los menús, pero sí que estableceremos las bases para otro apartado posterior de esta asignatura sobre el CSS.

9.1. Las herramientas de vuestro menú HTML: enlaces, anclas y listas

Con el HTML hay que considerar varios tipos diferentes de menús y sistemas de navegación, todos ellos muy relacionados con los elementos `link` y `a` (ancla). Para explicarlo rápidamente:

- Los elementos `link` describen relaciones a través de varios documentos. Por ejemplo, podéis decir a un agente de usuario que el documento actual forma parte de un curso más amplio que incluye varios documentos y qué otro documento es el contenido.
- Las anclas (también conocidas como elementos `a`) permiten enlazar con otro documento o con una sección concreta del documento actual. El agente de usuario no las sigue automáticamente; es necesario que los visitantes las activen con el medio que tengan a su disposición (ratón, teclado, reconocimiento de voz, un conmutador, etc.).

Si no habéis leído los apartados anteriores de esta asignatura sobre los enlaces y sobre las listas, os aconsejamos que los leáis, ya que aquí damos por conocida parte de la información que encontraréis en estos apartados para así evitar repeticiones.

Las anclas y los enlaces no se convierten en menús sin más; es necesario darles una estructura y aplicarles estilos para que tanto el navegador como los usuarios sepan que funcionan como un menú de navegación y no sólo como un grupo de enlaces aleatorios. Si el orden de las páginas no es importante, podéis utilizar una lista no ordenada igual que en el ejemplo de menú de lista no ordenada del archivo “`unordered.html`”^{*}.

Observad que hemos puesto un `id` a los elementos `ul`. Lo hemos hecho para ofrecer un enganche para aplicarles estilos con el CSS y para añadirles más adelante un comportamiento especial con JavaScript. Un `id` es una manera muy eficaz de permitir que otras tecnologías elijan un elemento concreto con el HTML.

! Podéis ver el apartado 6 del módulo “Conceptos básicos de CSS” de esta asignatura.

Nota

Podéis descargaros códigos de ejemplo en “`menu_examples.zip`”^{*} para ir siguiendo las explicaciones; iremos haciendo referencias a ellos a lo largo de este apartado.

^{*} http://mosaic.uoc.edu/ac/le/operafiles/U7/menu_examples.zip

! Podéis ver los enlaces y las listas en los apartados 2 y 4 de este módulo.

^{*} <http://mosaic.uoc.edu/ac/le/operafiles/U7/unordered.html>

Si el orden en el que los visitantes van pasando por todos los documentos es importante, entonces deberíais utilizar una lista ordenada. Si, por ejemplo, tenéis un curso en línea formado por múltiples documentos y cada uno parte de aquello que se ha explicado en el anterior, podéis utilizar una lista ordenada como la del ejemplo de lista ordenada del archivo “ordered.html”*.

* <http://mosaic.uoc.edu/ac/le/operafiles/U7/ordered.html>

El uso de listas para crear menús es ideal por varias razones:

- Todo el HTML se encuentra en un único elemento de lista (`ul`, por ejemplo), lo cual significa que podéis utilizar la cascada del CSS para aplicar un estilo diferente a cada uno de ellos y que es muy fácil llegar hasta los elementos en JavaScript bajando desde el nivel superior.
- Las listas se pueden anidar, lo cual significa que podéis crear fácilmente una navegación con varios niveles de jerarquía.
- Incluso si no se aplica ningún estilo a la lista, la representación del HTML por parte del navegador es lógica y para un visitante es muy fácil ver que estos enlaces van juntos y que forman una única unidad lógica.

Las listas se anidan incrustando la lista imbricada al elemento `li`, y no después de éste. Podéis ver un ejemplo correcto y uno incorrecto en el archivo “nesting.html”*.

* <http://mosaic.uoc.edu/ac/le/operafiles/U7/nesting.html>

Tened en cuenta que los navegadores muestran los dos ejemplos de la misma manera.

La visualización en el navegador no debería ser nunca un indicador de la calidad del código.

Una construcción HTML no válida como la del ejemplo erróneo anterior hará que sea muy difícil aplicarle estilos, añadirle comportamiento con JavaScript o convertirlo a otro formato. La estructura desde elementos UL anidados debe ser siempre `` y nunca ``.

9.2. La necesidad de flexibilidad

Es muy probable que el menú de un sitio cambie a menudo; los sitios web tienen tendencia a crecer orgánicamente a medida que se van añadiendo funciones y que crece la base de usuarios, por lo cual deberíais crear los menús teniendo en cuenta que se añadirán y eliminarán entradas a medida que el sitio vaya progresando, y también que quizá se traducirán a diferentes lenguajes (con lo cual cambiará la longitud de los enlaces). También os podéis encontrar trabajando en sitios

en los que el HTML de los menús se crea dinámicamente utilizando lenguajes del servidor y no HTML estático. Esto, sin embargo, no significa que el conocimiento del HTML ya no tenga ninguna utilidad. De hecho, será cada vez más importante porque estos conocimientos se seguirán necesitando para crear las plantillas HTML con las que trabajará el *script* del servidor.

9.3. Diferentes tipos de menús

Hay varios tipos de menú que deberéis crear en HTML cuando vais trabajando en diferentes sitios web. La mayoría de ellos se pueden crear con listas, aunque algunas veces las restricciones de la interfaz obligan a seguir otro camino (ya hablaremos de ello más adelante). Los menús basados en listas que muy probablemente crearéis son los siguientes:

- Navegación por la página: por ejemplo, un contenido para una única página con enlaces que conducen a diferentes secciones de ésta.
- Navegación por el sitio: una barra de navegación para todo el sitio web (o una subsección de éste) con enlaces que conducen a páginas diferentes del mismo sitio.
- Navegación contextual según el contenido: una lista de enlaces que conducen a páginas estrechamente relacionadas con el tema de la página actual, tanto del mismo sitio como de otros.
- Mapas del sitio: listas largas de enlaces que conducen a todas las páginas diferentes de un sitio web; los enlaces se agrupan en sublistas relacionadas para que sea más fácil entenderlo todo.
- Paginación: enlaces que conducen a otras páginas que constituyen otras secciones o partes de todo un conjunto junto con la página actual, como por ejemplo parte 1, parte 2 y parte 3 de un artículo.

9.3.1. Navegación por la página (tabla de contenidos)

Ya hemos hablado de esto un poco en el apartado sobre los enlaces, pero aquí encontraréis una descripción más detallada de qué significa la navegación por la página y qué hay que hacer para que funcione.

En el ejemplo relativo a este subapartado de navegación por la página (archivo “inpagenavigation.html”*) hemos utilizado una lista de enlaces que conducen a anclas que se encuentran más abajo en la misma página. Para conectar las anclas, es necesario utilizar un atributo `id` en los elementos hacia los cuales se navegará y un atributo `href` consistente en un símbolo de almohadilla seguido del mismo nombre que el valor `id` del ancla a la que queréis que conduzca este enlace. Cada

* <http://mosaic.uoc.edu/ac/1e/operafiles/U7/inpagenavigation.html>

una de las secciones de la página también tiene un enlace para volver al menú que funciona de la misma manera, pero que conduce de vuelta al menú.

9.3.2. Navegación por el sitio

La navegación por el sitio es probablemente el tipo de menú más habitual que deberéis crear. Es el menú de todo el sitio (o de una subsección) que muestra tanto las opciones que pueden elegir los visitantes como la jerarquía del sitio. Para estos menús, las listas son la opción perfecta, tal como veréis en el ejemplo de navegación por el sitio del archivo “home.html”*.

* <http://mosaic.uoc.edu/ac/le/operafiles/U7/home.html>

No encontraréis demasiadas sorpresas, por lo menos desde el punto de vista del HTML en sí. Más adelante en esta asignatura hablaremos sobre la aplicación de estilos a estos tipos de menús con CSS y, en otras asignaturas del grado de Multimedia, sobre la manera de añadir comportamiento con JavaScript. Algo importante que hay que tener en cuenta es el modo de destacar el documento actual del menú, con el fin de dar al usuario la sensación de estar en un sitio concreto y que se está moviendo de un sitio a otro (aunque en realidad no se mueve en absoluto, a no ser que utilice algún dispositivo móvil para navegar por la web). Eso es lo que veremos a continuación.

1) Dar a los visitantes la sensación de “Usted está aquí”

La regla principal para el desarrollo de webs y la navegación es que el documento actual no se debe enlazar nunca con él mismo, sino que debe ser claramente diferente de las otras entradas del menú. Esto es importante porque da a los visitantes algo a lo que aferrarse y porque les explica dónde se encuentran en su viaje por el sitio. Hay muchos casos diferentes, como aplicaciones web, los enlaces permanentes (*permalinks*) de los blogs y lo que se conoce como sitios web “de una página”, pero en el 99% de los casos un enlace hacia el documento que ya estáis mirando es redundante y resulta confuso para el visitante.

En el apartado sobre los enlaces ya hemos explicado que un enlace es un acuerdo y una responsabilidad: se ofrece a los visitantes una manera de llegar a más información que necesitan, pero hay que tener mucho cuidado porque si el enlace no da a los usuarios lo que quieren, o si tiene un comportamiento inesperado, perderéis credibilidad y su confianza. Si, por ejemplo, ofrecéis un enlace que apunta hacia el documento actual, al activarlo volverá a cargarse el documento. Como usuario esto es algo que no esperáis: ¿de qué sirve hacer clic en este enlace? Lo único que se consigue es que los usuarios queden perplejos.

Por ello, desde el menú no debe haber nunca ningún enlace hasta la página actual. Podríais eliminar el enlace o, mejor todavía, hacer que deje de actuar como un enlace pero destacarlo (por ejemplo con un elemento `strong`); con

ello los usuarios tienen una indicación visual y también indica a los visitantes ciegos que es importante y que es la entrada actual del menú.

Podéis ver un ejemplo de página actual destacada en: “about.html”

<http://mosaic.uoc.edu/ac/le/operafiles/U7/about.html>

2) ¿Cuántas opciones hay que dar a los usuarios en cada momento?

Otra cuestión que hay que tener en cuenta es cuántas opciones queréis dar a los visitantes. Muchos de los menús que veis en la web intentan garantizar que se podrá acceder a todas las páginas del sitio desde un único menú. Aquí es donde entran en juego los trucos de los *scripts* y del CSS; podéis hacer que los menús sean más fáciles de utilizar ocultando partes hasta que los usuarios seleccionen algunas áreas concretas (es lo que algunas veces se conoce como *menús rollover*). Desde un punto de vista técnico es perfecto, pero este enfoque presenta unos cuantos problemas:

- No todos los usuarios podrán utilizar este truco tan inteligente de la manera esperada; los usuarios del teclado, por ejemplo, deberán pasar por todos los enlaces de la página con el tabulador para llegar a lo que buscan.
- Para conseguirlo, deberéis añadir mucho HTML a cada documento de vuestro sitio, y una buena parte de éste puede ser redundante en muchas páginas. Si bajo tres niveles dentro de un menú de este tipo para llegar al documento que quiero leer, no es necesario que pueda ver las opciones que me llevan a una profundidad de 4, 5 y 6 niveles.
- Si ofrecéis demasiadas opciones a la vez, podéis agobiar a los visitantes; a los seres humanos no les gusta tomar decisiones. Sólo debéis pensar en todo el tiempo que necesitáis para elegir un plato en una carta de restaurante un tanto larga.
- Si en una página no hay mucho contenido pero sí muchos enlaces, los motores de búsqueda asumirán que no hay mucha información válida y no le harán demasiado caso, con lo cual será difícil de encontrar cuando se hagan búsquedas en la web.

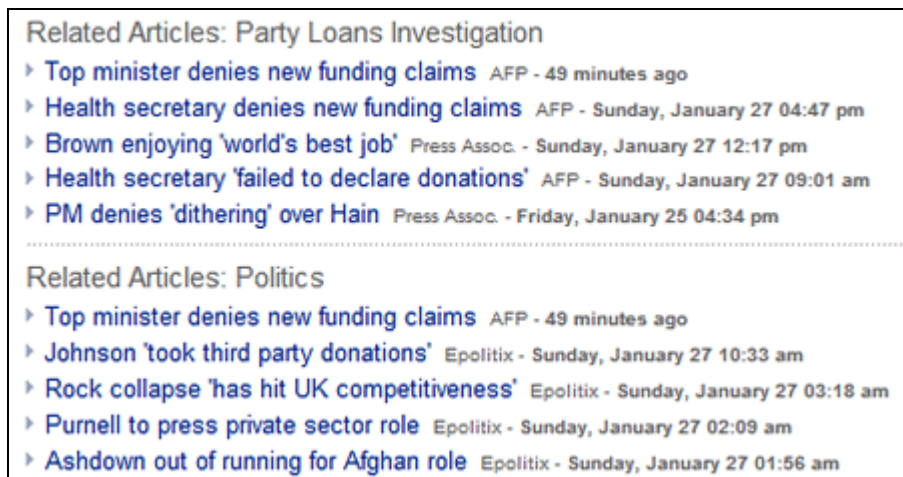
De hecho, la decisión sobre las entradas que pondréis en un menú es vuestra, ya que los diferentes diseños exigirán unas opciones diferentes; pero si tenéis alguna duda, siempre deberíais intentar reducir los menús a los enlaces hasta las secciones principales del sitio. Siempre podéis ofrecer otros menús en los sitios donde sea oportuno.

9.3.3. Menús contextuales

Los menús contextuales son enlaces que se basan en contenido del documento actual y ofrecen más información relacionada con la página en la que os en-

contráis. Un ejemplo clásico de esto son los enlaces “noticias relacionadas” que se suelen encontrar al final de las noticias que leéis, como podéis ver en la figura 1.

Figura 1



Ejemplo de menú contextual: una noticia que ofrece noticias relacionadas.

Estos menús son un poco diferentes de los menús contextuales de las interfaces de usuario, que ofrecen opciones diferentes según el sitio desde el que se accede (como los menús que aparecen con un clic del botón derecho del ratón o con “Ctrl” + clic en las aplicaciones de escritorio que ofrecen unas opciones concretas según el punto en el que se encuentra el puntero del ratón en cada momento).

Los menús contextuales de los sitios web son una manera perfecta de presentar el contenido de otras partes del sitio; y con respecto al HTML, no son nada más que otra lista de enlaces.

9.3.4. Mapas del sitio

Los mapas del sitio son exactamente lo que indica su nombre: mapas de todas las páginas diferentes (o de las secciones principales si hablamos de sitios realmente muy grandes) del sitio. Permiten a los visitantes del sitio ver rápidamente la estructura general e ir rápidamente donde quieran, incluso si la página que necesitan se encuentra en un nivel muy profundo de la jerarquía.

Tanto los mapas del sitio como las búsquedas en el sitio son una manera perfecta de ofrecer a los visitantes una opción alternativa cuando se encuentran perdidos, o de ofrecerles un acceso rápido cuando tienen prisa.

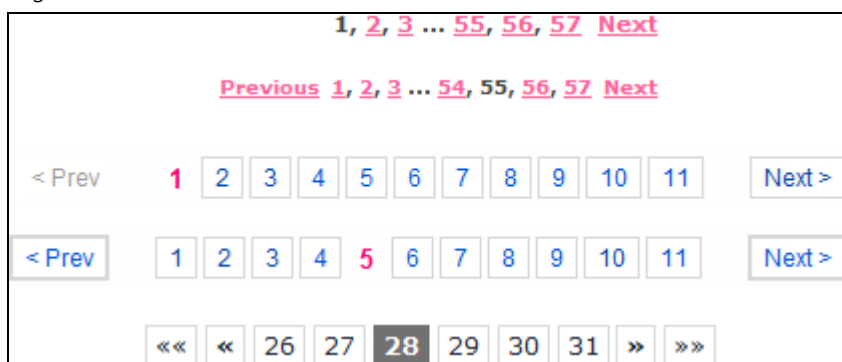
Desde el punto de vista del HTML, pueden ser una lista anidada muy grande llena de enlaces o, en el caso de sitios web muy grandes, títulos de sección con enlaces anidados de las jerarquías específicas de cada sección, o incluso formularios de búsqueda para cada una de las secciones.

9.3.5. Paginación

La paginación es necesaria cuando hay que ofrecer una manera de navegar por documentos muy grandes divididos en varias páginas. Encontraréis paginación, por ejemplo, en los archivos de imágenes muy grandes o en las páginas de resultados de búsqueda (como las de Google o Yahoo).

La paginación es diferente de otros tipos de navegación porque normalmente enlaza hacia el mismo documento, pero con un enlace que contiene más información, como por ejemplo la página desde la que hay que empezar. La figura 2 muestra algunos ejemplos de paginación:

Figura 2



Los menús de paginación permiten que los visitantes naveguen por grandes conjuntos de datos sabiendo en todo momento dónde se encuentran.

El HTML no es nada revolucionario, ya que no deja de ser otra lista de enlaces con el enlace actual (lo que indica los datos actuales y el nivel que ocupan en la paginación) desactivado y destacado (con un elemento `strong`, por ejemplo).

La diferencia principal con la navegación por el sitio es que la paginación incluye una lógica de programación importante. Según el sitio en el que os encontréis de todo el conjunto de datos, deberéis mostrar u ocultar los vínculos previo, siguiente, primero y último. Si tenéis un volumen de información realmente enorme por el que navegar, también es recomendable ofrecer enlaces a puntos de referencia como 100, 200 y muchas más opciones. Por esta razón, muy probablemente no codificaréis menús de este tipo en HTML, sino que los crearéis en el servidor. Esto, sin embargo, no hace que cambien las normas; los datos actuales no se deben enlazar con ellos mismos y no debéis ofrecer enlaces que no lleven a ningún sitio.

9.4. Cuando las listas no son suficientes: mapas de imagen y formularios

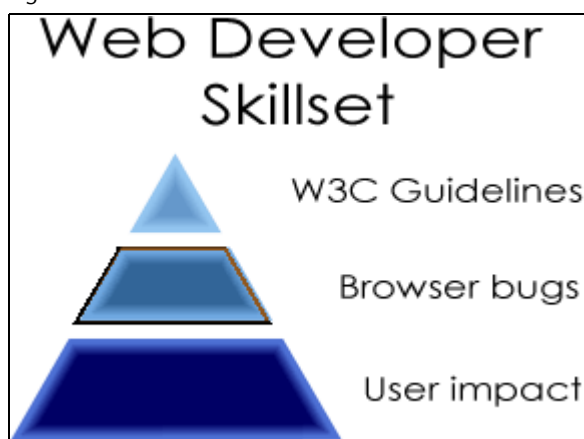
En el 99% de los casos, las listas ordenadas o no ordenadas son alternativas HTML suficientes para crear menús, especialmente porque el orden lógico y la anidación también permiten aplicar estilos muy interesantes con CSS. Sin embargo, hay algunas situaciones que pueden exigir algunas técnicas de diseño diferentes.

9.4.1. Definir zonas sensibles con mapas de imagen

Una de esas técnicas es la de los mapas de imagen en el lado del cliente. Los mapas de imagen transforman una imagen en un menú al convertir secciones de una imagen en áreas interactivas que se pueden enlazar con diferentes documentos. El ejemplo de mapa de imagen* asociado a esta sección convierte una imagen en un menú en el que se puede hacer clic. Probadlo siguiendo el enlace anterior y haciendo clic en las diferentes secciones del triángulo de la imagen que aparece en la figura 3.

* <http://mosaic.uoc.edu/ac/le/operafiles/U7/imagemap.html>

Figura 3



Definiendo un mapa con elementos de área podéis convertir partes de una imagen en elementos interactivos.

Archivo fuente: "imagemap.html"

<http://mosaic.uoc.edu/ac/le/operafiles/U7/imagemap.html>

Podéis convertir una imagen en un menú definiendo un mapa con áreas diferentes (también conocidas como *zonas sensibles*). Deberéis dar al mapa un atributo `name` y conectar la imagen y el mapa con el atributo `usemap` del elemento `img`. Observad que funciona exactamente igual que los enlaces en la misma página, lo que significa que el valor del atributo `usemap` deberá ir precedido de una almohadilla.

Cada una de las áreas debe tener varios atributos:

`href`

Define la URL a la que enlaza el área (que también podría ser un destino en el mismo documento).

`Alt`

Define un texto alternativo por si no se puede encontrar la imagen o si el agente de usuario no acepta imágenes.

`Shape`

Define la forma del área. Puede ser `rect` para rectángulos, `circle` para círculos o `poly` para formas irregulares definidas mediante polígonos.

Coords

Define las coordenadas en la imagen que se deben convertir en zonas sensibles; estos valores se miden desde el vértice superior izquierdo de la imagen y se pueden medir en píxeles o porcentajes. Para las áreas rectangulares, sólo hay que definir el vértice superior izquierdo y el inferior derecho; para los círculos deberéis definir el centro y el radio; para los polígonos deberéis ofrecer una lista de todos los puntos de vértice.

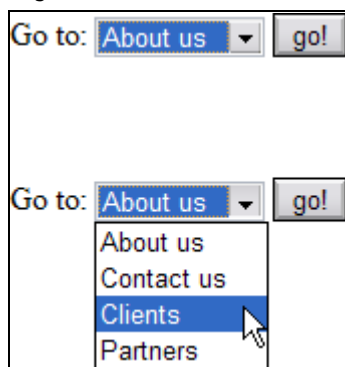
9.4.2. Ahorrar espacio de pantalla y evitar la sobrecarga de enlaces con formularios

Otra técnica que podéis utilizar es crear un formulario con el elemento `select`. Podéis definir las diferentes páginas como opciones del elemento `select`, y así los visitantes podrán elegir una opción y enviar el formulario para pasar a otras páginas. Encontraréis un ejemplo de menú de formulario en el archivo “selectnavigation.html”*.

* <http://mosaic.uoc.edu/ac/le/operafiles/U7/selectnavigation.html>

La ventaja más obvia del uso de este tipo de menú es que podéis ofrecer un montón de opciones sin ocupar demasiado espacio de la pantalla, ya que los navegadores muestran el menú en una línea; podéis ver la figura 4.

Figura 4



Los menús de selección ocupan sólo una línea en la pantalla.

Archivo fuente: “selectnavigationoptgroup.html”

<http://mosaic.uoc.edu/ac/le/operafiles/U7/selectnavigationoptgroup.html>

Podéis ir más lejos y agrupar las diferentes opciones del menú con el elemento `optgroup`, tal como podéis ver en el ejemplo del archivo “selectnavigationoptgroup.html”*.

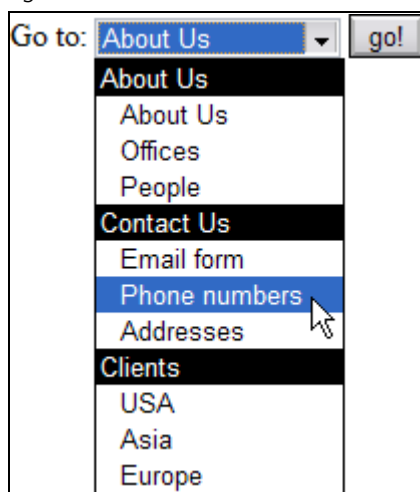
* <http://mosaic.uoc.edu/ac/le/operafiles/U7/selectnavigationoptgroup.html>

Entonces aparecerá un menú con opciones que no se podrán seleccionar (son los nombres de los grupos), tal como se muestra en la figura 5.

Esta técnica tiene la ventaja de que ocupa muy poco espacio, pero también os obliga a tener un *script* en el servidor para enviar a los visitantes a las páginas que han elegido. Para hacer que los enlaces funcionen, también podéis utilizar

JavaScript, pero no os podéis fiar de que el JavaScript esté disponible; deberéis comprobar que vuestros usuarios puedan utilizar los menús incluso con el JavaScript desactivado.

Figura 5



Los menús de selección pueden incorporar grupos de opciones que os permitirán decir a los visitantes qué opciones están relacionadas entre ellas. Así es tal como aparece en Opera 9.5.

Otra ventaja, aunque menos evidente, es que de esta manera no ofrecéis un exceso de enlaces en el mismo documento. Esto significa que no marearéis a los usuarios de tecnologías de asistencia (que normalmente suelen encontrar todos los enlaces en una lista muy larga). Y también significa que los motores de búsqueda no considerarán los enlaces de vuestra página como algo inútil, ya que la relación entre enlaces y texto hace que el documento parezca un mapa de sitio. No obstante, muchas tecnologías de asistencia pueden generar un mapa de los enlaces de vuestras páginas; si todos vuestros enlaces importantes se encuentran en un menú de selección, existe la posibilidad de que un visitante no los llegue a encontrar nunca. Así pues, es una buena idea ofrecer enlaces de ancla hacia las páginas de destino principales y los menús del elemento select para ofrecer más opciones. Los visitantes los podrán utilizar, pero las máquinas como los robots de los motores de búsqueda no necesitarán saber que existen.

9.5. Dónde poner el menú, y ofrecer opciones para saltárselo

Una última cosa que cabe mencionar sobre los menús HTML es que la situación del menú es muy importante. Pensad en los visitantes que no tienen ningún mecanismo de desplazamiento o que no pueden ver y que, por lo tanto, se fían de la navegación con el teclado para moverse por el sitio. Lo primero con lo que se encontrarán cuando carguen el documento es su ubicación y el título, y a continuación el documento se leerá de principio a fin deteniéndose en cada uno de los enlaces para pedir al visitante si los quieren seguir o no. Otras opciones pueden ser una lista de todos los enlaces o pasar de título a título. Si el menú se encuentra en la parte superior del documento, será lo pri-

mero que encontrará el usuario. El hecho de tener que pasar por 15 o 20 enlaces antes de llegar al contenido puede ser una tarea realmente molesta. Para ello hay dos soluciones disponibles. En primer lugar, podéis poner el menú después del contenido principal del documento (si queréis, también lo podéis poner en la parte superior de la pantalla utilizando el CSS). En segundo lugar, podéis ofrecer un enlace para pasar por alto el menú. Éste no es nada más que un enlace colocado antes del menú principal y que lleva al principio del contenido, con lo que el visitante, si lo desea, se puede saltar el menú e ir directamente al contenido. Podéis añadir otro enlace “ir al menú” al final del documento para así poder volver fácilmente a la parte superior.

Para conocer más detalles sobre este tipo de enlaces, podéis ver ejemplos en: “[skiplinks.html](http://mosaic.uoc.edu/ac/le/operafiles/U7/skiplinks.html)”
<http://mosaic.uoc.edu/ac/le/operafiles/U7/skiplinks.html>

Los enlaces para saltar el menú no sólo son útiles para las personas con este tipo de discapacidades, sino que también lo hacen todo más fácil cuando se navega por un sitio en un dispositivo móvil con una pantalla pequeña.

Resumen

En este apartado hemos explicado los diferentes tipos de menús que probablemente deberéis escribir en HTML. Hemos hablado de:

- Por qué las listas con anclas son la estructura HTML perfecta para definir un menú.
- Por qué es importante no considerar los menús como algo inamovible, sino que hay prever y planificar los cambios que puedan hacerse.
- Navegación por la página: enlaces hacia secciones del documento actual y de vuelta hacia el menú.
- Navegación por el sitio: ofrecer un menú que muestre tanto las páginas del sitio actual como su jerarquía; también hemos explicado por qué es importante destacar la página en la que se encuentra actualmente el usuario.
- Navegación contextual: ofrecer enlaces hacia páginas relacionadas de otros puntos del sitio (o de otros sitios).
- Mapas del sitio: como alternativa y herramienta de orientación para los visitantes que se sientan perdidos o que lleguen con una necesidad específica.
- Paginación: una herramienta que permite a los visitantes navegar por un documento dividido en múltiples páginas.
- Mapas de imagen: crear menús gráficos poniendo zonas sensibles sobre imágenes.

- Menús de formulario: ofrecer muchas opciones sin ocupar mucho espacio de pantalla y sin agobiar a los visitantes y los motores de búsqueda con demasiados enlaces.
- Enlaces para saltar el menú y ubicación del menú.

Más adelante volveremos a hablar de algunos de estos temas en la sección de CSS de esta asignatura y explicaremos cómo se puede conseguir que estas estructuras HTML tengan un aspecto atractivo y sean unos menús todavía más intuitivos para los visitantes.

Preguntas de repaso

1. ¿Por qué es una buena idea etiquetar los menús como listas?
2. Cuando diseñáis un menú de navegación, ¿qué hay que planificar de cara al futuro?
3. ¿Cuáles son las ventajas de utilizar elementos `select` para los menús, y cuáles son los problemas?
4. ¿Qué se define con los elementos `area`, y para qué sirve el atributo `nohref` de un elemento `area`? (no lo encontraréis aquí, deberéis investigar un poco en línea).
5. ¿Cuáles son las ventajas de los enlaces para saltarse el menú?

10. Validar el HTML

Mark Norman Francis

Ahora ya habéis escrito unas cuantas páginas en HTML y parece que se ven bastante bien, pero hay unas cuantas cosas que no son del todo correctas. ¿Cuál es la mejor manera de encontrar aquello que no funciona y garantizar que estas páginas (y todas las páginas que escribáis en el futuro) se vean correctamente con todos los navegadores sin ningún error?

La respuesta es la validación. Hay muchas herramientas disponibles, del W3C y de otros sitios, que os permiten validar el código de vuestros sitios web. Los tres validadores más habituales que utilizaréis son:

- El validador de marcado del W3C*: busca el doctype del (X)HTML que utilizáis para el documento que le dais para comprobar, revisa todo el documento y señala los lugares donde el HTML no sigue correctamente el doctype (es decir, donde hay errores en el HTML).
- El comprobador de enlaces de W3C*: esta herramienta revisa todo el documento que le deis para comprobar y prueba todos los enlaces que encuentra para garantizar que funcionen (es decir, que los valores `href` no apunten hacia recursos que no existen).
- El validador del CSS de W3C*: tal como probablemente ya os imagináis, esta herramienta revisa un documento CSS (o HTML/CSS) y comprueba que el CSS siga correctamente las especificaciones del CSS.

* <http://validator.w3.org/>

* <http://validator.w3.org/checklink>

* <http://jigsaw.w3.org/css-validator/>

En este apartado explicaremos el primero de estos tres validadores y la manera de utilizarlo y cómo interpretar los tipos de resultados típicos que da. El comprobador de enlaces es muy obvio, y sobre el validador del CSS no habrá que explicar mucho nada más una vez que hayáis leído este apartado y los apartados relativos al CSS que encontraréis más adelante en esta asignatura.

Podéis ver los contenidos del módulo "Conceptos básicos de CSS" de esta asignatura.



10.1. Errores

En general, en la programación informática hay dos tipos de problemas con el código:

- Errores sintácticos, que se dan cuando un error de escritura del código hace que el ordenador sea incapaz de ejecutar o compilar el programa adecuadamente.

- Errores de programación (o lógicos), que se dan cuando el código no refleja completamente la intención del programador.

La mayoría de los lenguajes de programación detectan muy fácilmente el primer error, ya que el programa no se ejecuta o no se compila hasta que se soluciona. Esto provoca que encontrar y solucionar estos tipos de problemas en los momentos de desconcierto en los que os preguntáis: “¿por qué no hace lo que debería hacer?” sea mucho más fácil.

HTML no es un lenguaje de programación. Los errores de sintaxis de una página web no hacen normalmente que el navegador web se niegue a abrir la página (a pesar de que XHTML suele ser más estricto que HTML, por lo menos cuando se sirve como datos `application/xhtml+xml` o `text/xml`, según se quiera; y algunos doctypes rechazarán el uso de ciertos tipos de elementos HTML). Ésta es una de las razones principales de la rápida adopción y difusión de la web.

El primer navegador web, WorldWideWeb* (creado por Tim Berners-Lee) era también un editor, y permitía crear páginas web sin tener que aprender HTML. Este editor creaba HTML no válido. Esto se podría haber solucionado, pero estableció un precedente importante que aún existe en todos los navegadores web actuales: es más importante que la gente pueda acceder al contenido que quejarse de errores que la gente no entenderá o que no podrá solucionar.

* <http://www.w3.org/People/Berners-Lee/WorldWideWeb.html>

10.2. ¿Qué es la validación?

Los navegadores web aceptarán las páginas web malas (el término que utilizamos aquí es *no válidas*) y harán todo lo que puedan para reproducir el código intentando adivinar cuáles eran las intenciones del autor, pero también se puede comprobar si el HTML se ha escrito correctamente; de hecho, tal como veréis a continuación, vale la pena hacerlo. Eso se conoce como **validar el HTML**.

El programa de validación compara el código HTML de la página web con las normas de su doctype y dice si no se han respetado estas normas y dónde.

Podéis ver el doctype en el apartado 3 del módulo “Fundamentos de HTML”.

10.3. ¿Por qué validar?

Algunos desarrolladores de webs opinan que si una página web se ve bien en los navegadores, el hecho de que no se valide no tiene ninguna importancia. Consideran que la validación es un objetivo ideal, pero ni mucho menos, obligatorio.



Podéis ver cómo empezar una lista ordenada en un punto diferente de 1 en el apartado 2 de este módulo.

Esta actitud no deja de ser sabia. La especificación HTML no es perfecta, y ahora incluso ha quedado un poco anticuada. Algunos detalles que son perfectamente correctos (como por ejemplo empezar una lista ordenada en un punto diferente de 1) no son válidas en HTML.

Sin embargo, tal como dice el dicho:

“Aprende las normas para así saber cómo infringirlas correctamente”.

Hay dos razones muy potentes para validar el HTML mientras lo vais escribiendo.

- Las personas no son perfectas, igual que tampoco lo es el código; todos cometemos errores, y vuestras páginas web serán de una calidad superior (es decir, funcionarán mejor) si los elimináis.
- Los navegadores cambian. En el futuro es muy probable que los navegadores sean menos condescendientes cuando analicen un código no válido, y no al revés.

La validación es vuestro sistema de alerta precoz ante la introducción de problemas en vuestras páginas que entonces se pueden manifestar de maneras realmente interesantes y difíciles de determinar. Cuando un navegador encuentra código HTML no válido, debe tomar una decisión bien fundamentada sobre cuáles eran vuestras intenciones, y los diferentes navegadores pueden llegar a respuestas diferentes.

10.4. Los diferentes navegadores interpretan el HTML no válido de una manera diferente

El HTML válido es el único contrato que tenéis con los fabricantes de los navegadores. La especificación de HTML dice cómo lo deberíais escribir vosotros y cómo deberían interpretar vuestro documento los navegadores. Actualmente, la conformidad de los navegadores con los estándares ha llegado al punto en el que, si escribís un código válido, todos los navegadores principales interpretarán vuestro código de la misma manera. Con el HTML eso sucede prácticamente siempre, mientras que los demás estándares se pueden encontrar con algunas otras diferencias aquí y allí con respecto a su aceptación.

Pero ¿qué sucede si enviáis un código no válido a un navegador? ¿Qué consecuencias tiene? La respuesta es que entra en acción la gestión de errores del navegador para decidir qué hay que hacer con el código. Básicamente, lo que hace es decir: “De acuerdo, este código no se puede validar, pero ¿cómo podemos presentar esta página al usuario final? Llenaremos los huecos de la manera siguiente”.

Suena muy bien, ¿no? Si la página contiene algunos errores, ¿el navegador llenará los huecos por vosotros? Pues no exactamente, porque cada navegador hace las cosas a su manera. Por ejemplo:

```
<p><strong>Este texto debería estar en negrita</strong></p>
<p>¿Este texto debería estar en negrita? ¿Cómo queda el HTML cuando se
representa?</p>

<p><a href="#"></strong>Este texto debería ser un enlace</p>
<p>¿Este texto debería ser un enlace? ¿Cómo queda el HTML cuando se representa?</p>
```

Los errores son que el elemento `strong` se encuentra anidado incorrectamente a través de múltiples elementos de bloque y que el elemento de ancla no se ha cerrado. Cuando se intenta verlo con diferentes navegadores, éstos interpretan el código de maneras muy diferentes:

- Opera convierte los elementos consecutivos en hijos del elemento negrita.
- Firefox añade elementos negrita adicionales entre los párrafos que no estaban presentes en el etiquetado.
- Internet Explorer pone el texto “Este texto debería ser un enlace” fuera de la etiqueta de ancla que crea el enlace.

Ninguno de los comportamientos de los diferentes navegadores es incorrecto; todos intentan reparar los errores de vuestro código incorrecto. La conclusión es, pues, que hay que evitar siempre que sea posible el etiquetado no válido en vuestra página.

10.4.1. El “quirks mode”

Otra cosa que deberíais conocer es lo que se conoce como “**Quirksmode**” o “**modo quirks**” (*peculiaridades*). Es el modo al que pasa el navegador cuando encuentra una página con un doctype incorrecto, o sin doctype. En este modo, el navegador hace todo lo que puede para saber qué serie de reglas debe utilizar para validar el código y repara los errores lo mejor que sabe. Este modo existe realmente para permitir la visualización de las páginas más antiguas, y no se debe utilizar nunca para crear una página nueva.

10.5. Cómo validar las páginas

Ahora que ya hemos explorado toda la parte teórica que hay detrás de la validación del HTML, ya podemos pasar a la parte más sencilla: la validación en sí. Bien, de hecho, no es exactamente así. Poner una URL en un validador y

Nota

Podéis encontrar la versión original de este ejemplo en el artículo de Hallvord Steen “Same DOM errors, different browser interpretations”^{*}; si lo leéis, encontraréis un tratamiento mucho más detallado de los errores del HTML e información sobre las herramientas de depuración.

^{*} <http://www.thinkvitamin.com/features/dev/same-dom-errors-different-browser-interpretations>

ver si la página es válida o no es muy fácil; descubrir qué es lo que está mal y solucionar los errores no es siempre tan fácil, porque los mensajes de error en ocasiones son algo crípticos. A continuación veremos algunos ejemplos.

Lo que miraremos en este subapartado es lo siguiente (también os lo podéis descargar o ver su HTML):

```
1  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
2  <html xmlns="http://www.w3.org/1999/xhtml" lang="en">
3    <head>
4      <title>Validating your HTML</title>
5    </head>
6    <body>
7      <h2>The tale of Herbet Gruel</h2>
8      <p>Welcome to my story. I am a slight whisp of a man, slender and fragile,
        features wrinkled and worn, eyes sunken into their sockets like rabbits
        cowering in their burrows. The <em>years have not been kind to me</em>,
        but yet I hold no regrets, as I have overcome all that sought to ail me,
        and have been allowed to bide my time, making mischief as I travel to and
        fro, 'cross the unyielding landscape of the <a href="http://outer-rim-
        rocks.co.uk" colspan="3">outer rim</a>.</p>
9      <h3>Buster</h3>
10     <p>Buster is my guardian angel. Before that, he was my dog. Before that,
        who knows? I lost my dog many moons ago while out hunting geese in the
        undergrowth. A shot rang out from my rifle, and I called for Buster to
        collect the goose I had felled. He ran off towards where the bird had
        landed, but never returned. I never found his body, but I comfort myself
        with the thought that he did not die; rather he transcended to a higher
        place, and now watches over me, to ensure my well-being.
11     <h3>My possessions</h3>
12     <p>A travelling man needs very little to accompany him on the road:</p>
13     <ul>
14       <li>My hat full of memories</li>
15       <li>My trusty walking cane</li>
16       <li>A purse that did contain gold at one time</li>
17       <li>A diary, from the year 1874</li>
18       <li>An empty glasses case</li>
19       <li>A newspaper, for when I need to look busy</li>
20     </ul>
21   </body>
```

Archivo fuente: "example_validation.html"

http://mosaic.uoc.edu/ac/le/operafiles/example_validation.html

Esta página tan sencilla está formada por tres títulos, tres párrafos, un enlace y una lista no ordenada. Utiliza el doctype XHTML 1.0 Strict como serie de reglas de validación. El documento contiene algunos errores, que descubriréis más adelante con el validador de HTML del W3C.

10.5.1. El validador de HTML del W3C

Como ya hemos comentado, el W3C ofrece un validador en línea. Os será muy útil poder cambiar entre diferentes pestañas para ir alternando entre el validador y este apartado mientras vais siguiendo este ejemplo.

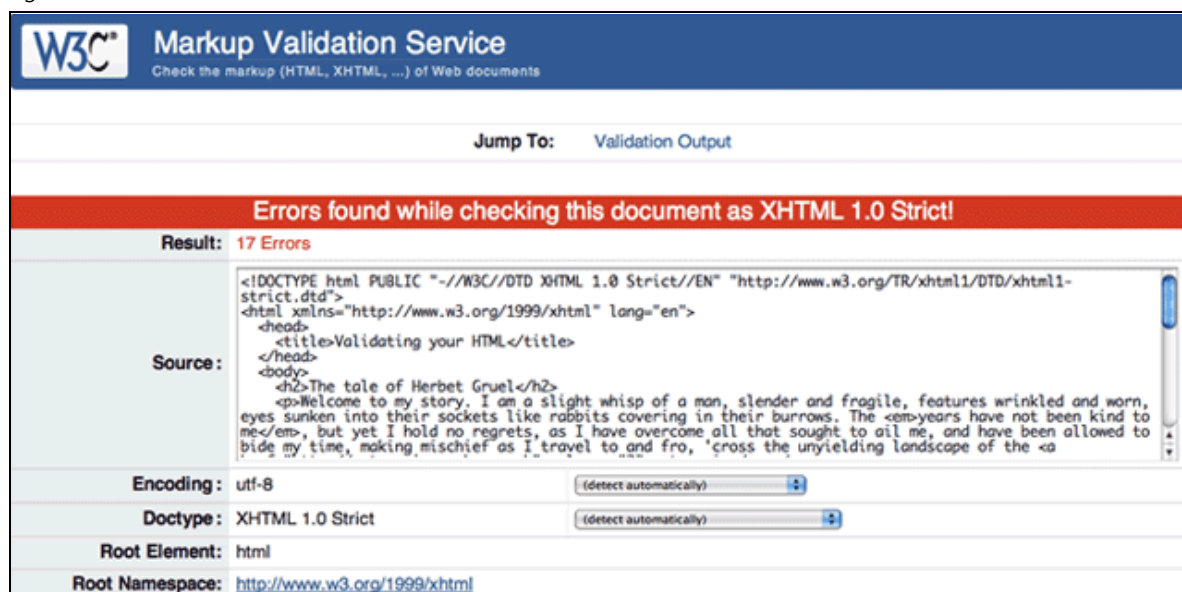
Tened en cuenta que también podéis validar páginas con el validador del W3C directamente desde el navegador Opera sencillamente haciendo clic con el botón derecho del ratón o con un “Ctrl” + clic y seleccionando la opción “Validate” (validar).

Veréis que el validador tiene tres pestañas en la parte superior del interfaz:

- Validate by URI (validar por URI): permite introducir la dirección de una página que ya se encuentra en Internet para su validación.
- Validate by File Upload (validar por carga de archivo): permite cargar un archivo HTML para su validación.
- Validate by Direct Input (validar por entrada directa): permite pegar el contenido de un archivo HTML en la ventana para su validación.

Sea cual sea el método que utilicéis, el resultado debería ser el mismo; creemos que lo más sencillo es comprobar la página de ejemplo que tenemos aquí copiando todo el código anterior y pegándolo en una tercera pestaña. Si lo hacéis, deberíais obtener el resultado que podéis ver en la figura 1:

Figura 1



Resultados de la validación del documento de muestra, ¡17 errores!

Puede parecer muy preocupante, especialmente si os digo que en este documento ¡no hay realmente 17 errores! No os desesperéis; dice que hay más errores de los que hay realmente porque normalmente un error al principio de la página tiene un efecto en cascada, que hace que el validador informe de más errores a medida que va bajando por la página, ya que parece que hay más elementos que no están cerrados o que no están correctamente anidados. Sólo es necesario que penséis en el significado del mensaje de error y que encontréis los errores obvios de etiquetado. La siguiente tabla muestra todos los errores que hemos reparado para poder validar la página, junto con las deducciones para averiguar qué era lo que no funcionaba y la solución aplicada.

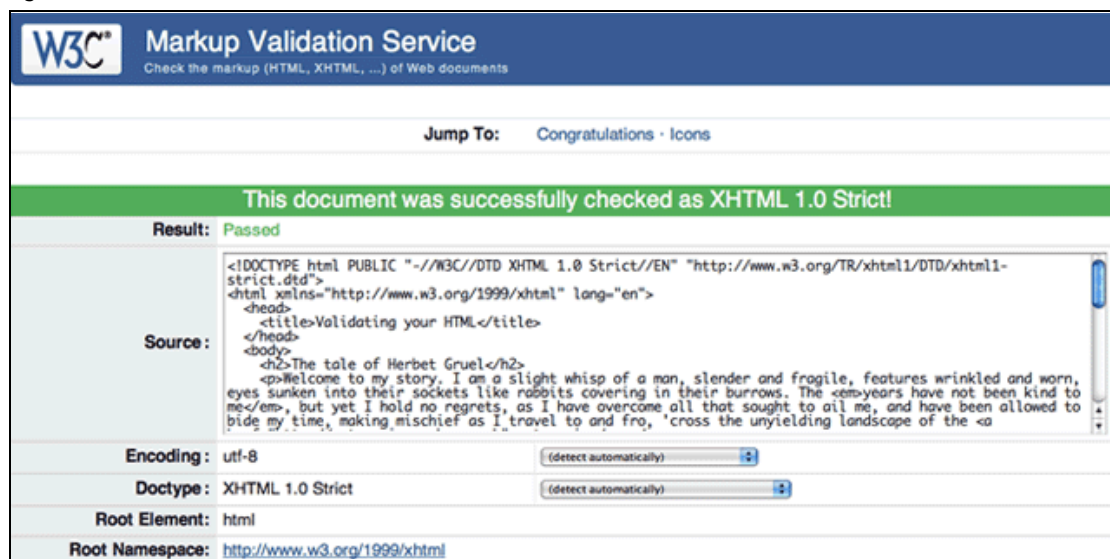
Tabla 1

Mensaje de error	Deducción	Reparación
Línea 8, columna 461: no hay ningún atributo "colspan"	Sabemos que hay un atributo <code>colspan</code> y que es un HTML válido; por lo tanto, ¿por qué dice que no lo hay? Esperad; quizá quiere decir que se utiliza en un elemento en el que no se debería utilizar. Evidentemente, se utiliza en un elemento <code>a</code> . ¡Mal!	Hemos eliminado el atributo <code>colspan</code> del elemento <code>a</code> .
Línea 13, columna 7: el tipo de documento no permite el elemento "h3" aquí; falta una etiqueta de apertura de "object", "applet", "map", "iframe", "button", "ins", "del". <h3>My possessions</h3>	A primera vista también parece extraña, ya que el elemento <code>h3</code> está cerrado y en este contexto está permitido. Debéis tener en cuenta que, a menudo, este mensaje de error significa que hay un elemento no cerrado cerca...	Hemos añadido una etiqueta de cierre <code></p></code> a la línea que hay sobre el título en cuestión.
Línea 19, columna 40: el tipo de documento no permite el elemento "li" aquí; falta una etiqueta de apertura de "ul", "ol", "menu", "div". A diary, from the year 1874	Éste es muy fácil; en la línea indicada podéis ver rápidamente que la etiqueta <code>li</code> final no tiene la barra inclinada de cierre (<code>/</code>)	Hemos añadido una barra inclinada de cierre a la línea en cuestión.
Línea 23, columna 9: no está la etiqueta de cierre para "html", pero se ha especificado OMITTAG NO. </body>	En este caso también es muy fácil ver que falta la etiqueta <code>html</code> final. Incluso la explicación del mensaje de error empieza diciendo que quizá habéis olvidado cerrar un elemento.	Hemos añadido el elemento <code>html</code> final que faltaba.

Los errores reparados para validar la página de ejemplo

Después de corregir estos errores, el validador da un mensaje bastante satisfactorio, tal como muestra la figura 2:

Figura 2



Un mensaje satisfactorio que dice que ya se han reparado todos los errores.

Eso es todo lo que había que hacer. Sólo necesitáis no poneros nerviosos y que recordéis el doctype que utilizáis para validar la página.

Descargaos o mirad la versión correcta del HTML en: “example_validation_fixed.html”
http://mosaic.uoc.edu/ac/le/operafiles/example_validation_fixed.html

Resumen

Después de leer este apartado, deberíais poder trabajar con comodidad con el validador en línea de W3C para validar vuestro HTML. Esto, sin embargo, es sólo la punta del iceberg con respecto a la validación; a continuación encontraréis una lista con unas herramientas más complicadas que os pueden ayudar cuando vuestras páginas se vayan haciendo grandes y se vayan complicando.

Preguntas de repaso

1. ¿Qué sucede cuando un navegador analiza un HTML no válido?
2. ¿Qué problema hay?
3. ¿El uso de un conjunto de marcos en un documento validado con el doctype HTML 4 Strict generará un error?

Otras herramientas que podéis utilizar

El menú de depuración de Opera

<http://dragonfly.opera.com/app/debugmenu/DebugMenu.ini>

Bookmarklet de validación general

<http://www.squarefree.com/bookmarklets/validation.html>

La extensión Web Developer de Firefox

<http://chrispederick.com/work/web-developer/>

La barra de herramientas del desarrollador de IE

<http://www.microsoft.com/downloads/details.aspx?FamilyID=e59c3964-672d-4511-bb3e-2d5e1db91038&displaylang=en>

Safari tidy

<http://zappatic.net/safaritidy/>

HTML tidy

<http://tidy.sourceforge.net/>