

INTRODUCCIÓN A XHTML

JAVIER EGUILUZ

LIBROSWEB

Introducción a XHTML

Javier Eguiluz

Fecha publicación: 14/03/2013

Este libro fue publicado con la aplicación **easybook** (<http://easybook-project.org>), una herramienta de software libre para la publicación de libros digitales.

Licencia

Este libro se publica bajo la licencia *Creative Commons Reconocimiento - No Comercial - Compartir Igual 3.0*, cuyos detalles puedes consultar en <http://creativecommons.org/licenses/by-nc-sa/3.0/es/>

Puedes copiar, distribuir y comunicar públicamente la obra, incluso transformándola, siempre que cumplas todas las condiciones siguientes:

- **Reconocimiento:** debes reconocer siempre la autoría de la obra original, indicando tanto el nombre del autor (Javier Eguiluz) como el nombre del sitio donde se publicó originalmente (librosweb.es). Este reconocimiento no debe hacerse de una manera que sugiera que el autor o el sitio apoyan el uso que haces de su obra.
- **No comercial:** no puedes utilizar esta obra con fines comerciales de ningún tipo. Entre otros, no puedes vender esta obra bajo ningún concepto y tampoco puedes publicar estos contenidos en sitios web que incluyan publicidad de cualquier tipo.
- **Compartir igual:** si alteras o transformas esta obra o si realizas una obra derivada, debes compartir tu trabajo obligatoriamente bajo esta misma licencia.

Índice de contenidos

Capítulo 1. Introducción	7
1.1. ¿Qué es HTML?	7
1.2. Breve historia de HTML.....	7
1.3. Especificación oficial	9
1.4. HTML y XHTML.....	9
1.5. HTML y CSS	10
Capítulo 2. Características básicas	11
2.1. Lenguajes de etiquetas.....	11
2.2. El primer documento HTML.....	13
2.3. Etiquetas y atributos	15
2.4. Elementos HTML.....	20
2.5. Sintaxis de las etiquetas XHTML	22
Capítulo 3. Texto	25
3.1. Estructurar.....	26
3.2. Marcado básico de texto	30
3.3. Marcado avanzado de texto	35
3.4. Marcado genérico de texto	38
3.5. Espacios en blanco y nuevas líneas	39
3.6. Codificación de caracteres	47
Capítulo 4. Enlaces	51
4.1. URL	51
4.2. Enlaces relativos y absolutos	54
4.3. Enlaces básicos	59
4.4. Enlaces avanzados	63
4.5. Otros tipos de enlaces	66
4.6. Ejemplos de enlaces habituales	68
Capítulo 5. Listas	73
5.1. Listas no ordenadas	73
5.2. Listas ordenadas	74
5.3. Listas de definición.....	75

Capítulo 6. Imágenes y objetos	79
6.1. Imágenes.....	79
6.2. Mapas de imagen	82
6.3. Objetos	84
Capítulo 7. Tablas	89
7.1. Tablas básicas	90
7.2. Tablas avanzadas	100
Capítulo 8. Formularios	107
8.1. Formularios básicos	107
8.2. Elementos de formulario.....	109
8.3. Formularios avanzados	116
8.4. Otros elementos de formulario	119
Capítulo 9. Estructura y layout	127
Capítulo 10. Metainformación	131
10.1. Estructura de la cabecera	131
10.2. Metadatos.....	133
10.3. DOCTYPE.....	134
Capítulo 11. Otras etiquetas importantes	137
11.1. Comentarios.....	137
11.2. JavaScript.....	138
11.3. CSS.....	139
11.4. Iframes	140
11.5. Otras etiquetas	142
Capítulo 12. Accesibilidad	145
12.1. Requisitos del nivel A de accesibilidad.....	146
Capítulo 13. Validación	149
13.1. Validación con Dreamweaver	149
13.2. Validador del W3C	151
13.3. Otros validadores	152
Capítulo 14. Fragmentos de código	155
14.1. Documento XHTML	160
14.2. Cabecera XHTML.....	161

14.3. Tabla	161
14.4. Formulario	162
Capítulo 15. Ejercicios resueltos	165
15.1. Ejercicio 1	165
15.2. Ejercicio 2	165
15.3. Ejercicio 3	166
15.4. Ejercicio 4	167
15.5. Ejercicio 5	168
15.6. Ejercicio 6	168
15.7. Ejercicio 7	170
15.8. Ejercicio 8	171
15.9. Ejercicio 9	172
15.10. Ejercicio 10	173
15.11. Ejercicio 11	174
15.12. Ejercicio 12	175
15.13. Ejercicio 13	176
15.14. Ejercicio 14	178
15.15. Ejercicio 15	179
15.16. Ejercicio 16	182

Esta página se ha dejado vacía a propósito

Capítulo 1.

Introducción

1.1. ¿Qué es HTML?

Definiéndolo de forma sencilla, *"HTML es lo que se utiliza para crear todas las páginas web de Internet"*. Más concretamente, HTML es el lenguaje con el que se *"escriben"* la mayoría de páginas web.

Los diseñadores utilizan el lenguaje HTML para crear sus páginas web, los programas que utilizan los diseñadores generan páginas escritas en HTML y los navegadores que utilizamos los usuarios muestran las páginas web después de leer su contenido HTML.

Aunque HTML es un lenguaje que utilizan los ordenadores y los programas de diseño, es muy fácil de aprender y escribir por parte de las personas. En realidad, HTML son las siglas de *HyperText Markup Language* y más adelante se verá el significado de cada una de estas palabras.

El lenguaje HTML es un estándar reconocido en todo el mundo y cuyas normas define un organismo sin ánimo de lucro llamado World Wide Web Consortium (<http://www.w3.org/>), más conocido como **W3C**. Como se trata de un estándar reconocido por todas las empresas relacionadas con el mundo de Internet, una misma página HTML se visualiza de forma muy similar en cualquier navegador de cualquier sistema operativo.

El propio **W3C** define el lenguaje HTML como *"un lenguaje reconocido universalmente y que permite publicar información de forma global"*. Desde su creación, el lenguaje HTML ha pasado de ser un lenguaje utilizado exclusivamente para crear documentos electrónicos a ser un lenguaje que se utiliza en muchas aplicaciones electrónicas como buscadores, tiendas online y banca electrónica.

1.2. Breve historia de HTML

La historia completa de HTML es tan interesante como larga, por lo que a continuación se muestra su historia resumida a partir de la información que se puede encontrar en la Wikipedia.

El origen de HTML se remonta a 1980, cuando el físico **Tim Berners-Lee**, trabajador del CERN (<http://www.cern.ch/>) (*Organización Europea para la Investigación Nuclear*) propuso un nuevo sistema de *"hipertexto"* para compartir documentos.

Los sistemas de "*hipertexto*" habían sido desarrollados años antes. En el ámbito de la informática, el "*hipertexto*" permitía que los usuarios accedieran a la información relacionada con los documentos electrónicos que estaban visualizando. De cierta manera, los primitivos sistemas de "*hipertexto*" podrían asimilarse a los enlaces de las páginas web actuales.

Tras finalizar el desarrollo de su sistema de "*hipertexto*", Tim Berners-Lee lo presentó a una convocatoria organizada para desarrollar un sistema de "*hipertexto*" para Internet. Después de unir sus fuerzas con el ingeniero de sistemas **Robert Cailliau**, presentaron la propuesta ganadora llamada *WorldWideWeb* (W3).

El primer documento formal con la descripción de HTML se publicó en 1991 bajo el nombre *HTML Tags* (<http://www.w3.org/History/19921103-hypertext/hypertext/WWW/MarkUp/Tags.html>) (*Etiquetas HTML*) y todavía hoy puede ser consultado online a modo de *reliquia informática*.

La primera propuesta oficial para convertir HTML en un estándar se realizó en 1993 por parte del organismo IETF (<http://www.ietf.org/>) (*Internet Engineering Task Force*). Aunque se consiguieron avances significativos (en esta época se definieron las etiquetas para imágenes, tablas y formularios) ninguna de las dos propuestas de estándar, llamadas HTML y HTML+ consiguieron convertirse en estándar oficial.

En 1995, el organismo IETF organiza un grupo de trabajo de HTML y consigue publicar, el 22 de septiembre de ese mismo año, el estándar HTML 2.0. A pesar de su nombre, HTML 2.0 es el primer estándar oficial de HTML.

A partir de 1996, los estándares de HTML los publica otro organismo de estandarización llamado W3C (<http://www.w3.org/>) (*World Wide Web Consortium*). La versión HTML 3.2 se publicó el 14 de Enero de 1997 y es la primera recomendación de HTML publicada por el W3C. Esta revisión incorpora los últimos avances de las páginas web desarrolladas hasta 1996, como *applets* de Java y texto que fluye alrededor de las imágenes.

HTML 4.0 se publicó el 24 de Abril de 1998 (siendo una versión corregida de la publicación original del 18 de Diciembre de 1997) y supone un gran salto desde las versiones anteriores. Entre sus novedades más destacadas se encuentran las hojas de estilos CSS, la posibilidad de incluir pequeños programas o *scripts* en las páginas web, mejora de la accesibilidad de las páginas diseñadas, tablas complejas y mejoras en los formularios.

La última especificación oficial de HTML se publicó el 24 de diciembre de 1999 y se denomina HTML 4.01. Se trata de una revisión y actualización de la versión HTML 4.0, por lo que no incluye novedades significativas.

Desde la publicación de HTML 4.01, la actividad de estandarización de HTML se detuvo y el W3C se centró en el desarrollo del estándar XHTML. Por este motivo, en el año 2004, las empresas Apple, Mozilla y Opera mostraron su preocupación por la falta de interés del W3C en HTML y decidieron organizarse en una nueva asociación llamada WHATWG (<http://www.whatwg.org/>) (*Web Hypertext Application Technology Working Group*).

La actividad actual del WHATWG se centra en el futuro estándar HTML 5, cuyo primer borrador oficial (<http://www.w3.org/TR/html5/>) se publicó el 22 de enero de 2008. Debido a la fuerza de las empresas

que forman el grupo WHATWG y a la publicación de los borradores de HTML 5.0, en marzo de 2007 el W3C decidió retomar la actividad estandarizadora de HTML (<http://www.w3.org/2007/03/html-pressrelease>).

De forma paralela a su actividad con HTML, W3C ha continuado con la estandarización de XHTML, una versión *avanzada* de HTML y basada en XML. La primera versión de XHTML se denomina XHTML 1.0 y se publicó el 26 de Enero de 2000 (y posteriormente se revisó el 1 de Agosto de 2002).

XHTML 1.0 es una adaptación de HTML 4.01 al lenguaje XML, por lo que mantiene casi todas sus etiquetas y características, pero añade algunas restricciones y elementos propios de XML. La versión XHTML 1.1 ya ha sido publicada en forma de borrador y pretende modularizar XHTML. También ha sido publicado el borrador de XHTML 2.0, que supondrá un cambio muy importante respecto de las anteriores versiones de XHTML.

1.3. Especificación oficial

El organismo W3C (<http://www.w3.org/>) (*World Wide Web Consortium*) elabora las normas que deben seguir los diseñadores de páginas web para crear las páginas HTML. Las normas oficiales están escritas en inglés y se pueden consultar de forma gratuita en las siguientes direcciones:

- Especificación oficial de HTML 4.01 (<http://www.w3.org/TR/html401/>)
- Especificación oficial de XHTML 1.0 (<http://www.w3.org/TR/xhtml1/>)

El estándar XHTML 1.0 incluye el 95% del estándar HTML 4.01, ya que sólo añade pequeñas mejoras y modificaciones menores. Afortunadamente, no es necesario leer las especificaciones y recomendaciones oficiales de HTML para aprender a diseñar páginas con HTML o XHTML. Las normas oficiales están escritas con un lenguaje bastante formal y algunas secciones son difíciles de comprender. Por ello, en los próximos capítulos se explica de forma sencilla y con decenas de ejemplos la especificación oficial de XHTML.

1.4. HTML y XHTML

El lenguaje XHTML es muy similar al lenguaje HTML. De hecho, XHTML no es más que una adaptación de HTML al lenguaje XML. Técnicamente, HTML es descendiente directo del lenguaje SGML, mientras que XHTML lo es del XML (que a su vez, también es descendiente de SGML).

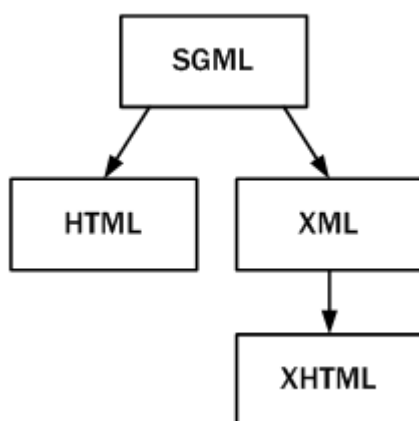


Figura 1.1 Esquema de la evolución de HTML y XHTML

Las páginas y documentos creados con XHTML son muy similares a las páginas y documentos HTML. Las discusiones sobre si HTML es mejor que XHTML o viceversa son recurrentes en el ámbito de la creación de contenidos web, aunque no existe una conclusión ampliamente aceptada.

Actualmente, entre HTML 4.01 y XHTML 1.0, la mayoría de diseñadores escogen XHTML. En un futuro cercano, si los diseñadores deben elegir entre HTML 5 y XHTML 1.1 o XHTML 2.0, quizás la elección sea diferente.

1.5. HTML y CSS

Originalmente, las páginas HTML sólo incluían información sobre sus contenidos de texto e imágenes. Con el desarrollo del estándar HTML, las páginas empezaron a incluir también información sobre el aspecto de sus contenidos: tipos de letra, colores y márgenes.

La posterior aparición de tecnologías como JavaScript, provocaron que las páginas HTML también incluyeran el código de las aplicaciones (llamadas *scripts*) que se utilizan para crear páginas web dinámicas.

Incluir en una misma página HTML los contenidos, el diseño y la programación complica en exceso su mantenimiento. Normalmente, los contenidos y el diseño de la página web son responsabilidad de diferentes personas, por lo que es conveniente separarlos.

CSS es el mecanismo que permite separar los contenidos definidos mediante XHTML y el aspecto que deben presentar esos contenidos:

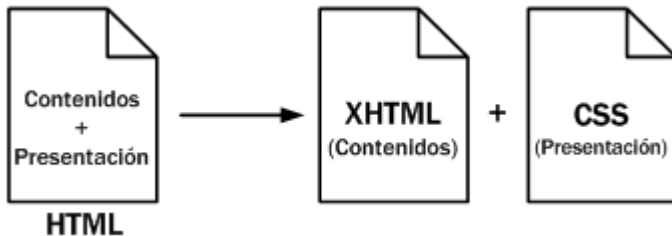


Figura 1.2 Esquema de la separación de los contenidos y su presentación

Una ventaja añadida de la separación de los contenidos y su presentación es que los documentos XHTML creados son más flexibles, ya que se adaptan mejor a las diferentes plataformas: pantallas de ordenador, pantallas de dispositivos móviles, impresoras y dispositivos utilizados por personas discapacitadas.

De esta forma, utilizando exclusivamente XHTML se crean páginas web "*feas*" pero correctas. Aplicando CSS, se pueden crear páginas "*bonitas*" a partir de las páginas XHTML correctas.

Capítulo 2.

Características básicas

2.1. Lenguajes de etiquetas

Uno de los retos iniciales a los que se tuvo que enfrentar la informática fue el de cómo almacenar la información en los archivos digitales. Como los primeros archivos sólo contenían texto sin formato, la solución utilizada era muy sencilla: se codificaban las letras del alfabeto y se transformaban en números.

De esta forma, para almacenar un contenido de texto en un archivo electrónico, se utiliza una tabla de conversión que transforma cada carácter en un número. Una vez almacenada la secuencia de números, el contenido del archivo se puede recuperar realizando el proceso inverso.

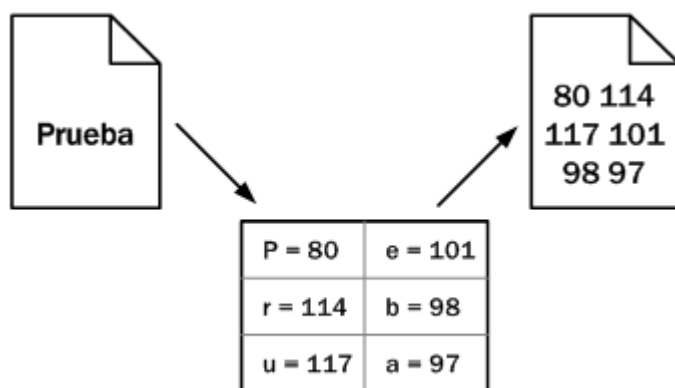


Figura 2.1 Ejemplo sencillo de codificación de caracteres

El proceso de transformación de caracteres en secuencias de números se denomina **codificación de caracteres** y cada una de las tablas que se han definido para realizar la transformación se conocen con el nombre de **páginas de código**. Una de las codificaciones más conocidas (y una de las primeras que se publicaron) es la codificación ASCII. La importancia de las codificaciones en HTML se verá más adelante.

Una vez resuelto el problema de almacenar el texto simple, se presenta el reto de almacenar los contenidos de texto con formato. En otras palabras, ¿cómo se almacena un texto en negrita? ¿y un texto de color rojo? ¿y otro texto azul, en negrita y subrayado?

Utilizar una tabla de conversión similar a las que se utilizan para textos sin formato no es posible, ya que existen infinitos posibles estilos para aplicar al texto. Una solución técnicamente viable consiste en almacenar la información sobre el formato del texto en una zona especial reservada dentro del propio archivo. En esta zona se podría indicar dónde comienza y dónde termina cada formato.

No obstante, la solución que realmente se emplea para guardar la información con formato es mucho más sencilla: el archivo electrónico almacena tanto los contenidos como la información sobre el formato de esos contenidos. Si por ejemplo se quiere dividir el texto en párrafos y se desea dar especial importancia a algunas palabras, se podría indicar de la siguiente manera:

```
<parrafo>
Contenido de texto con <importante>algunas palabras</importante> resaltadas de forma
especial.
</parrafo>
```

El principio de un párrafo se indica mediante la palabra `<parrafo>` y el final de un párrafo se indica mediante la palabra `</parrafo>`. De la misma manera, para asignar más importancia a ciertas palabras del texto, se encierran entre `<importante>` y `</importante>`.

El proceso de indicar las diferentes partes que componen la información se denomina **marcar** (*markup* en inglés). Cada una de las palabras que se emplean para marcar el inicio y el final de una sección se denominan **etiquetas**.

Aunque existen algunas excepciones, en general las etiquetas se indican por pares y se forman de la siguiente manera:

- Etiqueta de apertura: carácter `<`, seguido del nombre de la etiqueta (sin espacios en blanco) y terminado con el carácter `>`
- Etiqueta de cierre: carácter `<`, seguido del carácter `/`, seguido del nombre de la etiqueta (sin espacios en blanco) y terminado con el carácter `>`

Así, la estructura típica de las etiquetas HTML es:

```
<nombre_etiqueta> ... </nombre_etiqueta>
```

HTML es un **lenguaje de etiquetas** (también llamado **lenguaje de marcado**) y las páginas web habituales están formadas por cientos o miles de pares de etiquetas. De hecho, las letras "ML" de la sigla HTML significan "*markup language*", que es como se denominan en inglés a los *lenguajes de marcado*. Además de HTML, existen muchos otros lenguajes de etiquetas como XML, SGML, DocBook y MathML.

La principal ventaja de los lenguajes de etiquetas es que son muy sencillos de leer y escribir por parte de las personas y de los sistemas electrónicos. La principal desventaja es que pueden aumentar mucho el tamaño del documento, por lo que en general se utilizan etiquetas con nombres muy cortos.

2.2. El primer documento HTML

Las páginas HTML se dividen en dos partes: la cabecera y el cuerpo. La cabecera incluye información sobre la propia página, como por ejemplo su título y su idioma. El cuerpo de la página incluye todos sus contenidos, como párrafos de texto e imágenes.

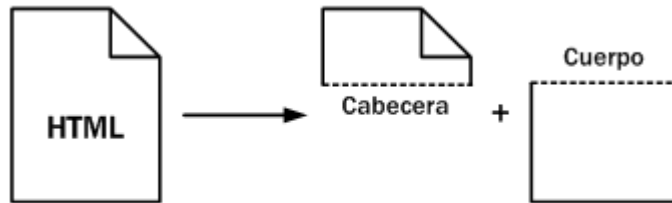


Figura 2.2 Esquema de las partes que forman un documento HTML

El cuerpo (llamado *body* en inglés) contiene todo lo que el usuario ve en su pantalla y la cabecera (llamada *head* en inglés) contiene todo lo que no se ve (con la única excepción del título de la página, que los navegadores muestran como título de sus ventanas).

A continuación se muestra el código HTML de una página web muy sencilla:

```
<html>

<head>
<title>El primer documento HTML</title>
</head>

<body>
<p>El lenguaje HTML es <strong>tan sencillo</strong> que
prácticamente se entiende sin estudiar el significado
de sus etiquetas principales.</p>
</body>

</html>
```

Si quieres probar este primer ejemplo, debes hacer lo siguiente:

1. Abre un editor de archivos de texto y crea un archivo nuevo
2. Copia el código HTML mostrado anteriormente y pégalo tal cual en el archivo que has creado
3. Guarda el archivo con el nombre que quieras, pero con la extensión `.html`

Para que el ejemplo anterior funcione correctamente, es imprescindible que utilices un editor de texto sin formato. Si tu sistema operativo es Windows, puedes utilizar el *Bloc de notas*, *Wordpad*, *EmEditor*, *UltraEdit*, *Notepad++*, etc. pero no puedes utilizar un procesador de textos como *Word* o *Open Office*. Si utilizas sistemas operativos tipo Linux, puedes utilizar editores como *Gedit*, *Kedit*, *Kate* e incluso *Vi*, pero no utilices *KOffice* ni *Open Office*.

Después de crear el archivo con el contenido HTML, ya se puede abrir con cualquier navegador para que se muestre con el siguiente aspecto:

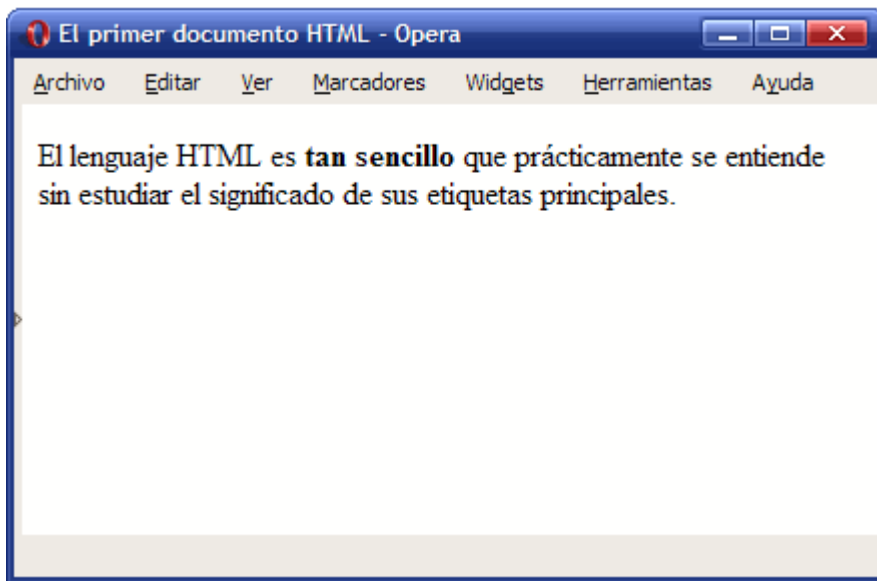


Figura 2.3 Aspecto que muestra el primer documento HTML en cualquier navegador

Si ya estás viendo tu primera página HTML en el navegador, prueba a pulsar sobre el menú [Ver > Código fuente](#) y podrás ver el código HTML de la página que está cargada en el navegador. De hecho, puedes ver el código HTML de cualquier página de Internet mediante la opción [Ver > Código fuente](#). Prueba a ver el código HTML de tu página preferida y verás cuantas etiquetas puede llegar a tener una página compleja.

Volviendo al código HTML del primer ejemplo, es importante conocer las tres etiquetas principales de un documento HTML (`<html>`, `<head>`, `<body>`):

- `<html>`: indica el comienzo y el final de un documento HTML. Ninguna etiqueta o contenido puede colocarse antes o después de la etiqueta `<html>` (con una sola excepción que se verá más adelante). En el interior de la etiqueta `<html>` se definen la cabecera y el cuerpo del documento HTML y todo lo que se coloque fuera de la etiqueta `<html>` se ignora.
- `<head>`: delimita la parte de la cabecera del documento. La cabecera contiene información sobre el propio documento HTML, como por ejemplo su título y el idioma de la página. Los contenidos indicados en la cabecera no son visibles para el usuario, con la excepción de la etiqueta `<title>`, que se utiliza para indicar el título del documento y que los navegadores lo visualizan en la parte superior izquierda de la ventana del navegador (si no te has fijado anteriormente, vuelve a abrir el primer ejemplo en cualquier navegador y observa dónde se muestra el título de la página).
- `<body>`: delimita el cuerpo del documento HTML. El cuerpo encierra todos los contenidos que se muestran al usuario (párrafos de texto, imágenes, tablas). En general, el `<body>` de un documento contiene cientos de etiquetas HTML, mientras que el `<head>` no contiene más que unas pocas.

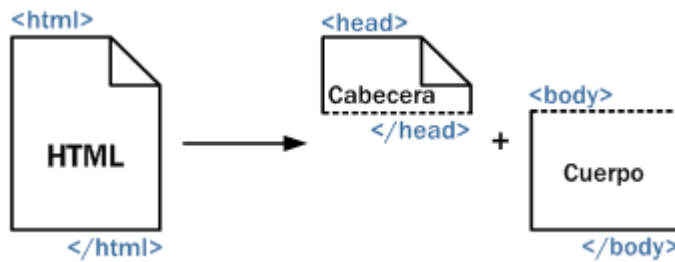


Figura 2.4 Esquema de las etiquetas principales que contiene un documento HTML

Ejercicio 1

Determinar el código HTML correspondiente a la siguiente página:

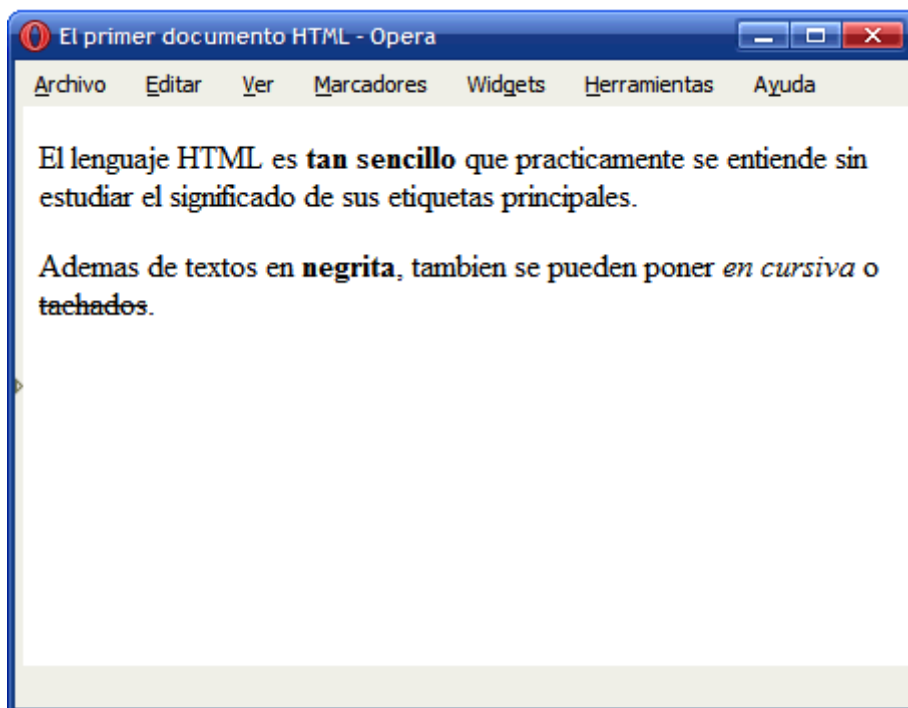


Figura 2.5 Página HTML sencilla que resalta algunas partes del texto

Pistas: `` y ``

2.3. Etiquetas y atributos

HTML define 91 etiquetas que los diseñadores pueden utilizar para *marcar* los diferentes elementos que componen una página:

a, abbr, acronym, address, applet, area, b, base, basefont, bdo, big, blockquote, body, br, button, caption, center, cite, code, col, colgroup, dd, del, dfn, dir, div, dl, dt, em, fieldset, font, form, frame, frameset, h1, h2, h3, h4, h5, h6, head, hr, html, i, iframe, img, input, ins, isindex, kbd, label, legend, li, link, map, menu, meta, noframes, noscript, object, ol, optgroup, option, p, param, pre, q, s, samp, script, select, small, span, strike, strong, style, sub, sup, table, tbody, td, textarea, tfoot, th, thead, title, tr, tt, u, ul, var.

De todas las etiquetas disponibles, las siguientes se consideran **obsoletas** y no se pueden utilizar: `applet`, `basefont`, `center`, `dir`, `font`, `isindex`, `menu`, `s`, `strike`, `u`.

A pesar de que se trata de un número de etiquetas muy grande, no es suficiente para crear páginas complejas. Algunos elementos como las imágenes y los enlaces requieren cierta información adicional para estar completamente definidos.

La etiqueta `<a>` por ejemplo se emplea para incluir un enlace en una página. Utilizando sólo la etiqueta `<a>` no es posible establecer la dirección a la que apunta cada enlace. Como no es viable crear una etiqueta por cada enlace diferente, la solución consiste en personalizar las etiquetas HTML mediante cierta información adicional llamada **atributos**.

De esta forma, se utiliza la misma etiqueta `<a>` para todos los enlaces de la página y se utilizan los atributos para indicar la dirección a la que apunta cada enlace.

```
<html>

<head>
<title>Ejemplo de atributos en las etiquetas</title>
</head>

<body>
<p>
  Los enlaces son muy fáciles de indicar:
  <a>Soy un enlace incompleto, porque no tengo dirección de destino</a>.
  <a href="http://www.google.com">Este otro enlace apunta a la página de Google</a>.
</p>
</body>

</html>
```

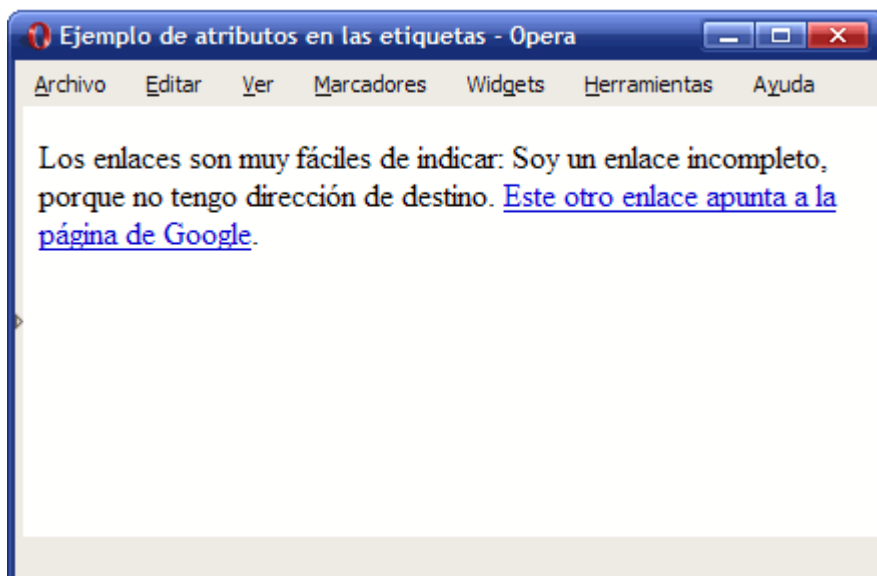


Figura 2.6 Los atributos permiten personalizar las etiquetas HTML

El primer enlace del ejemplo anterior no está completamente definido, ya que no apunta a ninguna dirección. El segundo enlace, utiliza la misma etiqueta `<a>`, pero añade información adicional mediante un atributo llamado `href`. Los atributos se incluyen dentro de la etiqueta de apertura. Por ahora no es importante comprender la etiqueta `<a>` ni el atributo `href`, ya que se explicarán con todo detalle más adelante.

No todos los atributos se pueden utilizar en todas las etiquetas. Por ello, cada etiqueta define su propia lista de atributos disponibles. Además, cada atributo también indica el tipo de valor que se le puede asignar. Si el valor de un atributo no es válido, el navegador ignora ese atributo.

Aunque cada una de las etiquetas HTML define sus propios atributos, algunos de los atributos son comunes a muchas o casi todas las etiquetas. De esta forma, es habitual explicar por separado los atributos comunes de las etiquetas para no tener que volver a hacerlo cada vez que se explica una nueva etiqueta. Los atributos comunes se dividen en cuatro grupos según su funcionalidad:

- **Atributos básicos:** se pueden utilizar prácticamente en todas las etiquetas HTML.
- **Atributos para internacionalización:** los utilizan las páginas que muestran sus contenidos en varios idiomas.
- **Atributos de eventos:** sólo se utilizan en las páginas web dinámicas creadas con JavaScript.
- **Atributos de foco:** relacionados principalmente con la accesibilidad de los sitios web.

2.3.1. Atributos básicos

Los siguiente cuatro atributos básicos se pueden aplicar prácticamente a todas las etiquetas HTML:

Atributo	Descripción
<code>id = "texto"</code>	Establece un identificador único a cada elemento dentro de una página HTML
<code>class = "texto"</code>	Establece la clase CSS que se aplica a los estilos del elemento
<code>style = "texto"</code>	Establece de forma directa los estilos CSS de un elemento
<code>title = "texto"</code>	Establece el título a un elemento (mejora la accesibilidad y los navegadores lo muestran cuando el usuario pasa el ratón por encima del elemento)

La mayoría de páginas web actuales utilizan los atributos `id` y `class` de forma masiva. Sin embargo, estos atributos sólo son realmente útiles cuando se trabaja con CSS y con Javascript.

Respecto al valor de los atributos `id` y `class`, sólo pueden contener guiones medios (-), guiones bajos (_), letras y/o números, pero no pueden empezar por números. Además, los navegadores distinguen mayúsculas de minúsculas y no se recomienda utilizar letras como ñ y acentos, ya que no es seguro que funcionen correctamente en todas las versiones de todos los navegadores.

2.3.2. Atributos para internacionalización

Estos atributos son útiles para aquellas páginas que muestran sus contenidos en varios idiomas y para aquellas que quieren indicar de forma explícita el idioma de sus contenidos:

Atributo	Descripción
<code>lang = "codigo de idioma"</code>	Indica el idioma del elemento mediante un código predefinido
<code>xml:lang = "codigo de idioma"</code>	Indica el idioma del elemento mediante un código predefinido
<code>dir</code>	Indica la dirección del texto (útil para los idiomas que escriben de derecha a izquierda)

En las páginas XHTML, el atributo `xml:lang` tiene más prioridad que `lang` y es obligatorio incluirlo siempre que se incluye el atributo `lang`. Los idiomas se indican mediante un código estandarizado (`es` para español, `en` para inglés, etc.), tal y como se explicará más adelante en la sección [idioma del enlace](#) (página 64) del capítulo 4.

Como la palabra [internacionalización](#) es muy larga, se suele sustituir por la abreviatura `i18n` (el número 18 se refiere al número de letras que existen entre la letra `i` y la letra `n` de la palabra [internacionalización](#)).

2.3.3. Atributos de eventos

Estos atributos sólo se utilizan en las páginas web que incluyen código JavaScript para realizar acciones dinámicas sobre los elementos de la página. Cada vez que el usuario pulsa una tecla, mueve su ratón o pulsa cualquier botón del ratón, se produce un evento dentro del navegador. Utilizando JavaScript y los siguientes atributos, es posible responder de forma adecuada a cada evento.

Atributo	Descripción	Elementos que pueden usarlo
<code>onblur</code>	Deseleccionar el elemento	<code><button></code> , <code><input></code> , <code><label></code> , <code><select></code> , <code><textarea></code> , <code><body></code>
<code>onchange</code>	Deseleccionar un elemento que se ha modificado	<code><input></code> , <code><select></code> , <code><textarea></code>
<code>onclick</code>	Pinchar y soltar el ratón	Todos los elementos
<code>ondblclick</code>	Pinchar dos veces seguidas con el ratón	Todos los elementos
<code>onfocus</code>	Seleccionar un elemento	<code><button></code> , <code><input></code> , <code><label></code> , <code><select></code> , <code><textarea></code> , <code><body></code>
<code>onkeydown</code>	Pulsar una tecla (sin soltar)	Elementos de formulario y <code><body></code>
<code>onkeypress</code>	Pulsar una tecla	Elementos de formulario y <code><body></code>
<code>onkeyup</code>	Soltar una tecla pulsada	Elementos de formulario y <code><body></code>
<code>onload</code>	La página se ha cargado completamente	<code><body></code>
<code>onmousedown</code>	Pulsar (sin soltar) un botón del ratón	Todos los elementos
<code>onmousemove</code>	Mover el ratón	Todos los elementos

Atributo	Descripción	Elementos que pueden usarlo
<code>onmouseout</code>	El ratón "sale" del elemento (pasa por encima de otro elemento)	Todos los elementos
<code>onmouseover</code>	El ratón "entra" en el elemento (pasa por encima del elemento)	Todos los elementos
<code>onmouseup</code>	Soltar el botón que estaba pulsado en el ratón	Todos los elementos
<code>onreset</code>	Inicializar el formulario (borrar todos sus datos)	<code><form></code>
<code>onresize</code>	Se ha modificado el tamaño de la ventana del navegador	<code><body></code>
<code>onselect</code>	Seleccionar un texto	<code><input></code> , <code><textarea></code>
<code>onsubmit</code>	Enviar el formulario	<code><form></code>
<code>onunload</code>	Se abandona la página (por ejemplo al cerrar el navegador)	<code><body></code>

2.3.4. Atributos de foco

Cuando el usuario selecciona un elemento en una aplicación, se dice que "el elemento tiene el foco del programa". Si por ejemplo un usuario pincha con su ratón sobre un cuadro de texto y comienza a escribir, ese cuadro de texto tiene el foco del programa, llamado "focus" en inglés. Si el usuario selecciona después otro elemento, el elemento original pierde el foco y el nuevo elemento es el que tiene el foco del programa.

Los elementos de las páginas web también pueden obtener el foco de la aplicación (en este caso, el foco del navegador) y HTML define algunos atributos específicos para controlar cómo se seleccionan los elementos.

Atributo	Descripción
<code>accesskey = "letra"</code>	Establece una tecla de acceso rápido a un elemento HTML
<code>tabindex = "numero"</code>	Establece la posición del elemento en el orden de tabulación de la página. Su valor debe estar comprendido entre 0 y 32.767
<code>onfocus</code> , <code>onblur</code>	Controlan los eventos JavaScript que se ejecutan cuando el elemento obtiene o pierde el foco

Cuando se pulsa repetidamente la tecla del tabulador sobre una página web, el navegador selecciona de forma alternativa todos los elementos de la página que se pueden seleccionar (principalmente los enlaces y los elementos de formulario). El atributo `tabindex` permite alterar el orden en el que se seleccionan los elementos, por lo que es muy útil cuando se quiere controlar de forma precisa cómo se seleccionan los campos de un formulario complejo.

Por su parte, el atributo `accesskey` permite establecer una tecla para acceder de forma rápida a cualquier elemento. Aunque la tecla de acceso rápido se establece mediante HTML, la combinación de teclas necesarias para activar ese acceso rápido depende del navegador. En el navegador Internet Explorer se pulsa la tecla `ALT` + la tecla definida; en el navegador Firefox se pulsa `Alt` + `Shift` + la tecla definida; en el navegador Opera se pulsa `Shift` + `Esc` + la tecla definida; en el navegador Safari se pulsa `Ctrl` + la tecla definida.

2.4. Elementos HTML

Además de etiquetas y atributos, HTML define el término **elemento** para referirse a las partes que componen los documentos HTML.

Aunque en ocasiones se habla de forma indistinta de "elementos" y "etiquetas", en realidad un elemento HTML es mucho más que una etiqueta, ya que está formado por:

- Una etiqueta de apertura.
- Cero o más atributos.
- Texto encerrado por la etiqueta.
- Una etiqueta de cierre.

El texto encerrado por la etiqueta es opcional, ya que algunas etiquetas de HTML no pueden encerrar ningún texto. El siguiente esquema muestra un elemento HTML, formado por una etiqueta `<p>`, atributos y contenidos de texto:

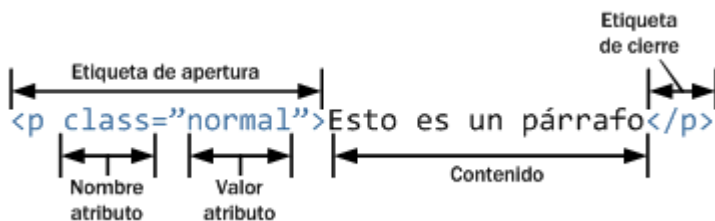


Figura 2.7 Esquema de las partes que componen un elemento HTML

La estructura mostrada en el esquema anterior es un elemento HTML ya que comienza con una etiqueta de apertura (`<p>`), contiene cero o más atributos (`class="normal"`), dispone de un contenido de texto (`Esto es un párrafo`) y finaliza con una etiqueta de cierre (`</p>`).

Por tanto, si una página web tiene dos párrafos de texto, la página contiene dos elementos y cuatro etiquetas (dos etiquetas `<p>` de apertura y dos etiquetas `</p>` de cierre). De todas formas, aunque estrictamente no son lo mismo, es habitual intercambiar las palabras "elemento" y "etiqueta".

Por otra parte, el lenguaje HTML clasifica a todos los elementos en dos grupos: elementos **en línea** (*inline elements* en inglés) y elementos de **bloque** (*block elements* en inglés).

La principal diferencia entre los dos tipos de elementos es la forma en la que ocupan el espacio disponible en la página. Los elementos de bloque siempre empiezan en una nueva línea y ocupan todo el espacio disponible hasta el final de la línea, aunque sus contenidos no lleguen hasta el final de la

línea. Por su parte, los elementos en línea sólo ocupan el espacio necesario para mostrar sus contenidos.

Si se considera el siguiente ejemplo:

```
<html>

<head>
  <title>Ejemplo de elementos en línea y elementos de bloque</title>
</head>

<body>
<p>Los párrafos son elementos de bloque.</p>
<a href="http://www.google.com">Los enlaces son elementos en línea</a>
<p>Dentro de un párrafo, <a href="http://www.google.com">los enlaces</a>
siguen siendo elementos en línea.</p>
</body>

</html>
```

La siguiente imagen muestra cómo visualizan los navegadores el código HTML anterior (mediante CSS se han añadido bordes que muestran el espacio ocupado por cada elemento):

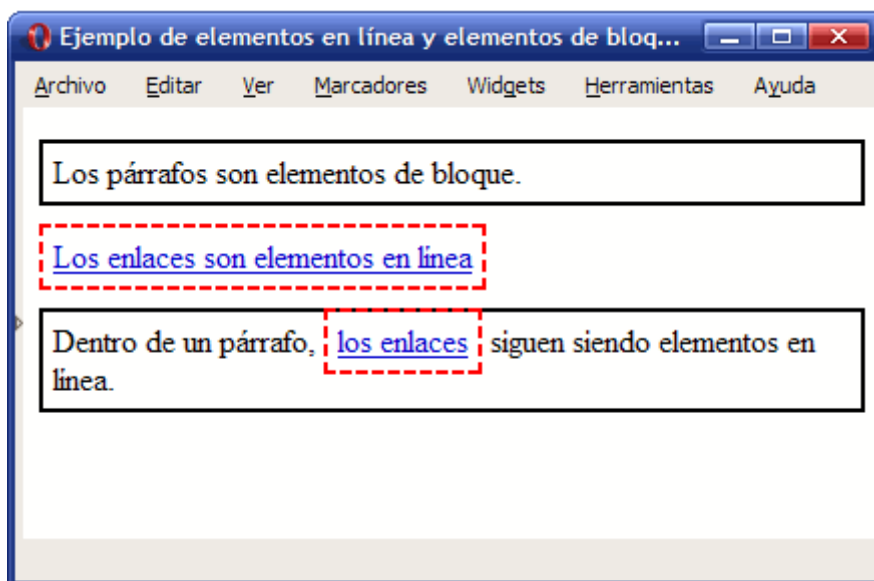


Figura 2.8 Diferencias entre elementos en línea y elementos de bloque

El primer párrafo contiene un texto corto que sólo ocupa la mitad de la anchura de la ventana del navegador. No obstante, el espacio reservado por el navegador para el primer párrafo llega hasta el final de esa línea, por lo que resulta evidente que los elementos `<p>` son elementos de bloque.

Por otra parte, el primer enlace del ejemplo anterior también tiene un texto corto que ocupa solamente la mitad de la anchura de la ventana del navegador. En este caso, el navegador sólo reserva para el enlace el sitio necesario para mostrar sus contenidos. Si se añade otro enlace en esa misma línea, se mostraría a continuación del primer enlace. Por tanto, los elementos `<a>` son elementos en línea.

Por último, el segundo párrafo sigue ocupando todo el espacio disponible hasta el final de cada línea (por ser un elemento de bloque) y el enlace que se encuentra dentro del párrafo sólo ocupa el sitio necesario para mostrar sus contenidos (por ser un elemento en línea).

La mayoría de elementos de bloque pueden contener en su interior elementos en línea y otros elementos de bloque. Los elementos en línea sólo pueden contener texto u otros elementos en línea. En otras palabras, un elemento de bloque no puede aparecer dentro de un elemento en línea. En cambio, un elemento en línea puede aparecer dentro de un elemento de bloque y dentro de otro elemento en línea.

Los elementos en línea definidos por HTML son: [a](#), [abbr](#), [acronym](#), [b](#), [basefont](#), [bdo](#), [big](#), [br](#), [cite](#), [code](#), [dfn](#), [em](#), [font](#), [i](#), [img](#), [input](#), [kbd](#), [label](#), [q](#), [s](#), [samp](#), [select](#), [small](#), [span](#), [strike](#), [strong](#), [sub](#), [sup](#), [textarea](#), [tt](#), [u](#), [var](#).

Los elementos de bloque definidos por HTML son: [address](#), [blockquote](#), [center](#), [dir](#), [div](#), [dl](#), [fieldset](#), [form](#), [h1](#), [h2](#), [h3](#), [h4](#), [h5](#), [h6](#), [hr](#), [isindex](#), [menu](#), [noframes](#), [nos-crypt](#), [ol](#), [p](#), [pre](#), [table](#), [ul](#).

Los siguientes elementos también se considera que son de bloque: [dd](#), [dt](#), [frame-set](#), [li](#), [tbody](#), [td](#), [tfoot](#), [th](#), [thead](#), [tr](#).

Los siguientes elementos pueden ser en línea y de bloque según las circunstancias: [button](#), [del](#), [iframe](#), [ins](#), [map](#), [object](#), [script](#).

2.5. Sintaxis de las etiquetas XHTML

El lenguaje HTML original era muy permisivo en su sintaxis, por lo que era posible escribir sus etiquetas y atributos de muchas formas diferentes. Las etiquetas por ejemplo podían escribirse en mayúsculas, en minúsculas e incluso combinando mayúsculas y minúsculas. El valor de los atributos de las etiquetas se podían indicar con y sin comillas ("). Además, el orden en el que se abrían y cerraban las etiquetas no era importante.

La flexibilidad de HTML puede parecer un aspecto positivo, pero el resultado final son páginas con un código HTML desordenado, difícil de mantener y muy poco profesional. Afortunadamente, XHTML soluciona estos problemas añadiendo ciertas normas en la forma de escribir las etiquetas y atributos.

A continuación se muestran las cinco restricciones básicas que introduce XHTML respecto a HTML en la sintaxis de sus etiquetas:

1) Las etiquetas se tienen que cerrar de acuerdo a como se abren:

Ejemplo correcto en XHTML:

```
| <p>Este es un párrafo con <a>un enlace</a></p>
```

Ejemplo incorrecto en XHTML (pero correcto en HTML):

```
| <p>Este es un párrafo con <a>un enlace</p></a>
```

2) Los nombres de las etiquetas y atributos siempre se escriben en minúsculas:

Ejemplo correcto en XHTML:

```
<p>Este es un párrafo con <a href="http://www.google.com">un enlace</a></p>
```

Ejemplo incorrecto en XHTML (pero correcto en HTML):

```
<P>Este es un párrafo con <A HREF="http://www.google.com">un enlace</A></P>
```

3) El valor de los atributos siempre se encierra con comillas:

Ejemplo correcto en XHTML:

```
<p>Este es un párrafo con <a href="http://www.google.com">un enlace</a></p>
```

Ejemplo incorrecto en XHTML (pero correcto en HTML):

```
<p>Este es un párrafo con <a href=http://www.google.com>un enlace</a></p>
```

4) Los atributos no se pueden comprimir:

Ejemplo correcto en XHTML:

```
<dl compact="compact">...</dl>
```

Ejemplo incorrecto en XHTML (pero correcto en HTML):

```
<dl compact>...</dl>
```

Este tipo de atributos en los que el nombre coincide con su valor no son muy habituales.

5) Todas las etiquetas deben cerrarse siempre:

La mayoría de etiquetas HTML encierran un contenido de texto entre la etiqueta de apertura y la etiqueta de cierre. Sin embargo, algunas etiquetas especiales llamadas "*etiquetas vacías*" no necesitan encerrar ningún texto.

La etiqueta `
` por ejemplo, se utiliza para indicar el comienzo de una nueva línea, tal y como se verá más adelante. Por sus características, la etiqueta `
` nunca encierra ningún contenido de texto.

Como el estándar XHTML obliga a cerrar todas las etiquetas abiertas, siempre que se incluya la etiqueta `
` se debería cerrar de forma seguida: `
</br>`. Para que el código resulte más cómodo de escribir, XHTML permite en estos casos escribir de forma abreviada una etiqueta que se abre y se cierra de forma consecutiva.

En lugar de abrir y cerrar de forma consecutiva la etiqueta (`
</br>`) se puede utilizar la sintaxis `
` para indicar que es una etiqueta vacía que se abre y se cierra en ese mismo punto. En la forma compacta es habitual equivocarse con la posición del carácter `/`.

Ejemplo correcto en XHTML:

```
<br/>
```

Ejemplo incorrecto en XHTML (pero correcto en HTML):

```
<br>
```

Además de estas cinco restricciones básicas, XHTML incluye otros cambios más avanzados respecto a HTML:

1. Antes de acceder al valor de un atributo, se eliminan todos los espacios en blanco que se encuentran antes y después del valor. Además, se eliminan todos los espacios en blanco sobrantes dentro del valor de un atributo. En otras palabras, si en el interior de un atributo se incluyen varios espacios en blanco seguidos, se eliminan todos salvo un único espacio en blanco utilizado para separar las diferentes palabras.
2. Como se explicará más adelante al hablar de la etiqueta `<script>`, el código JavaScript debe encerrarse entre unas etiquetas especiales (`<![CDATA[` y `]]>`) para evitar que el navegador interprete de forma errónea caracteres como `&` y `<`.
3. Las páginas XHTML deben prescindir del atributo `name` para identificar de forma única a los elementos. En su lugar, siempre debe utilizarse el atributo `id`. De hecho, en la versión 1.0 del estándar XHTML, el atributo `name` se ha declarado obsoleto para las etiquetas `a`, `applet`, `form`, `frame`, `iframe`, `img` y `map`.

Capítulo 3.

Texto

La mayor parte del contenido de las páginas HTML habituales está formado por texto, llegando a ser más del 90% del código de la página. Por este motivo, es muy importante conocer los elementos y etiquetas que define HTML para el manejo del texto.

El lenguaje HTML incorpora al tratamiento del texto muchas de las ideas y normas establecidas en otros entornos de publicación de contenidos. De esta forma, HTML define etiquetas para **estructurar** el contenido en secciones y párrafos y define otras etiquetas para **marcar** elementos importantes dentro del texto.

La tarea inicial del editor de contenidos HTML consiste en estructurar el texto original definiendo sus párrafos, titulares y títulos de sección, como se muestra en la siguiente imagen:

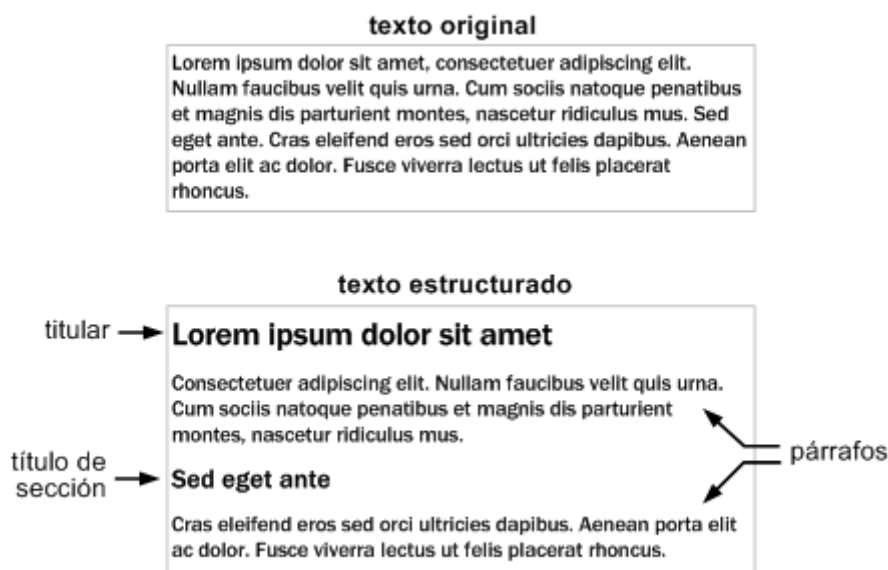


Figura 3.1 Resultado de estructurar un texto sencillo

El proceso de estructurar un texto simple consiste en indicar las diferentes zonas o secciones que componen el texto. De esta forma, los textos estructurados utilizan etiquetas para delimitar cada párrafo y títulos de sección para delimitar cada una de las secciones que forman el texto.

Una vez definida la estructura básica de los contenidos de la página, el siguiente paso consiste en marcar los diferentes elementos dentro del propio texto: definiciones, abreviaturas, textos importantes, textos modificados, citas a otras referencias, etc.

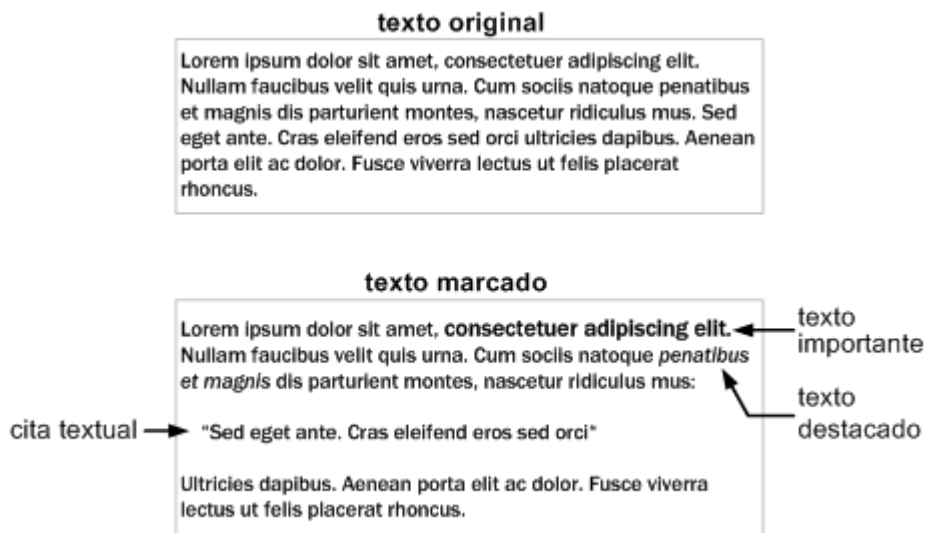


Figura 3.2 Resultado de marcar un texto sencillo

El anterior ejemplo muestra la transformación de un párrafo con un texto simple en un párrafo cuyo texto contiene elementos marcados de forma especial. Así, algunas palabras del texto se muestran en negrita porque se consideran importantes; otras palabras aparecen en cursiva, ya que se han marcado como destacadas e incluso una frase aparece tabulada y entre comillas, indicando que es una cita textual de otro contenido.

En las secciones siguientes se muestran todas las etiquetas que define HTML para estructurar y marcar el texto. Además, se hace una mención especial al tratamiento que hace HTML de los espacios en blanco y las nuevas líneas.

3.1. Estructurar

La forma más sencilla de estructurar un texto consiste en separarlo por párrafos. Además, HTML permite incluir títulos que delimitan cada una de las secciones.

3.1.1. Párrafos

Una de las etiquetas más utilizadas de HTML es la etiqueta `<p>`, que permite definir los párrafos que forman el texto de una página. Para delimitar el texto de un párrafo, se encierra ese texto con la etiqueta `<p>`, como muestra el siguiente ejemplo:

```
<html>
<head>
<title>Ejemplo de texto estructurado con párrafos</title>
```

```

</head>

<body>
<p>Este es el texto que forma el primer párrafo de la página.
Los párrafos pueden ocupar varias líneas y el navegador se encarga
de ajustar su longitud al tamaño de la ventana.</p>

<p>El segundo párrafo de la página también se define encerrando
su texto con la etiqueta p. El navegador también se encarga de
separar automáticamente cada párrafo.</p>
</body>

</html>

```

El ejemplo anterior se visualiza de la siguiente manera en cualquier navegador:

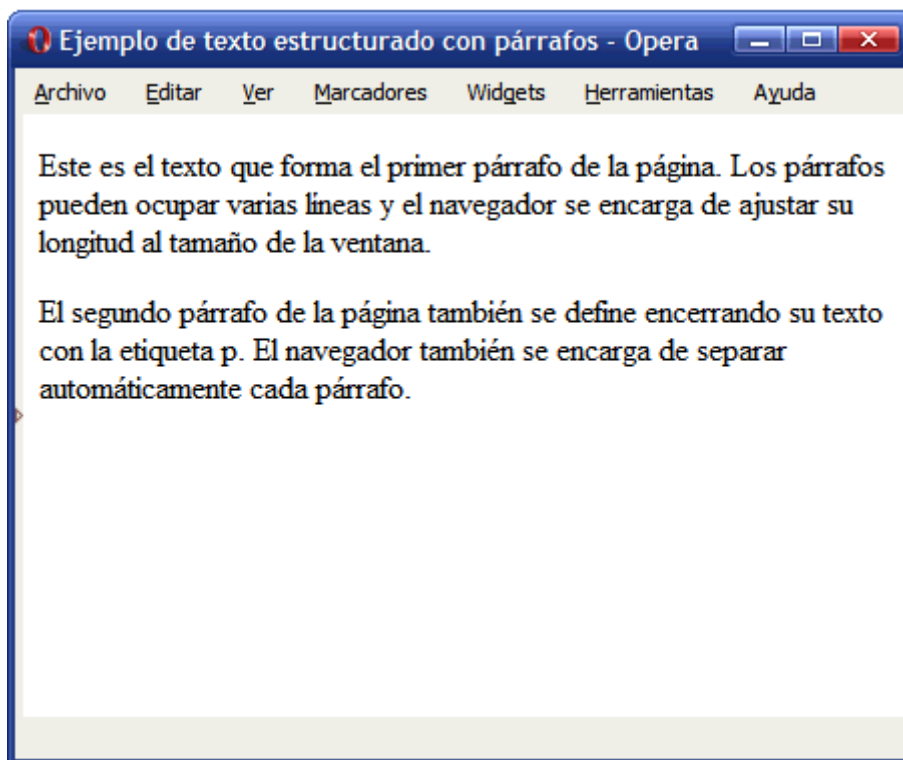


Figura 3.3 Ejemplo de texto HTML estructurado con párrafos

La siguiente tabla recoge la definición formal de la etiqueta `<p>`:

<code><p></code>	Delimita el contenido de un párrafo de texto		
Atributos comunes	básicos, internacionalización y eventos	Tipo de elemento	Bloque
Atributos propios	-		

Los párrafos creados con HTML son elementos de bloque, por lo que siempre ocupan toda la anchura de la ventana del navegador. Además, no tienen atributos específicos, pero sí que se les pueden asignar los atributos comunes de HTML básicos, de internacionalización y de eventos.

3.1.2. Secciones

Las páginas HTML habituales suelen tener una estructura más compleja que la que se puede crear solamente mediante párrafos. De hecho, es habitual que las páginas se dividan en diferentes secciones jerárquicas.

Los títulos de sección se utilizan para delimitar el comienzo de cada sección de la página. HTML permite crear secciones de hasta seis niveles de importancia. De esta forma, aunque una página puede definir cualquier número de secciones, sólo puede incluir seis niveles jerárquicos.

Las etiquetas que definen los títulos de sección son `<h1>`, `<h2>`, `<h3>`, `<h4>`, `<h5>` y `<h6>`. La etiqueta `<h1>` es la de mayor importancia y por tanto se utiliza para definir los titulares de la página. La importancia del resto de etiquetas es descendiente, de forma que la etiqueta `<h6>` es la que se utiliza para delimitar las secciones menos importantes de la página.

A continuación se muestra la definición formal de la etiqueta `<h1>`, siendo idéntica la definición del resto de etiquetas referidas a los títulos de sección:

<code><h1></code>	Define los títulos de las secciones de mayor importancia de la página.		
Atributos comunes	básicos, internacionalización y eventos	Tipo de elemento	Bloque
Atributos propios	-		

Al igual que la etiqueta `<p>`, las etiquetas de título de sección son elementos de bloque y no tienen atributos específicos.

Las etiquetas `<h1>`, ..., `<h6>` definen títulos de sección, no secciones completas. Por este motivo, no es necesario encerrar los contenidos de una sección con su etiqueta correspondiente. Solamente se debe encerrar con las etiquetas `<h1>`, ..., `<h6>` los títulos de cada sección.

El siguiente ejemplo muestra el uso de las etiquetas de título de sección:

```
<html>

<head>
<title>Ejemplo de texto estructurado con secciones</title>
</head>

<body>
<h1>Titular de la página</h1>

<p>Párrafo de introducción...</p>

<h2>La primera sub-sección</h2>
```

```
<p>Párrafo de contenido...</p>

<h2>Otra subsección</h2>

<p>Más párrafos de contenido...</p>
</body>

</html>
```

Los navegadores muestran el ejemplo anterior de la siguiente manera:

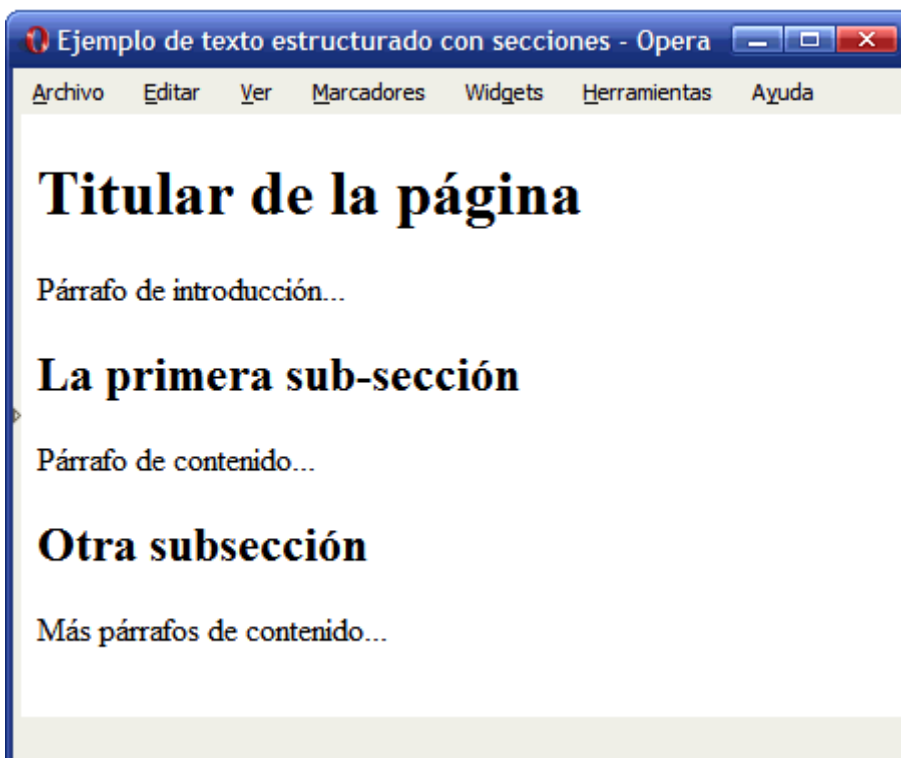


Figura 3.4 Ejemplo de texto HTML estructurado con párrafos y secciones

Los navegadores asignan de forma automáticamente el tamaño del título de cada sección en función de su importancia. Así, los títulos de sección `<h1>` se muestran con el tamaño de letra más grande, ya que son el nivel jerárquico superior, mientras que los títulos de sección `<h6>` se visualizan con un tamaño de letra muy pequeño, adecuado para el nivel jerárquico de menor importancia.

Evidentemente, el aspecto que los navegadores aplican por defecto a los títulos de sección se puede modificar utilizando las hojas de estilos de CSS. La siguiente imagen muestra el tamaño por defecto con el que los navegadores muestran cada titular:

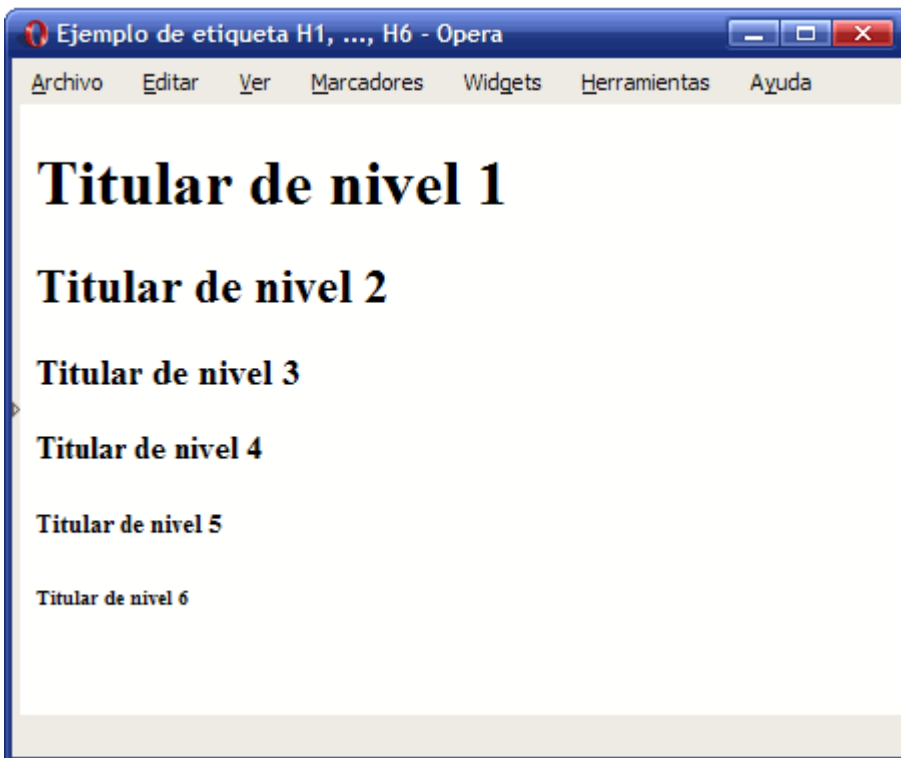


Figura 3.5 Ejemplo de uso de las etiquetas h1, h2, h3, h4, h5 y h6

3.2. Marcado básico de texto

Una vez estructurado el texto en párrafos y secciones, el siguiente paso es el marcado de los elementos que componen el texto. Los textos habituales están formados por elementos como palabras en negrita o cursiva, anotaciones y correcciones, citas a otros documentos externos, etc. HTML proporciona varias etiquetas para *marcar* cada uno de los diferentes tipos de texto.

Entre las etiquetas más utilizadas para marcar texto se encuentran `` y ``. La definición formal de estas dos etiquetas se muestra a continuación:

<code></code>	Realza la importancia del texto que encierra		
Atributos comunes	básicos, internacionalización y eventos	Tipo de elemento	En línea
Atributos propios	-		
<code></code>	Realza con la máxima importancia el texto que encierra		
Atributos comunes	básicos, internacionalización y eventos	Tipo de elemento	En línea
Atributos propios	-		

La etiqueta `` marca un texto indicando que su importancia es mayor que la del resto del texto. La etiqueta `` indica que un determinado texto es de la mayor importancia dentro de la página. Ejemplo:

```
<html>

<head>
<title>Ejemplo de etiqueta em y strong</title>
</head>

<body>
<p>El lenguaje HTML permite marcar algunos segmentos de texto
como <em>muy importantes</em> y otros segmentos como <strong>los
más importantes</strong>.</p>
</body>

</html>
```

Por defecto, los navegadores muestran los elementos `` en cursiva para hacer evidente su importancia y muestran los elementos `` en negrita, para indicar que son los más importantes:

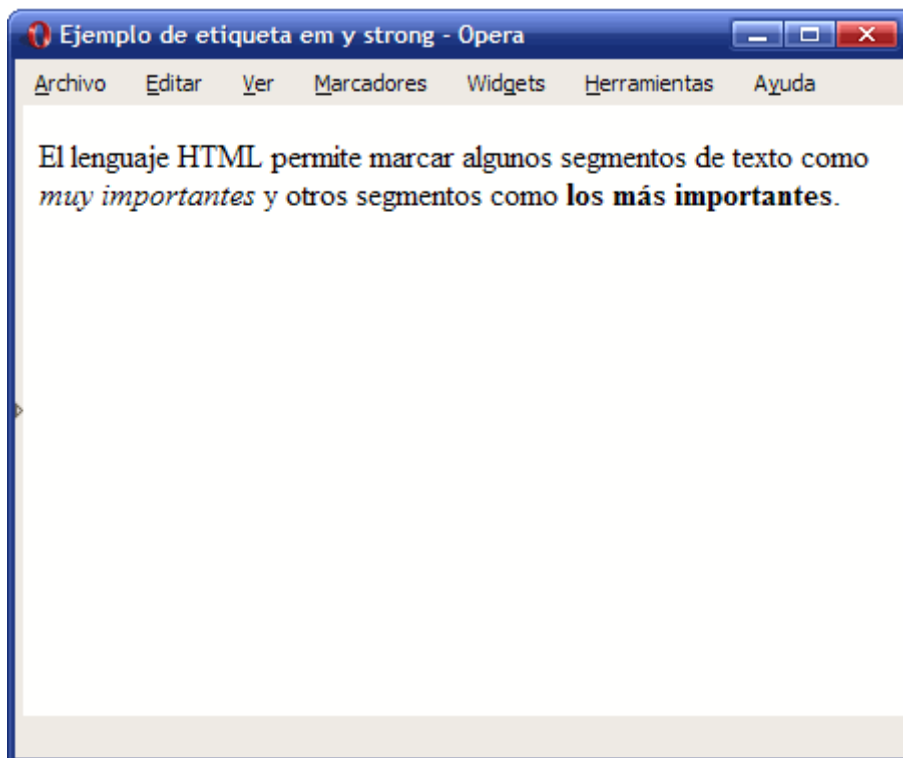


Figura 3.6 Ejemplo de uso de las etiquetas em y strong

Ejercicio 2

Estructurar y marcar el siguiente texto [extraído de la Wikipedia \(http://es.wikipedia.org/wiki/Exploraci3n_especial\)](http://es.wikipedia.org/wiki/Exploraci3n_especial) para que el navegador lo muestre con el aspecto de la siguiente imagen:

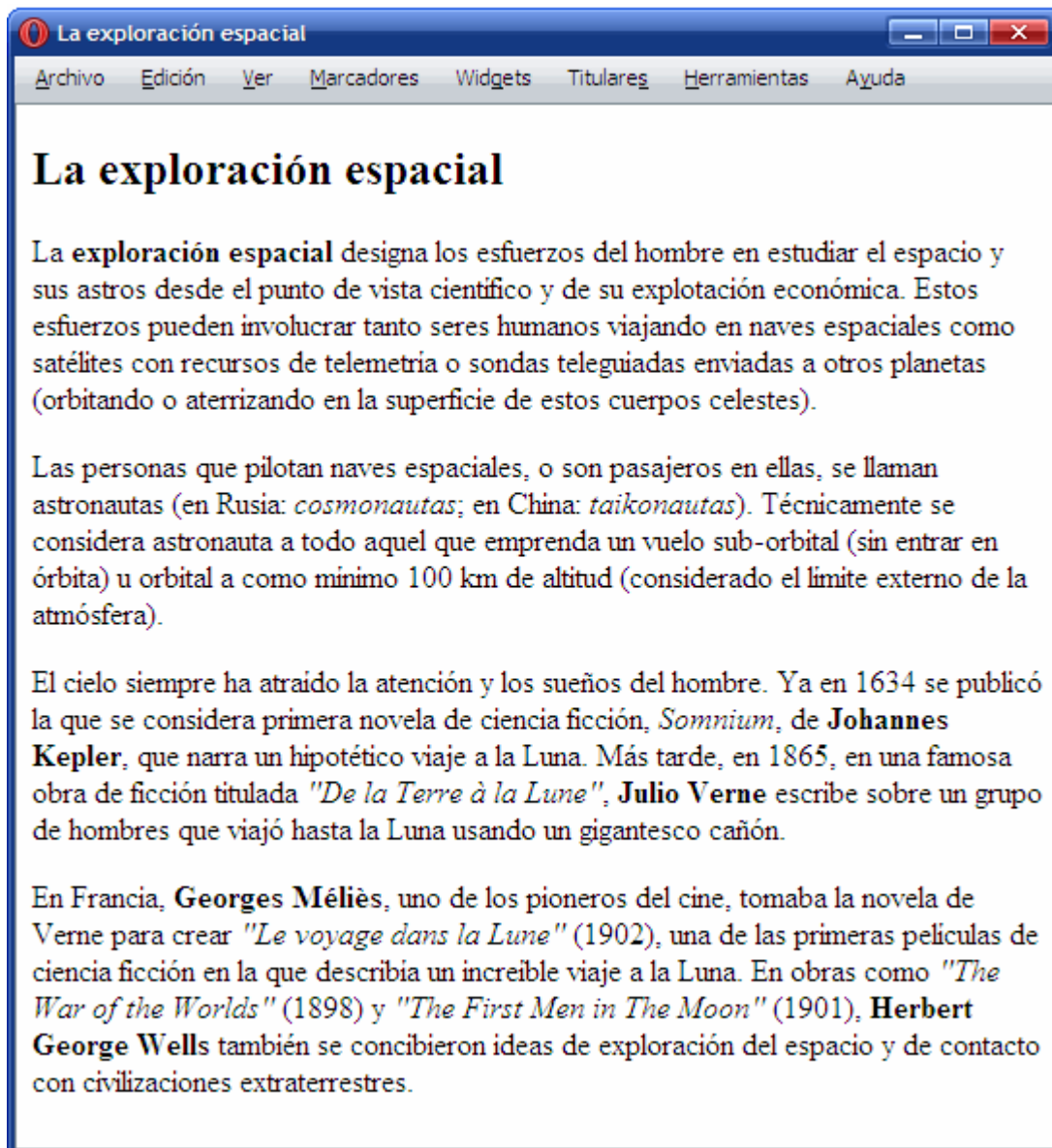


Figura 3.7 Resultado de estructurar y marcar el texto original

HTML también permite marcar de forma adecuada las modificaciones realizadas en el contenido de una página. En otras palabras, HTML permite indicar de forma clara el texto que ha sido eliminado y el texto que ha sido añadido a un determinado texto original. Las etiquetas utilizadas son `<ins>` y ``, cuya definición formal es la siguiente:

<ins>	Se emplea para marcar una modificación en los contenidos originales consistente en la inserción de un nuevo contenido		
Atributos comunes	básicos, internacionalización y eventos	Tipo de elemento	Bloque y en línea
Atributos propios	<ul style="list-style-type: none"> • <code>cite = "url"</code> - Indica la URL de la página en la que se puede obtener más información sobre el motivo por el que se realizó la modificación. • <code>datetime = "fecha"</code> - Especifica la fecha y hora en la que se realizó el cambio 		

	Se emplea para marcar una modificación en los contenidos originales consistente en el borrado de cierto contenido		
Atributos comunes	básicos, internacionalización y eventos	Tipo de elemento	Bloque y en línea
Atributos propios	<ul style="list-style-type: none"> • <code>cite = "url"</code> - Indica la URL de la página en la que se puede obtener más información sobre el motivo por el que se realizó la modificación. • <code>datetime = "fecha"</code> - Especifica la fecha y hora en la que se realizó el cambio 		

Las dos etiquetas cuentan con los mismos atributos específicos, que opcionalmente se pueden añadir para proporcionar más información sobre los cambios realizados. El atributo `cite` se emplea para indicar la dirección de un documento externo en el que se puede encontrar más información relacionada con la inserción o el borrado de texto. El atributo `datetime` puede utilizarse para indicar la fecha y la hora en la que se realizó cada cambio.

Ejemplo:

```
<html>
<head><title>Ejemplo de etiqueta ins y del</title></head>
<body>

<h3>Ejemplo de etiqueta ins y del</h3>

<p>El HTML, acrónimo inglés de Hyper Text Markup Language (lenguaje de <del
datetime="20091025" cite="http://librosweb.es/xhtml/capitulo_3/
marcado_basico_de_texto.html">marcado de hipertexto</del> <ins datetime="20091026"
cite="http://librosweb.es/xhtml/capitulo_3.html"> marcas hipertextuales</ins>) es un
lenguaje de marcación diseñado para estructurar textos y presentarlos en forma de
hipertexto.</p>

</body>
</html>
```

Los navegadores muestran el ejemplo anterior de la siguiente manera:

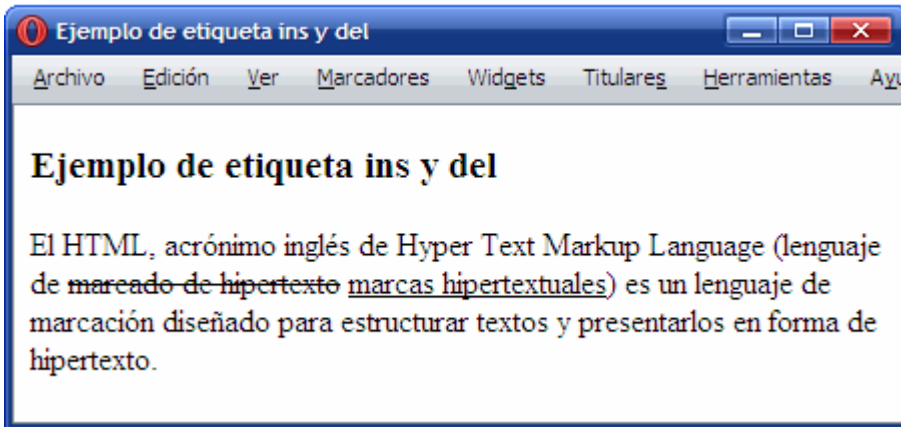


Figura 3.8 Ejemplo de uso de las etiquetas ins y del

Por defecto, el texto eliminado (marcado con la etiqueta ``) se muestra tachado de forma que el usuario pueda identificarlo fácilmente como un texto que formaba parte del texto original y que ya no tiene validez. El texto insertado (marcado con la etiqueta `<ins>`) se muestra subrayado, de forma que el usuario pueda identificarlo como un texto nuevo que no formaba parte del texto original.

Por otra parte, en muchos tipos de páginas (artículos, noticias) es habitual citar literalmente un texto externo. HTML define la etiqueta `<blockquote>` para incluir citas textuales en las páginas web. La definición de la etiqueta HTML con el nombre más largo se muestra a continuación:

<blockquote>	Se emplea para indicar que el texto que encierra es una cita textual de otro texto externo		
Atributos comunes	básicos, internacionalización y eventos	Tipo de elemento	Bloque
Atributos propios	<ul style="list-style-type: none"> <code>cite = "url"</code> - Indica la dirección de la página web original de la que se extrae la cita 		

Al igual que `<ins>` y ``, la etiqueta `<blockquote>` permite indicar mediante el atributo `cite` la dirección de un documento del que se ha extraído la cita. Ejemplo:

```
<html>
<head><title>Ejemplo de etiqueta blockquote</title></head>

<body>
<p>Según el W3C, el valor del
atributo <em>cite</em> en las etiquetas <strong>blockquote</strong> tiene el
siguiente significado:</p>

<blockquote cite="http://www.w3.org/TR/html401/struct/text.html">El valor de este
atributo
es una dirección URL que indica el documento original de la cita.</blockquote>
</body>

</html>
```

El aspecto que muestra el ejemplo anterior en cualquier navegador es el siguiente:

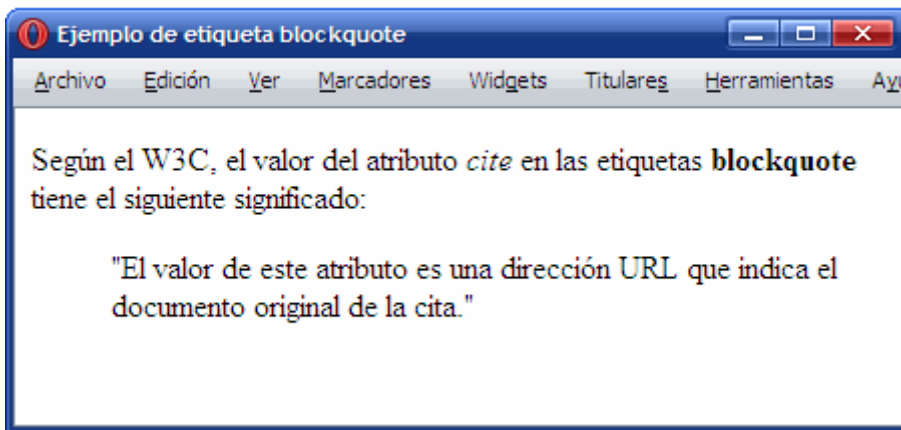


Figura 3.9 Ejemplo de uso de la etiqueta blockquote

Para indicar de forma clara que el texto es una cita externa, los navegadores muestran por defecto el texto del elemento `<blockquote>` con un gran margen en la parte izquierda.

3.3. Marcado avanzado de texto

Las páginas y documentos más avanzados suelen incluir otros elementos importantes que se deben marcar de forma adecuada. Por ello, HTML incluye muchas otras etiquetas que permiten marcar más elementos del texto.

La etiqueta `<abbr>` marca las abreviaturas de un texto y la etiqueta `<acronym>` se emplea para marcar las siglas o acrónimos del texto. Su definición es la siguiente:

<abbr>	Se emplea para marcar las abreviaturas del texto y proporcionar el significado de esas abreviaturas		
Atributos comunes	básicos, internacionalización y eventos	Tipo de elemento	En línea
Atributos propios	<ul style="list-style-type: none"> <code>title = "texto"</code> - Indica el significado completo de la abreviatura 		

<acronym>	Se emplea para marcar las siglas o acrónimos del texto y proporcionar el significado de esas siglas		
Atributos comunes	básicos, internacionalización y eventos	Tipo de elemento	En línea
Atributos propios	<ul style="list-style-type: none"> <code>title = "texto"</code> - Indica el significado completo del acrónimo o sigla 		

En ambos casos, el atributo `title` se puede utilizar para incluir el significado completo de la abreviatura o sigla. Ejemplo:

```
<html>

<head>
<title>Ejemplo de etiqueta acronym</title>
</head>

<body>
<p>El lenguaje <acronym title="HyperText Markup Language">HTML</acronym> es
estandarizado
por el <acronym title="World Wide Web Consortium">W3C</acronym>.</p>
</body>

</html>
```

La mayoría de navegadores muestran por defecto un borde inferior punteado para todos los elementos `<abbr>` y `<acronym>`. Al posicionar el puntero del ratón sobre la palabra subrayada, el navegador muestra un pequeño recuadro (llamado *tooltip* en inglés) con el valor del atributo `title`:

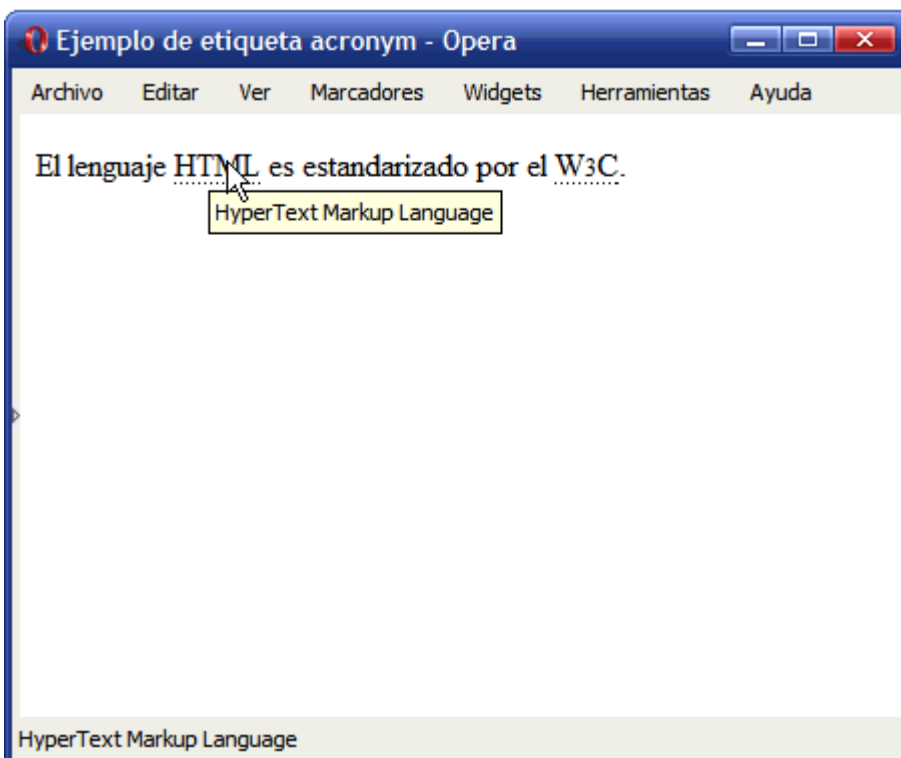


Figura 3.10 Ejemplo de uso de la etiqueta acronym

Por otra parte, en ocasiones resulta útil incluir la definición de una palabra extraña o cuyo uso está restringido a un entorno muy determinado. HTML incluye la etiqueta `<dfn>` para proporcionar al usuario la definición de todas las palabras para las que se considere apropiado. La definición formal de esta etiqueta se muestra a continuación:

<dfn>	Se emplea para marcar las definiciones de ciertos términos y proporcionar el significado de esos términos		
Atributos comunes	básicos, internacionalización y eventos	Tipo de elemento	En línea
Atributos propios	<ul style="list-style-type: none"> • <code>title = "texto"</code> - Indica el significado completo del término 		

El siguiente ejemplo muestra cómo se utiliza la etiqueta `<dfn>` para incluir la definición completa de una palabra cuyo uso no es habitual fuera de los ámbitos médicos y psicológicos:

```
<p>Con estos síntomas, podría tratarse de un caso de <dfn title="Imagen o sensación
subjetiva,
propia de un sentido, determinada por otra sensación que afecta a un sentido
diferente">sinestesia</dfn></p>
```

Por último, HTML incluye una etiqueta que se puede utilizar para marcar un texto como una citación:

<cite>	Se emplea para marcar una cita o una referencia a otras fuentes		
Atributos comunes	básicos, internacionalización y eventos	Tipo de elemento	En línea
Atributos propios	-		

En ocasiones, no está clara la diferencia entre `<cite>` y `<blockquote>`. El elemento `<cite>` marca el autor de la cita (persona, documento, etc.) y `<blockquote>` marca el contenido de la propia cita. En el siguiente ejemplo, `<blockquote>` encierra el contenido de una frase célebre y `<cite>` encierra el nombre de su autor:

```
Como dijo <cite>Mahatma Gandhi</cite>:
<blockquote>Vive como si fueras a morir mañana y aprende como si fueras a vivir para
siempre.</blockquote>
```

Ejercicio 3

Estructurar y marcar el siguiente texto para que el navegador lo muestre con el aspecto de la siguiente imagen:

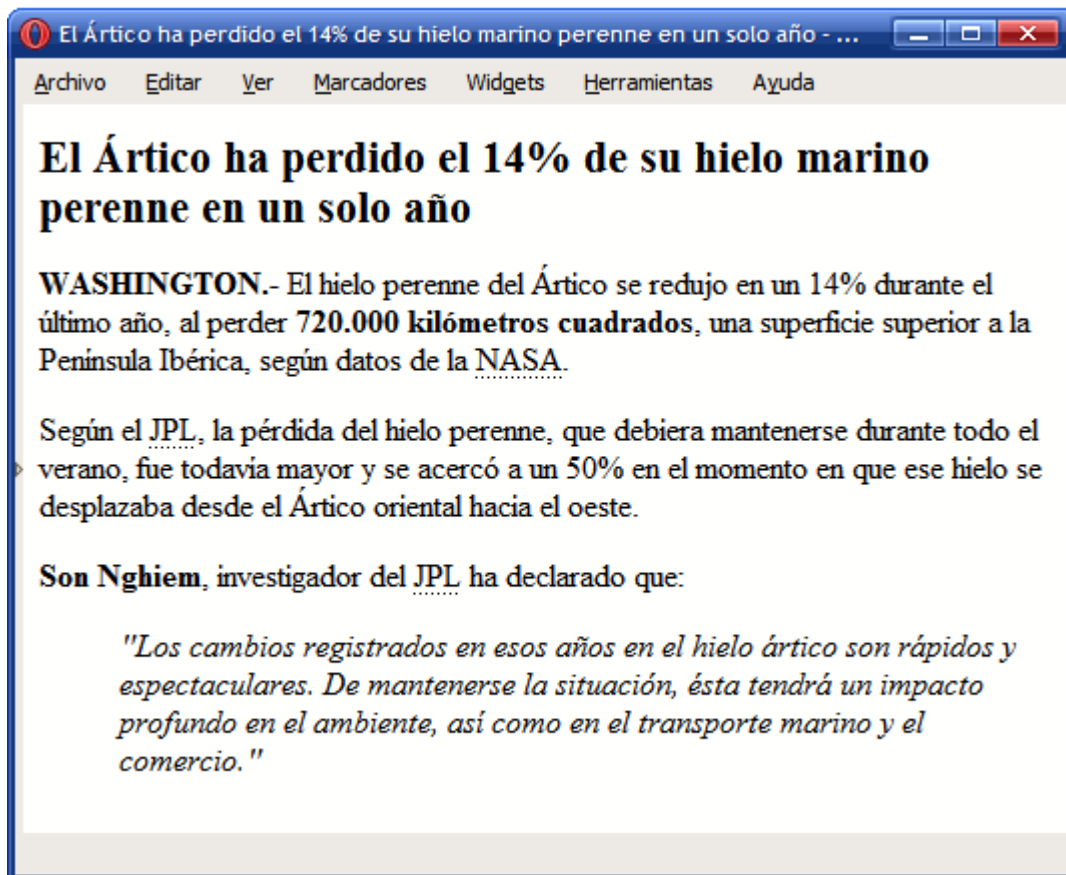


Figura 3.11 Texto HTML correctamente estructurado y marcado

3.4. Marcado genérico de texto

El estándar HTML/XHTML incluye numerosas etiquetas para marcar los contenidos de texto. No obstante, la infinita variedad de posibles contenidos textuales hace que no sean suficientes. Si se considera el siguiente ejemplo:

Importante: si quiere ponerse en contacto con la empresa ACME, puede hacerlo en el teléfono 900 555 555 o a través de la dirección de correo electrónico contacto@acme.org

El texto del ejemplo anterior contiene elementos de texto importantes, siglas, números de teléfono y direcciones de correo electrónico. XHTML define la etiqueta `` para marcar los elementos importantes y `<acronym>` para marcar las siglas:

```
<strong>Importante</strong>: si quiere ponerse en contacto con la empresa
<acronym>ACME</acronym>, puede hacerlo en el teléfono 900 555 555 o a través de la
dirección de correo electrónico contacto@acme.org
```

Desafortunadamente, XHTML no define ninguna etiqueta específica para marcar números de teléfono o direcciones de correo electrónico. De la misma forma, no define etiquetas para otros posibles elementos que se pueden encontrar en los contenidos de texto.

Por este motivo, el estándar HTML/XHTML incluye una etiqueta llamada `` que se emplea para marcar cualquier elemento que no se puede marcar con las otras etiquetas definidas. Siguiendo con

el ejemplo anterior, la etiqueta `` se utiliza para marcar el teléfono y la dirección de correo electrónico:

```
<strong>Importante</strong>: si quiere ponerse en contacto con la empresa
<acronym>ACME</acronym>, puede hacerlo en el teléfono <span>900 555 555</span> o a
través de la dirección de correo electrónico <span>contacto@acme.org</span>
```

La etiqueta `` se visualiza por defecto con el mismo aspecto que el texto normal. Por tanto es habitual utilizar esta etiqueta junto con los atributos `id` y `class` para modificar posteriormente su aspecto con CSS:

```
<strong>Importante</strong>: si quiere ponerse en contacto con la empresa
<acronym>ACME</acronym>, puede hacerlo en el teléfono <span class="telefono">900 555
555</span> o a través de la dirección de correo electrónico <span
class="email">contacto@acme.org</span>
```

La etiqueta `` sólo se puede utilizar para encerrar contenidos y etiquetas en línea. Cuando se quieren estructurar elementos de bloque, se utiliza la etiqueta `<div>`, tal y como se verá en capítulos posteriores.

3.5. Espacios en blanco y nuevas líneas

El aspecto más sorprendente del lenguaje HTML cuando se desarrollan los primeros documentos es el tratamiento especial de los "espacios en blanco" del texto. HTML considera *espacio en blanco* a los espacios en blanco, los tabuladores, los retornos de carro y el carácter de nueva línea (`ENTER` o `Intro`).

El siguiente ejemplo ilustra este comportamiento:

```
<html>
<head><title>Ejemplo de etiqueta p</title></head>
<body>
<p>Este primer párrafo no contiene saltos de línea ni otro tipo de espaciado.</p>

<p>Este segundo párrafo sí que contiene saltos
de
línea
y otro tipo de espaciado.</p>
</body>
</html>
```

El anterior código HTML se visualiza en cualquier navegador de la siguiente manera:

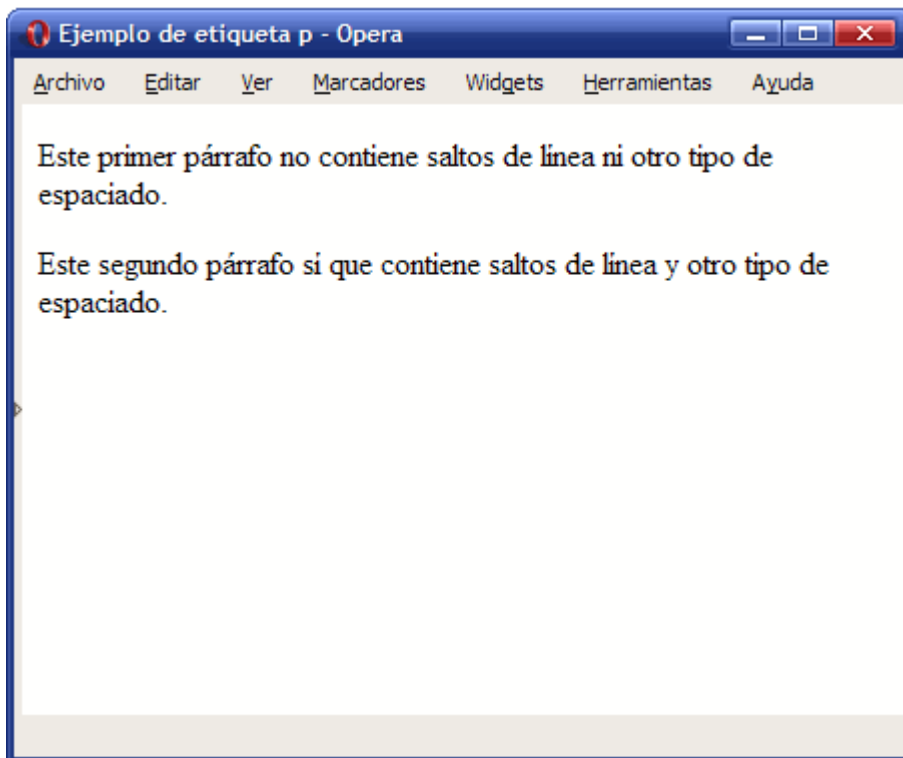


Figura 3.12 Ejemplo de comportamiento de HTML con los espacios en blanco

Los dos párrafos de la imagen anterior se ven idénticos, aunque el segundo párrafo incluye varios espacios en blanco y está escrito en varias líneas diferentes. La razón de este comportamiento es que HTML ignora todos los espacios en blanco *sobrantes*, es decir, todos los espacios en blanco que no son el espacio en blanco que separa las palabras.

No obstante, HTML proporciona varias alternativas para poder incluir tantos espacios en blanco y tantas nuevas líneas como sean necesarias dentro del contenido textual de las páginas.

3.5.1. Nuevas líneas

Para incluir una nueva línea en un punto y forzar a que el texto que sigue se muestre en la línea inferior, se utiliza la etiqueta `
`. En cierta manera, insertar la etiqueta `
` en un determinado punto del texto equivale a presionar la tecla **ENTER** (o **Intro**) en ese mismo punto.

La definición formal de `
` se muestra a continuación:

<code>
</code>	Fuerza al navegador a insertar una nueva línea		
Atributos comunes	básicos	Tipo de elemento	En línea y etiqueta vacía
Atributos propios	-		

La etiqueta `
` es una de las pocas *etiquetas especiales* de HTML. La particularidad de `
` es que es una etiqueta vacía, es decir, no encierra ningún texto. De esta forma, la etiqueta debe abrirse y cerrarse de forma consecutiva: `
</br>`.

En estos casos, HTML permite utilizar un atajo para indicar que una etiqueta se está abriendo y cerrando de forma consecutiva: `
` (también se puede escribir como `
`).

Utilizando la etiqueta `
` se puede rehacer el ejemplo anterior para que respete las líneas que forman el segundo párrafo:

```
<html>

<head>
<title>Ejemplo de etiqueta br</title>
</head>

<body>
<p>Este primer párrafo no contiene saltos de línea ni otro tipo de espaciado.</p>

<p>Este segundo párrafo sí que contiene saltos <br/>
de <br/>
línea <br/>
y otro tipo de espaciado.</p>
</body>

</html>
```

El navegador ahora sí que muestra correctamente las nuevas líneas que se querían insertar:

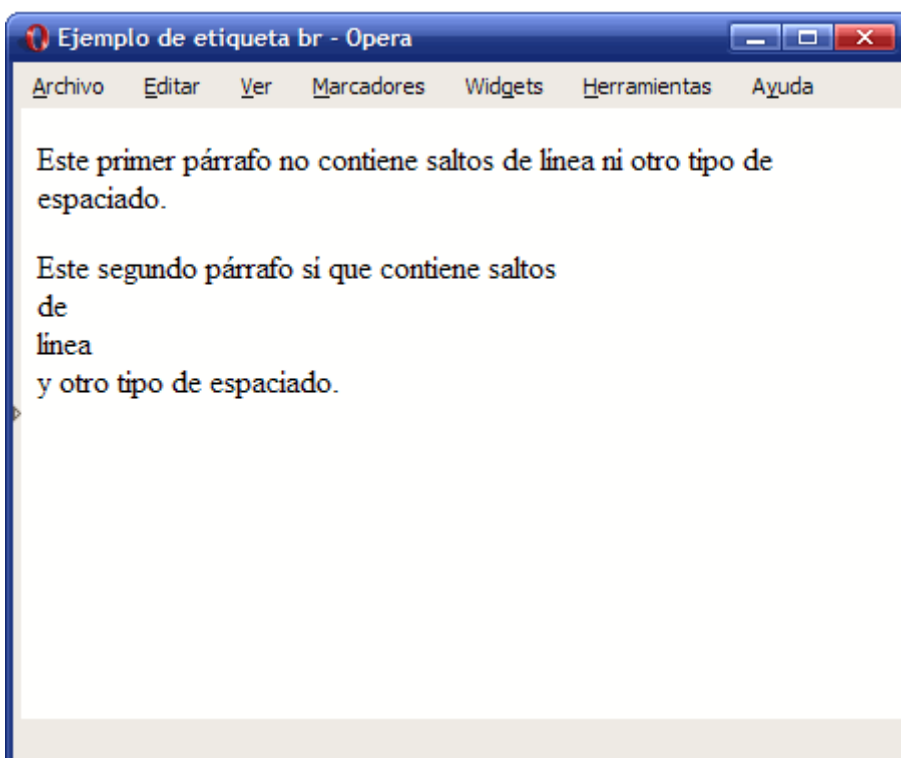


Figura 3.13 Ejemplo de uso de la etiqueta br

3.5.2. Espacios en blanco

La solución al problema de los espacios en blanco no es tan sencilla como el de las nuevas líneas. Para incluir espacios en blanco adicionales, se debe sustituir cada nuevo espacio en blanco por el texto ` `; (es importante incluir el símbolo `&` al principio y el símbolo `;` al final).

Así, el código HTML del ejemplo anterior se debe rehacer para incluir los espacios en blanco adicionales:

```
<html>

<head>
<title>Ejemplo de entidad &nbsp;</title>
</head>

<body>
<p>Este primer párrafo no contiene saltos de línea ni otro tipo de espaciado.</p>

<p>Este segundo párrafo sí que contiene saltos <br/>
de <br/>
línea <br/>
y &nbsp;&nbsp;&nbsp; otro &nbsp;&nbsp;&nbsp; tipo &nbsp;&nbsp;&nbsp; de &nbsp;&nbsp;&nbsp; espaciado.</p>
</body>

</html>
```

Ahora el navegador sí que muestra correctamente los espacios en blanco (y las nuevas líneas) del segundo párrafo:

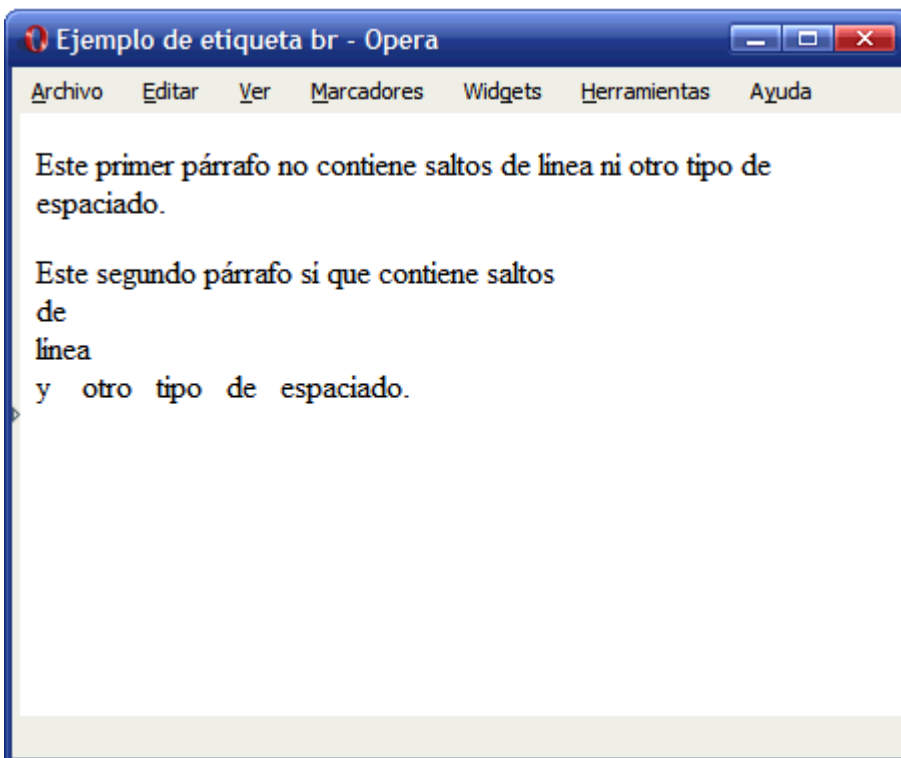


Figura 3.14 Ejemplo de uso de espacios en blanco en HTML

Cada texto ` ` solamente equivale a un espacio en blanco, por lo que se deben escribir tantos ` ` seguidos como espacios en blanco seguidos existan en el texto.

Más adelante se profundiza en el origen de ` `; y se comprenderá por qué es necesario incluir esa sucesión tan extraña de caracteres cada vez que se quiere incluir un espacio en blanco adicional.

Ejercicio 4

Determinar el código HTML que corresponde al siguiente documento:



Nombre	Diametro relativo	Periodo orbital	Número de lunas
Mercurio	0,382	0,24 años	0
Venus	0,949	0,62 años	0
Tierra	1	1 año	1
Marte	0,532	1,88 años	2
Júpiter	11,209	11,86 años	49
Saturno	9,449	29,46 años	52
Urano	4,007	84,01 años	27
Neptuno	3,883	164,80 años	13

Figura 3.15 Texto HTML con espacios en blanco y nuevas líneas

3.5.3. Texto preformateado

En ocasiones, es necesario mostrar los espacios en blanco de un texto que no se puede modificar. Se trata de un caso habitual cuando una página web debe mostrar directamente el texto generado por alguna aplicación.

En estos casos, se puede utilizar la etiqueta `<pre>`, que muestra el texto tal y como se ha escrito, respetando todos los espacios en blanco y todas las nuevas líneas. La definición formal de la etiqueta se muestra a continuación:

<code><pre></code>	Muestra el texto que encierra tal y como está escrito (respetando los espacios en blanco)		
Atributos comunes	básicos, internacionalización y eventos	Tipo de elemento	Bloque
Atributos propios	-		

El siguiente ejemplo muestra el uso de la etiqueta `<pre>`:

```
<html>
<head><title>Ejemplo de etiqueta pre</title></head>

<body>
<pre>
    La etiqueta pre
    respeta los espacios en blanco
    y
    muestra el texto
    tal y como
    está escrito
</pre>

<p>
    La etiqueta pre
    respeta los espacios en blanco
    y
    muestra el texto
    tal y como
    está escrito
</p>

</body>
</html>
```

El ejemplo anterior incluye el mismo texto (con espacios en blanco y varias líneas) dentro de una etiqueta `<pre>` y dentro de una etiqueta `<p>`. Las diferencias visuales en un navegador son muy evidentes:

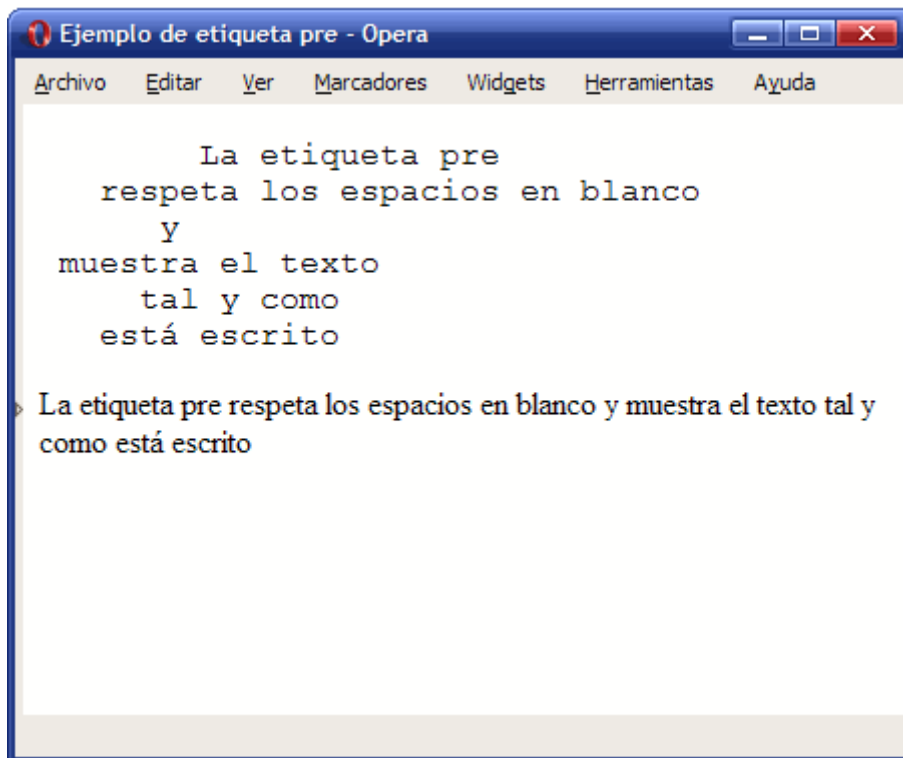


Figura 3.16 Ejemplo de uso de la etiqueta pre

El primer texto se ve en pantalla tal y como se ha escrito, respetando todos los espacios en blanco y todas las nuevas líneas. El segundo texto se ve como un párrafo normal, ya que HTML ha eliminado todos los espacios en blanco sobrantes.

Los elementos `<pre>` son *especiales*, ya que los navegadores les aplican las siguientes reglas:

- Mantienen todos los espacios en blanco (tabuladores, espacios y nuevas líneas)
- Muestra el texto con un tipo de letra especial, denominado "*de ancho fijo*", ya que todas sus letras son de la misma anchura
- No se ajusta la longitud de las líneas (las líneas largas producen un *scroll* horizontal en la ventana del navegador)

Esta última característica diferencia por completo a los párrafos de los elementos `<pre>`. Como se ha visto, los navegadores ajustan la anchura de los párrafos de texto para que ocupen todo el tamaño de la ventana. Sin embargo, los elementos `<pre>` se muestran tal y como son originalmente, por lo que una línea muy larga dentro de un elemento `<pre>` provoca que la anchura de la página sea superior a la anchura de la ventana del navegador.

Si en el ejemplo anterior se añade más texto al final de la segunda línea (para producir una línea larga), el navegador muestra un *scroll* horizontal ya que el texto completo no cabe en el tamaño de la ventana y las líneas de los elementos `<pre>` nunca se ajustan.

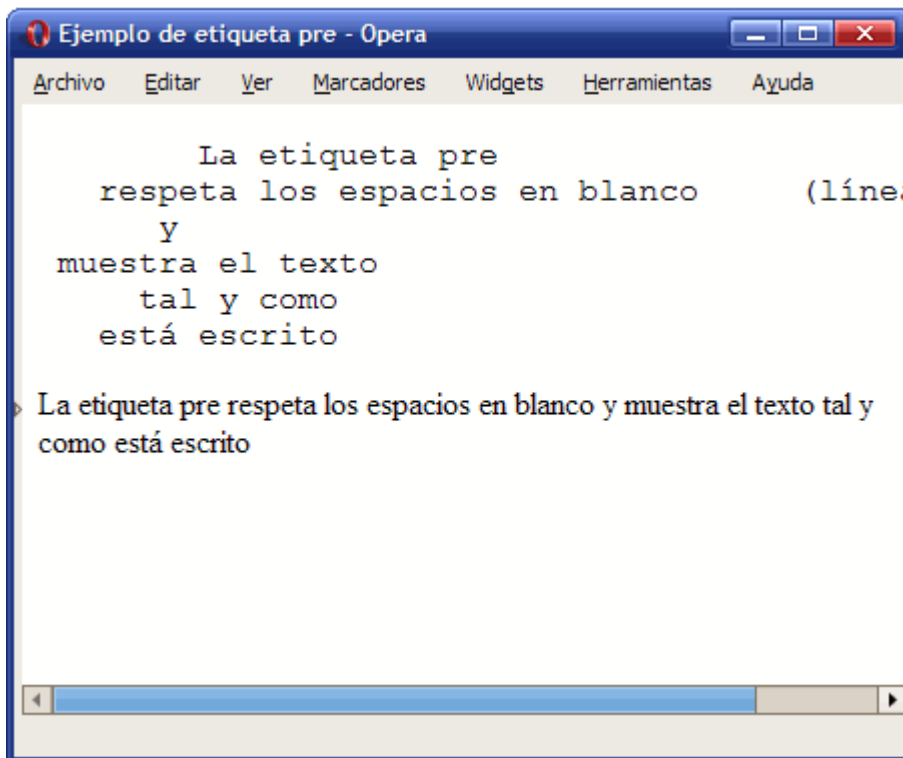


Figura 3.17 Ejemplo de aparición de scroll horizontal con la etiqueta pre

Otra etiqueta relacionada con `<pre>` es la etiqueta `<code>`, que se utiliza para mostrar código fuente de cualquier lenguaje de programación. La definición formal de `<code>` es la siguiente:

<code><code></code>	Delimita el texto considerado un fragmento de código fuente		
Atributos comunes	básicos, internacionalización y eventos	Tipo de elemento	En línea
Atributos propios	-		

En la mayoría de páginas web, no tiene sentido utilizar la etiqueta `<code>`. Sin embargo, en muchas páginas web técnicas que incluyen listados de programas, trozos de código o etiquetas HTML, lo correcto es emplear la etiqueta `<code>`.

Ejemplo:

```

<html>
<head><title>Ejemplo de etiqueta code</title></head>

<body>

<code>
    La etiqueta code
    no respeta los espacios en blanco
</code>

<p>La etiqueta <code>code</code> es similar a la
    
```

etiqueta `<code>pre</code>`, sobre todo en el formato del texto.

```
</body>
</html>
```

El navegador muestra claramente el comportamiento de `<code>` y sus diferencias con `<pre>`:

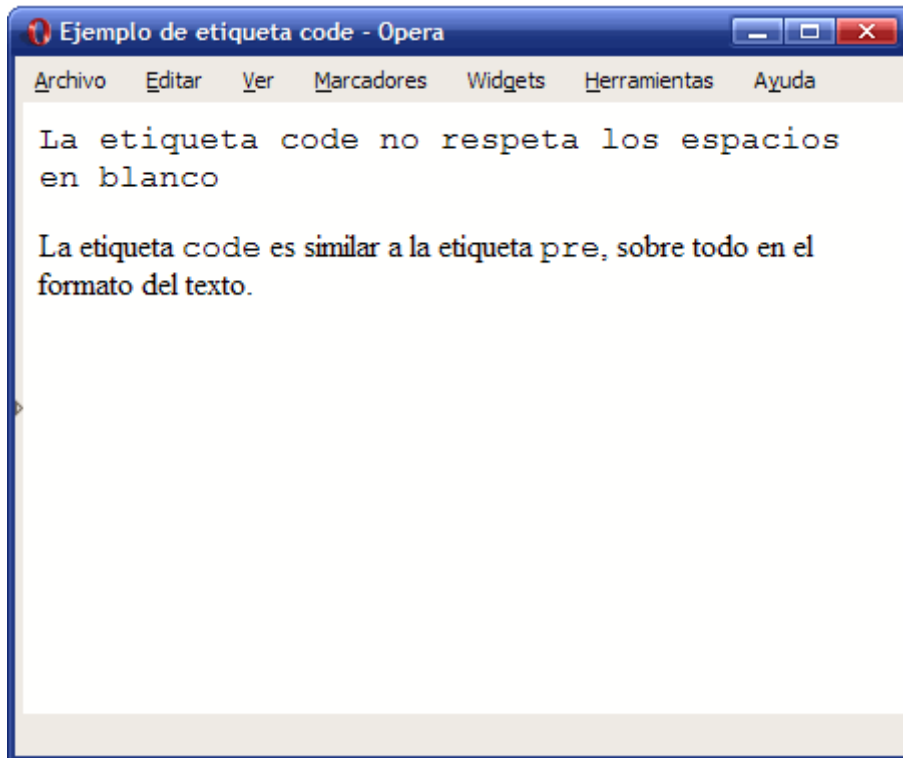


Figura 3.18 Ejemplo de uso de la etiqueta code

Al igual que sucede con los elementos `<pre>`, el texto encerrado por la etiqueta `<code>` se muestra con un tipo de letra especial de ancho fijo. Por el contrario, el elemento `<code>` no respeta los espacios en blanco ni las líneas, por lo que su comportamiento es similar a la etiqueta `<p>`. La última diferencia es que `<code>` es un elemento en línea, mientras que `<pre>` es un elemento de bloque.

3.6. Codificación de caracteres

Una consideración importante directamente relacionada con el texto de las páginas HTML es la codificación de los caracteres y la inserción de caracteres *especiales*. Algunos de los caracteres que se utilizan habitualmente en los textos no se pueden incluir directamente en las páginas web:

- Los caracteres que utiliza HTML para definir sus etiquetas (`<`, `>` y `"`) no se pueden utilizar libremente.
- Los caracteres propios de los idiomas que no son el inglés (`ñ`, `á`, `ç`, `¿`, etc.) pueden ser problemáticos dependiendo de la codificación de caracteres utilizada.

La solución a la primera limitación consiste en sustituir los caracteres reservados de HTML por unas expresiones llamadas *entidades HTML* y que representan a cada carácter:

Entidad	Carácter	Descripción	Traducción
<	<	less than	signo de menor que
>	>	more than	signo de mayor que
&	&	ampersand	ampersand
"	"	quotation mark	comillas
 	(espacio en blanco)	non-breaking space	espacio en blanco
'	'	apostrophe	apóstrofo

De esta forma, si se considera el siguiente texto:

Los caracteres <, >, " y & pueden dar problemas con los textos en HTML

Para mostrar correctamente el texto anterior en una página HTML, se debe sustituir cada carácter especial por su entidad HTML:

```
<p>Los caracteres &lt;;, &gt;;, &quot; y &amp; pueden dar problemas con los textos en HTML</p>
```

Ejercicio 5

Determinar el código HTML que corresponde al siguiente documento:

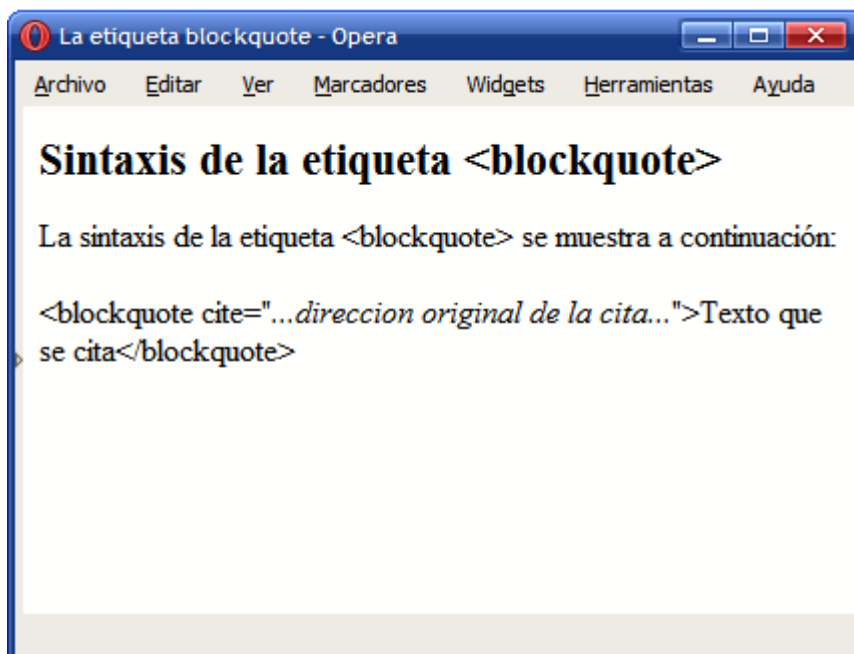


Figura 3.19 Texto HTML que incluye caracteres especiales

Por otra parte, los caracteres propios de los idiomas diferentes al inglés también pueden ser problemáticos. El motivo es que desde que se crea una página web hasta que llega al navegador del usuario, intervienen numerosos procesos:

- El diseñador crea la página web con su editor HTML (por ejemplo Dreamweaver).
- Si se trata de una aplicación dinámica, el programador recorta la página HTML del diseñador y la mezcla con el resto del código de la aplicación (por ejemplo PHP).
- El servidor web almacena las páginas HTML estáticas o el código de la aplicación web y sirve las páginas solicitadas por los usuarios.
- El usuario solicita y visualiza las páginas web a través de su navegador.

Si en todos los procesos anteriores se utiliza la misma codificación de caracteres, los caracteres propios de los idiomas se pueden escribir directamente:

```
<p>Este párrafo contiene caracteres acentuados y se almacena en formato UTF-8</p>
```

La palabra **párrafo** del ejemplo anterior incluye la letra **á**. Si el editor HTML del diseñador utiliza la codificación UTF-8, el entorno de desarrollo del programador también utiliza UTF-8, el servidor web sirve las páginas con esa codificación y el navegador del usuario es capaz de visualizar las páginas con formato UTF-8, el texto anterior se verá correctamente en el navegador del usuario.

Sin embargo, muchas veces no es posible que todos los procesos involucrados utilicen la misma codificación de caracteres. Por limitaciones técnicas o por decisiones de los diseñadores y programadores, los textos pueden pasar de codificación UTF-8 a codificación ISO-8859 en cualquier momento. Si se produce este cambio sin realizar una conversión correcta, el navegador del usuario mostrará caracteres extraños en todos los acentos y en todas las letras como la **ñ**.

La solución más sencilla para asegurar que todos estos caracteres potencialmente problemáticos se van a visualizar correctamente en el navegador del usuario consiste en sustituir cada carácter problemático por su entidad HTML:

Entidad	Carácter	Descripción oficial
<code>&ntilde;</code>	ñ	latin letter n with tilde
<code>&Ntilde;</code>	Ñ	latin capital n letter with tilde
<code>&aacute;</code>	á	a acute
<code>&eacute;</code>	é	e acute
<code>&iacute;</code>	í	i acute
<code>&oacute;</code>	ó	o acute
<code>&uacute;</code>	ú	u acute
<code>&Aacute;</code>	Á	A acute
<code>&Eacute;</code>	É	E acute
<code>&Iacute;</code>	Í	I acute
<code>&Oacute;</code>	Ó	O acute

Entidad	Carácter	Descripción oficial
<code>&Uacute;</code>	Ú	U acute
<code>&euro;</code>	€	euro

Así, el párrafo de texto del ejemplo anterior, se podría escribir de la siguiente manera:

```
<p>Este párrafo contiene caracteres acentuados y se almacena en formato UTF-8</p>
```

Si se utilizan las entidades HTML en vez de los caracteres problemáticos, es indiferente pasar de una codificación de caracteres a otra diferente. En la Wikipedia se puede consultar la lista completa de las 252 entidades HTML definidas (http://en.wikipedia.org/wiki/List_of_XML_and_HTML_character_entity_references).

Capítulo 4.

Enlaces

El lenguaje de marcado HTML se definió teniendo en cuenta algunas de las características que existían en ese momento para la publicación digital de contenidos. Entre los conceptos utilizados en su creación, se encuentra el mecanismo de "*hipertexto*".

De hecho, las letras "HT" de la sigla HTML significan "*hipertexto*" (*hypertext* en inglés), por lo que el significado completo de HTML podría traducirse como "lenguaje de marcado para hipertexto".

La incorporación del *hipertexto* fue una de las claves del éxito del lenguaje HTML, ya que permitió crear documentos interactivos que proporcionan información adicional cuando se solicita. El elemento principal del hipertexto es el "*hiperenlace*", también llamado "enlace web" o simplemente "enlace".

Los enlaces se utilizan para establecer relaciones entre dos recursos. Aunque la mayoría de enlaces relacionan páginas web, también es posible enlazar otros recursos como imágenes, documentos y archivos.

Una característica que no se suele tener en cuenta en los enlaces es que están formados por dos extremos y un sentido. En otras palabras, el enlace comienza en un recurso y apunta hacia otro recurso. Cada uno de los dos extremos se llaman "*anchors*" en inglés, que se puede traducir literalmente como "anclas".

4.1. URL

Antes de empezar a crear enlaces, es necesario comprender y dominar el concepto de URL. El acrónimo URL (del inglés *Uniform Resource Locator*) hace referencia al identificador único de cada recurso disponible en Internet. Las URL son esenciales para crear los enlaces, pero también se utilizan en otros elementos HTML como las imágenes y los formularios.

La URL de un recurso tiene dos objetivos principales:

- Identificar de forma única a ese recurso
- Permitir localizar de forma eficiente ese recurso

En primer lugar, las URL permiten que cada página HTML publicada en Internet tenga un nombre único que permita diferenciarla de las demás. De esta forma es posible crear enlaces que apunten de forma inequívoca a una determinada página.

Si se accede a la página principal de Google, la dirección que muestra el navegador es:

```
| http://www.google.com
```

La cadena de texto <http://www.google.com> es la URL completa de la página principal de Google. La URL de las páginas es imprescindible para crear los enlaces, ya que permite distinguir una página de otra.

El segundo objetivo de las URL es el de permitir la localización eficiente de cada recurso de Internet. Para ello es necesario comprender las diferentes partes que forman las URL. Una URL sencilla siempre está formada por las mismas tres partes. Si por ejemplo se considera la siguiente URL:

```
| http://www.librosweb.es/xhtml/capitulo4.html
```

Las partes que componen la URL anterior son:

- Protocolo (<http://>): el mecanismo que debe utilizar el navegador para acceder a ese recurso. Todas las páginas web utilizan <http://>. Las páginas web *seguras* (por ejemplo las de los bancos y las de los servicios de email) utilizan <https://> (se añade una letra *s*).
- Servidor (www.librosweb.es): simplificando mucho su explicación, se trata del ordenador en el que se encuentra guardada la página que se quiere acceder. Los navegadores son capaces de obtener la dirección de cada servidor a partir de su nombre.
- Ruta ([/xhtml/capitulo4.html](http://www.librosweb.es/xhtml/capitulo4.html)): *camino* que se debe seguir, una vez que se ha llegado al servidor, para localizar el recurso específico que se quiere acceder.

Por tanto, las URL no sólo identifican de forma única a cada recurso de Internet, sino que también proporcionan a los navegadores la información necesaria para poder llegar hasta ese recurso.

La mayoría de URL son tan sencillas como la URL mostrada anteriormente. No obstante, existen URL complejas formadas por más partes.

```
| http://www.alistapart.com/comments/webstandards2008?page=5#42
```

Las cinco partes que forman la URL anterior son:

- Protocolo (<http://>)
- Servidor (www.alistapart.com)
- Ruta ([/comments/webstandards2008](http://www.alistapart.com/comments/webstandards2008))
- Consulta ([?page=5](http://www.alistapart.com/comments/webstandards2008?page=5)): información adicional necesaria para que el servidor localice correctamente el recurso que se quiere acceder. Siempre comienza con el carácter *?* y contiene una sucesión de palabras separadas por *=* y *&*
- Sección ([#42](http://www.alistapart.com/comments/webstandards2008?page=5#42)): permite que el navegador se posicione automáticamente en una sección de la página web. Siempre comienza con el carácter *#*

Como las URL utilizan los caracteres `:`, `=`, `&` y `/` para separar sus partes, estos caracteres están reservados y no se pueden utilizar libremente. Además, algunos caracteres no están reservados pero pueden ser problemáticos si se utilizan en la propia URL.

Si es necesario incluir estos caracteres reservados y especiales en una URL, se sustituyen por combinaciones de caracteres seguros. Esta sustitución se denomina *codificación* de caracteres y el servidor realiza el proceso inverso (*decodificación*) cuando le llega una URL con los caracteres codificados.

A continuación se muestra la tabla para codificar los caracteres más comunes:

Carácter original	Carácter codificado	Carácter original	Carácter codificado
/	%2F	?	%3F
:	%3A	@	%40
=	%3D	&	%26
"	%22	\	%5C
'	%60	~	%7E
(espacio en blanco)	%20	#	%23

Por otra parte, aunque desde hace tiempo ya es posible incluir en las URL caracteres de otros idiomas que no sean el inglés, aún no es completamente seguro utilizar estos caracteres en las URL. Si se utilizan letras como ñ, á, é o ç, es posible que algunos navegadores no las interpreten de forma correcta.

La solución consiste en codificar todos los caracteres que no existen en inglés. La siguiente tabla muestra la codificación de los caracteres más utilizados:

Carácter original	Carácter codificado	Carácter original	Carácter codificado
ñ	%F1	Ñ	%D1
á	%E1	Á	%C1
é	%E9	É	%C9
í	%ED	Í	%CD
ó	%F3	Ó	%D3
ú	%FA	Ú	%DA
ç	%E7	Ç	%C7

Teniendo en cuenta las dos tablas anteriores de codificación de caracteres, es fácil crear las URL correctas sin caracteres problemáticos:

```
<!-- URL problemática -->
http://www.ejemplo.com/estaciones/otoño.html
```

```

<!-- URL correcta -->
http://www.ejemplo.com/estaciones/oto%F1o.html

<!-- URL problemática -->
http://www.ejemplo.com/ruta/nombre página.html

<!-- URL correcta -->
http://www.ejemplo.com/ruta/nombre%20p%E1gina.html
    
```

4.2. Enlaces relativos y absolutos

Las páginas web habituales suelen contener decenas de enlaces de diferentes tipos. La siguiente imagen muestra algunos de los tipos de enlaces de la página principal del sitio web www.thinkvitamin.com:



Figura 4.1 Ejemplo de diferentes tipos de enlaces en la página [456BereaStreet.com](http://www.456BereaStreet.com)

En esa página, cuando se pincha sobre algunos enlaces, el navegador abandona el sitio web para acceder a páginas que se encuentran en otros sitios. Estos enlaces se conocen como "enlaces externos". Sin embargo, la mayoría de enlaces de un sitio web apuntan a páginas del propio sitio web, por lo que se denominan "enlaces internos".

Además de internos/externos, la otra característica que diferencia a los enlaces (y por tanto, también a las URL) es si el enlace es absoluto o relativo. Las **URL absolutas** incluyen todas las partes de la URL (protocolo, servidor y ruta) por lo que no se necesita más información para obtener el recurso enlazado.

Las **URL relativas** prescinden de algunas partes de las URL para hacerlas más breves. Como se trata de URL incompletas, es necesario disponer de información adicional para obtener el recurso enlazado.

do. En concreto, para que una URL relativa sea útil es imprescindible conocer la URL del origen del enlace.

Las URL relativas se construyen a partir de las URL absolutas y prescinden de la parte del protocolo, del nombre del servidor e incluso de parte o toda la ruta del recurso enlazado. Aunque las URL relativas pueden ser difíciles de entender para los que comienzan con HTML, son tan útiles que todos los sitios web las utilizan.

Imagina que dispones de una página publicada en `http://www.ejemplo.com/ruta1/ruta2/pagina1.html` y quieres incluir en ella un enlace a otra página que se encuentra en `http://www.ejemplo.com/ruta1/ruta2/pagina2.html`. Como las URL identifican de forma única a los recursos de Internet y proporcionan la información necesaria para llegar hasta ellos, el enlace debería utilizar la URL completa de la segunda página.

Las URL completas también se llaman URL absolutas, ya que el navegador no necesita disponer de información adicional para localizar el recurso enlazado. Si se utilizan siempre las URL absolutas, los enlaces están completamente definidos.

Sin embargo, escribir siempre las URL completas es bastante aburrido, cuesta mucho tiempo y hace imposible cambiar la ubicación de los contenidos de un sitio web. Por ese motivo, casi todos los sitios web de Internet utilizan URL relativas siempre que es posible.

Una URL relativa es una versión abreviada de una URL absoluta. Su objetivo es eliminar todas las partes de la URL absoluta que se pueden *adivinar* a partir de la información de contexto de la página web. En otras palabras, las URL relativas aprovechan la *inteligencia* de los navegadores para crear URL incompletas que los navegadores pueden completar deduciendo la información que falta.

Considerando de nuevo el ejemplo anterior, la URL a la que se quiere enlazar utiliza el mismo protocolo y se encuentra en el mismo servidor que la página origen, por lo que la URL relativa puede prescindir de esas partes:

```
URL absoluta: http://www.ejemplo.com/ruta1/ruta2/pagina2.html
URL relativa: /ruta1/ruta2/pagina2.html
```

En el ejemplo anterior, las dos URL son equivalentes porque cuando no se indica el protocolo y el servidor de una URL, los navegadores suponen que son los mismos que los de la página origen. Por lo tanto, cuando el navegador encuentra la URL `/ruta1/ruta2/pagina2.html`, realiza el siguiente proceso:

1. La URL no es absoluta, por lo que se debe determinar la URL absoluta a partir de la URL relativa para poder cargar el recurso enlazado.
2. A la URL relativa le falta el protocolo y el servidor, por lo que se supone que son los mismos que los de la página origen (`http://` y `www.ejemplo.com`).
3. Se añaden las partes que faltan a la URL relativa para obtener la URL absoluta: `http:// + www.ejemplo.com + /ruta1/ruta2/pagina2.html = http://www.ejemplo.com/ruta1/ruta2/pagina2.html`.

Aunque el ejemplo mostrado es el caso más sencillo de URL relativa, existen otros casos más avanzados en los que se prescinde de parte o toda la ruta del recurso que se enlaza. A continuación se muestran los cuatro tipos diferentes de URL relativas:

1) El origen y el destino del enlace se encuentran en el mismo directorio

Si desde una página web se quiere enlazar un recurso que se encuentra en el mismo directorio del servidor, la URL relativa puede prescindir de todas las partes de la URL absoluta salvo el nombre del recurso enlazado.

Elemento	Valor
Página origen	http://www.ejemplo.com/ruta1/ruta2/ruta3/pagina1.html
Página enlazada	Página web llamada pagina2.html y que se encuentra en el mismo directorio
URL absoluta	http://www.ejemplo.com/ruta1/ruta2/ruta3/pagina2.html
URL relativa	pagina2.html

Cuando el navegador encuentra una URL relativa que sólo consiste en el nombre de un recurso, supone que el protocolo, servidor y directorio del recurso enlazado son los mismos que los del origen del enlace.

2) El destino del enlace se encuentra cerca de su origen y en un nivel superior

En este caso, el recurso que se enlaza no está en el mismo directorio que el origen del enlace pero sí que está *cerca* y en algún directorio superior. La URL relativa debe indicar de alguna manera que es necesario *subir* un nivel en la jerarquía de directorios para llegar hasta el recurso.

Para indicar al navegador que debe subir un nivel, se incluyen dos puntos y una barra (`../`) en la ruta del recurso enlazado. De esta forma, cada vez que aparece `../` en una URL relativa, significa que se debe subir un nivel.

Elemento	Valor
Página origen	http://www.ejemplo.com/ruta1/ruta2/ruta3/pagina1.html
Página enlazada	Página web llamada pagina2.html y que se encuentra en el directorio superior llamado <code>ruta2</code>
URL absoluta	http://www.ejemplo.com/ruta1/ruta2/pagina2.html
URL relativa	../pagina2.html

Cuando el navegador encuentra la URL relativa `../pagina2.html`, sabe que para encontrar el recurso enlazado ([pagina2.html](#)) tiene que subir un nivel desde el lugar en el que se encuentra esa URL relativa. La página que incluye esa URL se encuentra en el directorio `ruta1/ruta2/ruta3`, por lo que subir un nivel equivale entrar en el directorio `ruta1/ruta2`.

De la misma forma, si el destino se encuentra un par de niveles por encima, se debe incluir `../` dos veces seguidas:

Elemento	Valor
Página origen	http://www.ejemplo.com/ruta1/ruta2/ruta3/pagina1.html
Página enlazada	Página web llamada pagina2.html y que se encuentra en el directorio superior llamado <code>ruta1</code>
URL absoluta	http://www.ejemplo.com/ruta1/pagina2.html
URL relativa	<code>../../pagina2.html</code>

Además de subir niveles, también se puede entrar en otros directorios para obtener los recursos:

Elemento	Valor
Página origen	http://www.ejemplo.com/ruta1/ruta2/ruta3/pagina1.html
Página enlazada	Página web llamada pagina2.html y que se encuentra en un directorio llamado <code>ruta4</code> que se encuentra en la raíz del servidor
URL absoluta	http://www.ejemplo.com/ruta4/pagina2.html
URL relativa	<code>../../../../ruta4/pagina2.html</code>

Si se intentan subir más niveles de los que es posible, el navegador ignora todos los `../` sobrantes. Si la página que tiene el enlace es <http://www.ejemplo.com/ruta1/ruta2/ruta3/pagina1.html> y la URL relativa que se incluye es `../../../../../../../../pagina2.html`, el navegador realmente la interpreta como [../../../../pagina2.html](http://www.ejemplo.com/ruta4/pagina2.html).

Como el objetivo de las URL relativas es crear URL más cortas y sencillas que las URL absolutas, este método sólo se puede utilizar cuando el origen y el destino se encuentran cerca, porque de otro modo la URL relativa se complica demasiado.

3) El destino del enlace se encuentra cerca de su origen y en un nivel inferior

Este caso es muy similar al anterior, pero más sencillo. Si el recurso enlazado se encuentra en algún directorio inferior al que se encuentra el origen, sólo es necesario indicar el nombre de los directorios a los que debe entrar el navegador.

Elemento	Valor
Página origen	http://www.ejemplo.com/ruta1/ruta2/ruta3/pagina1.html

Elemento	Valor
Página enlazada	Página web llamada pagina2.html y que se encuentra en un directorio inferior llamado ruta4
URL absoluta	http://www.ejemplo.com/ruta1/ruta2/ruta3/ruta4/pagina2.html
URL relativa	ruta4/pagina2.html

De la misma forma, se pueden indicar varios directorios seguidos para que el navegador descienda jerárquicamente por la estructura de directorios:

Elemento	Valor
Página origen	http://www.ejemplo.com/ruta1/ruta2/ruta3/pagina1.html
Página enlazada	Página web llamada pagina2.html y que se encuentra en un directorio inferior llamado ruta6 que está dentro del directorio ruta5 y que a su vez está dentro del directorio ruta4
URL absoluta	http://www.ejemplo.com/ruta1/ruta2/ruta3/ruta4/ruta5/ruta6/pagina2.html
URL relativa	ruta4/ruta5/ruta6/pagina2.html

4) El origen y el destino del enlace se encuentran muy alejados

Cuando el origen y el destino de un enlace se encuentran muy alejados (pero en el mismo servidor) las URL relativas se pueden complicar en exceso. Aunque es posible utilizar `../` para subir por la jerarquía de directorios y se puede entrar en cualquier directorio indicando su nombre, las URL relativas que se obtienen son demasiado largas y complicadas.

En estos casos, lo más sencillo es indicar la ruta completa hasta el recurso enlazado comenzando desde la raíz del servidor web. Por lo tanto, estas URL relativas sólo omiten el protocolo y el nombre del servidor.

Elemento	Valor
Página origen	http://www.ejemplo.com/ruta1/ruta2/ruta3/pagina1.html
Página enlazada	Página web llamada pagina2.html y que se guarda en un directorio llamado ruta7 que se encuentra en la raíz del servidor
URL absoluta	http://www.ejemplo.com/ruta7/pagina2.html

Elemento	Valor
URL relativa	/ruta7/pagina2.html

Cuando la URL relativa comienza por /, el navegador considera que es la ruta completa comenzando desde la raíz del servidor, por lo que sólo le añade el protocolo y el nombre del servidor origen.

A continuación se resumen los cuatro posibles casos de URL relativas y el procedimiento que sigue el navegador para convertirlas en URL absolutas:

Si la URL relativa...	El navegador la transforma en URL absoluta...
...sólo consiste en el nombre de un recurso	...añadiendo el protocolo, servidor y ruta completa del origen del enlace
...comienza por ../	...añadiendo el protocolo y servidor del origen del enlace, subiendo un nivel en la jerarquía de directorios y añadiendo el resto de la ruta incluida en la URL relativa
...comienza por /	...añadiendo el protocolo y servidor del origen del enlace
En cualquier otro caso	...añadiendo el protocolo, servidor y ruta completa del origen del enlace, a la que se añade la ruta incluida en la URL relativa

4.3. Enlaces básicos

Los enlaces en HTML se crean mediante la etiqueta `<a>` (su nombre viene del inglés "anchor", literalmente traducido como "ancla"). A continuación se muestra la definición simplificada de `<a>` y más adelante se muestra su definición completa:

<code><a></code>	Se emplea para enlazar todo tipo de recursos		
Atributos comunes	básicos, internacionalización, eventos y foco	Tipo de elemento	En línea
Atributos propios	<ul style="list-style-type: none"> <code>name = "texto"</code> - Permite nombrar al enlace para que se pueda acceder desde otros enlaces <code>href = "url"</code> - Indica la URL del recurso que se quiere enlazar 		

El atributo más importante de la etiqueta `<a>` es `href`, que se utiliza para indicar la URL a la que apunta el enlace. Cuando el usuario pincha sobre un enlace, el navegador se dirige a la URL del recurso indicado mediante `href`. Las URL de los enlaces pueden ser absolutas, relativas, internas y externas.

Con la definición anterior, para crear un enlace que apunte a la página principal de Google solamente habría que incluir lo siguiente en un documento HTML:

```
<a href="http://www.google.com">Página principal de Google</a>
```

Como el atributo `href` indica una URL, un enlace puede apuntar a cualquier tipo de recurso al que pueda acceder el navegador. El siguiente enlace apunta a una imagen, que se mostrará en el navegador cuando el usuario pinche sobre el enlace:

```
<a href="http://www.ejemplo.com/fondo_escritorio.jpg">Imagen interesante para un fondo de escritorio</a>
```

De la misma forma, los enlaces pueden apuntar directamente a documentos PDF, Word, etc.

```
<a href="http://www.ejemplo.com/informe.pdf">Descargar informe completo [PDF]</a>  
<a href="http://www.ejemplo.com/informe.doc">Descargar informe completo [DOC]</a>
```

Un truco muy útil con los enlaces es el uso de URL relativas para poder volver al inicio del sitio web desde cualquier página web interior:

```
<a href="/">Volver al inicio</a>
```

El enlace anterior utiliza una URL relativa con una ruta que apunta directamente a la raíz del servidor. Para obtener la URL absoluta, el navegador añade el mismo protocolo y el mismo nombre de servidor de la página en la que se encuentra el enlace. El resultado es que cuando se pincha ese enlace, siempre se vuelve al inicio del sitio web, independientemente de la página en la que se incluya el enlace.

El otro atributo básico de la etiqueta `<a>` es `name`, que permite definir enlaces dentro de una misma página web. Si una página es muy larga, puede ser útil mostrar enlaces de tipo "[Saltar hasta la segunda sección](#)", "[Volver al principio de la página](#)", etc.

Este tipo de enlaces son especiales, ya que la URL de la página siempre es la misma para todas las secciones y por tanto, debe añadirse otra parte a las URL para identificar cada sección.

```
<a name="primera_seccion"></a>  
<p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Mauris id ligula eu felis  
adipiscing ultrices. Duis gravida leo ut lectus. Praesent condimentum mattis ligula.</p>  
...  
<a name="segunda_seccion"></a>  
<p>Pellentesque malesuada. In in lacus. Phasellus erat erat, lacinia a, convallis eu,  
nonummy et, odio. Aenean urna elit, ultrices id, placerat varius, facilisis eget,  
dolor.</p>  
...
```

El atributo `name` permite crear "enlaces vacíos" que hacen referencia a secciones dentro de una misma página. Una vez definidos los "enlaces vacíos", es posible crear un enlace que apunte directamente a una sección concreta de una página:

```
<!-- Enlace normal a la página -->  
<a href="http://www.ejemplo.com/pagina1.html">Enlace a la página 1</a>
```

```
<!-- Enlace directo a la segunda sección de la página -->
<a href="http://www.ejemplo.com/pagina1.html#segunda_seccion">Enlace a la sección 2 de
la página 1</a>
```

La sintaxis que se utiliza con estos enlaces es la misma que con los enlaces normales, salvo que se añade el símbolo # seguido del nombre de la sección a la que se apunta. Cuando el usuario pincha sobre uno de estos enlaces, el navegador accede a la página apuntada por la URL y baja directamente a la sección cuyo nombre se indica después del símbolo #.

También es posible utilizar este tipo de enlaces con URL relativas en una misma página. El siguiente ejemplo añade enlaces de tipo "Volver al inicio de la página" en varias secciones:

```
<a name="inicio"></a>

<p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Mauris id ligula eu felis
adipiscing ultrices. Duis gravida leo ut lectus. Praesent condimentum mattis ligula.</p>
<a href="#inicio">Volver al inicio de la página</a>
...

<p>Pellentesque malesuada. In in lacus. Phasellus erat erat, lacinia a, convallis eu,
nonummy et, odio. Aenean urna elit, ultrices id, placerat varius, facilisis eget,
dolor.</p>
<a href="#inicio">Volver al inicio de la página</a>
...
```

Los enlaces directos a secciones también funcionan con el atributo `id` de cualquier elemento. El siguiente ejemplo es equivalente al ejemplo anterior:

```
<h1 id="inicio">Título de la página</h1>

<p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Mauris id ligula eu felis
adipiscing ultrices. Duis gravida leo ut lectus. Praesent condimentum mattis ligula.</p>
<a href="#inicio">Volver al inicio de la página</a>
...

<p>Pellentesque malesuada. In in lacus. Phasellus erat erat, lacinia a, convallis eu,
nonummy et, odio. Aenean urna elit, ultrices id, placerat varius, facilisis eget,
dolor.</p>
<a href="#inicio">Volver al inicio de la página</a>
...
```

El nombre de la sección que se indica después del símbolo # puede utilizar el valor de los atributos `id` de cualquier elemento. De hecho, se recomienda utilizar los atributos `id` de los elementos ya existentes en la página en vez de crear "enlaces vacíos" de tipo ``.

Ejercicio 6

A partir de la estructura de directorios y archivos indicada en la siguiente imagen:

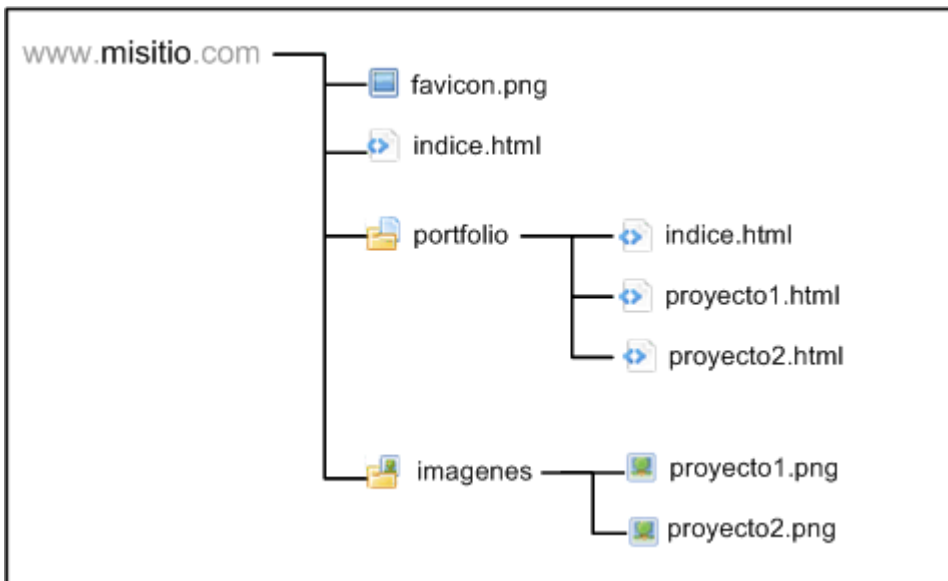


Figura 4.2 Estructura de archivos y directorios de un sitio web de ejemplo

1) Crear la siguiente página llamada `indice.html` que sirva como página principal del sitio:



Figura 4.3 Página principal del sitio web de ejemplo

2) Crear la página de índice del portfolio:



Figura 4.4 Página con la información sobre los proyectos realizados

4.4. Enlaces avanzados

Incluir enlaces básicos mediante la etiqueta `<a>` es muy sencillo. Sin embargo, la definición completa de `<a>` es muy compleja, ya que dispone de varios atributos específicos importantes. A continuación se muestra la definición completa de `<a>`:

<a>	Se emplea para enlazar todo tipo de recursos		
Atributos comunes	básicos, internacionalización, eventos y foco	Tipo de elemento	En línea
Atributos propios	<ul style="list-style-type: none"> • <code>name = "texto"</code> - Permite nombrar al enlace para que se pueda acceder desde otros enlaces • <code>href = "url"</code> - Indica la URL del recurso que se quiere enlazar • <code>hreflang = "codigo_idioma"</code> - Idioma del recurso enlazado • <code>type = "tipo_de_contenido"</code> - Permite "avisar" al navegador sobre el tipo de contenido que se enlaza (imágenes, archivos, etc.) para que pueda prepararse en caso de que no entienda ese contenido • <code>rel = "tipo_de_relacion"</code> - Describe la relación del documento actual con el recurso enlazado • <code>rev = "tipo_de_relacion"</code> - Describe la relación del recurso enlazado con el documento actual • <code>charset = "tipo_de_charset"</code> - Describe la codificación del recurso enlazado 		

4.4.1. Idioma del enlace (hreflang)

El enlace puede indicar al navegador el idioma del recurso que se enlaza. Para establecer el valor del idioma, se utiliza un código estandarizado de dos letras. Además del idioma genérico, también se puede indicar una variación idiomática. Ejemplo de códigos de idioma más utilizados:

Código	Idioma	Variación idiomática
<code>en</code>	Inglés	-
<code>en-AU</code>	Inglés	Australia
<code>en-US</code>	Inglés	Estados Unidos
<code>es</code>	Español	-
<code>es-AR</code>	Español	Argentina
<code>es-ES</code>	Español	España
<code>es-MX</code>	Español	México

Otros códigos utilizados son: `fr` (francés), `de` (alemán), `it` (italiano), `nl` (holandés), `el` (griego), `pt` (portugués), `ar` (árabe), `he` (hebreo), `ru` (ruso), `zh` (chino), `ja` (japonés).

La lista completa de códigos de idioma está definida en el estándar ISO 639 (<http://xml.coverpages.org/iso639a.html>).

4.4.2. Tipo de contenido (type)

Se utiliza para notificar al navegador sobre el tipo de contenido que se enlaza. Se indica mediante una cadena de texto cuyos posibles valores también están estandarizados. Los valores de los contenidos más utilizados son los siguientes: "text/html" (páginas HTML), "image/png" (imágenes con formato PNG), "image/gif" (imágenes con formato GIF), "text/css" (hojas de estilo CSS), "application/rss+xml" (archivos RSS).

La lista completa de tipos de contenido se define en los estándares RFC 2045 y RFC 2046 (<http://www.iana.org/assignments/media-types/>).

4.4.3. Tipo de relación (rel y rev)

Los enlaces pueden proporcionar información adicional muy útil para los navegadores y para los motores de búsqueda como Google. Los atributos `rel` y `rev` permiten indicar la relación que la página actual tiene con la página a la que se enlaza (atributo `rel`) y la relación que tiene la página enlazada con la página actual (atributo `rev`).

Los tipos de relación definidos son los siguientes:

- `alternate` – Indica que es una versión alternativa al documento actual (puede ser una versión en otro idioma o una versión preparada para otro medio, como una impresora o un dispositivo móvil)
- `stylesheet` – Indica que se ha enlazado una hoja de estilos
- `start` – Indica que se trata del primer documento de una colección de documentos (por ejemplo el primer capítulo de un libro)
- `next` – Indica que es el documento que sigue al actual dentro de una secuencia lógica de documentos (por ejemplo, los capítulos de un libro)
- `prev` – Indica que es el documento que precede al actual dentro de una secuencia lógica de documentos (por ejemplo, los capítulos de un libro)
- `contents` – Indica que el recurso enlazado es el documento que contiene la tabla de contenidos de la colección de documentos (por ejemplo, el índice de un libro).
- `bookmark` – Establece el enlace actual como un "marcador" o "favorito". Un marcador es un enlace que constituye un punto de entrada muy importante dentro del documento.

La especificación oficial de HTML define la lista completa de tipos de relaciones (<http://www.w3.org/TR/1999/REC-html401-19991224/types.html#type-links>) que se pueden utilizar.

4.4.4. Codificación de caracteres (charset)

Además del idioma, tipo de contenido y relación del recurso que se enlaza, los enlaces también pueden indicar la codificación de caracteres que utiliza la página web enlazada.

Los valores que se pueden utilizar también están estandarizados y las codificaciones más utilizadas son UTF-8 y ISO-8859-1, aunque existen decenas de códigos definidos (ISO-10646-UCS-2, IBM852, Big5-HKSCS, windows-1252, HZ-GB-2312).

El organismo IANA publica la lista completa de codificaciones de caracteres disponibles (<http://www.iana.org/assignments/character-sets>).

Los ejemplos anteriores de enlaces básicos se pueden rehacer utilizando algunos de los atributos definidos por la etiqueta `<a>`:

```
<a href="http://www.google.com" hreflang="en" type="text/html" charset="UTF-8">Página principal de Google</a>
```

```
<a href="http://www.ejemplo.com/fondo_escritorio.jpg" type="image/jpg">Imagen interesante para un fondo de escritorio</a>
```

4.5. Otros tipos de enlaces

Los enlaces mostrados en las secciones anteriores son los más utilizados por las páginas web. Los enlaces creados con la etiqueta `<a>` permiten enlazar cualquier tipo de recurso desde cualquier página. La característica más importante de estos enlaces es que el usuario debe activar la carga de los recursos. En otras palabras, el navegador no carga ningún recurso enlazado con la etiqueta `<a>` a menos que el usuario pinche sobre el enlace.

Además de estos enlaces, las páginas HTML pueden incluir otro tipo de enlaces que cargan los recursos automáticamente. Si una página HTML utiliza archivos CSS para aplicar estilos a sus contenidos, no es lógico que los enlace con la etiqueta `<a>` y espere a que el usuario pinche sobre el enlace para así cargar los archivos CSS. De la misma forma, muchas páginas web dinámicas necesitan que el navegador cargue varios archivos JavaScript para funcionar correctamente.

HTML define las etiquetas `<script>` y `<link>` para enlazar recursos que se deben cargar automáticamente. Cuando el navegador encuentra alguna de estas dos etiquetas, descarga los recursos enlazados y los aplica a la página web.

La etiqueta `<script>` tiene dos modos de funcionamiento, ya que se emplea tanto para insertar un bloque de código JavaScript en la página como para enlazar un archivo JavaScript externo.

<code><script></code>	Se emplea para enlazar o definir un bloque de código (normalmente JavaScript)	
Atributos comunes	- Tipo de elemento	Bloque y en línea (también puede ser una etiqueta vacía)
Atributos propios	<ul style="list-style-type: none"> • <code>src = "url"</code> - Indica la dirección del archivo que contiene el código • <code>type = "tipo_de_contenido"</code> - Permite "avisar" al navegador sobre el tipo de código que se incluye (normalmente JavaScript) • <code>defer = "defer"</code> - El código no va a modificar el contenido de la página web • <code>charset = "tipo_de_charset"</code> - Describe la codificación del código enlazado 	

Aunque la etiqueta `<script>` permite enlazar código de varios lenguajes de programación, el uso habitual de `<script>` consiste en enlazar un archivo JavaScript externo:

```
<head>
  <script type="text/javascript" src="http://www.ejemplo.com/js/
inicializar.js"></script>
</head>
```

El atributo `type` utilizado habitualmente para los archivos JavaScript es `"text/javascript"`. El atributo `src` es equivalente al atributo `href` de los enlaces creados con la etiqueta `<a>`. La URL indicada en el atributo `src` puede ser absoluta o relativa y externa o interna.

Además de enlazar un archivo JavaScript externo, la misma etiqueta `<script>` también permite incluir en la página web un bloque de código JavaScript:

```
<html>
<head>
  <script type="text/javascript">
    //
      window.onload = function() { alert("La página se ha cargado completamente"); }
    //]]&gt;
  &lt;/script&gt;
&lt;/head&gt;
&lt;body&gt;
  ...
&lt;/body&gt;
&lt;/html&gt;</pre></div><div data-bbox="138 492 913 564" data-label="Text"><p>Cuando se incluye código JavaScript en la propia página XHTML, se debe insertar dentro de una sección especial llamada <code>CDATA</code>. Para ello, el código JavaScript se debe encerrar entre <code>&lt;![CDATA[</code> y <code>]]&gt;</code>. Cuando el navegador encuentra una sección de este tipo, no procesa su contenido como si fuera XHTML y por tanto no tiene en cuenta los posibles errores de validación de XHTML.</p></div><div data-bbox="138 577 913 650" data-label="Text"><p>De esta forma, se pueden construir páginas XHTML válidas y código JavaScript correcto. En los capítulos posteriores se profundiza en el concepto de validación de páginas XHTML. Los caracteres <code>//</code> al comienzo y al final de la sección <code>CDATA</code> son necesarios para los navegadores que no son capaces de procesar correctamente estas secciones.</p></div><div data-bbox="138 664 913 718" data-label="Text"><p>La etiqueta <code>&lt;script&gt;</code> (tanto cuando enlaza, como cuando incluye directamente el código) puede aparecer en cualquier parte del documento HTML, aunque normalmente se incluye dentro de la cabecera de la página (<code>&lt;head&gt;...&lt;/head&gt;</code>).</p></div><div data-bbox="138 732 913 768" data-label="Text"><p>La segunda etiqueta de XHTML para enlazar recursos es <code>&lt;link&gt;</code>, que permite enlazar y relacionar la página con otros recursos externos.</p></div><div data-bbox="878 925 913 944" data-label="Page-Footer">67</div>
```

<code><link></code>	Se emplea para enlazar y establecer relaciones entre el documento y otros recursos		
Atributos comunes	básicos, internacionalización y eventos	Tipo de elemento	Etiqueta vacía
Atributos propios	<ul style="list-style-type: none"> Los siguientes con el mismo significado que para la etiqueta "a": <code>charset</code>, <code>href</code>, <code>hreflang</code>, <code>type</code>, <code>rel</code> y <code>rev</code> <code>media = "tipo_de_medio"</code> - Indica el medio para el que debe aplicarse la relación 		

Al contrario que `<script>`, la etiqueta `<link>` solamente se puede incluir dentro de la cabecera del documento. Se pueden añadir tantas etiquetas `<link>` como sean necesarias, pero siempre dentro de `<head>...</head>`.

El atributo `media` hace referencia al medio para el que es válida la relación con el recurso enlazado. Los medios disponibles también están estandarizados, siendo los más comunes `screen` para los contenidos mostrados en pantalla, `print` para las impresoras y `handheld` para los dispositivos móviles.

El uso habitual de la etiqueta `<link>` es el de enlazar las hojas de estilos CSS utilizadas por las páginas web:

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

En este caso, es habitual establecer los atributos `rel` y `type` para indicar el tipo de recurso enlazado y su relación con la página web. La URL del recurso enlazado se indica en el atributo `href`, que admite tanto URL absolutas como relativas.

4.6. Ejemplos de enlaces habituales

4.6.1. Enlace al inicio del sitio web

```
<a href="/">Inicio</a>
```

Al pulsar el enlace anterior desde cualquier página web, se vuelve directamente a la página de inicio, *home* o página principal del sitio web.

4.6.2. Enlace a un email

```
<a href="mailto:nombre@direccion.com" title="Dirección de email para solicitar más información">
Solicita más información
</a>
```

Al pinchar sobre el enlace anterior, se abre automáticamente el programa de correo electrónico del ordenador del usuario y se establece la dirección de envío al valor indicado después de `mailto:`. La sintaxis es la misma que la de un enlace normal, salvo que se cambia el prefijo `http://` por `mailto:`:

La sintaxis de `mailto:` permite utilizarlo para otros ejemplos más complejos:

```
<!-- Envío del correo electrónico a varias direcciones a la vez -->
<a href="mailto:nombre@direccion.com,otro_nombre@direccion.com">Solicita más
información</a>

<!-- Añadir un "asunto" inicial al correo electrónico -->
<a href="mailto:nombre@direccion.com?subject=Solicitud de más información">Solicita más
información</a>

<!-- Añadir un texto inicial en el cuerpo del correo electrónico -->
<a href="mailto:nombre@direccion.com?body=Estaría interesado en solicitar más
información sobre sus productos">Solicita más información</a>
```

Todas las opciones anteriores se pueden combinar entre sí para realizar ejemplos más avanzados. Aunque el uso de `mailto:` puede parecer una ventaja, su uso está desaconsejado. Si se incluye una dirección de correo electrónico directamente en una página web, es muy probable que en poco tiempo esa dirección de email se encuentre llena de correo electrónico basura o "*spam*", ya que existen programas automáticos encargados de rastrear sistemáticamente todas las páginas web de Internet para encontrar direcciones de correo electrónico válidas.

La forma de mostrar las direcciones de correo electrónico en las páginas web consiste en incluir la dirección en una imagen o indicarla de forma que solamente los usuarios puedan entenderlo:

```
<p>La dirección de correo es <strong>nombre (arroba) direccion.com</strong></p>
<p>La dirección de correo es <strong>nombre_arroba_direccion.com</strong></p>
<p>La dirección de correo es <strong>nombreQUITAESTO@direccion.com</strong></p>
<p>La dirección de correo es <strong>nombre(ARROBA)direccion.com</strong></p>
<p>La dirección de correo es <strong>nombre @ direccion . com</strong></p>
```

4.6.3. Enlace a un archivo FTP

Para enlazar un archivo almacenado en un servidor FTP, la parte del protocolo de la URL debe cambiar de `http://` a `ftp://`:

```
<a href="ftp://ftp.ejemplo.com/ruta/archivo.zip" title="Archivo comprimido de los
contenidos">
Descarga un ZIP con todos los contenidos
</a>
```

4.6.4. Enlazar varias hojas de estilos CSS

```
<link rel="stylesheet" type="text/css" href="/css/comun.css" />
<link rel="stylesheet" type="text/css" href="/css/secciones.css" />
```

4.6.5. Enlazar hojas de estilos CSS para diferentes medios

```
<link rel="stylesheet" type="text/css" href="/css/comun.css" media="screen, projection" />
<link rel="stylesheet" type="text/css" href="/css/impresora.css" media="print" />
<link rel="stylesheet" type="text/css" href="/css/movil.css" media="handheld" />
```

4.6.6. Enlazar el favicon

El *favicon* o icono para favoritos es el pequeño icono que muestran las páginas en varias partes del navegador. Dependiendo del navegador que se utilice, este icono se muestra en la barra de direcciones, en la barra de título del navegador y/o en el menú de favoritos/marcadores.

```
<link rel="shortcut icon" href="/favicon.ico" type="image/ico" />
```

Aunque en principio la imagen debería ser de tipo **.ICO** (formato gráfico de los iconos), algunos navegadores soportan favicons en otros formatos gráficos más habituales (como por ejemplo **.PNG**).

4.6.7. Enlazar un archivo RSS

```
<link rel="alternate" type="application/rss+xml" title="Resumen de todos los artículos del blog" href="/feed.xml" />
```

4.6.8. Enlazar hojas de estilos, favicon y RSS

En una misma página se pueden incluir varias etiquetas `<link>`, por lo que es habitual que las páginas enlacen hojas de estilos, favicon y archivos RSS de forma conjunta:

```
<head>
...
<link rel="stylesheet" type="text/css" href="/css/impresora.css" media="print" />
<link rel="stylesheet" type="text/css" href="/css/movil.css" media="handheld" />
<style type="text/css" media="screen,projection">
  @import '/css/main.css';
</style>
<link rel="shortcut icon" href="/favicon.ico" type="image/ico" />
<link rel="alternate" type="application/rss+xml" title="Resumen de todos los artículos del blog" href="/feed.xml" />
...
</head>
```

4.6.9. Indicar que existe una versión de la página en otro idioma

```
<head>
<title>English tutorial</title>
<link lang="es" xml:lang="es" title="El tutorial en español" type="text/html"
rel="alternate" hreflang="es" href="http://www.ejemplo.com/tutorial/espanol.html" />
</head>
```

4.6.10. Indicar que existe una versión de la página preparada para imprimir

```
<head>
<link media="print" title="El tutorial en PDF" type="application/pdf" rel="alternate"
```

```
href="http://www.ejemplo.com/tutorial/documento.pdf" />
</head>
```

4.6.11. Indicar que existe una página que es índice de la página actual

```
<head>
<title>Tutorial - Capítulo 5</title>
<link rel="start" title="El índice del tutorial" type="text/html"
href="http://www.ejemplo.com/tutorial/indice.html" />
</head>
```

Ejercicio 7

Enlazar el favicon en todas las páginas del ejercicio 6 y añadir todos los atributos posibles a los enlaces.

Esta página se ha dejado vacía a propósito

Capítulo 5.

Listas

En ocasiones, es posible agrupar determinadas palabras o frases en un conjunto de elementos que tienen más significado de forma conjunta. El menú de navegación de un sitio web por ejemplo está formado por un grupo de palabras. Aunque cada palabra por separado tiene sentido, de forma conjunta constituyen el menú de navegación de la página, por lo que su significado conjunto es mayor que por separado.

El lenguaje HTML define tres tipos diferentes de listas para agrupar los elementos: listas no ordenadas (se trata de una colección simple de elementos en la que no importa su orden), listas ordenadas (similar a la anterior, pero los elementos están numerados y por tanto, importa su orden) y listas de definición (un conjunto de términos y definiciones similar a un diccionario).

5.1. Listas no ordenadas

Las listas no ordenadas son las más sencillas y las que más se utilizan. Una lista no ordenada es un conjunto de elementos relacionados entre sí pero para los que no se indica un orden o secuencia determinados. La etiqueta `` encierra todos los elementos de la lista y la etiqueta `` cada uno de sus elementos.

<code></code>	Se emplea para definir listas no ordenadas		
Atributos comunes	básicos, internacionalización y eventos	Tipo de elemento	Bloque
Atributos propios	-		

<code></code>	Se emplea para definir los elementos de las listas (ordenadas y no ordenadas)		
Atributos comunes	básicos, internacionalización y eventos	Tipo de elemento	Bloque
Atributos propios	-		

El siguiente código HTML muestra un ejemplo sencillo de lista no ordenada:

```
<html>
<head><title>Ejemplo de etiqueta UL</title></head>
<body>

<h1>Menú</h1>

<ul>
  <li>Inicio</li>
  <li>Noticias</li>
  <li>Artículos</li>
  <li>Contacto</li>
</ul>

</body>
</html>
```

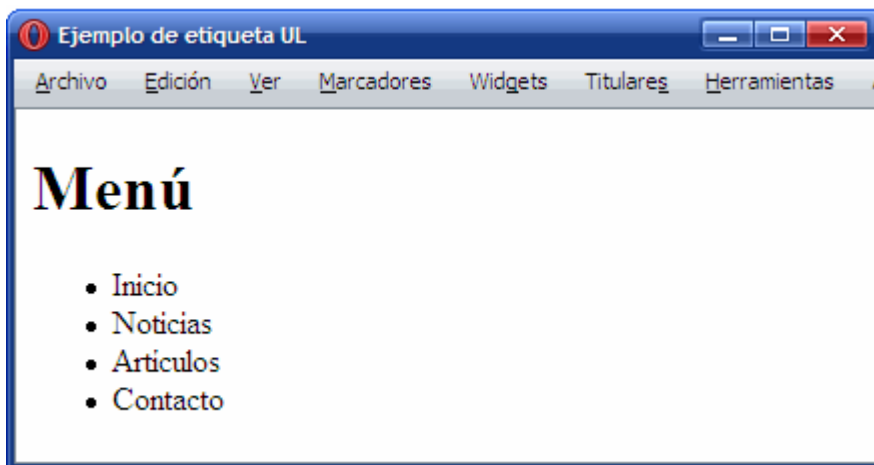


Figura 5.1 Ejemplo de uso de la etiqueta ul

El navegador por defecto muestra los elementos de la lista tabulados y con una pequeña viñeta formada por un círculo negro. Como ya se sabe, el aspecto con el que se muestran los elementos de las listas se puede modificar mediante las hojas de estilos CSS.

5.2. Listas ordenadas

Las listas ordenadas son casi idénticas a las listas no ordenadas, salvo que en este caso los elementos relacionados se muestran siguiendo un orden determinado. Cuando se crea por ejemplo una lista con las instrucciones de un producto, es importante el orden en el que se realiza cada paso. Cuando se muestra un índice o tabla de contenidos en un libro, es importante el orden de cada elemento del índice.

En todos estos casos, la lista más adecuada es la lista ordenada, que se define mediante la etiqueta ``. Los elementos de la lista se definen mediante la etiqueta ``, la misma que se utiliza en las listas no ordenadas.

	Se emplea para definir listas ordenadas		
Atributos comunes	básicos, internacionalización y eventos	Tipo de elemento	Bloque
Atributos propios	-		

El siguiente código HTML muestra un ejemplo sencillo de lista ordenada:

```
<html>
<head><title>Ejemplo de etiqueta OL</title></head>
<body>

<h1>Instrucciones</h1>

<ol>
  <li>Enchufar correctamente</li>
  <li>Comprobar conexiones</li>
  <li>Encender el aparato</li>
</ol>

</body>
</html>
```

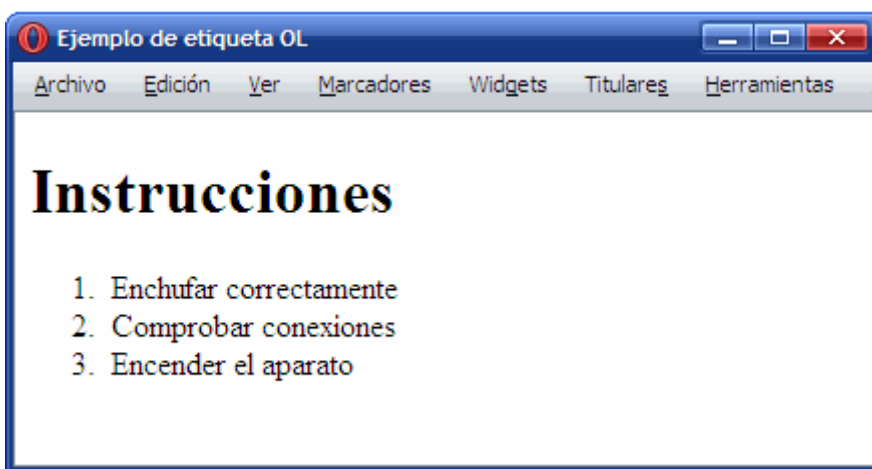


Figura 5.2 Ejemplo de uso de la etiqueta ol

El navegador muestra la lista de forma muy parecida a las listas no ordenadas, salvo que en este caso no se emplean viñetas gráficas en los elementos, sino que se numeran de forma consecutiva. El tipo de numeración empleada también se puede modificar aplicando hojas de estilos CSS a los elementos de la lista.

5.3. Listas de definición

Las listas de definición apenas se utilizan en la mayoría de páginas HTML. Su funcionamiento es similar al de un diccionario, ya que cada elemento de la lista está formado por términos y definiciones. La etiqueta `<dl>` crea la lista de definición y las etiquetas `<dt>` y `<dd>` definen respectivamente el término y la descripción de cada elemento de la lista.

<dl>	Se emplea para definir listas de definición		
Atributos comunes	básicos, internacionalización y eventos	Tipo de elemento	Bloque
Atributos propios	-		

<dt>	Se emplea para definir los términos de los elementos de una lista de definición		
Atributos comunes	básicos, internacionalización y eventos	Tipo de elemento	Bloque
Atributos propios	-		

<dd>	Se emplea para indicar las definiciones de los elementos de una lista de definición		
Atributos comunes	básicos, internacionalización y eventos	Tipo de elemento	Bloque
Atributos propios	-		

El siguiente código HTML muestra un ejemplo sencillo de lista de definición:

```

<html>
<head><title>Ejemplo de etiqueta DL</title></head>
<body>
<h1>Metalenguajes</h1>

<dl>
  <dt>SGML</dt>
  <dd>Metalenguaje para la definición de otros lenguajes de marcado</dd>

  <dt>XML</dt>
  <dd>Lenguaje basado en SGML y que se emplea para describir datos</dd>

  <dt>RSS</dt>
  <dt>GML</dt>
  <dt>XHTML</dt>
  <dt>SVG</dt>
  <dt>XUL</dt>
  <dd>Lenguajes derivados de XML para determinadas aplicaciones</dd>
</dl>

</body>
</html>

```

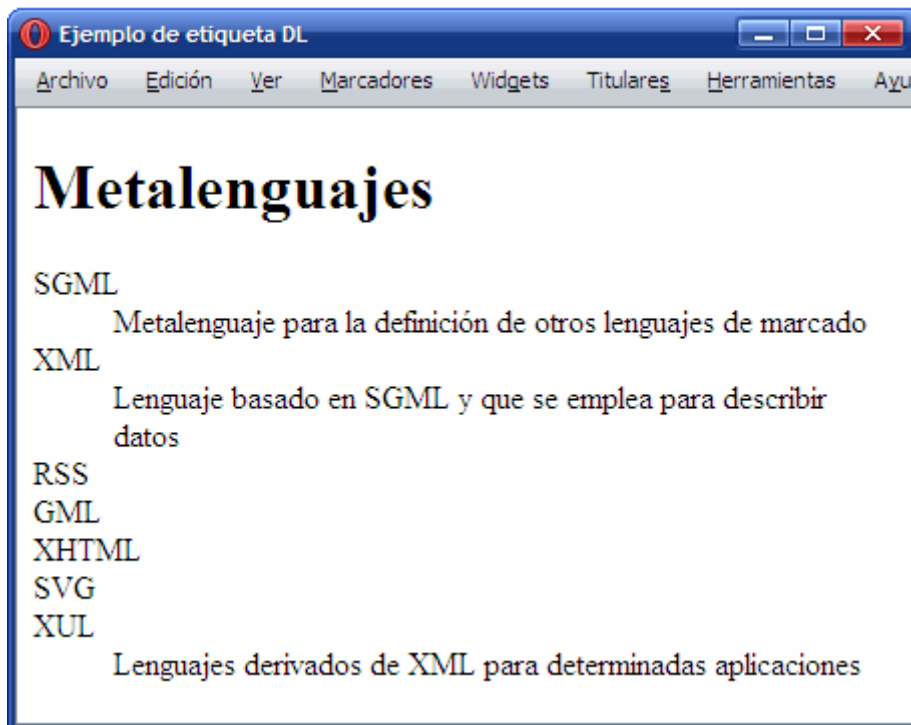


Figura 5.3 Ejemplo de uso de la etiqueta dl

Los navegadores formatean las listas de definición de forma similar a las otras listas, tabulando la definición y alineando a la izquierda los términos. Aunque no es habitual, cada término puede tener asociada más de una definición y cada definición puede tener asociada varios términos.

Ejercicio 8

Determinar el código HTML que corresponde a la siguiente lista anidada simple



Figura 5.4 Ejemplo de lista anidada simple de dos niveles

Ejercicio 9

Determinar el código HTML que corresponde a la siguiente lista anidada compleja



Figura 5.5 Ejemplo de lista anidada compleja de dos niveles

Capítulo 6.

Imágenes y objetos

6.1. Imágenes

Las imágenes son uno de los elementos más importantes de las páginas web. De hecho, prácticamente todas las páginas web contienen alguna imagen y la mayoría incluyen decenas de imágenes. Dentro de las imágenes que se pueden incluir en una página HTML se deben distinguir dos tipos: las imágenes de contenido y las imágenes *de adorno*.

Las imágenes de contenido son las que proporcionan información y complementan la información textual. Las imágenes *de adorno* son las que se utilizan para hacer bordes redondeados, para mostrar pequeños iconos en las listas de elementos, para mostrar fondos de página, etc. Las imágenes de contenido se incluyen directamente en el código HTML mediante la etiqueta `` y las imágenes de adorno no se deberían incluir en el código HTML, sino que deberían emplearse hojas de estilos CSS para mostrarlas.

A continuación se muestra la definición de la etiqueta ``, utilizada para incluir las imágenes en las páginas HTML:

	Se emplea para incluir imágenes en los documentos		
Atributos comunes	básicos, internacionalización y eventos	Tipo de elemento	En línea y etiqueta vacía
Atributos propios	<ul style="list-style-type: none"> • <code>src = "url"</code> - Indica la URL de la imagen que se muestra • <code>alt = "texto"</code> - Descripción corta de la imagen • <code>longdesc = "url"</code> - Indica una URL en la que puede encontrarse una descripción más detallada de la imagen • <code>name = "texto"</code> - Nombre del elemento imagen • <code>height = "unidad_de_medida"</code> - Indica la altura con la que se debe mostrar la imagen (no es obligatorio que coincida con la altura original de la imagen) • <code>width = "unidad_de_medida"</code> - Indica la anchura con la que se debe mostrar la imagen (no es obligatorio que coincida con la anchura original de la imagen) 		

Los dos atributos requeridos son `src` y `alt`. El atributo `src` es similar al atributo `href` de los enlaces, ya que establece la URL de la imagen que se va a mostrar en la página. Las URL indicadas pueden ser absolutas o relativas. El atributo `alt` permite describir el contenido de la imagen mediante un texto breve. Las descripciones deben tener una longitud inferior a 1024 caracteres y son útiles para las personas y dispositivos discapacitados que no pueden acceder a las imágenes.

Ejemplo sencillo para incluir una imagen:

```

```

Como `` es una etiqueta vacía, no tiene etiqueta de cierre. No obstante, para que la página XHTML sea válida, todas las etiquetas deben estar cerradas. Como ya se explicó anteriormente, para cerrar una etiqueta vacía se incluyen los caracteres `</>` al final de la etiqueta.

HTML no impone ninguna restricción sobre el formato gráfico que se puede utilizar en las imágenes, por lo que en principio la etiqueta `` puede incluir cualquier formato gráfico existente. Sin embargo, si la imagen utiliza un formato poco habitual, todos o algunos navegadores no serán capaces de mostrar esa imagen.

La recomendación es utilizar uno de los tres siguientes formatos gráficos que entienden todos los navegadores modernos: GIF, JPG y PNG. El formato PNG presenta el inconveniente de que los navegadores obsoletos como Internet Explorer 6 no muestran correctamente las imágenes con transparencias de 24 bits.

El atributo `longdesc` no se utiliza de forma habitual, pero permite indicar la URL en la que se puede encontrar más información sobre la imagen. Como el atributo `alt` sólo permite incluir descripciones de hasta 1024 caracteres, el atributo `longdesc` se emplea con las imágenes complejas que necesitan mucha información para ser descritas:


```



```

En el ejemplo anterior, las dos imágenes se encuentran en el mismo directorio del servidor (`/imagenes/`). Se trata de una estrategia habitual en la mayoría de sitios web: guardar todas las imágenes de contenido en un directorio especial independiente del resto de contenidos HTML. Además, el directorio siempre suele llamarse de la misma manera: `"imagenes"` o `"images"` en inglés.

Los atributos `width` y `height` se utilizan para indicar la anchura y altura con la que se muestran las imágenes, por lo que son los más contradictorios. Como ya se ha comentado, HTML estructura de forma correcta los contenidos de la página y CSS define el aspecto gráfico con el que se muestran los contenidos. En principio, la anchura y la altura con la que se muestra una imagen es parte de su aspecto gráfico, por lo que debería ser propio de CSS y no de XHTML.

Sin embargo, en la práctica no es viable establecer la anchura y altura de todas las imágenes de contenidos mediante CSS. Si el sitio web dispone de muchas imágenes, la sobrecarga de estilos diferentes que debería definir CSS sería contraproducente. Por este motivo, los atributos `width` y `height` son la excepción a la norma de que el código HTML no haga referencia al aspecto de los contenidos.

```



```

Si el valor del atributo `width` o `height` se indica mediante un número entero, el navegador supone que hace referencia a la unidad de medida píxel. Por tanto, en el ejemplo anterior, la primera foto se muestra con una anchura de 200 píxel y una altura de 350 píxel.

También es posible indicar la anchura y altura en forma de porcentaje. En este caso, el porcentaje hace referencia a la altura/anchura del elemento en el que está contenida la imagen. Si la imagen no se encuentra dentro de ningún otro elemento, hace referencia a la anchura/altura total de la página.

```
<div>
  
</div>
```

El ejemplo anterior mezcla los dos tipos de medidas que se pueden utilizar, para indicar que la foto tiene una anchura igual al 30% de la anchura del elemento `<div>` que la contiene y una altura de 350 píxel.

La anchura/altura con la que se muestra una imagen no tiene que coincidir obligatoriamente con la anchura/altura real de la imagen. Sin embargo, cuando estos valores no coinciden, las imágenes se muestran deformadas y el aspecto final es muy desagradable.

Si solamente se establece la altura de la imagen, el navegador calcula la anchura necesaria para que se mantenga la proporción de la imagen. De la misma forma, si sólo se establece la anchura de la imagen, el navegador calcula la altura que hace que la imagen se siga viendo con las mismas proporciones.

Ejercicio 10

Modificar la página de índice del portfolio de los ejercicios 6 y 7 para mostrar directamente las imágenes de los proyectos.



Figura 6.1 Nueva página del portfolio que muestra la imagen de cada uno de los proyectos

6.2. Mapas de imagen

Aunque el uso de los mapas de imagen se ha reducido drásticamente en los últimos años, aún se utilizan en algunos sitios especializados. Muchas agencias de viaje y sitios relacionados utilizan mapas geográficos para seleccionar el destino del viaje. La mayoría de mapas se realiza hoy en día mediante Flash, aunque algunos sitios siguen recurriendo a los mapas de imagen.

Un mapa de imagen permite definir diferentes zonas "pinchables" dentro de una imagen. El usuario puede pinchar sobre cada una de las zonas definidas y cada una de ellas puede apuntar a una URL diferente. Siguiendo el ejemplo anterior, una sola imagen que muestre un mapa de todos los continentes puede definir una zona diferente para cada continente. De esta forma, el usuario puede pinchar sobre la zona correspondiente a cada continente para que el navegador muestre la página que contiene los viajes disponibles a ese destino.

Las zonas o regiones que se pueden definir en una imagen se crean mediante rectángulos, círculos y polígonos. Para crear un mapa de imagen, en primer lugar se inserta la imagen original mediante la etiqueta ``. A continuación, se utiliza la etiqueta `<map>` para definir las zonas o regiones de la imagen. Cada zona se define mediante la etiqueta `<area>`.

<map>	Se emplea para definir mapas de imagen		
Atributos comunes	básicos, internacionalización y eventos	Tipo de elemento	Bloque y en línea
Atributos propios	<ul style="list-style-type: none"> <code>name = "texto"</code> - Nombre que identifica de forma única al mapa definido (es obligatorio indicar un nombre único) 		

<area>	Se emplea para definir las distintas áreas que forman un mapa de imagen		
Atributos comunes	básicos, internacionalización, eventos y foco	Tipo de elemento	Etiqueta vacía
Atributos propios	<ul style="list-style-type: none"> <code>href = "url"</code> - URL a la que se accede al pinchar sobre el área <code>nohref = "nohref"</code> - Se emplea para las áreas que no son seleccionables <code>shape = "default rect circle poly"</code> - Indica el tipo de área que se define (toda la imagen, rectangular, circular o poligonal) <code>coords = "lista de números"</code> - Se trata de una lista de números separados por comas que representan las coordenadas del área. Rectangular = X1,Y1,X2,Y2 (coordenadas X e Y del vértice superior izquierdo y coordenadas X e Y del vértice inferior derecho). Circular = X1,Y1,R (coordenadas X e Y del centro y radio del círculo). Poligonal = X1,Y1,X2,Y2,...,Xn,Yn (coordenadas de los vértices del polígono. Si las últimas coordenadas no son iguales que las primeras, se cierra automáticamente el polígono uniendo ambos vértices) 		

Si una imagen utiliza un mapa de imagen, debe indicarlo mediante el atributo `usemap`. El valor del atributo debe ser el nombre del mapa de imagen definido en otra parte del mismo documento HTML:

```

...
<map name="continentes">
  ...
</map>
```

Las áreas se definen mediante el atributo `shape` que indica el tipo de área y `coords` que es una lista de coordenadas cuyo significado depende del tipo de área definido. El enlace de cada área se define mediante el atributo `href`, con la misma sintaxis y significado que para los enlaces normales.

El siguiente ejemplo muestra una imagen sencilla que contiene cuatro figuras geométricas:

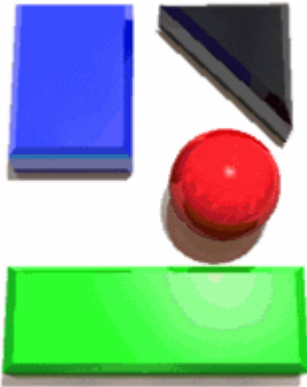


Figura 6.2 Ejemplo de imagen que incluye un mapa de imagen

Utilizando un círculo, dos rectángulos y un polígono se pueden definir fácilmente cuatro zonas *pinchables* en la imagen mediante el siguiente código HTML:

```


<map name="mapa_zonas">
  <area shape="rect" coords="20,25,84,113" href="rectangulo.html" />
  <area shape="polygon" coords="90,25,162,26,163,96,89,25,90,24" href="triangulo.html" />
  <area shape="circle" coords="130,114,29" href="circulo.html" />
  <area shape="rect" coords="19,156,170,211" href="mailto:rectangulo@direccion.com" />
  <area shape="default" nohref="nohref" />
</map>
```

6.3. Objetos

Además de las imágenes, HTML permite incluir en las páginas web otros elementos mucho más complejos, como *applets* de Java y vídeos en formato QuickTime o Flash. La mayoría de este tipo de contenidos no los interpreta el navegador directamente, sino que hace uso de pequeños programas llamados *plugins* y que se encargan de tratar con este tipo de elementos complejos.

La etiqueta `<object>` es la que permite "*embeber*" o incluir en las páginas HTML cualquier tipo de contenido complejo:

<object>	Se emplea para embeber objetos en los documentos		
Atributos comunes	básicos, internacionalización y eventos	Tipo de elemento	Bloque y en línea
Atributos propios	<ul style="list-style-type: none"> • <code>data = "url"</code> - Indica la URL de los datos que utiliza el objeto • <code>classid</code>, <code>codebase</code>, <code>codetype</code> - Información específica que depende del tipo de objeto • <code>type</code> - Indica el tipo de contenido de los datos • <code>height = "unidad_de_medida"</code> - Indica la altura con la que se debe mostrar el objeto • <code>width = "unidad_de_medida"</code> - Indica la anchura con la que se debe mostrar el objeto 		

El atributo `data` se emplea para indicar la URL del recurso que se va a incluir. El atributo `type` indica el tipo de contenido de los datos del objeto. Los posibles valores de `type` están estandarizados y coinciden con los del atributo `type` de la etiqueta `<a>` que se explicó anteriormente.

El propio estándar de HTML incluye ejemplos de uso de esta etiqueta. Incluir un vídeo en formato MPEG:

```
<object data="PlanetaTierra.mpeg" type="application/mpeg" />
```

También se pueden incluir varias versiones alternativas de un mismo contenido. Así, si el navegador no es capaz de interpretar el formato por defecto, puede optar por cualquiera de los otros formatos alternativos:

```
<object title="La Tierra vista desde el espacio" classid="http://www.observer.mars/
TheEarth.py">
  <!-- Formato alternativo en forma de vídeo -->
  <object data="PlanetaTierra.mpeg" type="application/mpeg">
    <!-- Otro formato alternativo mediante una imagen GIF -->
    <object data="PlanetaTierra.gif" type="image/gif">
      <!-- Si el navegador no soporta ningún formato, se muestra el siguiente texto -->
      La <strong>Tierra</strong> vista desde el espacio.
    </object>
  </object>
</object>
```

A los objetos también se les puede pasar información adicional en forma de parámetros mediante la etiqueta `<param>`:

<param>	Se emplea para indicar el valor de los parámetros del objeto		
Atributos comunes	id	Tipo de elemento	Etiqueta vacía
Atributos propios	<ul style="list-style-type: none"> • <code>name = "texto"</code> - Indica el nombre del parámetro • <code>value = "texto"</code> - Indica el valor del parámetro 		

Las etiquetas `<param>` siempre se incluyen en el interior de las etiquetas `<object>`:

```
<object data="..." type="...">
  <param name="parametro1" value="40" />
  <param name="parametro2" value="20" />
  <param name="parametro3" value="texto de prueba" />
</object>
```

Uno de los principales inconvenientes de `<object>` es la forma de incluir vídeos en formato Flash en las páginas HTML. Si se utiliza el siguiente código:

```
<object data="nombre_video.swf" type="application/x-shockwave-flash"></object>
```

El elemento anterior es correcto desde el punto de vista técnico, pero provoca que algunos navegadores como Internet Explorer no visualicen el vídeo hasta que se ha descargado completamente. Si se trata de un vídeo largo, esta solución no es válida para el usuario.

Por este motivo, se utiliza una solución alternativa para incluir vídeos Flash en las páginas HTML: el uso de la etiqueta `<embed>`. Aunque esta solución funciona correctamente, no se trata de una solución válida desde el punto de vista del estándar de XHTML, por lo que las páginas que incluyan esta solución no pasarán correctamente el proceso de validación.

<embed>	Se emplea para embeber objetos en los documentos		
Atributos comunes	básicos, internacionalización y eventos	Tipo de elemento	Bloque
Atributos propios	<ul style="list-style-type: none"> • <code>src = "url"</code> - Indica la URL del archivo u objeto que se incluye en la página • <code>type = "tipo_de_contenido"</code> - Indica el tipo de contenido del objeto (flash, quicktime, java, etc.) • <code>height = "unidad_de_medida"</code> - Indica la altura con la que se debe mostrar el objeto • <code>width = "unidad_de_medida"</code> - Indica la anchura con la que se debe mostrar el objeto 		

Este es el motivo por el que los sitios web más populares de vídeos en formato Flash proporcionan un código similar al siguiente para incluir sus vídeos en las páginas HTML:

```
<object width="425" height="350">
  <param name="movie" value="http://www.youtube.com/v/MsH0rBWCYjs"></param>
  <param name="wmode" value="transparent"></param>
  <embed src="http://www.youtube.com/v/MsH0rBWCYjs" type="application/
x-shockwave-flash" wmode="transparent" width="425" height="350"></embed>
</object>
```

Una vez más, se debe tener en cuenta que la solución anterior de utilizar la etiqueta `<embed>` es correcta desde el punto de vista del usuario (no tiene que esperar a que el vídeo se descargue completamente para poder verlo) pero no es una solución técnicamente válida, ya que la etiqueta `<embed>` no es parte del estándar XHTML.

Esta página se ha dejado vacía a propósito

Capítulo 7.

Tablas

Desde sus primeras versiones, HTML incluyó el soporte para crear tablas de datos en las páginas web. Además de ser sencillo, el modelo definido por HTML es muy flexible y bastante completo.

Las tablas en HTML utilizan los mismos conceptos de filas, columnas, cabeceras y títulos que los que se utilizan en cualquier otro entorno de publicación de documentos:

Cursos de diseño gráfico				
Nombre	Horas	Plazas	Horario	
Introducción a XHTML	20	20	09:00 – 13:00	
CSS avanzado	40	15	16:00 – 20:00	
Taller de usabilidad	40	10	16:00 – 20:00	
Introducción a AJAX	60	20	08:30 – 12:30	

Figura 7.1 Partes que componen una tabla compleja

Las tablas de HTML puede contener elementos simples, agrupaciones de filas y de columnas, cabeceras y pies de tabla, subdivisiones, cabeceras múltiples y otros elementos complejos.

A pesar de que las tablas HTML son fáciles de comprender y utilizar, son uno de los elementos más polémicos de HTML. El problema de las tablas es que no siempre se utilizan adecuadamente. Aunque parezca obvio, las tablas se deben utilizar para mostrar información tabular.

Hasta hace unos años, las tablas también se utilizaban para definir la estructura de las páginas web. La cabecera de la página era una fila de una gran tabla, el pie de página era otra fila de esta tabla y la zona de contenidos estaba formada por varias columnas dentro de esa gran tabla.

Aunque algunos malos diseñadores siguen utilizando hoy en día las tablas para definir la estructura completa de las páginas web, se trata de una técnica obsoleta y nada recomendable. El motivo es que

se complica en exceso el código HTML y su mantenimiento es muy complejo. La solución correcta para definir la estructura de las páginas consiste en la utilización de hojas de estilos CSS.

7.1. Tablas básicas

Las tablas más sencillas de HTML se definen con tres etiquetas: `<table>` para crear la tabla, `<tr>` para crear cada fila y `<td>` para crear cada columna.

A continuación se muestra el código HTML de una tabla sencilla:

```
<html>
<head><title>Ejemplo de tabla sencilla</title></head>
<body>

<h1>Listado de cursos</h1>

<table>
<tr>
  <td><strong>Curso</strong></td>
  <td><strong>Horas</strong></td>
  <td><strong>Horario</strong></td>
</tr>

<tr>
  <td>CSS</td>
  <td>20</td>
  <td>16:00 - 20:00</td>
</tr>

<tr>
  <td>HTML</td>
  <td>20</td>
  <td>16:00 - 20:00</td>
</tr>

<tr>
  <td>Dreamweaver</td>
  <td>60</td>
  <td>16:00 - 20:00</td>
</tr>
</table>

</body>
</html>
```

Si se visualiza el código anterior en cualquier navegador, se obtiene una tabla como la que muestra la siguiente imagen:

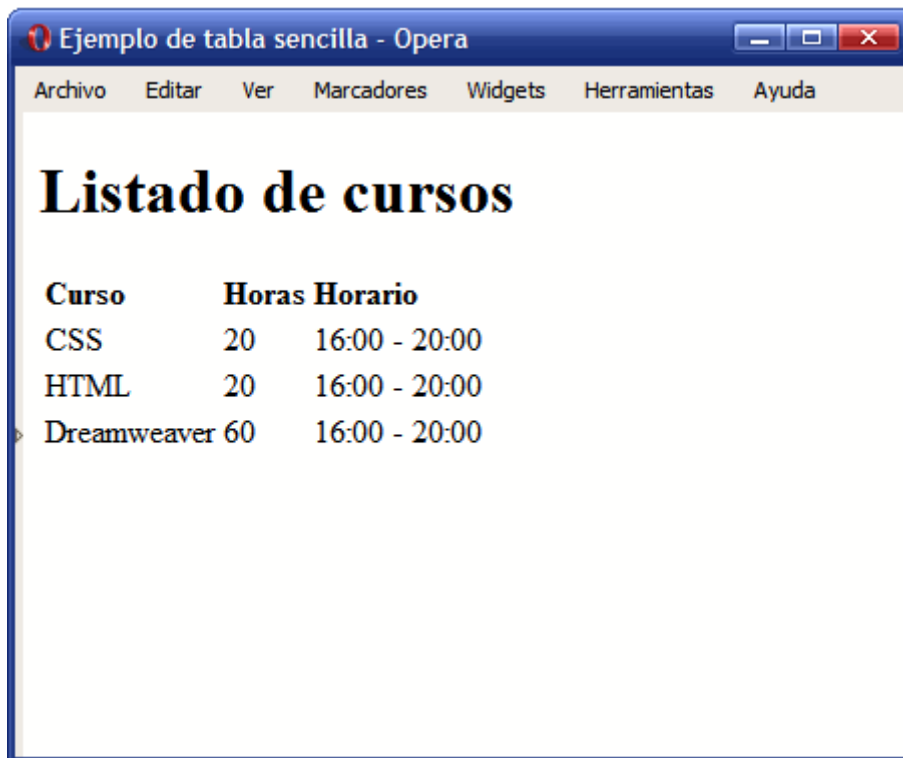


Figura 7.2 Ejemplo de tabla sencilla creada con las etiquetas `table`, `tr` y `td`

La etiqueta `<table>` encierra todas las filas y columnas de la tabla. Las etiquetas `<tr>` (del inglés "table row") definen cada fila de la tabla y encierran todas las columnas. Por último, la etiqueta `<td>` (del inglés "table data cell") define cada una de las columnas de las filas, aunque realmente HTML no define columnas sino *celdas de datos*.

Al definir una tabla, se debe pensar en primer lugar en las filas que la forman y a continuación en las columnas. El motivo es que HTML procesa primero las filas y por eso las etiquetas `<tr>` aparecen antes que las etiquetas `<td>`.

<code><table></code>	Se emplea para definir tablas de datos		
Atributos comunes	básicos, internacionalización y eventos	Tipo de elemento	Bloque
Atributos propios	<ul style="list-style-type: none"> <code>summary = "texto"</code> - Permite describir el contenido de la tabla (lo utilizan los buscadores y las personas discapacitadas) 		

<code><tr></code>	Se emplea para definir cada fila de las tablas de datos		
Atributos comunes	básicos, internacionalización y eventos	Tipo de elemento	Bloque
Atributos propios	-		

<td>	Se emplea para definir cada una de las celdas que forman las filas de una tabla, es decir, las columnas de la tabla		
Atributos comunes	básicos, internacionalización y eventos	Tipo de elemento	Bloque
Atributos propios	<ul style="list-style-type: none"> • abbr = "texto" - Permite describir el contenido de la celda (empleado sobre todo en los navegadores utilizados por personas discapacitadas) • headers = "lista_de_id" - Indica las celdas que actúan como cabeceras para esta celda (los títulos de las columnas y filas). Se indica como una lista de valores del atributo "id" de celdas • scope = "col, row, colgroup, rowgroup" - Indica las celdas para las que esta celda será su cabecera. Ej: scope="col" indica que esta celda es la cabecera de todas las demás celdas que están en la misma columna • colspan = "numero" - Número de columnas que ocupa esta celda • rowspan = "numero" - Número de filas que ocupa esta celda 		

De todos los atributos disponibles para las celdas, los más utilizados son **rowspan** y **colspan**, que se emplean para construir tablas complejas como las que se ven más adelante. Entre los demás atributos, sólo se utiliza de forma habitual el atributo **scope**, sobre todo con las celdas de cabecera que se ven a continuación.

Normalmente, algunas de las celdas de la tabla se utilizan como cabecera de las demás celdas de la fila o de la columna. En este caso, HTML define la etiqueta **<th>** (del inglés "table header cell") para indicar que una celda es cabecera de otras celdas.

<th>	Se emplea para definir las celdas que son cabecera de una fila o de una columna de la tabla		
Atributos comunes	básicos, internacionalización y eventos	Tipo de elemento	Bloque
Atributos propios	<ul style="list-style-type: none"> • abbr = "texto" - Permite describir el contenido de la celda (empleado sobre todo en los navegadores utilizados por personas discapacitadas) • headers = "lista_de_id" - Indica las celdas que actúan como cabeceras para esta celda (los títulos de las columnas y filas). Se indica como una lista de ID de celdas • scope = "col, row, colgroup, rowgroup" - Indica las celdas para las que esta celda será su cabecera. Ej: scope="col" indica que esta celda es la cabecera de todas las demás celdas que están en la misma columna • colspan = "numero" - Número de columnas que ocupa esta celda • rowspan = "numero" - Número de filas que ocupa esta celda 		

Los atributos de la etiqueta `<th>` son idénticos que los atributos definidos para la etiqueta `<td>`. En este caso, el atributo más utilizado es `scope`, que permite indicar si la celda es cabecera de la fila o de la columna (`<th scope="row">` y `<th scope="col">` respectivamente).

Por otra parte, HTML define la etiqueta `<caption>` para establecer la leyenda o título de una tabla. La etiqueta debe colocarse inmediatamente después de la etiqueta `<table>` y cada tabla sólo puede incluir una etiqueta `<caption>`.

<caption>	Se emplea para definir la leyenda o título de una tabla		
Atributos comunes	básicos, internacionalización y eventos	Tipo de elemento	En línea
Atributos propios	-		

Ejercicio 11

Determinar el código HTML necesario para crear la tabla que se muestra en la siguiente imagen:



The screenshot shows a browser window titled "Ejemplo de tabla sencilla - Opera". The browser's menu bar includes "Archivo", "Editar", "Ver", "Marcadores", "Widgets", "Herramientas", and "Ayuda". The main content area displays the heading "Su pedido" above a table. The table has four columns: "Nombre producto", "Precio unitario", "Unidades", and "Subtotal". The rows are as follows:

Nombre producto	Precio unitario	Unidades	Subtotal
Reproductor MP3 (80 GB)	192.02	1	192.02
Fundas de colores	2.50	5	12.50
Reproductor de radio & control remoto	12.99	1	12.99
TOTAL	-	7	207.51

Figura 7.3 Tabla sencilla con celdas de cabecera

Utilizar las celdas de cabecera donde sea necesario y añadir todos los atributos posibles.

Ejercicio 12

Determinar el código HTML necesario para crear la tabla que se muestra en la siguiente imagen. Utilizar las celdas de cabecera donde sea necesario y añadir todos los atributos posibles.



Figura 7.4 Tabla con los resultados de una búsqueda

Las tablas complejas suelen disponer de una estructura irregular que junta varias columnas para formar una columna ancha o une varias filas para formar una fila más alta que las demás. Para fusionar filas o columnas, se utilizan los atributos `rowspan` y `colspan` respectivamente.

La siguiente imagen muestra una tabla compleja que ha fusionado dos columnas simples para formar una columna más ancha:

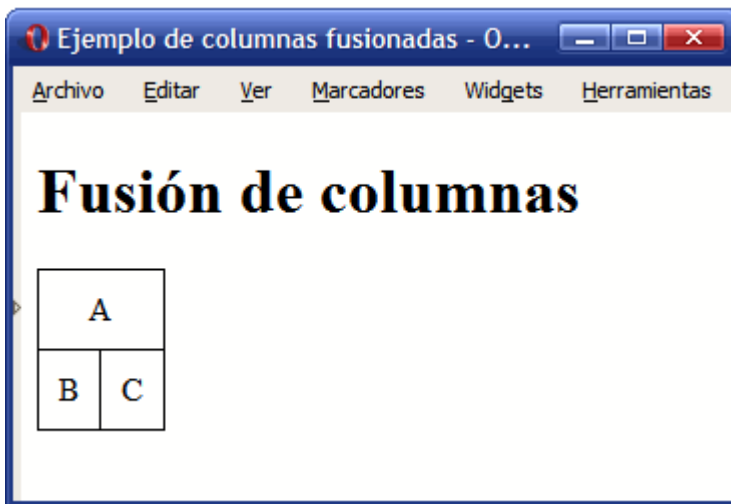


Figura 7.5 Ejemplo sencillo de fusión de columnas

Para obtener una tabla como la de la imagen anterior, se debe utilizar el siguiente código:

```
<table>
<tr>
  <td colspan="2">A</td>
</tr>

<tr>
  <td>B</td>
  <td>C</td>
</tr>
</table>
```

La primera fila de la tabla está formada sólo por una columna, mientras que la segunda fila está formada por dos columnas. En principio, podría pensarse en utilizar el siguiente código HTML para definir la tabla:

```
<table>
<tr>
  <td>A</td>
</tr>

<tr>
  <td>B</td>
  <td>C</td>
</tr>
</table>
```

Sin embargo, si se utiliza el código anterior, el navegador visualiza de forma incorrecta la tabla, ya que las tablas en HTML deben disponer de una estructura regular. En otras palabras, todas las filas de una tabla HTML deben tener el mismo número de columnas. Por lo tanto, si se quieren mostrar menos columnas en una fila, se fusionan mediante el atributo `colspan`, que indica el número de columnas simples que va a ocupar una determinada celda.

En el ejemplo anterior, la celda de la primera fila debe ocupar el espacio de dos columnas simples, por lo que el código HTML debe ser `<td colspan="2">A</td>`.

De forma equivalente, si se quiere diseñar una tabla HTML que fusiona filas como la de la siguiente imagen:



Figura 7.6 Ejemplo sencillo de fusión de filas

El código HTML que se debe utilizar para obtener la tabla de la imagen anterior es:

```
<table>
<tr>
  <td>A</td>
  <td rowspan="2">B</td>
</tr>

<tr>
  <td>C</td>
</tr>
</table>
```

De forma análoga a la fusión de columnas del ejemplo anterior, la fusión de filas debe indicarse de forma especial. Como las tablas HTML tienen que ser regulares, todas las columnas deben tener el mismo número de filas. Así, si en el ejemplo anterior se utilizara el siguiente código:

```
<table>
<tr>
  <td>A</td>
  <td>B</td>
</tr>

<tr>
  <td>C</td>
</tr>
</table>
```


La tabla anterior no se visualizaría correctamente. Como la segunda columna de la tabla ocupa el espacio de las dos filas, el código HTML debe indicar claramente que esa celda va a ocupar dos filas, de manera que todas las columnas de la tabla cuenten con el mismo número de filas.

Utilizando los atributos `rowspan` y `colspan`, es posible diseñar tablas tan complejas como las que se muestran en los siguientes ejemplos.

Ejemplo de fusión de múltiples columnas:

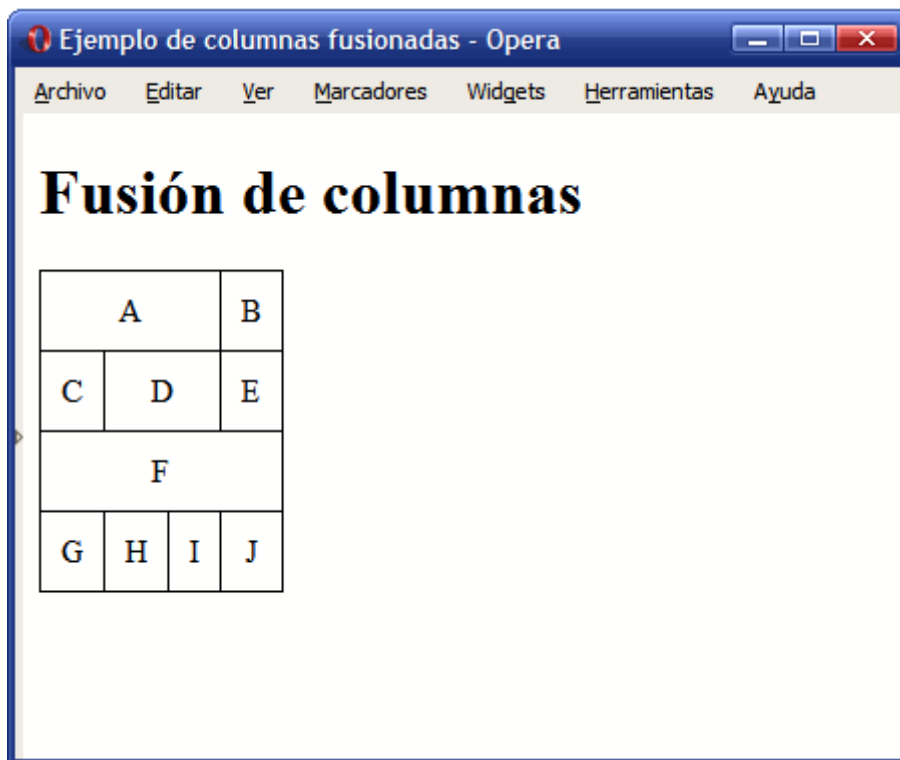


Figura 7.7 Ejemplo complejo de fusión de columnas

El código HTML necesario para fusionar las columnas de la tabla anterior se muestra a continuación:

```
<html>
<head><title>Ejemplo de columnas fusionadas</title></head>
<body>

<h1>Fusión de columnas</h1>

<table>
<tr>
  <td colspan="3">A</td>
  <td>B</td>
</tr>

<tr>
  <td>C</td>
  <td colspan="2">D</td>
  <td>E</td>
```

```

</tr>

<tr>
  <td colspan="4">F</td>
</tr>

<tr>
  <td>G</td>
  <td>H</td>
  <td>I</td>
  <td>J</td>
</tr>
</table>

</body>
</html>

```

Ejemplo de fusión de múltiples filas:

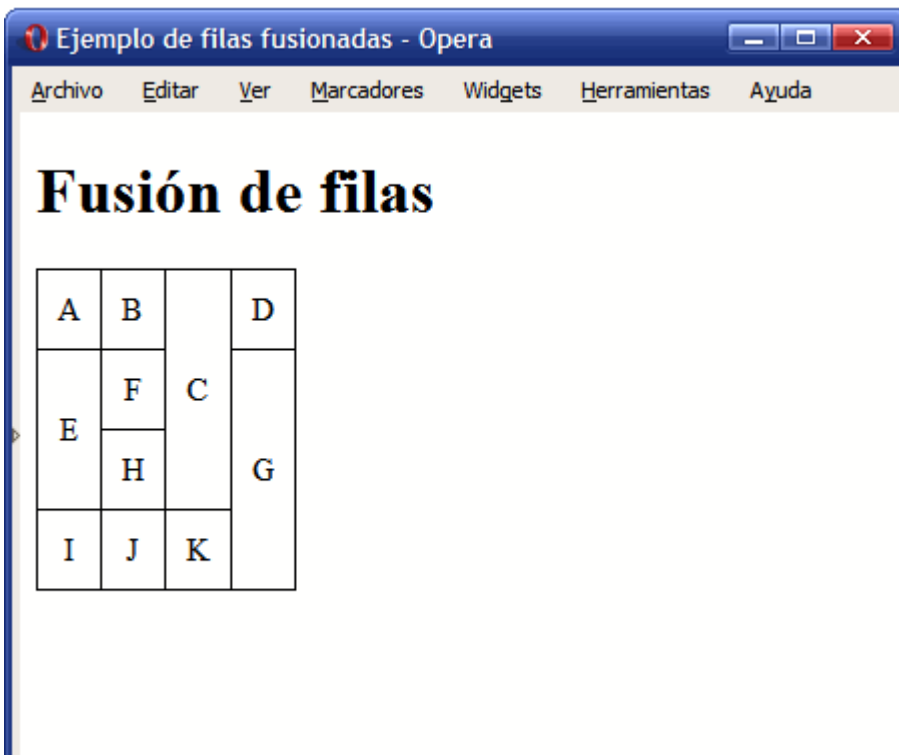


Figura 7.8 Ejemplo complejo de fusión de filas

El código HTML necesario para fusionar las filas de la tabla anterior se muestra a continuación:

```

<html>
<head><title>Ejemplo de filas fusionadas</title></head>
<body>

<h1>Fusión de filas</h1>

<table>

```

```
<tr>
  <td>A</td>
  <td>B</td>
  <td rowspan="3">C</td>
  <td>D</td>
</tr>
<tr>
  <td rowspan="2">E</td>
  <td>F</td>
  <td rowspan="3">G</td>
</tr>
<tr>
  <td>H</td>
</tr>
<tr>
  <td>I</td>
  <td>J</td>
  <td>K</td>
</tr>
</table>

</body>
</html>
```

Ejercicio 13

Determinar el código HTML necesario para crear la tabla que se muestra en la siguiente imagen:

Comparativa de reproductores MP3

Tabla comparativa de las características técnicas de los reproductores MP3







	 MP3 mini			 MP3 grande	
Capacidad de almacenamiento	4GB (1.000 canciones)	8GB (2.000 canciones)	16GB (4.000 canciones)	30GB (7.500 canciones)	80GB (20.000 canciones)
Colores					
Pantalla	LCD de 3 cm (diagonal) con retroiluminación			LCD de 6 cm (diagonal) con retroiluminación	
Tiempo de carga	Unas 3 horas			Unas 4 horas	
				Unas 2 horas para alcanzar el 80% de la capacidad	

Figura 7.9 Ejemplo de tabla con una estructura compleja

Emplear las etiquetas `<table>`, `<tr>`, `<td>`, `<th>`, `<caption>` y los atributos `colspan`, `rowspan`, `abbr`, `scope`, `summary`.

7.2. Tablas avanzadas

Algunas tablas complejas están formadas por más elementos que filas y celdas de datos. Así, es común que las tablas más avanzadas dispongan de una sección de cabecera, una sección de pie y varias secciones de datos. Además, también es posible agrupar varias columnas de forma lógica para poder aplicar estilos similares a un determinado grupo de columnas.

Un ejemplo clásico de tablas avanzadas es el de las tablas utilizadas en contabilidad, como por ejemplo la tabla que muestra el balance de una empresa:

CONSOLIDATED BALANCE SHEET	2008			
	March	June	September	December
Non-current assets	87.249	87.126	90.426	91.269
Intangible assets	21.810	21.145	20.986	20.758
Goodwill	17.914	19.660	21.828	21.739
Property, plant and equipment and investment property	33.245	32.332	33.428	33.888
Long-term financial assets and other non-current assets	5.723	5.687	5.981	6.183
Deferred tax assets	8.557	8.303	8.202	8.702
Current assets	18.042	17.979	19.128	17.713
Inventories	1.154	1.134	1.052	1.012
Trade and other receivables	9.244	9.495	9.709	9.666
Current tax receivable	1.288	1.565	1.468	1.555
Short-term financial investments	1.877	1.803	1.788	1.679
Cash and cash equivalents	4.468	3.557	5.101	3.792
Non-current assets classified as held for sale	11	425	9	9
Total Assets = Total Equity and Liabilities	105.291	105.106	109.554	108.982
Equity	15.714	15.072	19.185	20.001
Equity attributable to equity holders of the parent	11.932	12.085	16.397	17.178
Minority interest	3.782	2.987	2.788	2.823
Non-current liabilities	54.053	66.406	63.908	62.644,0
Long-term financial debt	41.665	54.263	51.647	50.675
Deferred tax liabilities	4.868	4.617	4.727	4.700
Long-term provisions	6.466	6.507	6.545	6.287
Other long-term liabilities	1.054	1.020	988	982
Current liabilities	35.523	23.628	26.462	26.337,0
Short-term financial debt	19.507	7.466	8.975	8.382
Trade and other payables	8.792	8.259	8.782	8.533
Current tax payable	2.007	2.324	2.529	2.841
Short-term provisions and other liabilities	5.218	5.212	6.176	6.580
Liabilities associated with non-current assets classified as held for sale	0	367	0	0
Financial Data				
Net Financial Debt (1)	53.510	54.922	52.239	52.145

Figura 7.10 Ejemplo de tabla compleja correspondiente al balance de una empresa

Las partes que componen las tablas complejas se definen mediante las etiquetas `<thead>`, `<tbody>` y `<tfoot>`. La cabecera de la tabla se define con la etiqueta `<thead>`, el pie de la tabla se define mediante `<tfoot>` y cada sección de datos se define con una etiqueta `<tbody>`.

<code><thead></code> <code><tbody></code> <code><tfoot></code>	Se emplean para agrupar varias filas en una cabecera (thead) un pie (tfoot) o una sección (tbody) de una tabla		
Atributos comunes	básicos, internacionalización y eventos	Tipo de elemento	Bloque
Atributos propios	-		

Cada tabla puede contener solamente una cabecera y un pie, pero puede incluir un número ilimitado de secciones. Si se define una cabecera y/o un pie, las etiquetas `<thead>` y/o `<tfoot>` deben colocarse inmediatamente antes que cualquier etiqueta `<tbody>`.

La siguiente imagen muestra una tabla avanzada con cabecera, pie y una sección de datos:

Análisis de ventas anuales				
AÑO	Expansión de ventas			
	Producto A	Producto B	Producto C	Producto D
N-3	-	-	-	-
N-2	3	5	8	4
N-1	4	4	7	3
N	5	7	6	2
AÑO	Expansión de ventas			
	Producto A	Producto B	Producto C	Producto D

Figura 7.11 Ejemplo de tabla avanzada con cabecera, pie y secciones

El código HTML necesario para crear la tabla de la imagen anterior hace uso de las etiquetas `<thead>`, `<tbody>` y `<tfoot>`:

```

<html>
<head><title>Ejemplo de tabla avanzada</title></head>
<body>

<h3>Análisis de ventas</h3>

<table summary="Análisis de ventas anuales">
  <caption>Análisis de ventas anuales</caption>
  <thead>
    <tr>
      <th rowspan="2" scope="col">AÑO</th>
      <th colspan="4" scope="col">Expansión de ventas</th>
    </tr>
    <tr>
      <th scope="col">Producto A</th>
      <th scope="col">Producto B</th>
      <th scope="col">Producto C</th>
  
```

```

        <th scope="col">Producto D</th>
    </tr>
</thead>

<tfoot>
    <tr>
        <th rowspan="2" scope="col">AÑO</th>
        <th scope="col">Producto A</th>
        <th scope="col">Producto B</th>
        <th scope="col">Producto C</th>
        <th scope="col">Producto D</th>
    </tr>
    <tr>
        <th colspan="4" scope="col">Expansión de ventas</th>
    </tr>
</tfoot>

<tbody>
    <tr>
        <th scope="row">N-3</th><td>-</td><td>-</td><td>-</td><td>-</td>
    </tr>
    <tr>
        <th scope="row">N-2</th><td>3</td><td>5</td><td>8</td><td>4</td>
    </tr>
    <tr>
        <th scope="row">N-1</th><td>4</td><td>4</td><td>7</td><td>3</td>
    </tr>
    <tr>
        <th scope="row">N</th><td>5</td><td>7</td><td>6</td><td>2</td>
    </tr>
</tbody>
</table>

</body>

</html>

```

Aunque al principio resulta extraño, el elemento `<tfoot>` siempre se escribe antes que cualquier elemento `<tbody>` en el código HTML. De hecho, si la etiqueta `<tfoot>` aparece después de un elemento `<tbody>`, la página no se considera válida.

La etiqueta `<tbody>` permite realizar agrupaciones de filas, pero en ocasiones se necesitan agrupar columnas. Aunque su uso no es muy común, HTML define dos etiquetas similares para agrupar columnas: `<col>` y `<colgroup>`.

La etiqueta `<col>` se utiliza para asignar los mismos atributos a varias columnas de forma simultánea. De esta forma, la etiqueta `<col>` no agrupa columnas, sino que sólo asigna atributos comunes a varias columnas.

La siguiente imagen muestra una tabla que hace uso de la etiqueta `<col>`:



Análisis de ventas anuales				
AÑO	Producto A	Producto B	Producto C	Producto D
N-3	-	-	-	-
N-2	3	5	8	4
N-1	4	4	7	3
N	5	7	6	2

Figura 7.12 Ejemplo de tabla avanzada que usa la etiqueta `col`

El código HTML necesario para crear la tabla anterior se muestra a continuación:

```
<table summary="Análisis de ventas anuales">
  <caption>Análisis de ventas anuales</caption>

  <col style="width:10%;" />
  <col style="width:30%;" />

  <thead>
    <tr>
      <th scope="col">AÑO</th>
      <th scope="col">Producto A</th>
      <th scope="col">Producto B</th>
      <th scope="col">Producto C</th>
      <th scope="col">Producto D</th>
    </tr>
  </thead>

  <tbody>
    <tr>
      <th scope="row">N-3</th><td>-</td><td>-</td><td>-</td><td>-</td>
    </tr>
    <tr>
      <th scope="row">N-2</th><td>3</td><td>5</td><td>8</td><td>4</td>
    </tr>
    <tr>
      <th scope="row">N-1</th><td>4</td><td>4</td><td>7</td><td>3</td>
    </tr>
    <tr>
      <th scope="row">N</th><td>5</td><td>7</td><td>6</td><td>2</td>
    </tr>
  </tbody>
</table>
```



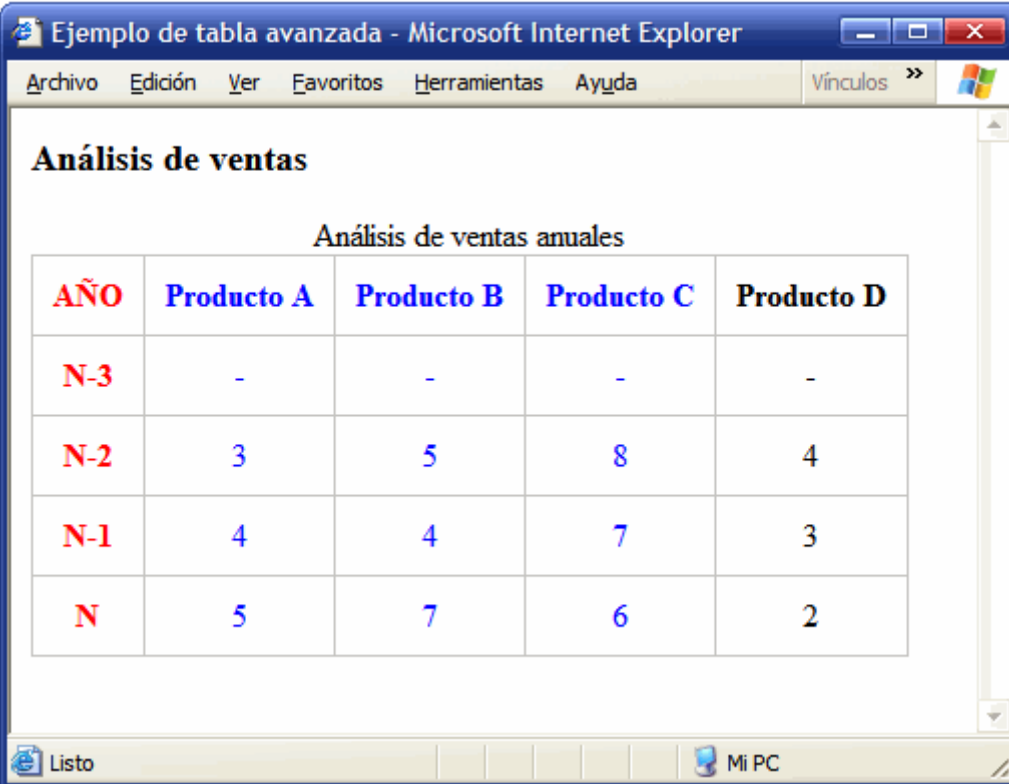
```

    <th scope="row">N-1</th><td>4</td><td>4</td><td>7</td><td>3</td>
  </tr>
  <tr>
    <th scope="row">N</th><td>5</td><td>7</td><td>6</td><td>2</td>
  </tr>
</tbody>
</table>

```

Por otra parte, la etiqueta `<colgroup>` se emplea para agrupar de forma estructural varias columnas de la tabla. La forma habitual de indicar el número de columnas que abarca la agrupación es utilizar el atributo `span`, que establece el número de columnas de cada agrupación.

La siguiente imagen muestra una tabla avanzada con una agrupación de columnas realizada con la etiqueta `<colgroup>`:



Análisis de ventas anuales				
AÑO	Producto A	Producto B	Producto C	Producto D
N-3	-	-	-	-
N-2	3	5	8	4
N-1	4	4	7	3
N	5	7	6	2

Figura 7.13 Ejemplo de tabla avanzada que usa la etiqueta `colgroup`

El código HTML necesario para crear la tabla anterior se muestra a continuación:

```

<table summary="Análisis de ventas anuales">
  <caption>Análisis de ventas anuales</caption>

  <colgroup span="1" style="color:red;" />
  <colgroup span="3" style="color:blue;" />

  <thead>
    <tr>
      <th scope="col">AÑO</th>
      <th scope="col">Producto A</th>

```

```
<th scope="col">Producto B</th>
<th scope="col">Producto C</th>
<th scope="col">Producto D</th>
</tr>
</thead>

<tbody>
<tr>
<th scope="row">N-3</th><td>-</td><td>-</td><td>-</td><td>-</td>
</tr>
<tr>
<th scope="row">N-2</th><td>3</td><td>5</td><td>8</td><td>4</td>
</tr>
<tr>
<th scope="row">N-1</th><td>4</td><td>4</td><td>7</td><td>3</td>
</tr>
<tr>
<th scope="row">N</th><td>5</td><td>7</td><td>6</td><td>2</td>
</tr>
</tbody>
</table>
```

El uso de las etiquetas `<col>` y `<colgroup>` no está muy extendido, debido a que la mayoría de navegadores no soportan muchas de sus funcionalidades.

Capítulo 8.

Formularios

HTML es un lenguaje de marcado cuyo propósito principal consiste en estructurar los contenidos de los documentos y páginas web. Sin embargo, HTML también incluye elementos para crear aplicaciones web. El estándar HTML/XHTML permite crear formularios para que los usuarios interactúen con las aplicaciones web.

Los años transcurridos desde la publicación de los estándares de HTML y XHTML ha provocado que no estén disponibles todos los elementos utilizados por los formularios más avanzados y modernos. No obstante, HTML/XHTML incluye los suficientes elementos de formulario para crear desde los formularios sencillos que utilizan los buscadores hasta los formularios complejos de las aplicaciones más avanzadas.

8.1. Formularios básicos

Los formularios más sencillos se pueden crear utilizando solamente dos etiquetas: `<form>` y `<input>`. Si se considera el formulario que muestra la siguiente imagen:



Figura 8.1 Formulario sencillo definido con las etiquetas form e input

El código HTML necesario para definir el formulario anterior se muestra a continuación:

```
<html>
<head><title>Ejemplo de formulario sencillo</title></head>
```

```

<body>

<h3>Formulario muy sencillo</h3>

<form action="http://www.librosweb.es/maneja_formulario.php" method="post">
  Escribe tu nombre:
  <input type="text" name="nombre" value="" />

  <br/>

  <input type="submit" value="Enviar" />
</form>

</body>
</html>

```

La etiqueta `<form>` encierra todos los contenidos del formulario (botones, cuadros de texto, listas desplegables) y la etiqueta `<input>` permite definir varios tipos diferentes de elementos (botones y cuadros de texto).

<form>	Se emplea para insertar un formulario en la página		
Atributos comunes	básicos, internacionalización y eventos	Tipo de elemento	Bloque
Atributos propios	<ul style="list-style-type: none"> • <code>action = "url"</code> - Indica la URL que se encarga de procesar los datos del formulario • <code>method = "POST o GET"</code> - Método HTTP empleado al enviar el formulario • <code>enctype = "application/x-www-form-urlencoded o multipart/form-data"</code> - Tipo de codificación empleada al enviar el formulario al servidor (sólo se indica de forma explícita en los formularios que permiten adjuntar archivos) • <code>accept = "tipo_de_contenido"</code> - Lista separada por comas de todos los tipos de archivos aceptados por el servidor (sólo para los formularios que permiten adjuntar archivos) • Otros: <code>accept-charset</code>, <code>onsubmit</code>, <code>onreset</code> 		

La mayoría de formularios utilizan sólo los atributos `action` y `method`. El atributo `action` indica la URL de la aplicación del servidor que se encarga de procesar los datos introducidos por los usuarios. Esta aplicación también se encarga de generar la respuesta que muestra el navegador.

El atributo `method` establece la forma en la que se envían los datos del formulario al servidor. Este atributo hace referencia al método HTTP, por lo que no es algo propio de HTML. Los dos valores que se utilizan en los formularios son `GET` y `POST`. De esta forma, casi todos los formularios incluyen el atributo `method="get"` o el atributo `method="post"`.

Al margen de otras diferencias técnicas, el método `POST` permite el envío de mucha más información que el método `GET`. En general, el método `GET` admite como máximo el envío de unos 500 bytes de in-

formación. La otra gran limitación del método `GET` es que no permite el envío de archivos adjuntos con el formulario. Además, los datos enviados mediante `GET` se ven en la barra de direcciones del navegador (se añaden al final de la URL de la página), mientras que los datos enviados mediante `POST` no se pueden ver tan fácilmente.

Si no sabes que método elegir para un formulario, existe una regla general que dice que el método `GET` se debe utilizar en los formularios que no modifican la información (por ejemplo en un formulario de búsqueda). Por su parte, el método `POST` se debería utilizar cuando el formulario modifica la información original (insertar, modificar o borrar alguna información).

El ejemplo más común de formulario con método `GET` es el de los buscadores. Si realizas una búsqueda con tu buscador favorito, verás que las palabras que has introducido en tu búsqueda aparecen como parte de la URL de la página de resultados.

Del resto de atributos de la etiqueta `<form>`, el único que se utiliza ocasionalmente es `enctype`. Como se explica más adelante, este atributo es imprescindible en los formularios que permiten adjuntar archivos.

8.2. Elementos de formulario

Los elementos de formulario como botones y cuadros de texto también se denominan "*campos de formulario*" y "*controles de formulario*". La mayoría de controles se crean con la etiqueta `<input>`, por lo que su definición formal y su lista de atributos es muy extensa:

<input>	Se emplean para insertar un control en un formulario		
Atributos comunes	básicos, internacionalización, eventos y foco	Tipo de elemento	En línea y etiqueta vacía
Atributos propios	<ul style="list-style-type: none"> • <code>type = "text password checkbox radio submit reset file hidden image button"</code> - Indica el tipo de control que se incluye en el formulario • <code>name = "texto"</code> - Asigna un nombre al control (es imprescindible para que el servidor pueda procesar el formulario) • <code>value = "texto"</code> - Valor inicial del control • <code>size = "unidad_de_medida"</code> - Tamaño inicial del control (para los campos de texto y de password se refiere al número de caracteres, en el resto de controles se refiere a su tamaño en píxel) • <code>maxlength = "numero"</code> - Máximo número de caracteres para los controles de texto y de password • <code>checked = "checked"</code> - Para los controles checkbox y radiobutton permite indicar qué opción aparece pre-seleccionada • <code>disabled = "disabled"</code> - El control aparece deshabilitado y su valor no se envía al servidor junto con el resto de datos • <code>readonly = "readonly"</code> - El contenido del control no se puede modificar • <code>src = "url"</code> - Para el control que permite crear botones con imágenes, indica la URL de la imagen que se utilizará como botón de formulario • <code>alt = "texto"</code> - Descripción del control 		

A continuación se muestran ejemplos para los diez controles que se pueden crear con la etiqueta `<input>`.

8.2.1. Cuadro de texto

Se trata del elemento más utilizado en los formularios. En el caso más sencillo, se muestra un cuadro de texto vacío en el que el usuario puede escribir cualquier texto:

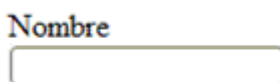


Figura 8.2 Ejemplo de etiqueta input (type=text)

A continuación se muestra el código HTML correspondiente al ejemplo anterior:

```
Nombre <br/>
<input type="text" name="nombre" value="" />
```

El atributo `type` diferencia a cada uno de los diez controles que se pueden crear con la etiqueta `<input>`. Para los cuadros de texto, su valor es `text`. El atributo `name` es el más importante en los campos del formulario. De hecho, si un campo no incluye el atributo `name`, sus datos no se envían al servidor. El valor que se indica en el atributo `name` es el nombre que utiliza la aplicación del servidor para obtener el valor de este campo de formulario.

Cuando el usuario pulsa el botón de envío del formulario, el navegador envía los datos a una aplicación del servidor para que procese la información y genere una respuesta adecuada. En el servidor, la aplicación que procesa los datos debe obtener en primer lugar toda la información introducida por el usuario. Para ello, utiliza el valor del atributo `name` para obtener los datos de cada control del formulario.

Como el valor del atributo `name` se utiliza en aplicaciones programadas, es esencial ponerse de acuerdo con el programador de la aplicación, no se debe modificar su valor sin modificar la aplicación y no se deben utilizar caracteres problemáticos en programación (espacios en blanco, acentos y caracteres como ñ o ç).

El atributo `value` se emplea para establecer el valor inicial del cuadro de texto. Si se crea un formulario para insertar datos, los cuadros de texto deberían estar vacíos. Por lo tanto, o no se añade el atributo `value` o se incluye con un valor vacío `value=""`. Si por el contrario se crea un formulario para modificar datos, lo lógico es que se muestren inicialmente los datos guardados en el sistema. En este caso, el atributo `value` incluirá el valor que se desea mostrar: `<input type="text" name="nombre" value="Juan Pérez" />`

Si no se especifica un tamaño, el navegador muestra el cuadro de texto con un tamaño predeterminado. El atributo `size` permite establecer el tamaño, en caracteres, con el que se muestra el cuadro de texto. Su uso es imprescindible en muchos formularios, en los que algunos campos como la dirección deben mostrar más caracteres de lo normal (`<input size="100" ...>`) y otros campos como el código postal deben mostrar menos caracteres de lo normal (`<input size="5" ...>`).

Además de controlar el tamaño con el que se muestra un cuadro de texto, también se puede limitar el tamaño del texto introducido. El atributo `maxlength` permite establecer el máximo número de caracteres que el usuario puede introducir en un cuadro de texto. Su uso es imprescindible para campos como el código postal, el número de la Seguridad Social y cualquier otro dato con formato predefinido y limitado.

Por último, el atributo `readonly` permite que el usuario pueda ver los contenidos del cuadro de texto pero no pueda modificarlos y el atributo `disabled` deshabilita un cuadro de texto de forma que el usuario no pueda modificarlo y además, el navegador no envía sus datos al servidor.

8.2.2. Cuadro de contraseña

La única diferencia entre este control y el cuadro de texto normal es que el texto que el usuario escribe en un cuadro de contraseña no se ve en la pantalla. En su lugar, los navegadores ocultan el texto utilizando asteriscos o círculos, por lo que es ideal para escribir contraseñas y otros datos sensibles.



Figura 8.3 Ejemplo de etiqueta input (type=password)

```
Contraseña <br/>  
<input type="password" name="contrasena" value="" />
```

Cambiando el valor del atributo `type` por `password` se transforma el cuadro de texto normal en un cuadro de contraseña. Todos los demás atributos se utilizan de la misma forma y tienen el mismo significado.

8.2.3. Checkbox

Los checkbox o "casillas de verificación" son controles de formulario que permiten al usuario seleccionar y deseleccionar opciones individualmente. Aunque en ocasiones se muestran varios checkbox juntos, cada uno de ellos es completamente independiente del resto. Por este motivo, se utilizan cuando el usuario puede activar y desactivar varias opciones relacionadas pero no excluyentes.

Puestos de trabajo buscados

Dirección

Técnico

Empleado

Figura 8.4 Ejemplo de etiqueta input (type=checkbox)

```
Puestos de trabajo buscados <br/>
<input name="puesto_directivo" type="checkbox" value="direccion"/> Dirección
<input name="puesto_tecnico" type="checkbox" value="tecnico"/> Técnico
<input name="puesto_empleado" type="checkbox" value="empleado"/> Empleado
```

El valor del atributo `type` para estos controles de formulario es `checkbox`. Como se muestra en el ejemplo anterior, el texto que se encuentra al lado de cada `checkbox` no se puede establecer mediante ningún atributo, por lo que es necesario añadirlo manualmente fuera del control del formulario. Si no se añade un texto al lado de la etiqueta `<input />` del `checkbox`, el usuario sólo ve un pequeño cuadrado sin ninguna información relativa a la finalidad de ese `checkbox`.

El valor del atributo `value`, junto con el valor del atributo `name`, es la información que llega al servidor cuando el usuario envía el formulario.

Si se quiere mostrar un `checkbox` seleccionado por defecto, se utiliza el atributo `checked`. Si el valor del atributo es `checked`, el `checkbox` se muestra seleccionado. En cualquier otro caso, el `checkbox` permanece sin seleccionar. Aunque resulta redundante que el nombre y el valor del atributo sean idénticos, es obligatorio indicarlo de esta forma porque los atributos en XHTML no pueden tener valores vacíos:

```
<input type="checkbox" checked="checked" ... /> Checkbox seleccionado por defecto
```

8.2.4. Radiobutton

Los controles de tipo `radiobutton` son similares a los controles de tipo `checkbox`, pero presentan una diferencia muy importante: son mutuamente excluyentes. Los `radiobutton` se utilizan cuando el usuario solamente puede escoger una opción entre las distintas opciones relacionadas que se le presentan. Cada vez que se selecciona una opción, automáticamente se deselectiona la otra opción que estaba seleccionaba.

Sexo

Hombre

Mujer

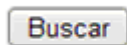
Figura 8.5 Ejemplo de etiqueta input (type=radio)

```
Sexo <br/>  
<input type="radio" name="sexo" value="hombre" checked="checked" /> Hombre  
<input type="radio" name="sexo" value="mujer" /> Mujer
```

El valor del atributo `type` para estos controles de formulario es `radio`. El atributo `name` se emplea para indicar los `radiobutton` que están relacionados. Por lo tanto, cuando varios `radiobutton` tienen el mismo valor en su atributo `name`, el navegador sabe que están relacionados y puede deseleccionar una opción del grupo de `radiobutton` cuando se seleccione otra opción.

8.2.5. Botón de envío de formulario

La mayoría de formularios dispone de un botón para enviar al servidor los datos introducidos por el usuario:

**Figura 8.6** Ejemplo de etiqueta input (type=submit)

```
<input type="submit" name="buscar" value="Buscar" />
```

El valor del atributo `type` para este control de formulario es `submit`. El navegador se encarga de enviar automáticamente los datos cuando el usuario pincha sobre este tipo de botón. El valor del atributo `value` es el texto que muestra el botón. Si no se establece el atributo `value`, el navegador muestra el texto predefinido `Enviar consulta`.

8.2.6. Botón de reseteo del formulario

Aunque su uso era muy popular hace unos años, la mayoría de formularios modernos ya no utilizan este tipo de botón. Se trata de un botón especial que borra todos los datos introducidos por el usuario y devuelve el formulario a su estado original:

**Figura 8.7** Ejemplo de etiqueta input (type=reset)

```
<input type="reset" name="limpiar" value="Borrar datos del formulario" />
```

El valor del atributo `type` para este control de formulario es `reset`. Cuando el usuario pulsa este botón, el navegador borra toda la información introducida y muestra el formulario en su estado original. Si el formulario no contenía originalmente ningún valor, el botón de `reset` lo vuelve a mostrar vacío. Si el formulario contenía información, el botón `reset` vuelve a mostrar la misma información original.

Como es habitual en los botones de formulario, el atributo `value` permite establecer el texto que muestra el botón. Si no se utiliza este atributo, el navegador muestra el texto predefinido del botón, que en este caso es `Restablecer`.

8.2.7. Ficheros adjuntos

Los formularios también permiten adjuntar archivos para subirlos al servidor. Aunque desde el punto de vista de HTML y del navegador no existe ninguna limitación sobre el número, tipo o tamaño total de los archivos que se pueden adjuntar, todos los servidores añaden restricciones por motivos de seguridad.

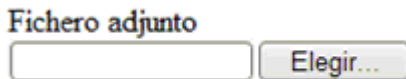


Figura 8.8 Ejemplo de etiqueta input (type=file)

```
Fichero adjunto
<input type="file" name="adjunto" />
```

El valor del atributo `type` para este control de formulario es `file`. El navegador se encarga de mostrar un cuadro de texto donde aparece el nombre del archivo seleccionado y un botón que permite navegar por los directorios y archivos del ordenador del usuario.

Si se incluye un control para adjuntar archivos, es obligatorio añadir el atributo `enctype` en la etiqueta `<form>` del formulario. El valor del atributo `enctype` debe ser `multipart/form-data`, por lo que la etiqueta `<form>` de los formularios que permiten adjuntar archivos siempre es:

```
<form action="..." method="post" enctype="multipart/form-data">
...
</form>
```

8.2.8. Campos ocultos

Los campos ocultos se emplean para añadir información oculta en el formulario:

*Los campos ocultos
no se ven en pantalla

Figura 8.9 Ejemplo de etiqueta input (type=hidden)

```
<input type="hidden" name="url_previa" value="/articulo/primero.html" />
```

El valor del atributo `type` para este control de formulario es `hidden`. Los campos ocultos no se muestran por pantalla, de forma que el usuario desconoce que el formulario los incluye. Normalmente los campos ocultos se utilizan para incluir información que necesita el servidor pero que no es necesario o no es posible que la establezca el usuario.

8.2.9. Botón de imagen

El aspecto de los botones de formulario se puede personalizar por completo, ya que incluso es posible utilizar una imagen como botón:



Figura 8.10 Ejemplo de etiqueta input (type=image)

```
<input type="image" name="enviar" src="accept.png" />
```

El valor del atributo `type` para este control de formulario es `image`. El atributo `src` indica la URL de la imagen que debe mostrar el navegador en lugar del botón normal.

Su principal ventaja es que permite personalizar por completo la estética de los botones y mostrarlos con un aspecto homogéneo en todos los navegadores. El principal inconveniente es que ralentiza la carga del formulario y que si se quiere modificar su aspecto, es necesario crear una nueva imagen.

8.2.10. Botón

Algunos formularios complejos necesitan botones más avanzados que los de enviar datos (`type="submit"`) y resetear el formulario (`type="reset"`). Por ese motivo, el estándar HTML/XHTML define un botón de tipo genérico:

**Figura 8.11** Ejemplo de etiqueta input (type=button)

```
<input type="button" name="guardar" value="Guardar Cambios" />
```

El valor del atributo `type` para este control de formulario es `button`. Si pruebas a pulsar un botón de este tipo, verás que el navegador no hace nada: no envía los datos al servidor y no borra los datos introducidos. Este tipo de botones sólo son útiles si se utilizan junto con el lenguaje de programación JavaScript. Si la página incluye código JavaScript, los botones de este tipo se pueden programar para que realicen cualquier tarea compleja cuando se pulsa sobre ellos.

Ejercicio 14

Determinar el código HTML necesario para crear el formulario que se muestra en la siguiente imagen:

Figura 8.12 Formulario con controles de varios tipos

1. Elegir el método más adecuado para el formulario ([GET](#) o [POST](#)) y cualquier otro atributo necesario.
2. La aplicación que se encarga de procesar el formulario se encuentra en la raíz del servidor, carpeta "[php](#)" y archivo "[insertar_cv.php](#)".
3. El nombre puede tener 30 caracteres como máximo, los apellidos 80 caracteres y la contraseña 10 caracteres como máximo.
4. Asignar los atributos adecuados al campo del DNI.
5. Por defecto, debe estar marcada la casilla de suscripción al boletín de novedades.

8.3. Formularios avanzados

Utilizando solamente las etiquetas `<form>` y `<input>` es posible diseñar la mayoría de formularios de las aplicaciones web. No obstante, HTML define algunos elementos adicionales para mejorar la estructura de los formularios creados.

La siguiente imagen muestra un formulario que agrupa sus elementos y añade etiquetas a cada campo para mejorar su estructura:

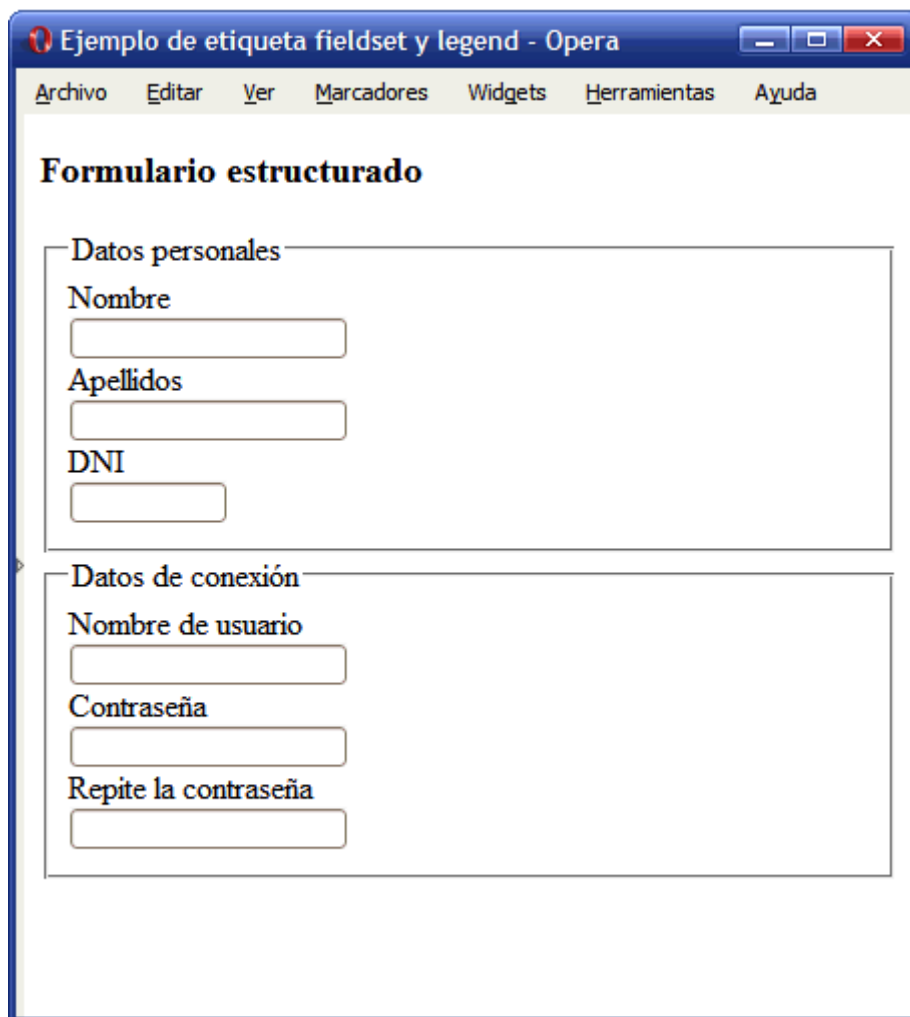


Figura 8.13 Ejemplo de uso de las etiquetas fieldset y legend

La etiqueta `<fieldset>` agrupa campos del formulario y la etiqueta `<legend>` asigna un nombre a cada grupo.

<fieldset>	Se emplea para agrupar de forma lógica varios campos de un formulario		
Atributos comunes	básicos, internacionalización y eventos	Tipo de elemento	Bloque
Atributos propios	-		

<legend>	Se emplea para definir el título o leyenda de un conjunto de campos de formulario agrupados con la etiqueta fieldset		
Atributos comunes	básicos, internacionalización y eventos	Tipo de elemento	En línea
Atributos propios	-		

A continuación se muestra el código HTML del formulario correspondiente a la imagen anterior y que hace uso de `<fieldset>` y `<legend>` para agrupar los campos del formulario:

```
<form action="maneja_formulario.php" method="post">
  <fieldset>
    <legend>Datos personales</legend>
    Nombre <br/>
    <input type="text" name="nombre" value="" />
    <br/>
    Apellidos <br/>
    <input type="text" name="apellidos" value="" />
    <br/>
    DNI <br/>
    <input type="text" name="dni" value="" size="10" maxlength="9" />
  </fieldset>

  <fieldset>
    <legend>Datos de conexión</legend>
    Nombre de usuario<br/>
    <input type="text" name="nombre" value="" maxlength="10" />
    <br/>
    Contraseña<br/>
    <input type="password" name="password" value="" maxlength="10" />
    <br/>
    Repite la contraseña<br/>
    <input type="password" name="password2" value="" maxlength="10" />
  </fieldset>
</form>
```

La etiqueta `<fieldset>` agrupa todos los controles de formulario a los que encierra. El navegador muestra por defecto un borde resaltado para cada agrupación. La etiqueta `<legend>` se incluye dentro de cada etiqueta `<fieldset>` y establece el título que muestra el navegador para cada agrupación de elementos.

Por otra parte, todos los controles de formulario salvo los botones presentan una carencia muy importante: no disponen de la opción de establecer el título o texto que se muestra junto al control. En el código HTML del ejemplo anterior, el nombre de cada campo se incluye en forma de texto normal, sin ninguna relación con el campo al que hace referencia.

Afortunadamente, el lenguaje HTML incluye una etiqueta denominada `<label>` y que se utiliza para establecer el título de cada campo del formulario. Su definición formal es la siguiente:

<label>	Se emplea para definir el título o leyenda de los campos definidos en un formulario		
Atributos comunes	básicos, internacionalización y eventos	Tipo de elemento	En línea
Atributos propios	<ul style="list-style-type: none"> • <code>for = "id_de_elemento"</code> - Indica el ID del campo del formulario para el que este elemento es su título • Otros: <code>accesskey</code>, <code>onfocus</code> y <code>onblur</code> 		

El único atributo que suele utilizarse con la etiqueta `<label>` es `for`, que indica el identificador (atributo `id`) del campo de formulario para el que esta etiqueta hace de título.

En el anterior ejemplo, el nombre de los campos de formulario se incluía mediante un texto normal:

```
Nombre <br/>
<input type="text" name="nombre" value="" />

Apellidos <br/>
<input type="text" name="apellidos" value="" />

DNI <br/>
<input type="text" name="dni" value="" size="10" maxlength="9" />
```

Utilizando la etiqueta `<label>`, cada campo de formulario puede disponer de su propio título:

```
<label for="nombre">Nombre</label> <br/>
<input type="text" id="nombre" name="nombre" value="" />

<label for="apellidos">Apellidos</label> <br/>
<input type="text" id="apellidos" name="apellidos" value="" />

<label for="dni">DNI</label> <br/>
<input type="text" id="dni" name="dni" value="" size="10" maxlength="9" />
```

La principal ventaja de utilizar `<label>` es que el código HTML está mejor estructurado y se mejora su accesibilidad. Además, al pinchar sobre el texto del `<label>`, el puntero del ratón se posiciona automáticamente para poder escribir sobre el campo de formulario asociado. Este comportamiento es especialmente útil para los campos de tipo `radiobutton` y `checkbox`.

8.4. Otros elementos de formulario

La etiqueta `<input>` permite crear diez tipos diferentes de controles de formulario. Sin embargo, algunas aplicaciones web utilizan otros elementos de formulario que no se pueden crear con `<input>`. Las listas desplegables y las áreas de texto disponen de sus propias etiquetas (`<select>` y `<textarea>` respectivamente).

Las áreas de texto son útiles cuando se debe introducir una gran cantidad de texto, ya que es mucho más cómodo de introducir que en un campo de texto normal:

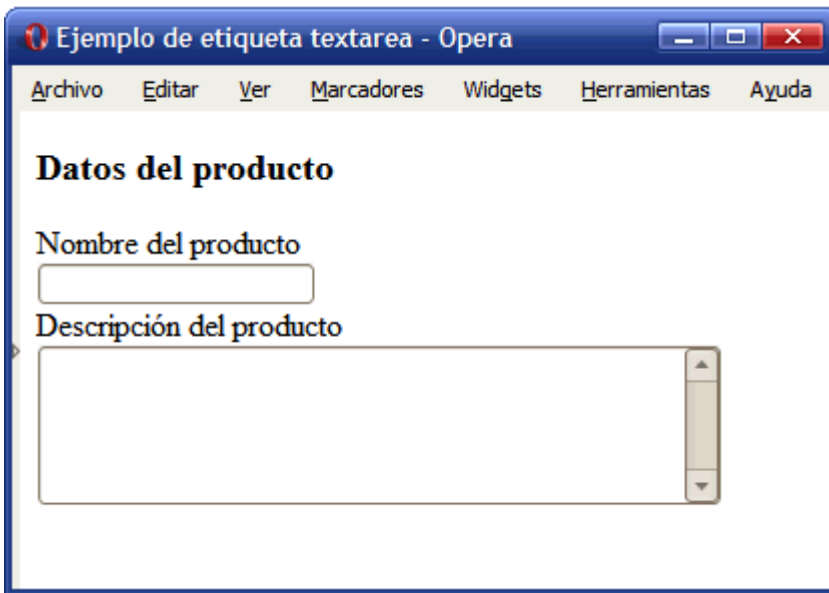


Figura 8.14 Ejemplo de uso de la etiqueta `textarea`

El código HTML del ejemplo anterior se muestra a continuación:

```
<form action="insertar_producto.php" method="post">

<label for="nombre">Nombre del producto</label> <br/>
<input type="text" id="nombre" name="nombre" value="" />

<label for="descripcion">Descripción del producto</label> <br/>
<textarea id="descripcion" name="descripcion" cols="40" rows="5"></textarea>

</form>
```

La definición formal de la etiqueta `<textarea>` es:

<textarea>	Se emplea para incluir un área de texto en un formulario		
Atributos comunes	básicos, internacionalización, eventos y foco	Tipo de elemento	En línea
Atributos propios	<ul style="list-style-type: none"> <code>rows = "numero"</code> - Número de filas de texto que mostrará el textarea <code>cols = "numero"</code> - Número de caracteres que se muestran en cada fila del textarea Otros: <code>name</code>, <code>disabled</code>, <code>readonly</code>, <code>onselect</code>, <code>onchange</code>, <code>onfocus</code>, <code>onblur</code> 		

Los atributos más utilizados en las etiquetas `<textarea>` son los que controlan su anchura y altura. La anchura del área de texto se controla mediante el atributo `cols`, que indica las *columnas* o número de

caracteres que se podrán escribir como máximo en cada fila. La altura del área de texto se controla mediante `rows`, que indica directamente las filas de texto que serán visibles.

El principal inconveniente de los elementos `<textarea>` es que el lenguaje HTML no permite limitar el número máximo de caracteres que se pueden introducir. Mientras los elementos `<input type="text">` disponen del atributo `maxlength`, las áreas de texto no disponen de un atributo equivalente, por lo que sólo es posible limitar el número de caracteres mediante su programación con JavaScript.

Por otra parte, el otro control disponible para los formularios es el de las listas desplegables:

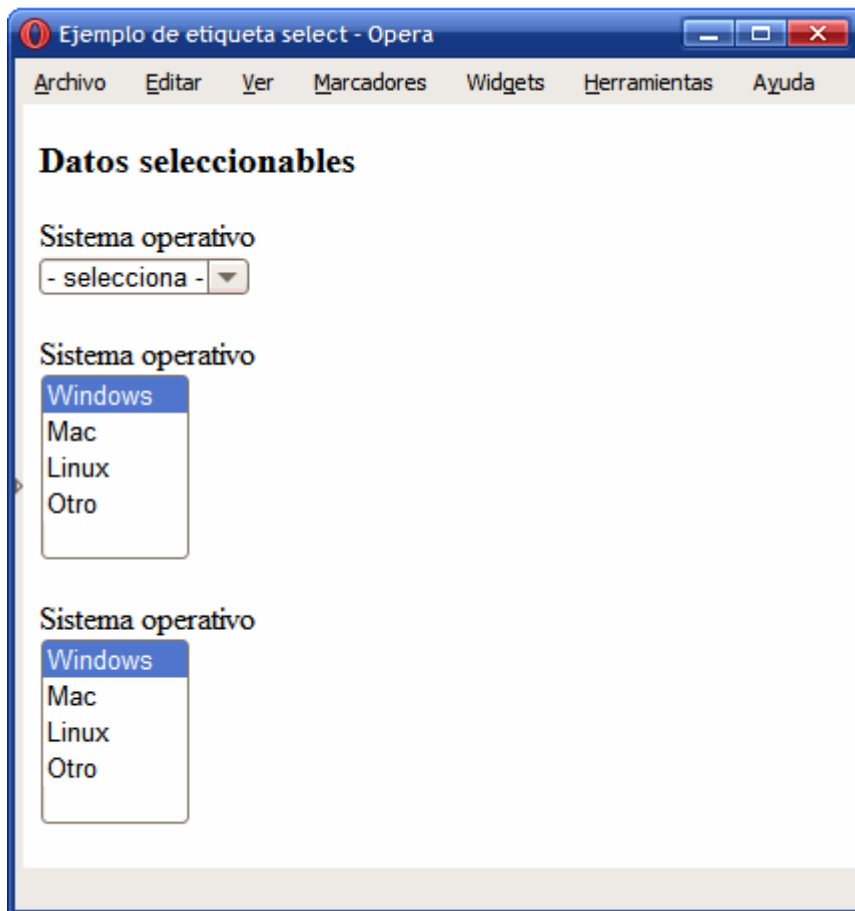


Figura 8.15 Ejemplo de uso de la etiqueta `select`

La imagen anterior muestra los tres tipos de listas desplegables disponibles. El primero es el de las listas más utilizadas que sólo muestran un valor cada vez y sólo permiten seleccionar un valor. El segundo tipo de lista es el que sólo permite seleccionar un valor pero muestra varios a la vez. Por último, el tercer tipo de lista desplegable es aquella que muestra varios valores y permite realizar selecciones múltiples.

El código HTML del ejemplo anterior se muestra a continuación:

```
<label for="so">Sistema operativo</label> <br/>
<select id="so" name="so">
  <option value="" selected="selected">- selecciona -</option>
  <option value="windows">Windows</option>
```

```

<option value="mac">Mac</option>
<option value="linux">Linux</option>
<option value="otro">Otro</option>
</select>

<label for="so2">Sistema operativo</label> <br/>
<select id="so2" name="so2" size="5">
  <option value="windows" selected="selected">Windows</option>
  <option value="mac">Mac</option>
  <option value="linux">Linux</option>
  <option value="otro">Otro</option>
</select>

<label for="so3">Sistema operativo</label> <br/>
<select id="so3" name="so3" size="5" multiple="multiple">
  <option value="windows" selected="selected">Windows</option>
  <option value="mac">Mac</option>
  <option value="linux">Linux</option>
  <option value="otro">Otro</option>
</select>

```

Los tres tipos de listas desplegables se definen con la misma etiqueta `<select>` y cada elemento de la lista se define mediante la etiqueta `<option>`:

<select>	Se emplea para incluir una lista desplegable en un formulario		
Atributos comunes	básicos, internacionalización y eventos	Tipo de elemento	En línea
Atributos propios	<ul style="list-style-type: none"> <code>size = "numero"</code> - Número de filas que se muestran de la lista (por defecto sólo se muestra una) <code>multiple = "multiple"</code> - Si se incluye, se permite seleccionar más de un elemento Otros: name, disabled, onchange, onfocus, onblur 		

<option>	Se emplea para definir cada elemento de una lista desplegable		
Atributos comunes	básicos, internacionalización y eventos	Tipo de elemento	-
Atributos propios	<ul style="list-style-type: none"> • <code>selected = "selected"</code> - Indica si el elemento aparece seleccionado por defecto al cargarse la página • <code>value = "texto"</code> - El valor que se envía al servidor cuando el usuario elige esa opción • Otros: label, disabled 		

La inmensa mayoría de listas desplegadas que utilizan las aplicaciones web son simples, por lo que el código HTML habitual de las listas desplegadas es:

```
<label for="so">Sistema operativo</label> <br/>
<select id="so" name="so">
  <option value="" selected="selected">- selecciona -</option>
  <option value="windows">Windows</option>
  <option value="mac">Mac</option>
  <option value="linux">Linux</option>
  <option value="otro">Otro</option>
</select>
```

La etiqueta `<select>` define la lista y encierra todas las opciones que muestra la lista. Cada una de las opciones de la lista se define mediante una etiqueta `<option>`. El atributo `value` de cada opción es obligatorio, ya que es el dato que se envía al servidor cuando el usuario envía el formulario. Para seleccionar por defecto una opción al mostrar la lista, se añade el atributo `selected` a la opción deseada.

Por otra parte, las listas desplegadas permiten agrupar sus opciones de forma que el usuario pueda encontrar fácilmente las opciones cuando la lista es muy larga:

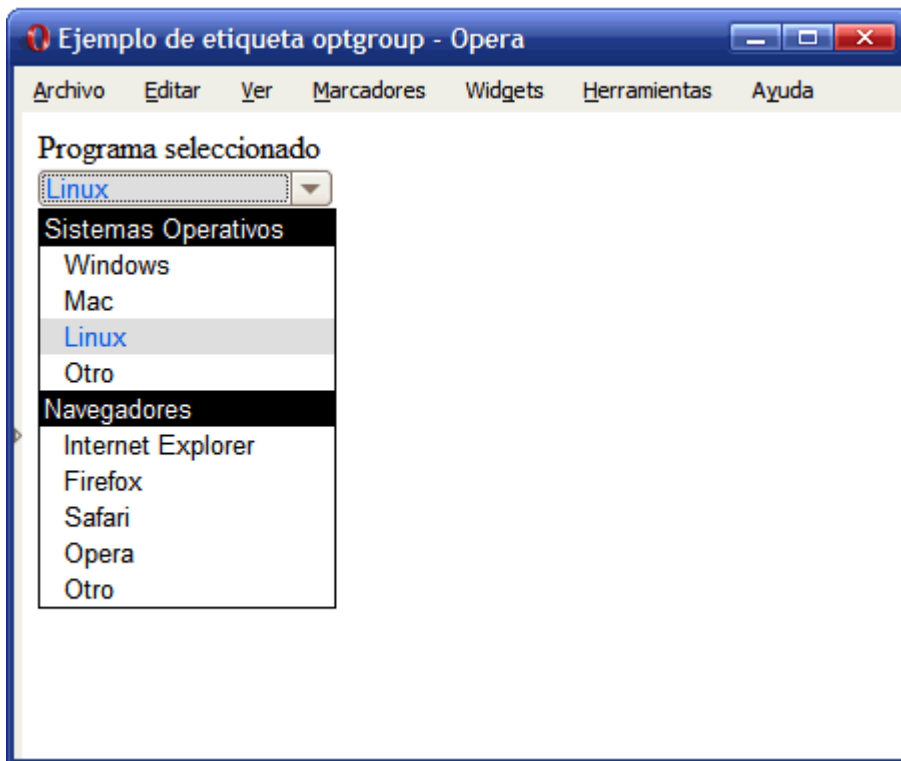


Figura 8.16 Ejemplo de uso de la etiqueta `optgroup`

El código HTML correspondiente a la imagen anterior se muestra a continuación:

```
<form id="formulario" method="post" action="">

<label for="programa">Programa seleccionado</label> <br/>
<select id="programa" name="programa">
  <optgroup label="Sistemas Operativos">
    <option value="Windows" selected="selected">Windows</option>
    <option value="Mac">Mac</option>
    <option value="Linux">Linux</option>
    <option value="Other">Otro</option>
  </optgroup>
  <optgroup label="Navegadores">
    <option value="Internet Explorer" selected="selected">Internet Explorer</option>
    <option value="Firefox">Firefox</option>
    <option value="Safari">Safari</option>
    <option value="Opera">Opera</option>
    <option value="Other">Otro</option>
  </optgroup>
</select>

</form>
```

La etiqueta `<optgroup>` permite agrupar opciones relacionadas dentro de una lista desplegable. Su definición formal se muestra a continuación:

<optgroup>	Se emplea para definir una agrupación lógica de opciones de una lista desplegable		
Atributos comunes	básicos, internacionalización y eventos	Tipo de elemento	-
Atributos propios	<ul style="list-style-type: none"> • <code>label = "texto"</code> - Texto que se muestra como título de la agrupación de opciones • Otros: <code>disabled</code>, <code>selected</code> 		

El único atributo que suele utilizarse con la etiqueta `<optgroup>` es `label`, que indica el nombre de cada agrupación. Los navegadores muestran de forma destacada el título de cada agrupación, de forma que el usuario pueda localizar más fácilmente la opción deseada.

Ejercicio 15

Determinar el código HTML necesario para crear el formulario que se muestra en la siguiente imagen:

Figura 8.17 Formulario con controles de tipo lista desplegable

Ejercicio 16

Determinar el código HTML necesario para crear el formulario que se muestra en la siguiente imagen:

The screenshot shows a browser window with the title "Información sobre el producto - Opera". The menu bar includes "Archivo", "Editar", "Ver", "Marcadores", "Widgets", "Herramientas", and "Ayuda". The main content area is titled "Información sobre el producto" and is divided into two sections:

- Datos básicos:** Contains a "Nombre" text input field, a "Descripción" text area with a vertical scrollbar, a "Foto" text input field next to an "Elegir..." button, and a checkbox labeled "Añadir contador de visitas".
- Datos económicos:** Contains a "Precio" text input field followed by a "€" symbol, an "Impuestos" dropdown menu currently showing "4%", and a "Promoción" section with radio buttons for "Ninguno" (selected), "Transporte gratuito", and "Descuento 5%". The dropdown menu is open, showing options for "4%", "7%", "16%", and "25%".

Figura 8.18 Formulario complejo

Capítulo 9.

Estructura y layout

Los capítulos anteriores muestran las decenas de etiquetas XHTML disponibles para marcar y estructurar cada elemento individual de las páginas web: tablas, listas, enlaces, párrafos, imágenes, etc. Aunque combinando esas etiquetas es posible crear cualquier página web, no es posible hacer que las páginas muestren estructuras complejas.

La mayoría de páginas HTML disponen de estructuras complejas formadas por varias columnas de contenidos y otro tipo de divisiones. Utilizando exclusivamente HTML no es posible crear estas estructuras complejas, ya que es imprescindible emplear las hojas de estilos CSS.

No obstante, los estilos de CSS necesitan la ayuda de HTML/XHTML para crear los diseños más avanzados. En concreto, el código HTML se encarga de agrupar los elementos de la página en diferentes divisiones en función de su finalidad: la zona de la cabecera de la página, la zona de contenidos, una zona lateral para el menú y otras secciones menores, la zona del pie de página, etc.

La siguiente imagen muestra algunas de las zonas definidas en la página principal del sitio www.alistapart.com:

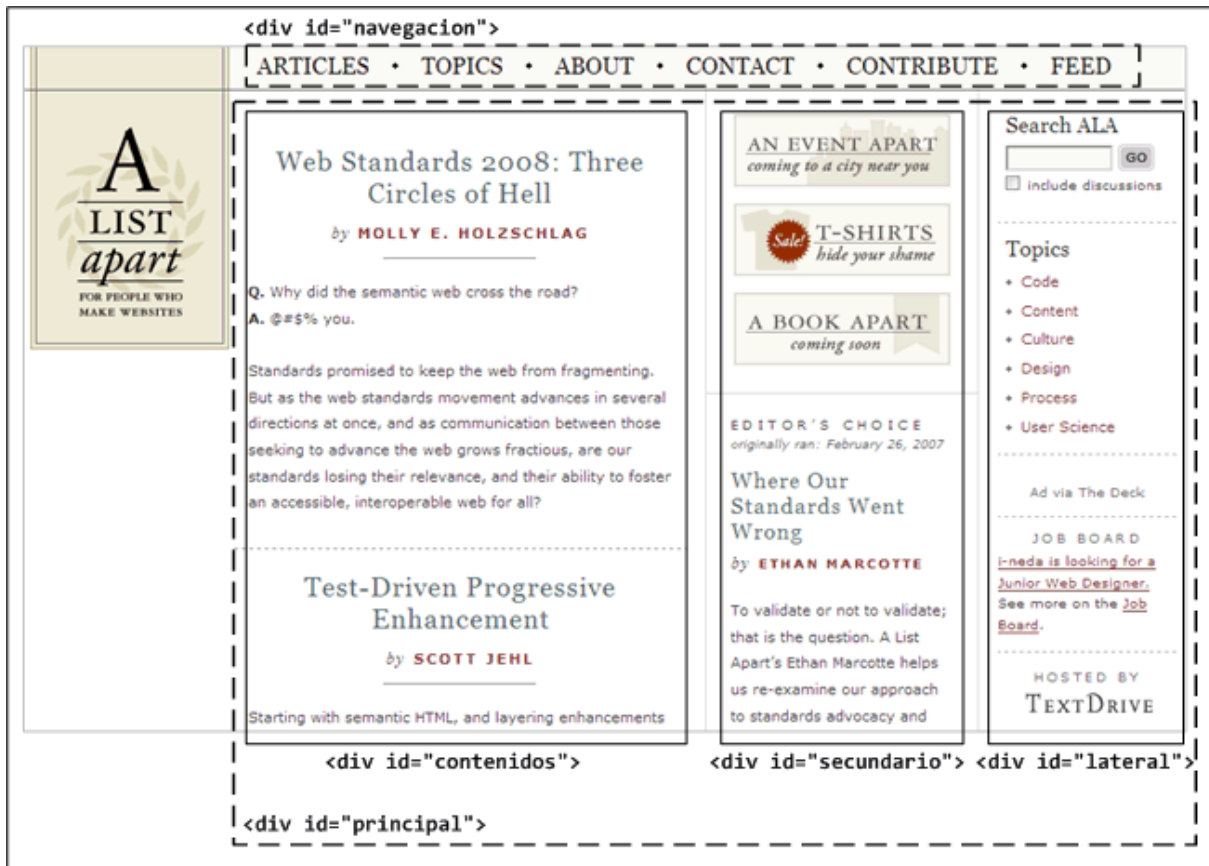


Figura 9.1 Ejemplo de página compleja estructurada con etiquetas div

Para agrupar los elementos que forman cada zona o división de la página se utiliza la etiqueta `<div>`:

<code><div></code>	Agrupa elementos de bloque		
Atributos comunes	básicos, internacionalización y eventos	Tipo de elemento	Bloque
Atributos propios	-		

El nombre de la etiqueta `div` tiene su origen en la palabra *división*, ya que esta etiqueta define zonas o divisiones dentro de una página HTML. En cualquier caso, casi todos los diseñadores web utilizan la palabra "capa" para referirse a una "división". Aunque se trata de un error grave (las capas se crean mediante una propiedad de CSS llamada *z-index*) es preferible seguir llamando "capas" a las zonas definidas con la etiqueta `<div>` para poder entenderse con el resto de diseñadores.

Las páginas web complejas que están bien diseñadas utilizan decenas de etiquetas `<div>`. Con mucha diferencia, los atributos más utilizados con esta etiqueta son `id` (para identificar la capa de forma única) y `class` (para aplicar a la capa estilos CSS).

No se va a profundizar en el proceso de diseñar una página web mediante `<div>`, ya que no es posible diseñar una página web compleja utilizando elementos `<div>` sin utilizar hojas de estilos CSS.

Por último, si observas el código HTML de algunas páginas web complejas, verás que la mayoría utilizan los mismos nombres para identificar sus divisiones. Los nombres más comunes, y sus equivalentes en inglés, se muestran a continuación:

- **contenedor** (*wrapper*) suele encerrar la mayor parte de los contenidos de la página y se emplea para definir las características básicas de la página: su anchura, sus bordes, imágenes laterales, si se centra o no respecto de la ventana del navegador, etc.
- **cabecera** (*header*) que incluye todos los elementos invariantes de la parte superior de la página (logotipo, imagen o banner, cuadro de búsqueda superior, etc.)
- **contenido** (*content*) engloba el contenido principal del sitio (la zona de noticias, la zona de artículos, la zona de productos, etc. dependiendo del tipo de sitio web)
- **menu** (*menu*) se emplea para agrupar todos los elementos del menú lateral de navegación de la página
- **pie** (*footer*) que incluye todos los elementos invariantes de la parte inferior de la página (aviso de copyright, política de privacidad, términos de uso, etc.)
- **lateral** (*sidebar*) se emplea para agrupar los elementos de las columnas laterales y secundarias de la página.

De esta forma, el esqueleto de una página HTML compleja suele ser similar al siguiente:

```
...
<div id="contenedor">
  <div id="cabecera">
    ...
  </div>

  <div id="contenido">
    <div id="menu">
      ..
    </div>
    ...
  </div>

  <div id="pie">
    ...
  </div>
</div>
...
```

El equivalente para las páginas en inglés sería el siguiente:

```
...
<div id="wrapper">
  <div id="header">
    ...
  </div>
```

```
<div id="content">
  <div id="menu">
    ..
  </div>
  ...
</div>

<div id="footer">
  ...
</div>
</div>
```

Capítulo 10.

Metainformación

Las páginas y documentos HTML incluyen más información de la que los usuarios ven en sus pantallas. Estos datos adicionales siempre están relacionados con la propia página, por lo que se denominan *metainformación* o *metadatos*. La metainformación siempre se incluye en la sección de la cabecera, es decir, dentro de la etiqueta `<head>`.

Aunque la metainformación más conocida y utilizada es el título de la propia página, se puede incluir mucha otra información útil para los navegadores y para los buscadores. En las próximas secciones se explica cómo incluir la metainformación y se introduce un concepto relacionado llamado **DOCTYPE**.

10.1. Estructura de la cabecera

Como ya se explicó anteriormente, las páginas XHTML se dividen en dos partes denominadas cabecera y cuerpo. La sección de la cabecera está formada por todas las etiquetas encerradas por la etiqueta `<head>`:

<head>	Define la cabecera del documento HTML		
Atributos comunes	internacionalización	Tipo de elemento	-
Atributos propios	<ul style="list-style-type: none">• <code>profile = "url"</code> - Especifica la URL del perfil o perfiles que utilizan los metadatos• <code>lang = "codigo_de_idioma"</code> - Especifica el idioma principal de los contenidos de la página		

La cabecera típica de una página HTML completa presenta la siguiente estructura:

```
<head>
  <!-- Zona de etiquetas META -->
  <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
```

```

<!-- Zona de título -->
<title>El título del documento</title>

<!-- Zona de recursos enlazados (CSS, RSS, JavaScript) -->
<link rel="stylesheet" href="#" type="text/css" media="screen" />
<link rel="stylesheet" href="#" type="text/css" media="print" />

<link rel="alternate" type="application/rss+xml" title="RSS 2.0" href="#" />

<script src="#" type="text/javascript"></script>
</head>

```

La etiqueta `<title>` establece el título de la página. Los navegadores muestran este título como título de la propia ventana del navegador. Los buscadores utilizan este título como título de sus resultados de búsqueda.

Por tanto, el valor de `<title>` no sólo es importante para los usuarios, sino que también es importante para que los usuarios encuentren las páginas a través de los buscadores. Un error común de muchos sitios web consiste en mostrar un mismo título genérico en todas sus páginas. Cada página debe mostrar un título corto, adecuado, único y que describa inequívocamente los contenidos de la página.

Las páginas XHTML deben tener definido un título y sólo uno, por lo que todas las páginas web deben incluir obligatoriamente una etiqueta `<title>`, cuya definición formal se muestra a continuación:

<title>	Define el título del documento HTML		
Atributos comunes	internacionalización	Tipo de elemento	-
Atributos propios	<ul style="list-style-type: none"> <code>lang = "codigo_de_idioma"</code> - Especifica el idioma principal del título de la página 		

Por último, la etiqueta `<head>` permite definir en el atributo `profile` la URL de un documento externo que contiene el perfil que siguen los metadatos de la cabecera. Los blogs creados con el programa WordPress incluyen por ejemplo el siguiente perfil en su cabecera:

```

<head profile="http://gmpg.org/xfn/11">
  ...
</head>

```

El documento `http://gmpg.org/xfn/11` es un perfil que define atributos adicionales para establecer la relación entre sitios web.

10.2. Metadatos

Una de las partes más importantes de la metainformación de la página son los metadatos, que permiten incluir cualquier información relevante sobre la propia página.

La especificación oficial de HTML no define la lista de metadatos que se pueden incluir, por lo que las páginas tienen libertad absoluta para definir los metadatos que consideren adecuados. La etiqueta empleada para la definición de los metadatos es `<meta>`.

<code><meta></code>	Permite definir el valor de los metadatos que forman la metainformación del documento		
Atributos comunes	internacionalización	Tipo de elemento	-
Atributos propios	<ul style="list-style-type: none"> • <code>name = "texto"</code> - El nombre de la propiedad que se define (no existe una lista oficial de propiedades) • <code>content = "texto"</code> - El valor de la propiedad definida (no existe una lista de valores permitidos) • <code>http-equiv = "texto"</code> - En ocasiones, reemplaza al atributo "name" y lo emplean los servidores para adaptar sus respuestas al documento • <code>scheme = "texto"</code> - Indica el esquema que se debe emplear para interpretar el valor de la propiedad 		

Los metadatos habituales utilizan solamente los atributos `name` y `content` para definir el nombre y el valor del metadato:

```
<meta name="autor" content="Juan Pérez" />
```

No obstante, algunas etiquetas `<meta>` muy utilizadas hacen uso del atributo `http-equiv`. Este atributo se utiliza para indicar que el valor establecido por este metadato puede ser utilizado por el servidor al entregar la página al navegador del usuario. El siguiente metadato indica al servidor que el contenido de la página es código HTML y su codificación de caracteres es UTF-8:

```
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
```

El atributo `scheme` no suele utilizarse, aunque permite proporcionar información de contexto para que el navegador interprete correctamente el valor del metadato. En el siguiente ejemplo, el atributo `scheme` indica al navegador que el valor del metadato hace referencia al código ISBN:

```
<meta scheme="ISBN" name="identificador" content="789-1392349610">
```

Aunque no existe una lista oficial con los metadatos que se pueden definir, algunos de ellos se utilizan en tantas páginas que se han convertido prácticamente en un estándar. A continuación se muestran los metadatos más utilizados:

Definir el autor del documento:

```
<meta name="author" content="Juan Pérez" />
```

Definir el programa con el que se ha creado el documento:

```
<meta name="generator" content="WordPress 2.8.4" />
```

Definir la codificación de caracteres del documento:

```
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
```

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

Definir el copyright del documento:

```
<meta name="copyright" content="librosweb.es" />
```

Definir el comportamiento de los buscadores:

```
<meta name="robots" content="index, follow" />
```

Definir las palabras clave que definen el contenido del documento:

```
<meta name="keywords" content="diseño, css, hojas de estilos, web, html" />
```

Definir una breve descripción del sitio:

```
<meta name="description" content="Artículos sobre diseño web, usabilidad y  
accesibilidad" />
```

La etiqueta que define la codificación de los caracteres (`http-equiv="Content-Type"`) se emplea prácticamente en todas las páginas y las etiquetas que definen la descripción (`description`) y las palabras clave (`keywords`) también son muy utilizadas.

10.3. DOCTYPE

El estándar XHTML deriva de XML, por lo que comparte con el muchas de sus normas y sintaxis. Uno de los conceptos fundamentales de XML es la utilización del DTD o *Document Type Definition* ("Definición del Tipo de Documento").

Un DTD es un documento que recoge el conjunto de normas y restricciones que deben cumplir los documentos de un determinado tipo. Si por ejemplo se define un DTD para los documentos relacionados con libros, se puede fijar como norma que cada libro tenga un título y sólo uno, que tenga uno o más autores, que la información sobre el número de páginas pueda ser opcional, etc.

El conjunto de normas, obligaciones y restricciones que se deben seguir al crear un documento de un determinado tipo, se recogen en su correspondiente DTD. El estándar XHTML define el DTD que deben seguir las páginas y documentos XHTML. En este documento se definen las etiquetas que se pueden utilizar, los atributos de cada etiqueta y el tipo de valores que puede tener cada atributo.

En realidad, la versión 1.0 del estándar de XHTML define tres DTD diferentes. Para indicar el DTD utilizado al crear una determinada página, se emplea una etiqueta especial llamada `doctype`. La etique-

ta **doctype** es el único elemento que se incluye fuera de la etiqueta `<html>` de la página. De hecho, la declaración del **doctype** es lo primero que se debe incluir en una página web, antes incluso que la etiqueta `<html>`.

Como se verá más adelante, para que una página XHTML sea correcta y válida es imprescindible que incluya el correspondiente **doctype** que indica el DTD utilizado. A continuación se muestran los tres DTD que se pueden utilizar al crear páginas XHTML:

XHTML 1.0 Estricto

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

Se trata de la variante con las normas más estrictas y las restricciones más severas. Las páginas web que incluyan este **doctype**, no pueden utilizar atributos relacionados con el aspecto de los contenidos, por lo que requiere una separación total de código HTML y estilos CSS.

XHTML 1.0 Transitorio

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

Se trata de una variante menos estricta que la anterior, ya que permite el uso de algunos atributos HTML relacionados con el aspecto de los elementos.

XHTML 1.0 Frameset

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

Esta última variante la utilizan las páginas que están formadas por *frames*, una práctica completamente desaconsejada y que hoy en día sólo utilizan los sitios web obsoletos.

Si no tienes claro el DTD que más te conviene, deberías utilizar el XHTML 1.0 transitorio, ya que es más fácil crear páginas web válidas. Si tienes conocimientos avanzados de XHTML, puedes utilizar XHTML 1.0 estricto.

Por otra parte, además del **DOCTYPE** apropiado, también es necesario que las páginas web indiquen el **namespace** asociado. Un **namespace** en un documento XML permite diferenciar las etiquetas y atributos que pertenecen a cada lenguaje.

Si en un mismo documento se mezclan etiquetas de dos o más lenguajes derivados de XML (XHTML y SVG por ejemplo) y que tienen el mismo nombre, no se podría determinar a qué lenguaje pertenece cada etiqueta y por tanto, no se podría interpretar esa etiqueta o ese atributo. Los **namespaces** se indican mediante una URL.

El **namespace** que utilizan todas las páginas XHTML (independientemente de la versión y del **DOCTYPE**) es <http://www.w3.org/1999/xhtml> y se indica de la siguiente manera:

```
<html xmlns="http://www.w3.org/1999/xhtml">
...
</html>
```

De esta forma, es habitual que las páginas XHTML comiencen con el siguiente código:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="es" xml:lang="es">
```

Aunque el código anterior es mucho más complicado que una simple etiqueta `<html>`, es imprescindible para que las páginas XHTML creadas sean correctas y superen satisfactoriamente el proceso de validación que se muestra en los capítulos siguientes.

Afortunadamente, si utilizas un editor avanzado como Dreamweaver para crear las páginas, todo el código anterior se incluye de forma automática. Si creas las páginas a mano, sólo tienes que copiar y pegar ese código en cada nueva página.

Capítulo 11.

Otras etiquetas importantes

11.1. Comentarios

Al igual que la mayoría de lenguajes de marcado, HTML permite incluir comentarios dentro de su código para añadir información que no se debe mostrar por pantalla.

Normalmente, los diseñadores y programadores incluyen comentarios para marcar el comienzo y el final de las secciones de las páginas, para incluir avisos y notas para otros diseñadores o para incluir explicaciones sobre la forma en la que se ha creado el código HTML.

Aunque los comentarios no se muestran por pantalla y por tanto son *invisibles* para los usuarios, sí que se descargan con el código HTML de la página. Por este motivo, nunca debe incluirse información sensible o confidencial en los comentarios.

La sintaxis de los comentarios es la siguiente:

- Apertura del comentario: `<!--`
- Contenido del comentario: `(cualquier texto)`
- Cierre del comentario: `-->`

El siguiente ejemplo muestra el uso de los comentarios HTML para indicar el comienzo y final de cada sección. Recuerda que los comentarios no se muestran por pantalla y que no influyen en la forma en la que se ven las páginas:

```
<!-- Inicio del menú -->
<div id="menu">
<ul>
  <li>...</li>
  <li>...</li>
  <li>...</li>
  <li>...</li>
</ul>
<!-- Fin del menú -->
```

```
<!-- Inicio de la publicidad -->
<div id="publicidad"> ... </div>
<!-- Fin de la publicidad -->
```

Los comentarios de HTML puede ocupar tantas líneas como sea necesario. Sin embargo, los comentarios no se pueden anidar, es decir, no se puede incluir un comentario dentro de otro comentario.

11.2. JavaScript

Como ya se explicó en los capítulos anteriores, la etiqueta `<script>` se utiliza para enlazar archivos JavaScript externos y para incluir bloques de código JavaScript en las páginas. Sin embargo, algunos navegadores no disponen de soporte completo de JavaScript, otros navegadores permiten bloquearlo parcialmente e incluso algunos usuarios bloquean completamente el uso de JavaScript porque creen que así navegan de forma más segura.

Si JavaScript está bloqueado o deshabilitado y la página web requiere su uso para un correcto funcionamiento, es habitual incluir un mensaje de aviso al usuario indicándole que debería activar JavaScript para disfrutar completamente de la página.

El siguiente ejemplo muestra una misma página web que requiere JavaScript tanto cuando se accede con JavaScript activado y como cuando se accede con JavaScript completamente desactivado.

Imagen de www.netvibes.com con JavaScript activado

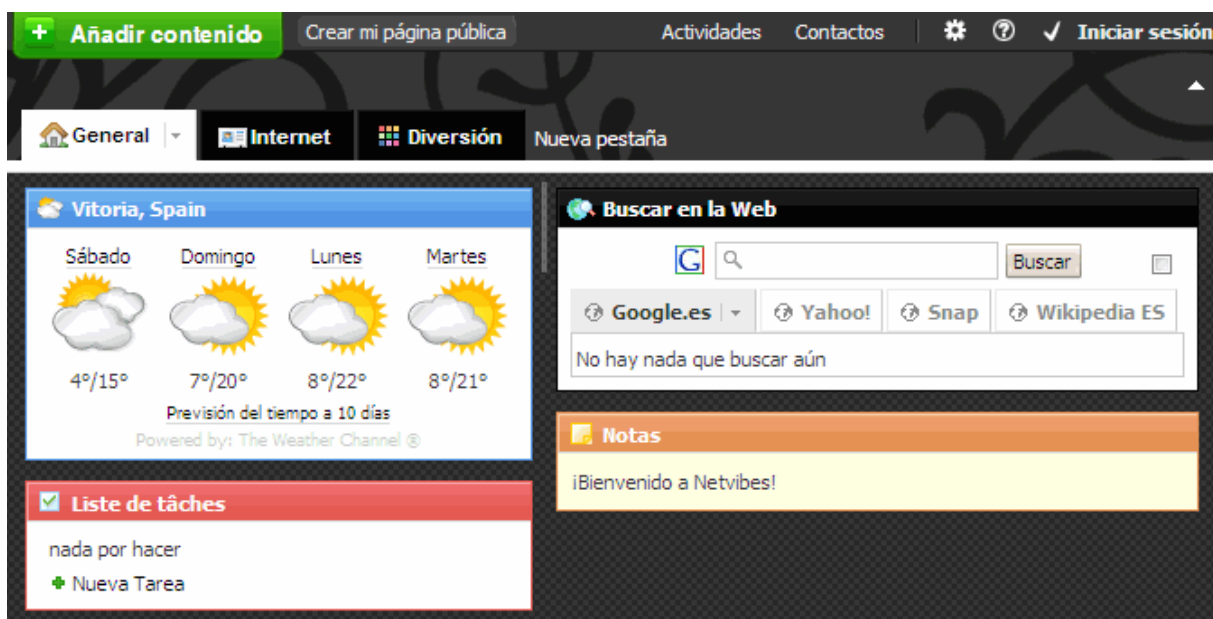


Figura 11.1 Ejemplo de página compleja con JavaScript activado

Imagen de www.netvibes.com con JavaScript deshabilitado



Figura 11.2 Ejemplo de página compleja con JavaScript desactivado

HTML define la etiqueta `<noscript>` para incluir un mensaje que los navegadores muestran cuando JavaScript se encuentra bloqueado o deshabilitado.

<code><noscript></code>	Define un mensaje alternativo que se muestra al usuario cuando su navegador no soporta la ejecución de scripts		
Atributos comunes	básicos, internacionalización y eventos	Tipo de elemento	Bloque
Atributos propios	-		

De esta forma, incluir un mensaje de aviso que solamente sea visible en los navegadores que tienen bloqueado JavaScript es tan sencillo como incluir la etiqueta `<noscript>` dentro del `<body>`.

```
<head> ... </head>
<body>
<noscript>
  <p>Bienvenido a Mi Sitio</p>
  <p>La página que estás viendo requiere para su funcionamiento el uso de JavaScript.
  Si lo has deshabilitado intencionadamente, por favor vuelve a activarlo.</p>
</noscript>
</body>
```

11.3. CSS

Algunos de los atributos más utilizados en la creación de páginas web son `id`, `class` y `style`. Los tres atributos están muy relacionados con CSS, sobre todo `class` y `style`.

El atributo `id` se emplea para asignar un identificador único a cada elemento de la página, lo que es útil tanto para aplicar estilos CSS a ese elemento como para programar aplicaciones con JavaScript.

Por otra parte, el atributo `class` se emplea para definir la clase CSS que se aplica a un elemento. La clase CSS es el nombre de un conjunto de estilos que se definen en la hoja de estilos y que se quieren aplicar a un elemento:

```
<p class="resumen">Lorem ipsum dolor sit amet, consectetur adipiscing elit.  
Maecenas at diam id enim viverra semper. Nulla id urna. Donec sodales.</p>
```

El párrafo del ejemplo anterior se muestra por pantalla con el aspecto definido por el conjunto de estilos llamado `resumen` y que se define en la hoja de estilos CSS enlazada por la página web.

El atributo `style` se emplea para definir estilos CSS directamente sobre los elementos HTML, tal y como se muestra en el siguiente ejemplo:

```
<p>Algunas palabras de esta frase se muestran de <span style="color:red">color  
rojo</span></p>
```

No se debe confundir el atributo `style` con la etiqueta `<style>` que se explicó anteriormente. La etiqueta `<style>` se utiliza para incluir bloques de código CSS:

```
<head>  
...  
<style type="text/css">  
  span {color:red;}  
</style>  
</head>
```

11.4. Iframes

Aunque su uso no es muy común, la etiqueta `<iframe>` puede ser muy útil en determinadas ocasiones, ya que permite insertar un documento HTML dentro de otro documento HTML. Un `iframe` puede considerarse como un *agujero* que se abre en una página web y a través del cual se muestra otra página web.

En ocasiones se utiliza para mostrar contenidos externos al sitio web como si fueran parte del mismo sitio. Otra vez se emplea para incluir una aplicación común a varios sitios web de una misma empresa.

La página principal de **Google Analytics** emplea un `<iframe>` para incluir en un pequeño recuadro la página correspondiente a la validación de usuario.



Figura 11.3 Ejemplo de página con un iframe

<iframe>	Se emplea para incluir en la página un marco que muestra otro documento HTML		
Atributos comunes	básicos	Tipo de elemento	Bloque y en línea
Atributos propios	<ul style="list-style-type: none"> <code>src = "url"</code> - URL del documento HTML que se visualiza en el iframe <code>height = "longitud"</code> - Altura que ocupará el iframe en el documento <code>width = "longitud"</code> - Anchura que ocupará el iframe en el documento <code>name = "texto"</code> - Nombre que identifica al iframe <code>longdesc = "url"</code> - Dirección en la que puede encontrarse una descripción larga del contenido del iframe <code>scrolling = "yes no auto"</code> - Indica si el iframe debe mostrar barras de scroll (horizontal y vertical) cuando el contenido incluido no cabe en el iframe 		

El siguiente ejemplo define la altura y anchura del `iframe`, indica la URL que se debe mostrar y mediante el atributo `scrolling` se indica que el `iframe` no debe mostrar barras de `scroll` ni siquiera en el caso de que el contenido mostrado no quepa en el `iframe` definido:

```
<iframe src="/ruta/documento.html" width="250" height="250" scrolling="no" />
```

11.5. Otras etiquetas

La etiqueta `<address>` es una de las etiquetas más desconocidas de HTML, por lo que uso no está muy extendido. La etiqueta `<address>` se utiliza para proporcionar información de contacto.

<address>	Define la información de contacto de un documento		
Atributos comunes	básicos, internacionalización y eventos	Tipo de elemento	Bloque
Atributos propios	-		

El siguiente ejemplo sencillo muestra directamente el nombre, dirección y teléfono de contacto de una empresa:

```
<address>
  Nombre de la empresa
  Dirección completa
  Teléfono y Fax
</address>
```

La especificación oficial de HTML muestra un ejemplo complejo del uso de la etiqueta `<address>`:

```
<address>
  <a href="../People/Raggett/">Dave Raggett</a>,
  <a href="../People/Arnaud/">Arnaud Le Hors</a>,
  contact persons for the <a href="Activity">W3C HTML Activity</a><br/>
  $Date: 1999/12/24 23:37:50 $
</address>
```

Hasta hace unos años, la etiqueta `<hr>` era una de las más utilizadas, ya que permite mostrar una línea horizontal de separación. Sin embargo, hoy en día apenas se utiliza, ya que se considera un elemento puramente estético, del que no debería preocuparse HTML y para el que CSS ofrece alternativas mucho mejores.

<hr>	Permite incluir una línea horizontal de separación		
Atributos comunes	básicos, internacionalización y eventos	Tipo de elemento	Bloque
Atributos propios	-		

La siguiente imagen muestra el aspecto con el que los navegadores muestran por defecto las líneas horizontales creadas con `<hr>`:

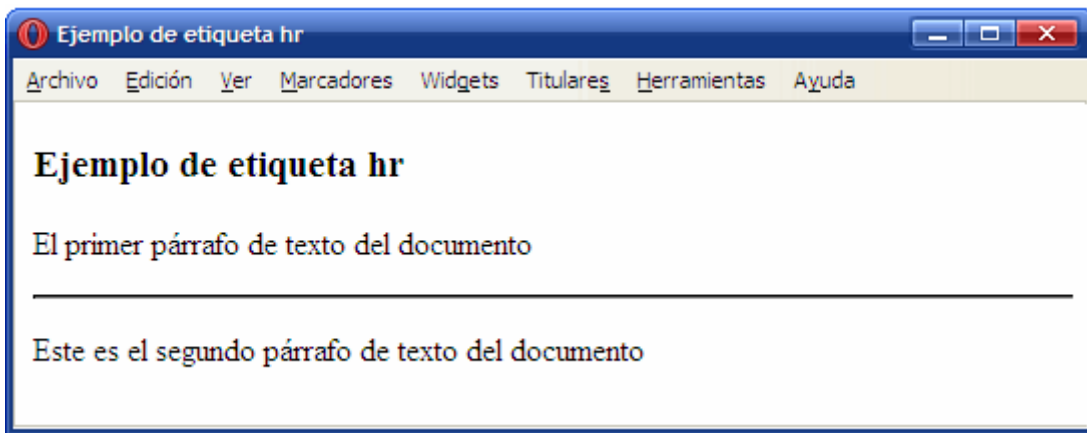


Figura 11.4 Ejemplo de uso de la etiqueta hr

El código HTML del ejemplo anterior se muestra a continuación:

```
<html>
<head><title>Ejemplo de etiqueta hr</title></head>

<body>
<h3>Ejemplo de etiqueta hr</h3>

<p>El primer párrafo de texto del documento</p>

<hr/>

<p>Este es el segundo párrafo de texto del documento</p>

</body>
</html>
```

Esta página se ha dejado vacía a propósito

Capítulo 12.

Accesibilidad

El principal objetivo de la accesibilidad web es el de permitir a cualquier usuario, independientemente del tipo de discapacidad que sufra, el acceso a los contenidos del sitio y permitirle la navegación necesaria para realizar las acciones deseadas.

Los sitios web accesibles no solamente facilitan el acceso de sus contenidos a los usuarios discapacitados, sino que también permiten ofrecer la misma funcionalidad con dispositivos muy limitados (dispositivos sin pantalla o con pantallas minúsculas, dispositivos sin teclado ni ratón, etc.).

Las cuatro principales ventajas de diseñar un sitio web completamente accesible son las siguientes:

- Los sitios accesibles separan completamente diseño y contenido.
- Un sitio accesible puede ser accedido por multitud de dispositivos diferentes sin necesidad de reescribir el código HTML.
- Los sitios accesibles son los únicos con una audiencia potencial global, ya que permiten el acceso a todos los usuarios y a todos los dispositivos.
- Generalmente, los sitios accesibles son más fáciles de utilizar también para los usuarios sin discapacidades.

La creación de sitios accesibles puede realizarse siguiendo las recomendaciones establecidas por el W3C y que se recogen en el documento *Web Content Accessibility Guidelines* (<http://www.w3.org/WAI/intro/wcag.php>) (WCAG).

La versión WCAG 1.0 (<http://www.w3.org/TR/WCAG10/>) que se utiliza en la actualidad se publicó en 1999, mientras que la versión WCAG 2.0 (<http://www.w3.org/TR/WCAG20/>) se encuentra todavía en borrador y se actualizó por última vez el día 30 de abril de 2008.

Las recomendaciones del WCAG 1.0 están formadas por 65 requisitos que un sitio web debe cumplir para considerarse accesible. Los requerimientos se agrupan en prioridades.

Los requisitos de prioridad 1 son de obligado cumplimiento, los de prioridad 2 son recomendables y los de prioridad 3 son deseables. Si un sitio web cumple con todos los requisitos de prioridad 1, se considera que el sitio es conforme al nivel A de accesibilidad.

El nivel AA de accesibilidad está reservado para los sitios que cumplan todos los requisitos de prioridad 1 y prioridad 2. Finalmente, los sitios que cumplen los 65 requisitos, son conformes al nivel AAA de accesibilidad.

12.1. Requisitos del nivel A de accesibilidad

Los requisitos de accesibilidad que exige el nivel A son los siguientes:

12.1.1. Generales

1.1 Proporcionar un texto alternativo para todas las imágenes, objetos y otros elementos no textuales (mediante los atributos `alt` y `longdesc`).

2.1 Asegurar que toda la información que utilice el color como elemento informativo pueda ser entendida por las personas o dispositivos que no pueden distinguir los colores.

4.1 Marcar claramente (mediante los atributos `lang` y `xml:lang`) las variaciones del idioma del texto o de los elementos textuales (`<caption>`) respecto del idioma principal de la página.

6.1 El documento debe poder leerse completamente cuando no se utilicen hojas de estilos.

6.2 La información equivalente para los contenidos dinámicos debe adaptarse a los cambios de los contenidos dinámicos.

7.1 Ningún elemento debe parpadear en la pantalla.

14.1 El contenido del sitio se debe escribir con un lenguaje sencillo y limpio.

12.1.2. Si se utilizan mapas de imagen

1.2 Proporcionar un enlace textual por cada una de las regiones del mapa de imagen.

9.1 Utilizar mapas de imagen en el cliente, en vez de mapas de imagen de servidor.

12.1.3. Si se utilizan tablas

5.1 Utilizar cabeceras de fila y de columna.

5.2 Si la tabla tiene varios niveles de cabeceras, utilizar las agrupaciones disponibles (`<thead>`, `<tfoot>`).

12.1.4. Si se utilizan frames

12.1 Indicar un título a cada *frame* para su identificación y facilitar la navegación.

12.1.5. Si se utilizan applets y scripts

6.3 Asegurar que la página también se pueda utilizar cuando no se ejecutan los applets y los scripts. Si no es posible, proporcionar informaciones equivalente o páginas alternativas que sean accesibles.

12.1.6. Si se utilizan contenidos multimedia (audio y vídeo)

1.3 Incluir una descripción textual del contenido multimedia.

1.4 Para los contenidos basados en vídeo o animaciones, sincronizar las alternativas textuales con la presentación.

12.1.7. Si no se pueden cumplir los anteriores requisitos

11.4 Proporcionar una página alternativa con la mayor cantidad posible de contenidos y que cumpla con los requisitos anteriores.

La lista completa con los 65 requisitos de los tres niveles de accesibilidad se puede consultar en <http://www.w3.org/TR/WCAG10/full-checklist.html>

Esta página se ha dejado vacía a propósito

Capítulo 13.

Validación

La validación es el proceso que asegura que un documento escrito en un determinado lenguaje (por ejemplo XHTML) cumple con las normas y restricciones de ese lenguaje. Las normas y restricciones de los documentos escritos en XML (y en sus lenguajes derivados, como XHTML) se definen en el DTD o *Document Type Definition* ("Definición del Tipo de Documento").

El concepto de validación es objeto de controversia en el ámbito del diseño web. Por una parte, la validación no es obligatoria y las páginas web se pueden ver bien sin que sean válidas. Por otra parte, una página válida es más correcta que otra página que no lo sea, ya que cumple con las normas y restricciones impuestas por XHTML.

Debido a esta controversia, algunos diseñadores consideran que se da demasiada importancia a la validación de las páginas y a la creación de páginas válidas. El resto de diseñadores argumentan que si la especificación de XHTML impone una serie de normas y restricciones, lo más correcto es que las páginas web las cumplan, aunque no sea obligatorio.

En cualquier caso, el proceso de validación consiste en probar página a página si su código HTML pasa la prueba de validación. Los validadores son las herramientas que se utilizan para validar cada página. Algunos editores de páginas web incluyen sus propios validadores y el organismo W3C ha creado una herramienta gratuita para la validación de las páginas.

En las próximas secciones de este capítulo se muestran las diferentes herramientas de validación disponibles para validar las páginas web.

13.1. Validación con Dreamweaver

Si utilizas Dreamweaver para crear las páginas web, el validador se encuentra integrado en la propia herramienta. En primer lugar, accede a la configuración de la herramienta de validación desde la opción [Edición > Preferencias > Validador](#):

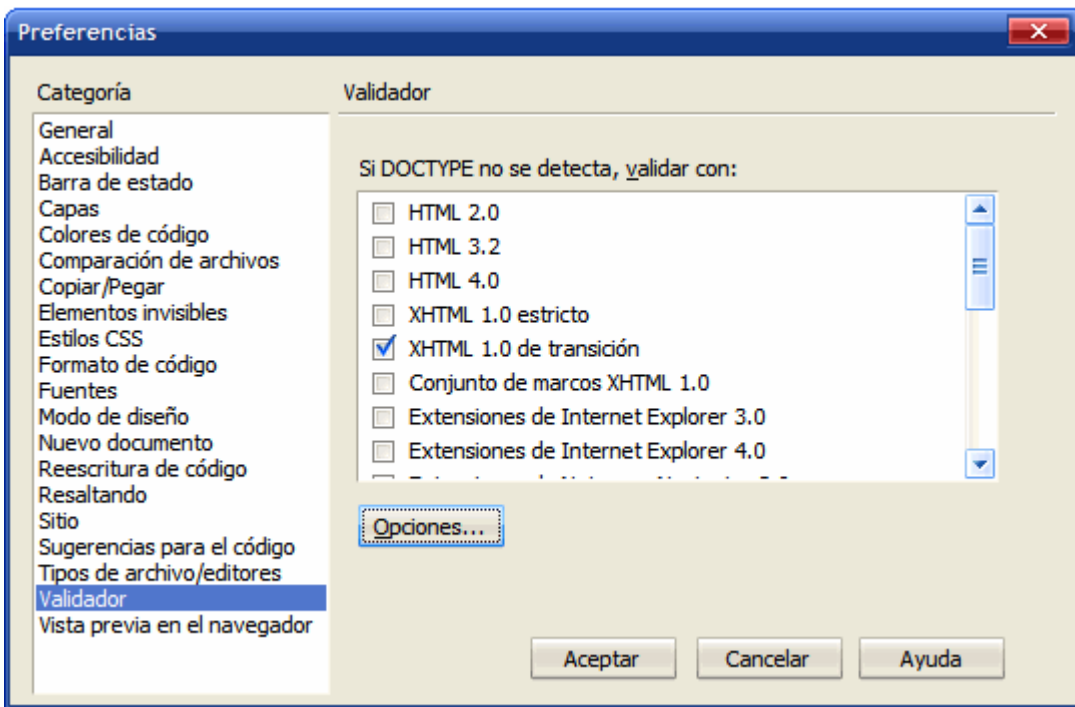


Figura 13.1 Configuración del validador de Dreamweaver

En esta ventana de configuración se puede elegir el DTD que se utiliza en caso de que la página web no indique el DTD que utiliza. Las páginas declaran el DTD que utilizan mediante el `doctype`, tal y como se explicó en capítulos anteriores. Una vez seleccionado el DTD por defecto (en la imagen anterior, se ha elegido el DTD de XHTML 1.0 de transición), se puede acceder a la herramienta de validación de Dreamweaver desde el icono que se muestra en la siguiente imagen:

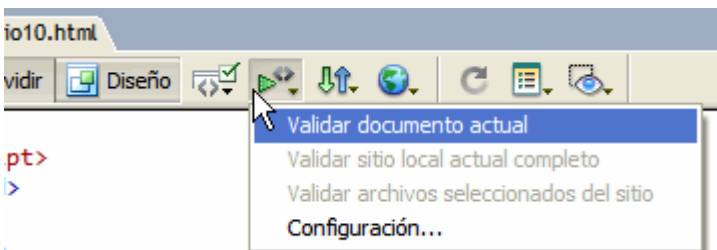


Figura 13.2 Icono que permite acceder a la herramienta de validación de Dreamweaver

Si no se han producido errores al validar la página, Dreamweaver lo indica mediante un mensaje que declara a la página como válida. Si se produce algún error, la página no es válida y Dreamweaver muestra la lista de todos los errores encontrados junto con sus posibles soluciones:

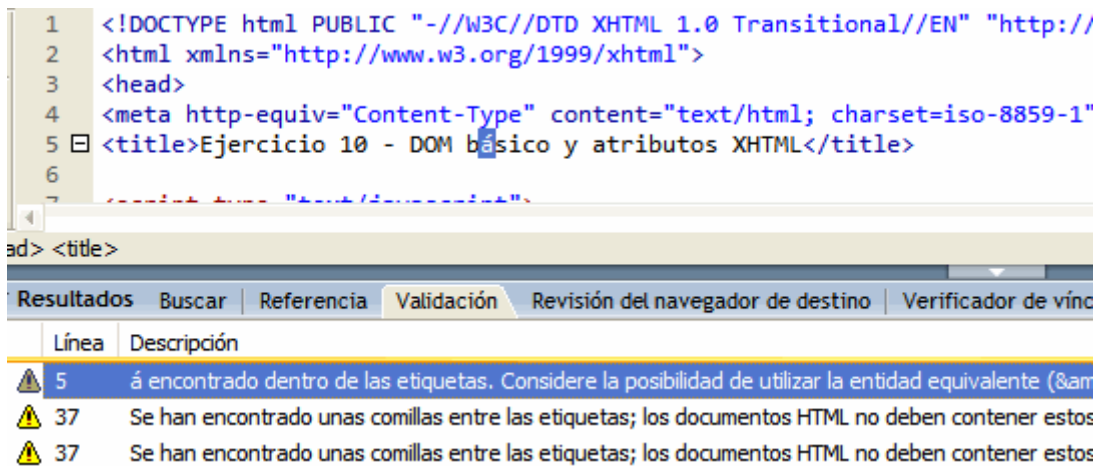


Figura 13.3 Resultado de validar una página con Dreamweaver

Después de corregir todos los errores, se puede pasar otra vez la prueba de validación para comprobar que la página cumple con todas las restricciones que impone el tipo de `doctype` que utiliza.

13.2. Validador del W3C

La validación de las páginas web no requiere el uso de editores avanzados como Dreamweaver, ya que el organismo W3C ha creado una herramienta que se puede utilizar gratuitamente a través de Internet: <http://validator.w3.org/>

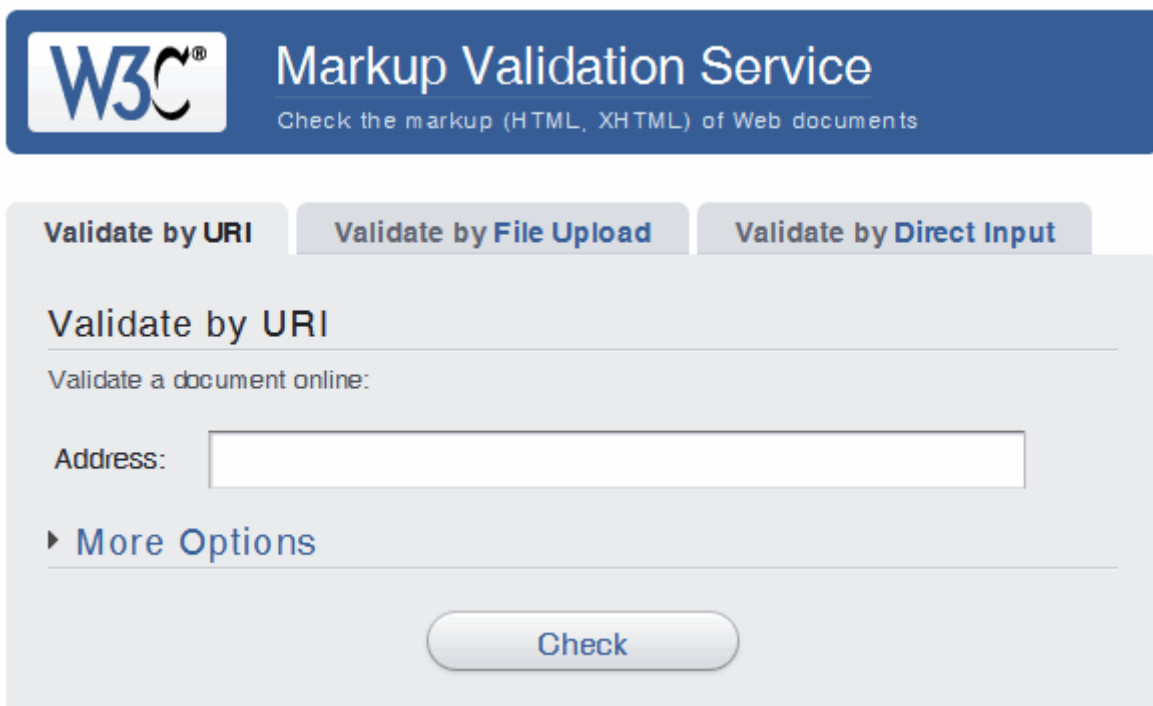


Figura 13.4 Página principal del validador del W3C

Aunque la herramienta sólo está disponible en inglés, su uso es muy intuitivo:

- **Validate by URI**, permite escribir la URL de la página que se quiere validar. Esta opción es la más sencilla para validar las páginas que ya están publicadas en Internet.

- **Validate by File Upload**, muestra un formulario mediante el que se puede subir el archivo HTML correspondiente a la página que se quiere validar. Esta opción es la mejor para validar las páginas web que has desarrollado y que aún no has publicado en Internet.
- **Validate by Direct Input**, permite validar código HTML de forma directa. Se trata de la opción más rápida para validar trozos o páginas HTML completas. Esta opción es la mejor cuando estás desarrollando las páginas y quieres asegurarte que el código sea válido.

La siguiente imagen muestra el resultado de la validación de la página principal de Google realizada mediante la opción **Validate by URI**:

This page is not Valid (no Doctype found)!	
Result:	Failed validation, 32 Errors
Address :	<input type="text" value="http://www.google.com/"/>
Encoding :	iso-8859-1 <input type="text" value="(detect automatically)"/>
Doctype :	(no Doctype found) <input type="text" value="(detect automatically)"/>
Root Element:	html

Figura 13.5 Resultado de validar una página con el validador de W3C

Si la página no pasa correctamente la prueba de validación, se muestra el listado completo de fallos junto con la ayuda necesaria para resolver cada uno de los errores.

Como se observa en la imagen anterior, incluso una página tan sencilla como la portada de Google contiene decenas de errores que impiden considerarla válida. Por lo tanto, la página principal de Google no es una página válida, aunque eso no impide que se vea bien en todos los navegadores y que los usuarios la consideren correcta.

13.3. Otros validadores

Además de los validadores disponibles en herramientas como Dreamweaver y de los validadores gratuitos disponibles en Internet, existe otro método de validación sencillo, gratuito y muy rápido. La única limitación de este validador es que necesariamente se debe utilizar el navegador Firefox.

Si ya dispones del navegador Firefox, puedes instalar el validador mediante un complemento llamado HTML Validator (<https://addons.mozilla.org/es-ES/firefox/addon/249>). La instalación se realiza como cualquier otro complemento, aunque en este caso la descarga dura un poco más de lo normal porque ocupa más de 2 MB.

Tras su instalación, la primera vez que se reinicia Firefox se muestra la siguiente ventana:

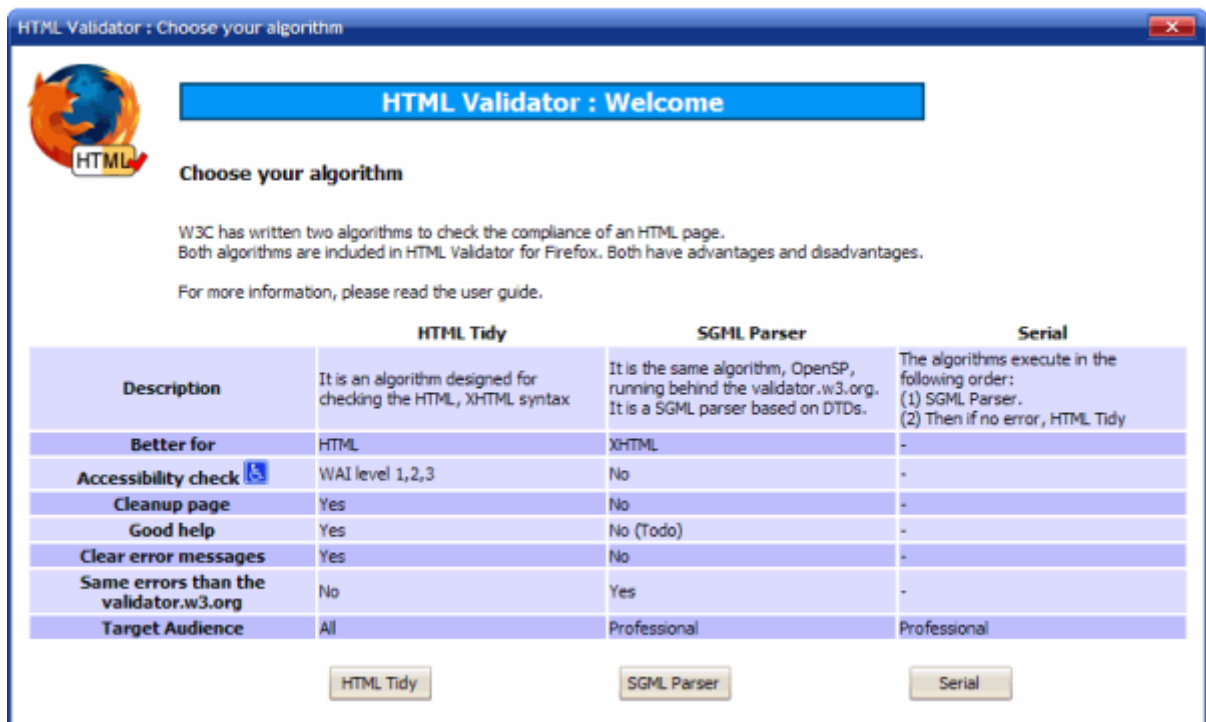


Figura 13.6 Configuración del validador Html Validator

En la ventana que se muestra se solicita al usuario que configure el tipo de validación que se va a realizar. Las opciones para elegir son: **HTML Tidy** (que ofrece ayuda para resolver los errores y es mejor para HTML), **SGML Parser** (ofrece menos ayuda, pero es el mismo que el validador del W3C) o **Serial** (que realiza las dos validaciones de forma seguida).

Si no sabes cual elegir, la opción **Serial** es una buena alternativa, ya que primero realiza la validación **SGML Parser** y a continuación, si no se han producido errores, realiza la validación **HTML Tidy**.

Una vez configurado el validador, abre cualquier página web y verás cómo en la esquina inferior derecha de Firefox se muestra un pequeño icono que indica si la página es válida o no. Cuando la página no es válida, aparece un icono correspondiente a un error. Si colocas el puntero del ratón sobre el icono, se muestra la información específica sobre los errores encontrados:



Figura 13.7 Información de error proporcionada por el validador Html Validator

Si pulsas dos veces sobre el icono, aparece una nueva ventana en la que se muestra la lista completa de errores, el lugar exacto en el que se han producido y las posibles soluciones para corregirlos.

Para ver directamente el número de errores de la página, puedes pulsar el botón derecho del ratón sobre el icono del validador y seleccionar la opción **Show** y después **Icon and Text**. Después de activar esta opción, cada vez que cargues una página web se muestra toda la información de validación.

Aunque existen muchos otros validadores, el uso de Firefox junto con Html Validator es la única alternativa que permite validar las páginas web sin esfuerzo. Abriendo cualquier página en el navegador Firefox, es posible visualizar al instante si la página es válida o no y el número de errores que se han encontrado.

Capítulo 14.

Fragmentos de código

Cuando se crean páginas web, es habitual repetir una y otra vez algunos trozos de código HTML, como por ejemplo las tablas, los formularios y la cabecera de las páginas. Para no tener que reescribir continuamente el mismo código, se utilizan los *fragmentos de código*, llamados "*snippets*" en inglés.

Crear *fragmentos de código* y trabajar con ellos es un proceso muy sencillo. A continuación se muestran los pasos necesarios para crear un fragmento correspondiente a una tabla completa de XHTML:

1. Se escribe el código XHTML completo de una tabla vacía (con sus etiquetas `<caption>`, `<thead>`, `<tbody>`, `<tfoot>`, sus atributos `summary`, `scope`, etc.)
2. Se guarda el código anterior en un archivo de texto con un nombre fácil de identificar (por ejemplo, "Tabla XHTML").
3. Cuando se necesite insertar una tabla en una página XHTML, se copia y se pega todo el código creado anteriormente y se completa con los datos necesarios.

El método descrito anteriormente no es muy eficiente si se realiza de forma manual. Afortunadamente, muchos programas utilizados para el diseño web permiten gestionar de forma más cómoda los fragmentos de código o *snippets*.

A continuación se muestra la forma de trabajar con los fragmentos de código en el conocido programa Dreamweaver:

- 1) Si no está visible, se debe mostrar la ventana correspondiente a los fragmentos de código mediante la opción de menú **Ventana > Fragmentos** o pulsando **Mayúsc. + F9**. Una vez abierta esta ventana, el aspecto que muestra se ve en la parte inferior izquierda de la siguiente imagen:

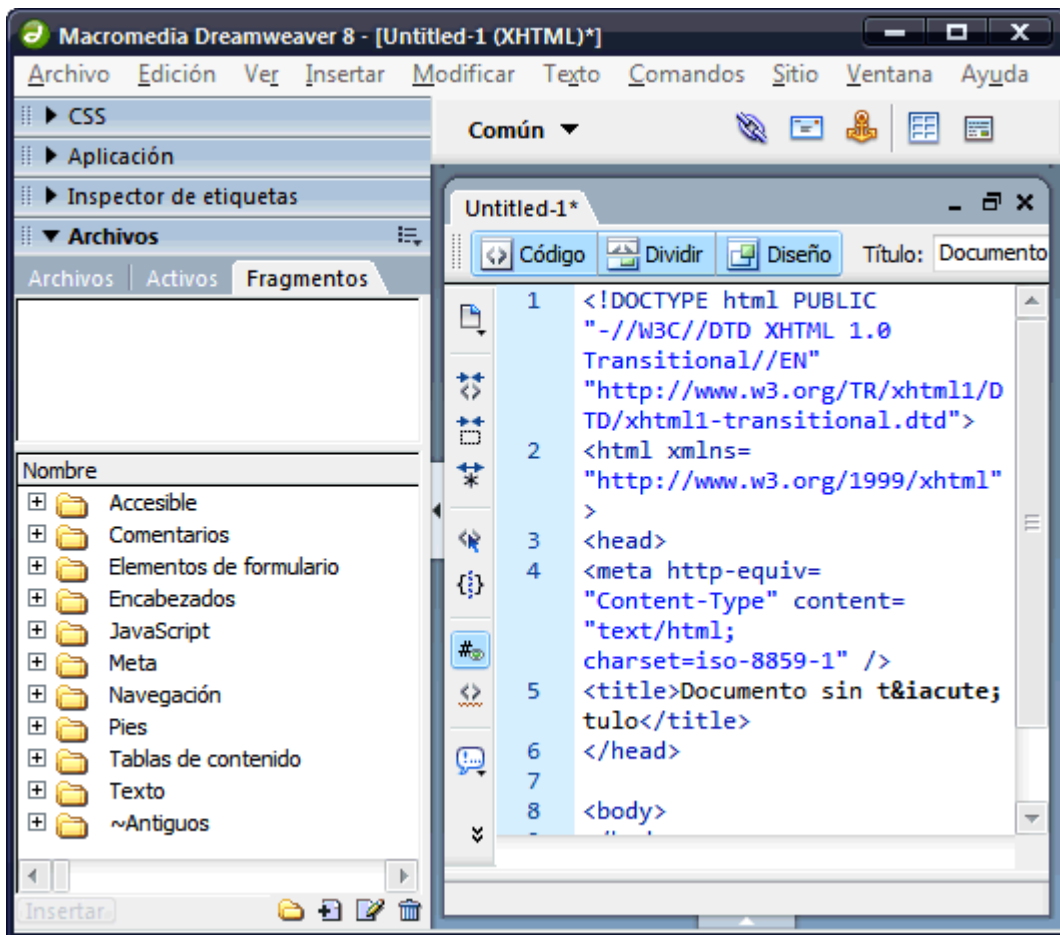


Figura 14.1 Visualizando la ventana de fragmentos de código en Dreamweaver

2) Dreamweaver ya dispone por defecto de muchos fragmentos de código útiles. Si por ejemplo se pulsa sobre la carpeta *Meta*, Dreamweaver muestra un *snippet* llamado *No crear caché*. Al pinchar dos veces sobre el nombre del fragmento, se inserta su contenido en el lugar en el que se encuentre el cursor dentro de la página, tal y como muestra la siguiente imagen:

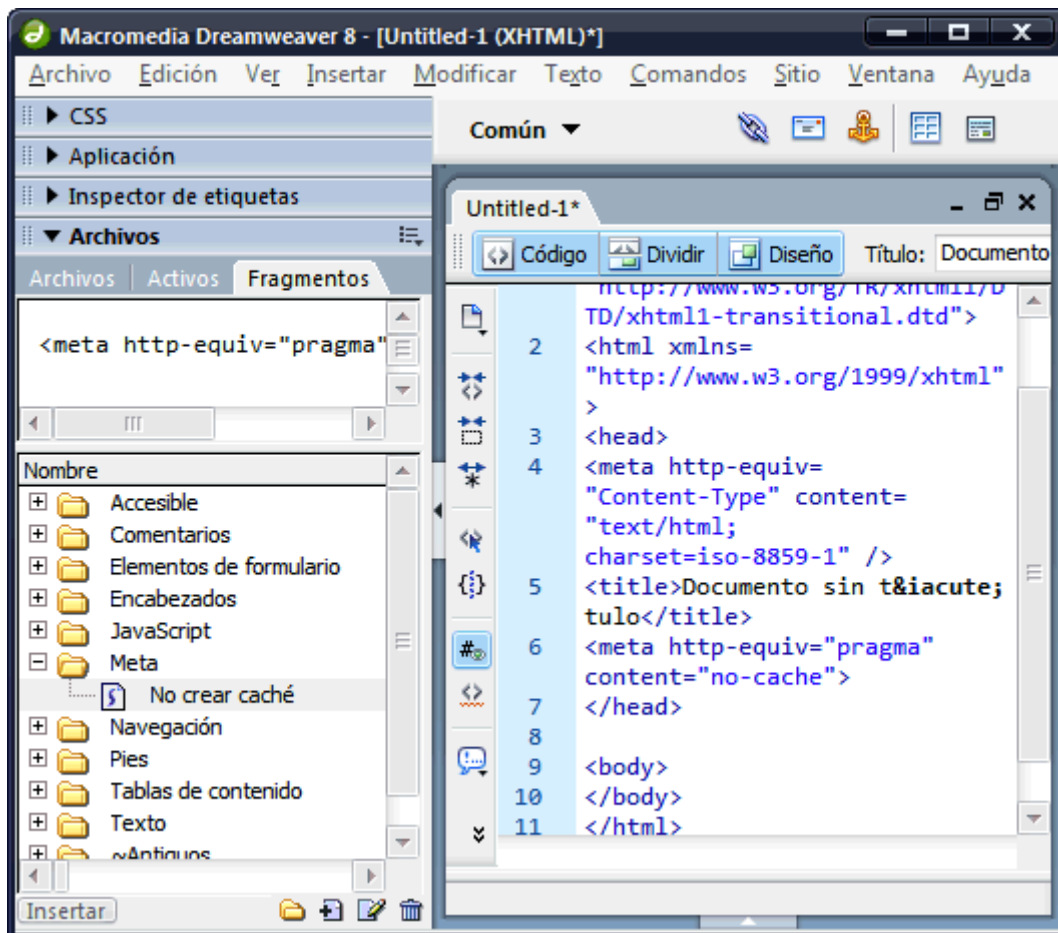


Figura 14.2 Insertando un fragmento de código de Dreamweaver en una página web

3) Para añadir un fragmento de código propio, se crea en primer lugar una carpeta en la que se guardarán todos los *snippets* propios. El nombre elegido para esta carpeta es **Propios** y la forma de crearla es pinchando sobre el pequeño icono que simboliza una carpeta:

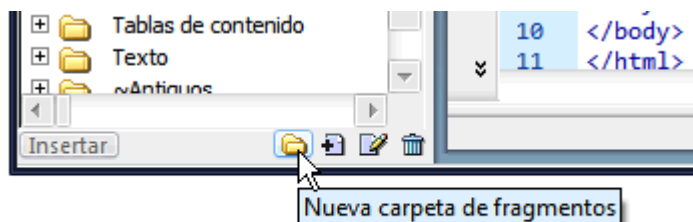


Figura 14.3 Creando una nueva carpeta para guardar los fragmentos de código propios

Una vez escrito el nombre de la nueva carpeta, el aspecto que muestra la ventana de *snippets* es el siguiente:

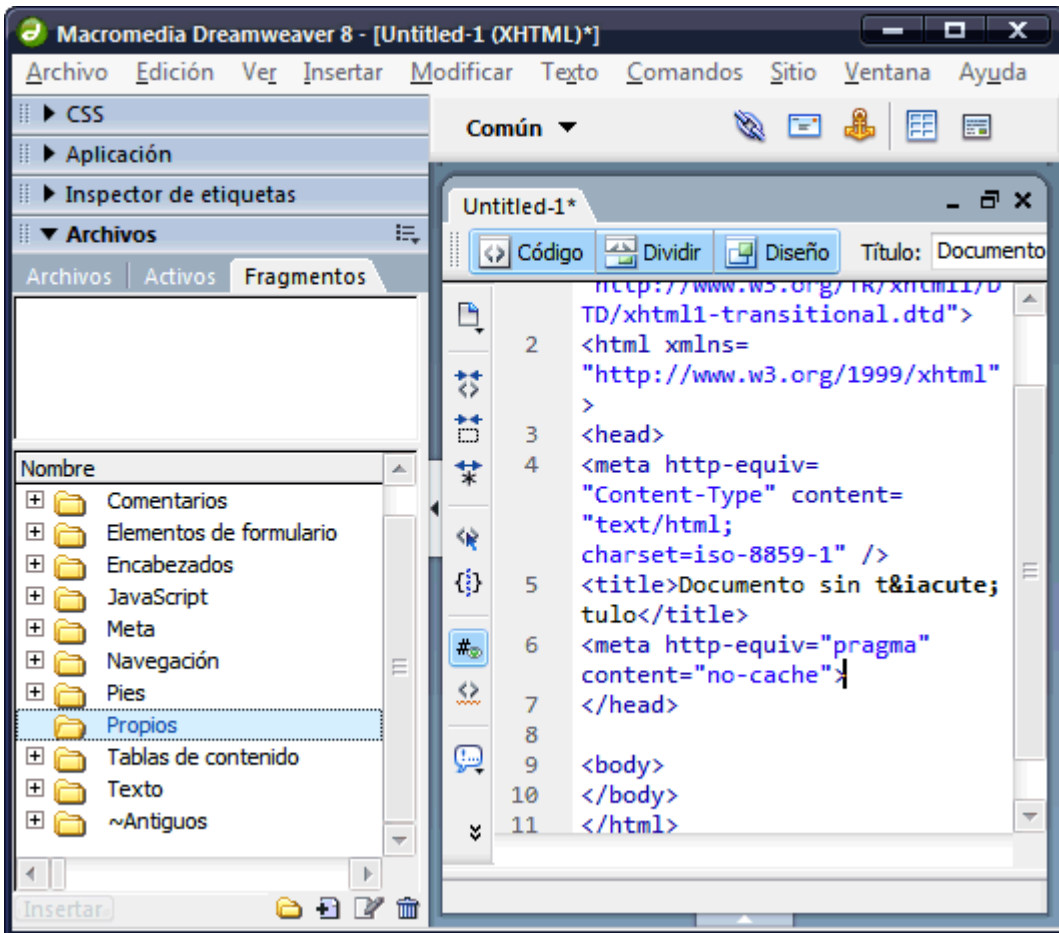


Figura 14.4 Nueva carpeta creada para guardar los fragmentos de código propios

4) Para crear un fragmento de código propio, se pulsa sobre el pequeño icono que simboliza un nuevo fragmento:



Figura 14.5 Creando un nuevo fragmento de código

Después de pulsar sobre ese icono, aparece la ventana en la que se puede escribir el nombre y el contenido del fragmento:

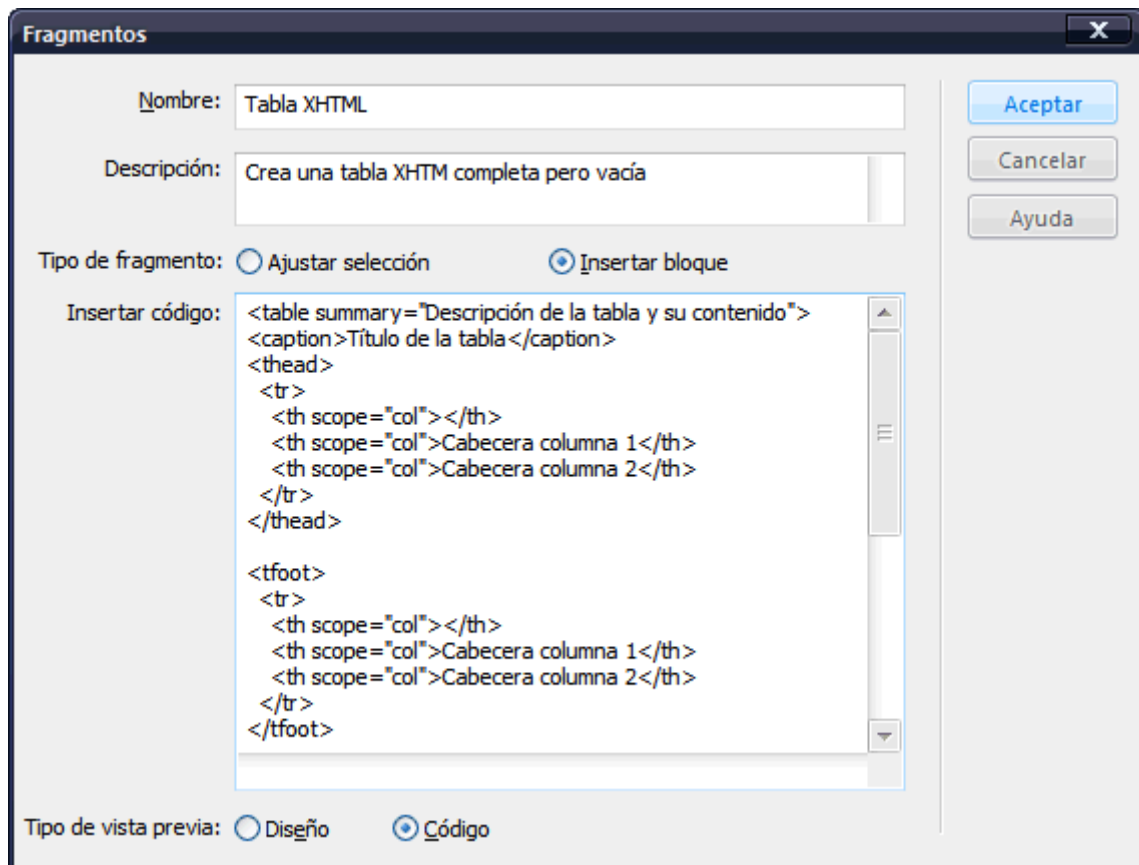


Figura 14.6 Propiedades (nombre, descripción y contenido) del fragmento de código propio

En este caso, se trata de un fragmento de código que inserta un bloque completo de código XHTML. También es posible crear fragmentos que añaden código XHTML antes y después del texto que ha sido seleccionado previamente.

Una vez creado el *snippet*, ya se puede insertar en cualquier zona de la página XHTML simplemente pulsando dos veces sobre su nombre:

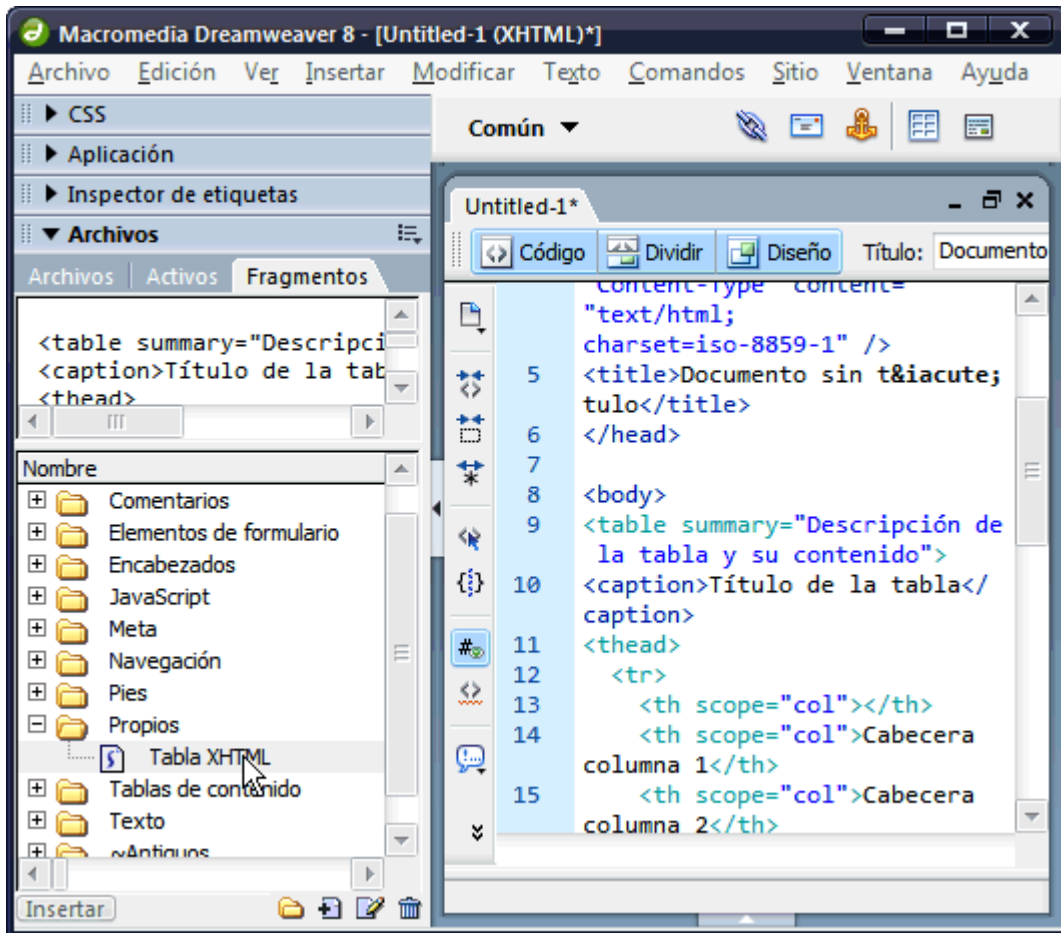


Figura 14.7 Insertando un fragmento de código propio en una página XHTML

En las siguientes secciones de este capítulo, se incluyen algunos fragmentos de código imprescindibles para crear páginas web.

14.1. Documento XHTML

La estructura básica de cualquier página o documento XHTML válido incluye al menos las siguientes etiquetas:

```

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="es" xml:lang="es">
  <head>
    <title>...</title>
    ...
  </head>
  <body>
    ...
  </body>
</html>
    
```

El DOCTYPE también puede ser:

```

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


14.2. Cabecera XHTML

Las cabeceras HTML pueden llegar a ser muy grandes, ya que incluyen toda la metainformación de la página y los enlaces a todos los recursos externos (hojas de estilos CSS, archivos JavaScript y archivos RSS).

```
<head>
<title>Título de la página</title>

<meta http-equiv="content-type" content="text/html; charset=iso-8859-1" />
<meta name="description" content="..." />
...
<meta name="author" content="..." />

<style type="text/css" media="screen,projection">
  @import '/css/archivo.css';
</style>
<link rel="stylesheet" type="text/css" href="/css/impresora.css" media="print" />
<link rel="stylesheet" type="text/css" href="/css/movil.css" media="handheld" />

<link rel="shortcut icon" href="/favicon.ico" type="image/ico" />

<link rel="alternate" type="application/rss+xml" title="..." href="/archivo_rss.xml" />

<script type="text/javascript" src="/js/archivo.js"></script>

<style type="text/css">
  /* Código CSS */
</style>

<script type="text/javascript">
  //
  // Código JavaScript
  //]]&gt;
&lt;/script&gt;
&lt;/head&gt;</pre>
</div>
<div data-bbox="139 686 268 707" data-label="Section-Header">
<h2>14.3. Tabla</h2>
</div>
<div data-bbox="139 710 913 746" data-label="Text">
<p>Las tablas XHTML complejas están formadas por cabecera, pie y uno o más cuerpos. Además, es necesario incluir atributos como <code>summary</code> y <code>scope</code> para mejorar la accesibilidad.</p>
</div>
<div data-bbox="156 761 632 908" data-label="Text">
<pre>&lt;table summary="Descripción de la tabla y su contenido"&gt;
&lt;caption&gt;Título de la tabla&lt;/caption&gt;
&lt;thead&gt;
  &lt;tr&gt;
    &lt;th scope="col"&gt;&lt;/th&gt;
    &lt;th scope="col"&gt;Cabecera columna 1&lt;/th&gt;
    &lt;th scope="col"&gt;Cabecera columna 2&lt;/th&gt;
  &lt;/tr&gt;
&lt;/thead&gt;</pre>
</div>
<div data-bbox="867 925 912 944" data-label="Page-Footer">161</div>
```

```

<tfoot>
  <tr>
    <th scope="col"></th>
    <th scope="col">Cabecera columna 1</th>
    <th scope="col">Cabecera columna 2</th>
  </tr>
</tfoot>

<tbody>
  <tr>
    <th scope="row">Cabecera fila 1</th>
    <td>...</td>
    <td>...</td>
  </tr>
  <tr>
    <th scope="row">Cabecera fila 2</th>
    <td>...</td>
    <td>...</td>
  </tr>
</tbody>

</table>

```

14.4. Formulario

Los formularios complejos agrupan sus campos mediante las etiquetas `<fieldset>` y `<legend>`. Además, cada campo debe incluir su título mediante la etiqueta `<label>`.

```

<form id="identificador" method="post" action="" enctype="multipart/form-data">

<fieldset>
<legend>Título del formulario</legend>

<label for="campo_texto">Campo de texto</label>
<input id="campo_texto" name="campo_texto" type="text" maxlength="255" value=""/>

<label for="campo_areatexto">Area de texto</label>
<textarea id="campo_areatexto" name="campo_areatexto" rows="10" cols="50"></textarea>

<label for="campo_desplegable">Lista desplegable</label>
<select id="campo_desplegable" name="campo_desplegable">
  <option value="0" selected="selected">- selecciona -</option>
  <option value="1">Valor 1</option>
  <option value="2">Valor 2</option>
  <option value="3">Valor 3</option>
</select>

<label for="campo_fichero">Subir un fichero</label>

```

```
<input id="campo_fichero" name="campo_fichero" type="file" value="" />

<input type="radio" id="valor1" name="campo_radio" value="valor1" checked="checked" />
<label for="valor1">Valor 1</label>
<input type="radio" id="valor2" name="campo_radio" value="valor2" />
<label for="valor2">Valor 2</label>

<input id="campo_check" name="campo_check" type="checkbox" value="valor1"
checked="checked"/>
<label for="campo_check">Campo check 1</label>

<input id="boton_enviar" type="submit" value="Enviar formulario" />

</fieldset>
</form>
```

Esta página se ha dejado vacía a propósito

Capítulo 15.

Ejercicios resueltos

15.1. Ejercicio 1

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/
DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="es" xml:lang="es">

<head>
<title>El primer documento HTML</title>
</head>

<body>
<p>El lenguaje HTML es <strong>tan sencillo</strong> que practicamente se entiende sin
estudiar el significado de sus etiquetas principales.</p>

<p>Ademas de textos en <strong>negrita</strong>, tambien se pueden poner <em>en
cursiva</em> o <del>tachados</del>.</p>

</body>

</html>
```

15.2. Ejercicio 2

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/
DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="es" xml:lang="es">

<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
<title>La exploración espacial</title>
</head>

<body>
```

```
<h2>La exploración espacial</h2>
```

```
<p>La <strong>exploración espacial</strong> designa los esfuerzos del hombre en estudiar el espacio y sus astros desde el punto de vista científico y de su explotación económica. Estos esfuerzos pueden involucrar tanto seres humanos viajando en naves espaciales como satélites con recursos de telemetría o sondas teleguiadas enviadas a otros planetas (orbitando o aterrizando en la superficie de estos cuerpos celestes).</p>
```

```
<p>Las personas que pilotan naves espaciales, o son pasajeros en ellas, se llaman astronautas (en Rusia: <em>cosmonautas</em>; en China: <em>taikonautas</em>). Técnicamente se considera astronauta a todo aquel que emprenda un vuelo sub-orbital (sin entrar en órbita) u orbital a como mínimo 100 km de altitud (considerado el límite externo de la atmósfera).</p>
```

```
<p>El cielo siempre ha atraído la atención y los sueños del hombre. Ya en 1634 se publicó la que se considera primera novela de ciencia ficción, <em>Somnium</em>, de <strong>Johannes Kepler</strong>, que narra un hipotético viaje a la Luna. Más tarde, en 1865, en una famosa obra de ficción titulada <em>"De la Terre à la Lune"</em>, <strong>Julio Verne</strong> escribe sobre un grupo de hombres que viajó hasta la Luna usando un gigantesco cañón.</p>
```

```
<p>En Francia, <strong>Georges Méliès</strong>, uno de los pioneros del cine, tomaba la novela de Verne para crear <em>"Le voyage dans la Lune"</em> (1902), una de las primeras películas de ciencia ficción en la que describía un increíble viaje a la Luna. En obras como <em>"The War of the Worlds"</em> (1898) y <em>"The First Men in The Moon"</em> (1901), <strong>Herbert George Wells</strong> también se concibieron ideas de exploración del espacio y de contacto con civilizaciones extraterrestres.</p>
```

```
</body>
```

```
</html>
```

15.3. Ejercicio 3

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

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

```
<head>
```

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

```
<title>El Ártico ha perdido el 14% de su hielo marino perenne en un solo año</title>
```

```
</head>
```

```
<body>
```

```
<h1>El Ártico ha perdido el 14% de su hielo marino perenne en un solo año</h1>
```

```
<p><strong>WASHINGTON.</strong> El hielo perenne del Ártico se redujo en un 14% durante el último año, al perder <strong>720.000 kilómetros cuadrados</strong>, una superficie superior a la Península Ibérica, según datos de la <acronym title="National
```

```
Aeronautics and Space Administration">NASA</acronym>.</p>
```

```
<p>Según el <acronym title="Laboratorio de Propulsión a Chorro">JPL</acronym>, la
pérdida del hielo perenne, que debiera mantenerse durante todo el verano, fue todavía
mayor y se acercó a un 50% en el momento en que ese hielo se desplazaba desde el Ártico
oriental hacia el oeste.</p>
```

```
<p><strong>Son Nghiem</strong>, investigador del <acronym title="Laboratorio de
Propulsión a Chorro">JPL</acronym> ha declarado que:</p>
```

```
<blockquote><em>"Los cambios registrados en esos años en el hielo ártico son rápidos y
espectaculares. De mantenerse la situación, ésta tendrá un impacto profundo en el
ambiente, así como en el transporte marino y el comercio."</em></blockquote>
```

```
</body>
```

```
</html>
```

15.4. Ejercicio 4

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

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

```
<head>
```

```
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
```

```
<title>Espacios en blanco arbitrarios</title>
```

```
</head>
```

```
<body>
```

```
<h2>Características de los planetas</h2>
```

```
<pre>
```

```
Nombre          Diametro relativo   Período orbital     Número de lunas
-----
Mercurio         0,382               0,24 años           0
Venus            0,949               0,62 años           0
Tierra           1                   1 año               1
Marte            0,532               1,88 años           2
Júpiter          11,209              11,86 años          49
Saturno          9,449               29,46 años          52
Urano            4,007               84,01 años          27
Neptuno          3,883               164,80 años         13
```

```
</pre>
```

```
</body>
```

```
</html>
```

15.5. Ejercicio 5

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/
DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="es" xml:lang="es">

<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
<title>La etiqueta <code>blockquote</code></title>
</head>

<body>

<h1>Sintaxis de la etiqueta <code>blockquote</code></h1>

<p>
La sintaxis de la etiqueta <code>blockquote</code> se muestra a continuación: <code><br/><br/>
<code>blockquote cite="<code><em>...direccion original de la cita...</em></code>"</code>Texto
que se cita<code></code></p>

</body>

</html>
```

15.6. Ejercicio 6

Página principal

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/
DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="es" xml:lang="es">

<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
<title>Mi Sitio</title>
</head>

<body>

<h1>Mi Sitio</h1>

<p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Donec iaculis posuere
justo. Nam vel neque. Proin sagittis mauris sit amet nisl. Sed ipsum. Aliquam vitae
justo.</p>

<h2>Ultimos proyectos</h2>
```



```
<p>Etiam consectetur, mauris vitae cursus scelerisque, dui turpis dignissim justo, et
eismod enim odio sit amet erat. Aliquam dui ligula, porttitor eu, facilisis vitae,
ornare sed, tortor.</p>

<p><a href="portfolio/indice.html" title="Ultimos proyectos realizados por Mi
Sitio">Acceder a los ultimos proyectos de Mi Sitio</a></p>

</body>

</html>
```

Página principal del portfolio

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/
DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="es" xml:lang="es">

<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
<title>Mi Sitio</title>
</head>

<body>

<p><a href="../indice.html" title="Página principal de Mi Sitio">Volver a la pagina
principal</a></p>

<h1>Ultimos proyectos</h1>

<p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Donec iaculis posuere
justo. Nam vel neque.</p>

<h3>Proyecto 1</h3>

<p>Etiam consectetur, mauris vitae cursus scelerisque, dui turpis dignissim justo, et
eismod enim odio sit amet erat.</p>

<p><a href="../imagenes/proyecto1.png" title="Imagen del Proyecto 1">Ver imagen del
Proyecto 1</a></p>

<h3>Proyecto 2</h3>

<p>Etiam consectetur, mauris vitae cursus scelerisque, dui turpis dignissim justo, et
eismod enim odio sit amet erat.</p>

<p><a href="../imagenes/proyecto2.png" title="Imagen del Proyecto 2">Ver imagen del
Proyecto 2</a></p>

</body>
```

```
</html>
```

Descargar ZIP con la solución completa (/ejercicios/xhtml/soluciones/ejercicio6/ejercicio6.zip)

15.7. Ejercicio 7

Página principal

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/
DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="es" xml:lang="es">

<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
<link rel="shortcut icon" href="favicon.ico" type="image/x-icon" />
<title>Mi Sitio</title>
</head>

<body>

<h1>Mi Sitio</h1>

<p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Donec iaculis posuere
justo. Nam vel neque. Proin sagittis mauris sit amet nisl. Sed ipsum. Aliquam vitae
justo.</p>

<h2>Ultimos proyectos</h2>

<p>Etiam consectetur, mauris vitae cursus scelerisque, dui turpis dignissim justo, et
euismod enim odio sit amet erat. Aliquam dui ligula, porttitor eu, facilisis vitae,
ornare sed, tortor.</p>

<p><a href="portfolio/indice.html" title="Ultimos proyectos realizados por Mi
Sitio">Acceder a los ultimos proyectos de Mi Sitio</a></p>

</body>

</html>
```

Página principal del portfolio

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/
DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="es" xml:lang="es">

<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
<link rel="shortcut icon" href="../favicon.ico" type="image/x-icon" />
<title>Mi Sitio</title>
```

```
</head>

<body>

<p><a href="portfolio/indice.html" title="Página principal de Mi Sitio"
rel="index">Volver a la pagina principal</a></p>

<h1>Ultimos proyectos</h1>

<p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Donec iaculis posuere
justo. Nam vel neque.</p>

<h3>Proyecto 1</h3>

<p>Etiam consectetur, mauris vitae cursus scelerisque, dui turpis dignissim justo, et
eismod enim odio sit amet erat.</p>

<p><a href="../imagenes/proyecto1.png" title="Imagen del Proyecto 1" type="image/
png">Ver imagen del Proyecto 1</a></p>

<h3>Proyecto 2</h3>

<p>Etiam consectetur, mauris vitae cursus scelerisque, dui turpis dignissim justo, et
eismod enim odio sit amet erat.</p>

<p><a href="../imagenes/proyecto2.png" title="Imagen del Proyecto 2" type="image/
png">Ver imagen del Proyecto 2</a></p>

</body>

</html>
```

[Descargar ZIP con la solución completa \(/ejercicios/xhtmll/soluciones/ejercicio7/ejercicio7.zip\)](/ejercicios/xhtmll/soluciones/ejercicio7/ejercicio7.zip)

15.8. Ejercicio 8

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/
DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="es" xml:lang="es">

<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
<title>Lista simple anidada</title>
</head>

<body>

<h1>Menú</h1>

<ul>
```

```

</li>Inicio</li>
<li>
  <strong>Noticias</strong>
  <ul>
    <li><a href="#" title="Ver las noticias más recientes">Recientes</a></li>
    <li><strong><a href="#" title="Ver las noticias más leídas">Más
leídas</a></strong></li>
    <li><a href="#" title="Ver las noticias más valoradas">Más valoradas</a></li>
  </ul>
</li>
</ul>

</body>

</html>

```

15.9. Ejercicio 9

```

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/
DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="es" xml:lang="es">

<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
<title>Lista compleja anidada</title>
</head>

<body>

<h1>Menú</h1>

<ul>
<li>Inicio</li>
<li>
  <strong>Noticias</strong>
  <ul>
    <li><a href="#" title="Ver las noticias más recientes">Recientes</a></li>
    <li><strong><a href="#" title="Ver las noticias más leídas">Más
leídas</a></strong></li>
    <li><a href="#" title="Ver las noticias más valoradas">Más valoradas</a></li>
  </ul>
</li>
<li>
  Artículos
  <ol>
    <li><strong>XHTML</strong></li>
    <li>CSS</li>
    <li>JavaScript</li>
    <li>Otros</li>
  </ol>

```

```

</li>
<li>
  Contacto
  <dl>
    <dt><em>Email</em></dt>
    <dd><strong>nombre@direccion.com</strong></dd>
    <dt><em>Teléfono</em></dt>
    <dd>900 900 900</dd>
    <dt><em>Fax</em></dt>
    <dd>900 900 900</dd>
  </dl>
</li>
</ul>

</body>
</html>

```

15.10. Ejercicio 10

Página principal del portfolio

```

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/
DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="es" xml:lang="es">

<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
<link rel="shortcut icon" href="../favicon.ico" type="image/x-icon" />
<title>Mi Sitio</title>
</head>

<body>

<p><a href="portfolio/indice.html" title="Página principal de Mi Sitio"
rel="index">Volver a la pagina principal</a></p>

<h1>Ultimos proyectos</h1>

<p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Donec iaculis posuere
justo. Nam vel neque.</p>

<h3>Proyecto 1</h3>

<p>Etiam consectetur, mauris vitae cursus scelerisque, dui turpis dignissim justo, et
eismod enim odio sit amet erat.</p>

<p></p>

<h3>Proyecto 2</h3>

```

```

<p>Etiam consectetuer, mauris vitae cursus scelerisque, dui turpis dignissim justo, et
euismod enim odio sit amet erat.</p>

<p></p>

</body>

</html>

```

Descargar ZIP con la solución completa (/ejercicios/xhtml/soluciones/ejercicio10/ejercicio10.zip)

15.11. Ejercicio 11

```

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/
DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="es" xml:lang="es">

<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
<title>Ejemplo de tabla sencilla</title>
</head>

<body>

<h1>Su pedido</h1>
<table>
  <tr>
    <th scope="col">Nombre producto</th>
    <th scope="col">Precio unitario</th>
    <th scope="col">Unidades</th>
    <th scope="col">Subtotal</th>
  </tr>

  <tr>
    <td>Reproductor MP3 (80 GB)</td>
    <td>192.02</td>
    <td>1</td>
    <td>192.02</td>
  </tr>

  <tr>
    <td>Fundas de colores</td>
    <td>2.50</td>
    <td>5</td>
    <td>12.50</td>
  </tr>

  <tr>
    <td>Reproductor de radio & control remoto</td>
    <td>12.99</td>

```

```

    <td>1</td>
    <td>12.99</td>
</tr>

<tr>
  <th scope="row">TOTAL</th>
  <td>-</td>
  <td>7</td>
  <td><strong>207.51</strong></td>
</tr>
</table>

</body>

</html>

```

15.12. Ejercicio 12

```

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/
DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="es" xml:lang="es">

<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
<title>Ejemplo de tabla avanzada</title>
</head>

<body>

<h1>Resultado de la búsqueda</h1>

<table summary="Tabla con los datos de los resultados de la búsqueda">
  <caption>Resultado de la búsqueda</caption>
  <tr>
    <th abbr="Imagen del producto" scope="col">Imagen</th>
    <th abbr="Datos del producto" scope="col">Datos</th>
  </tr>

  <tr>
    <td>
      
    </td>
    <td>
      <h4><a href="#" title="Ver más información sobre el portátil">Portátil - 3 GHz -
4 GB RAM</a></h4>
      <p><a href="#" title="Comprar el portátil">Comprar:</a> <del>2.990 &euro;</del>
<strong>2.599 &euro;</strong></p>
    </td>
  </tr>

```

```

<tr>
  <td></td>
  <td>
    <h4><a href="#" title="Ver más información sobre la videocámara">Videocámara -
Alta definición 1080p - 60 GB</a></h4>
    <p><a href="#" title="Comprar la videocámara">Comprar:</a> <del>1.099
&euro;</del> <strong>999 &euro;</strong></p>
  </td>
</tr>

<tr>
  <td></td>
  <td>
    <h4><a href="#" title="Ver más información sobre el televisor">Televisor - 46" -
Full HD</a></h4>
    <p><a href="#" title="Comprar el televisor">Comprar:</a> <del>1.999 &euro;</del>
<strong>1.799 &euro;</strong></p>
  </td>
</tr>

<tr>
  <td></td>
  <td>
    <h4><a href="#" title="Ver más información sobre el móvil">Móvil - 3G - Wi-Fi - 8
GB</a></h4>
    <p><a href="#" title="Comprar el móvil">Comprar:</a> <del>399 &euro;</del>
<strong>349 &euro;</strong></p>
  </td>
</tr>
</table>

</body>

</html>

```

Descargar ZIP con la solución completa (</ejercicios/xhtmll/soluciones/ejercicio12/ejercicio12.zip>)

15.13. Ejercicio 13

```

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/
DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="es" xml:lang="es">

<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
<title>Ejemplo de tabla compleja</title>
</head>

<body>

```



```

<h3>Comparativa de reproductores MP3</h3>

<table summary="Tabla comparativa de las características técnicas de los reproductores
MP3">
  <caption>Tabla comparativa de las características técnicas de los reproductores
MP3</caption>
  <tr>
    <th></th>
    <th abbr="Reproductor mini" scope="col" colspan="3">
      
      
      <br/><strong>MP3 mini</strong>
    </th>
    <th abbr="Reproductor estándar" scope="col" colspan="2">
      
      
      <br/><strong>MP3 grande</strong>
    </th>
  </tr>

  <tr>
    <th scope="row" abbr="Capacidad">Capacidad de almacenamiento</th>
    <td>4GB <br/> (1.000 canciones)</td>
    <td>8GB <br/> (2.000 canciones)</td>
    <td>16GB <br/> (4.000 canciones)</td>
    <td>30GB <br/> (7.500 canciones)</td>
    <td>80GB <br/> (20.000 canciones)</td>
  </tr>

  <tr>
    <th scope="row" abbr="Colores disponibles">Colores</th>
    <td>
      
    </td>
    <td>
      
      
      
      
    </td>
    <td>
      
    </td>
    <td colspan="2">
      
      
    </td>
  </tr>

```

```

    </td>
</tr>

<tr>
  <th scope="row" abbr="Tamaño de pantalla">Pantalla</th>
  <td colspan="3">LCD de 3 cm (diagonal) con retroiluminación</td>
  <td colspan="2">LCD de 6 cm (diagonal) con retroiluminación</td>
</tr>

<tr>
  <th rowspan="2" scope="row" abbr="Tiempo de carga">Tiempo de carga</th>
  <td rowspan="2" colspan="3">Unas 3 horas</td>
  <td colspan="2">Unas 4 horas</td>
</tr>

<tr>
  <td colspan="2">Unas 2 horas para alcanzar el 80% de la capacidad</td>
</tr>
</table>

</body>

</html>

```

Descargar ZIP con la solución completa (</ejercicios/xhtml/soluciones/ejercicio13/ejercicio13.zip>)

15.14. Ejercicio 14

```

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/
DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="es" xml:lang="es">

<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
<title>Rellena tu CV</title>
</head>

<body>

<h3>Rellena tu CV</h3>

<form action="/php/insertar_cv.php" method="post" enctype="multipart/form-data">
Nombre <br/>
<input type="text" name="nombre" value="" size="20" maxlength="30" />

<br/>

Apellidos <br/>
<input type="text" name="apellidos" value="" size="50" maxlength="80" />

```

```
<br/>

Contraseña <br/>
<input type="password" name="contrasena" value="" maxlength="10" />

<br/>

DNI <br/>
<input type="text" name="dni" value="" size="10" maxlength="9" />

<br/>

Sexo <br/>
<input type="radio" name="sexo" value="hombre" checked="checked" /> Hombre <br/>
<input type="radio" name="sexo" value="mujer" /> Mujer

<br/><br/>

Incluir mi foto <input type="file" name="foto" />

<br/><br/>

<input name="suscribir" type="checkbox" value="suscribir" checked="checked"/>
Suscribirme al boletín de novedades

<br/><br/>

<input type="submit" name="enviar" value="Guardar cambios" />
<input type="reset" name="limpiar" value="Borrar los datos introducidos" />

</form>

</body>

</html>
```

15.15. Ejercicio 15

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/
DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="es" xml:lang="es">

<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
<title>Rellena tu CV</title>
</head>

<body>
```

```
<h3>Rellena tu CV</h3>

<form action="/php/insertar_cv.php" method="post" enctype="multipart/form-data">

<fieldset>
  <legend>Datos personales</legend>

  <label for="provincia">Provincia</label> <br/>
  <select id="provincia" name="provincia">
    <option value="" selected="selected">- selecciona -</option>
    <option value="01">Álava/Araba</option>
    <option value="02">Albacete</option>
    <option value="03">Alicante/Alacant</option>
    <option value="04">Almería</option>
    <option value="33">Asturias</option>
    <option value="05">Ávila</option>
    <option value="06">Badajoz</option>
    <option value="07">Balears (Illes)</option>
    <option value="08">Barcelona</option>
    <option value="09">Burgos</option>
    <option value="10">Cáceres</option>
    <option value="11">Cádiz</option>
    <option value="39">Cantabria</option>
    <option value="12">Castellón/Castelló</option>
    <option value="51">Ceuta</option>
    <option value="13">Ciudad Real</option>
    <option value="14">Córdoba</option>
    <option value="15">Coruña (A)</option>
    <option value="16">Cuenca</option>
    <option value="17">Girona</option>
    <option value="18">Granada</option>
    <option value="19">Guadalajara</option>
    <option value="20">Guipúzcoa/Gipuzkoa</option>
    <option value="21">Huelva</option>
    <option value="22">Huesca</option>
    <option value="23">Jaén</option>
    <option value="24">León</option>
    <option value="27">Lugo</option>
    <option value="25">Lleida</option>
    <option value="28">Madrid</option>
    <option value="29">Málaga</option>
    <option value="52">Melilla</option>
    <option value="30">Murcia</option>
    <option value="31">Navarra</option>
    <option value="32">Ourense</option>
    <option value="34">Palencia</option>
    <option value="35">Palmas (Las)</option>
    <option value="36">Pontevedra</option>
    <option value="26">Rioja (La)</option>
    <option value="37">Salamanca</option>
```

```
<option value="38">Santa Cruz de Tenerife</option>
<option value="40">Segovia</option>
<option value="41">Sevilla</option>
<option value="42">Soria</option>
<option value="43">Tarragona</option>
<option value="44">Teruel</option>
<option value="45">Toledo</option>
<option value="46">Valencia/València</option>
<option value="47">Valladolid</option>
<option value="48">Vizcaya/Bizkaia</option>
<option value="49">Zamora</option>
<option value="50">Zaragoza</option>
</select>

<br/><br/>

<label for="fecha_dia">Fecha de nacimiento</label> <br/>
<input type="text" size="3" maxlength="2" id="fecha_dia" name="fecha_dia" />
de
<select id="fecha_mes" name="fecha_mes">
  <option value="1">Enero</option>
  <option value="2">Febrero</option>
  <option value="3">Marzo</option>
  <option value="4">Abril</option>
  <option value="5">Mayo</option>
  <option value="6">Junio</option>
  <option value="7">Julio</option>
  <option value="8">Agosto</option>
  <option value="9">Septiembre</option>
  <option value="10">Octubre</option>
  <option value="11">Noviembre</option>
  <option value="12">Diciembre</option>
</select>
de
<input type="text" size="5" maxlength="4" id="fecha_ano" name="fecha_ano" />

<br/><br/>

<label for="temasDeInteres">Temas de interés</label> <br/>
<select multiple="multiple" size="5" id="temasDeInteres" name="temasDeInteres">
  <option value="3105">Administración de bases de datos</option>
  <option value="3106">Análisis y programación</option>
  <option value="3107">Arquitectura</option>
  <option value="3108">Calidad</option>
  <option value="3109">ERP, CRM, Business Intelligence</option>
  <option value="3110">Gestión de proyectos</option>
  <option value="3111">Hardware, redes y seguridad</option>
  <option value="3112">Helpdesk</option>
  <option value="3113">Sistemas</option>
  <option value="3114">Telecomunicaciones</option>
```

```

    </select>
</fieldset>

</form>

</body>

</html>

```

15.16. Ejercicio 16

```

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/
DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="es" xml:lang="es">

<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
<title>Información sobre el producto</title>
</head>

<body>

<h3>Información sobre el producto</h3>

<form action="/php/insertar_subasta.php" method="post" enctype="multipart/form-data">

<fieldset>
  <legend>Datos básicos</legend>

  <label for="nombre">Nombre</label> <br/>
  <input type="text" name="nombre" id="nombre" size="50" maxlength="250" />

  <br/><br/>

  <label for="descripcion">Descripción</label> <br/>
  <textarea name="descripcion" id="descripcion" cols="40" rows="5"></textarea>

  <br/><br/>

  Foto <input type="file" name="foto" />

  <br/><br/>

  <input name="contador" type="checkbox" value="si" /> Añadir contador de visitas
</fieldset>

<fieldset>
  <legend>Datos económicos</legend>

  <label for="precio">Precio</label>

```


Descarga más libros sobre diseño y programación web en:

librosweb.es

LIBROSWEB