

Módulo I

XHTML y CSS

U.N.E.D.

XHTML

1.1 INTRODUCCIÓN

HTML nació en 1980 como un proyecto de Tim Berners-Lee basado en el concepto de hipertexto, que ayudaría a investigadores a compartir información en forma de documentos sobre Internet. Fue implementado más tarde en 1989 en la CERN (organización europea para la investigación nuclear), el nodo más grande en Europa. Desde allí, HTML comenzó su evolución que no está aún concluida, pasando por las versiones 2.0, 3.2, 4.0 y 4.01, todas ellas basadas en SGML (lenguaje de etiquetado estándar generalizado: un metalenguaje usado para crear otros lenguajes como sublenguajes del mismo).

Por otro lado, XML (lenguaje de marcas extensible) es también un metalenguaje (usado para crear otros lenguajes) y es también un sublenguaje de SGML, diseñado para ser más simple de procesar. En estos días, XML es ampliamente utilizado en diferentes formas para construir documentos y organizar información (por ejemplo, RSS (redifusión realmente simple), Atom, etc.) ya que provee una forma estándar de lograrlo que es más fácil de procesar que SGML.

En el año 2000, XHTML es recomendado por el World Wide Web Consortium (W3C) como la nueva versión estándar de HTML basada en XML en lugar de SGML. De esta forma, podemos considerar a XHTML como el resultado de mezclar HTML y XML. Hecho esto, todos los beneficios de XML son ahora heredados por HTML lo que lo hace más fácil de procesar, y por lo tanto estar disponible en más plataformas con capacidades de procesamiento reducidas (por ejemplo, PDAs (asistente digital personal) y teléfonos celulares).

Otro motivo para actualizar las versiones de HTML y para la creación del W3C es el re establecimiento del propósito original de HTML como un lenguaje semántico. Desde que fue implementado, muchos fabricantes de navegadores comenzaron a transformar el estándar con el objeto de agregarle más funcionalidad. Esto lo convirtió lentamente en un lenguaje más visual que semán-

tico, lo que inspiró al W3C a crear nuevos estándares pensados para revertir este efecto y retor-
narlo a su origen semántico. XHTML 1.1 es la más reciente de estas actualizaciones pero hay
más por venir.

En este tema se estudiarán los principios del XHTML y el uso de este para comenzar a diseñar
páginas web basadas en los estándares actuales.

Introducción a **XHTML**

Javier Eguíluz Pérez

Sobre este libro...

- Los contenidos de este libro están bajo una licencia Creative Commons Reconocimiento - No Comercial - Sin Obra Derivada 3.0 (<http://creativecommons.org/licenses/by-nc-nd/3.0/deed.es>)
- **Esta versión impresa se creó el 4 de octubre de 2008 y todavía está incompleta.** La versión más actualizada de los contenidos de este libro se puede encontrar en <http://www.librosweb.es/xhtml>
- Si quieras aportar sugerencias, comentarios, críticas o informar sobre errores, puedes enviarnos un mensaje a **contacto@librosweb.es**

Capítulo 1. Introducción	5
1.1. ¿Qué es HTML?.....	5
1.2. Breve historia de HTML	5
1.3. Especificación oficial.....	7
1.4. HTML y XHTML	7
1.5. HTML y CSS	8
Capítulo 2. Características básicas	9
2.1. Lenguajes de etiquetas	9
2.2. El primer documento HTML	10
2.3. Etiquetas y atributos.....	13
2.4. Elementos HTML.....	16
2.5. Sintaxis de las etiquetas XHTML.....	19
Capítulo 3. Texto	21
3.1. Estructurar	22
3.2. Marcado básico de texto	26
3.3. Marcado avanzado de texto	32
3.4. Marcado genérico de texto	35
3.5. Espacios en blanco y nuevas líneas	36
3.6. Codificación de caracteres.....	44
Capítulo 4. Enlaces	48
4.1. URL.....	48
4.2. Enlaces relativos y absolutos	51
4.3. Enlaces básicos	55
4.4. Enlaces avanzados	59
4.5. Otros tipos de enlaces	62
4.6. Ejemplos de enlaces habituales.....	64
Capítulo 5. Listas	67
5.1. Listas no ordenadas	67
5.2. Listas ordenadas	68
5.3. Listas de definición	69
Capítulo 6. Imágenes y objetos	73
6.1. Imágenes	73
6.2. Mapas de imagen	76
6.3. Objetos	78
Capítulo 7. Tablas	82
7.1. Tablas básicas	82
7.2. Tablas avanzadas	94
Capítulo 8. Formularios	100
8.1. Formularios básicos	100
8.2. Elementos de formulario	102
8.3. Formularios avanzados	109
8.4. Otros elementos de formulario	111

Capítulo 9. Estructura y layout	119
Capítulo 10. Metainformación.....	122
10.1. Estructura de la cabecera	122
10.2. Metadatos	123
10.3. DOCTYPE	125
Capítulo 11. Otras etiquetas importantes	127
11.1. Comentarios	127
11.2. JavaScript.....	127
11.3. CSS	129
11.4. Iframes.....	130
11.5. Otras etiquetas	131
Capítulo 12. Accesibilidad.....	134
12.1. Requisitos del nivel A de accesibilidad	134
Capítulo 13. Validación.....	137
13.1. Validación con Dreamweaver	137
13.2. Validador del W3C	139
13.3. Otros validadores	140
Capítulo 14. Fragmentos de código.....	143
14.1. Documento XHTML.....	148
14.2. Cabecera XHTML	149
14.3. Tabla	149
14.4. Formulario	150
Capítulo 15. Ejercicios resueltos	152

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).

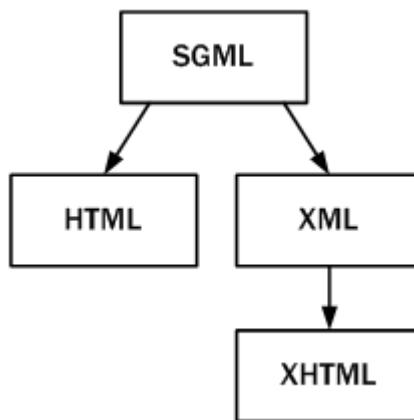


Figura 1.1. Esquema de la evolución de HTML y XHTML

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:

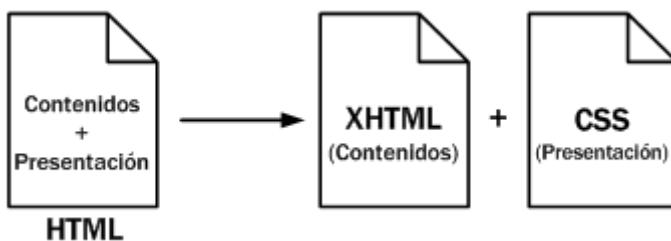


Figura 1.2. Esquema de la separación de los contenidos y su presentación

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.

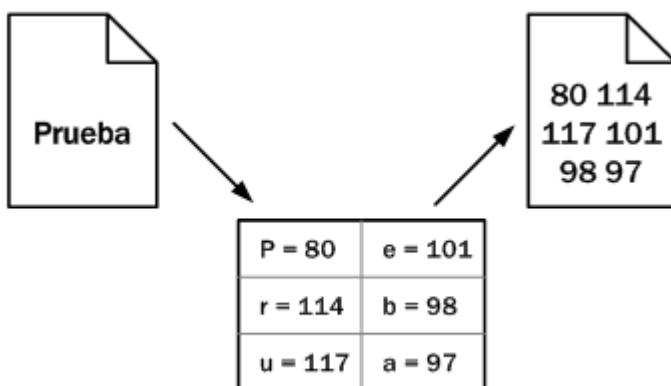


Figura 2.1. Ejemplo sencillo de codificación de caracteres

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.

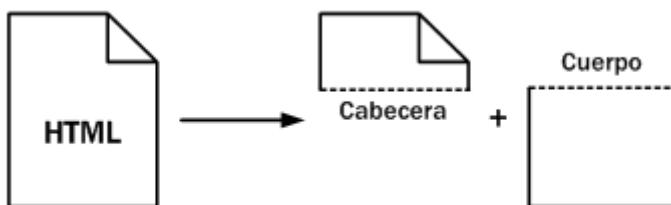


Figura 2.2. Esquema de las partes que forman un documento HTML

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 quieras 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:

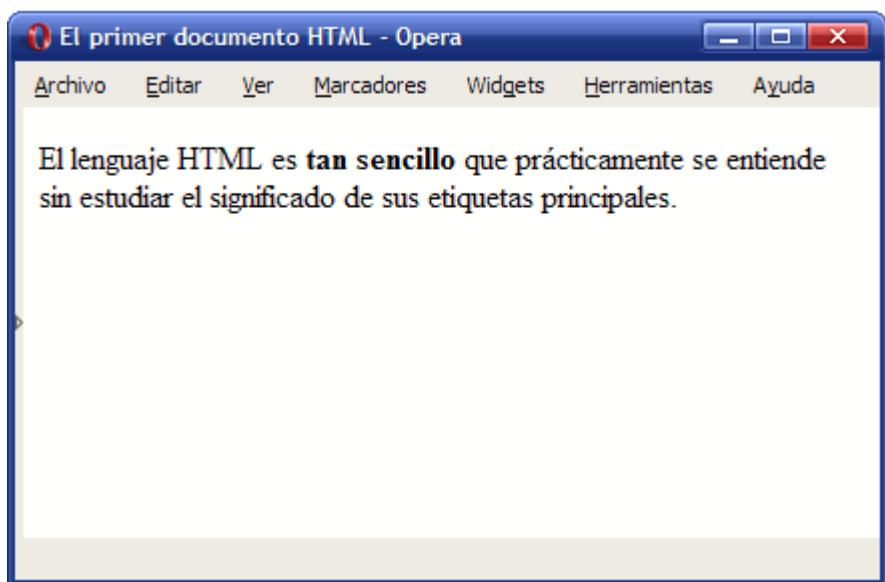


Figura 2.3. Aspecto que muestra el primer documento HTML en cualquier navegador

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.

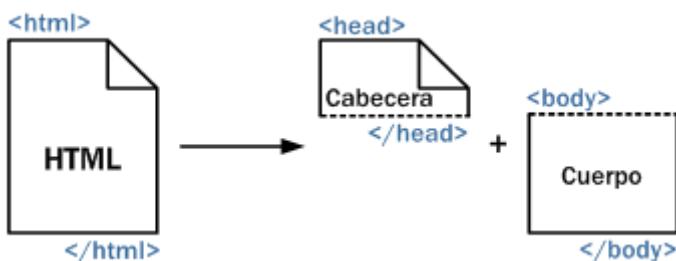


Figura 2.4. Esquema de las etiquetas principales que contiene un documento HTML

Ejercicio 1

Determinar el código HTML correspondiente a la siguiente página:

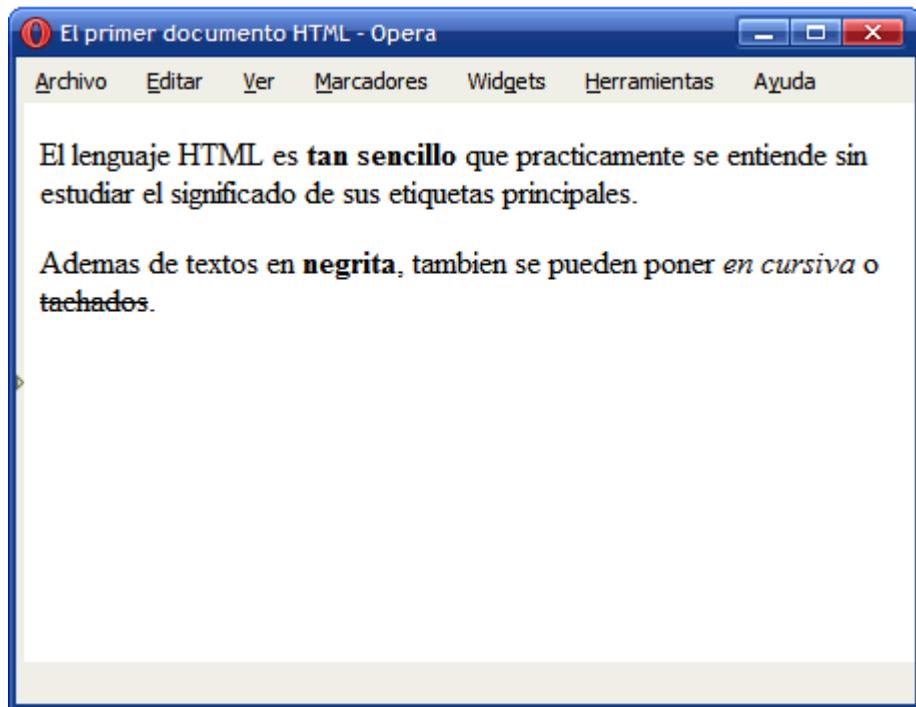


Figura 2.5. Página HTML sencilla que resalta algunas partes del texto

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>
```

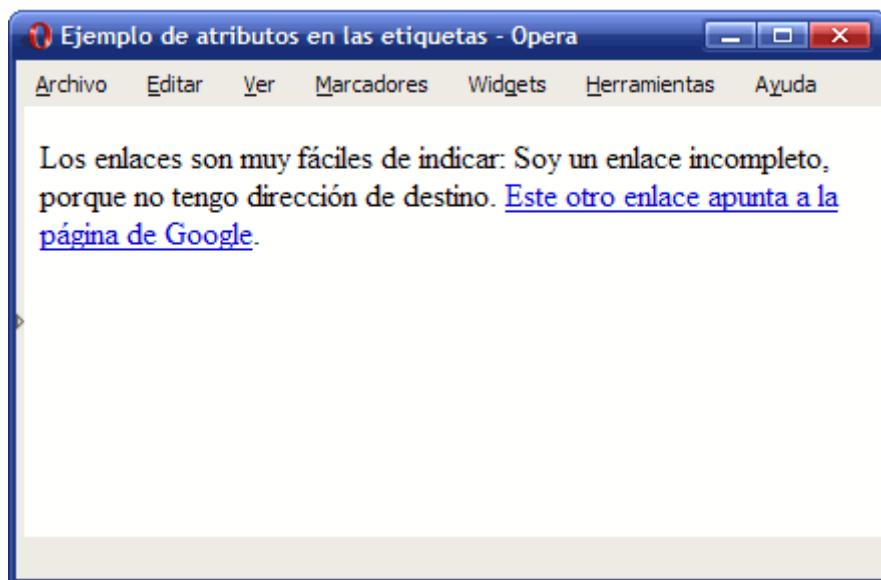


Figura 2.6. Los atributos permiten personalizar las etiquetas HTML

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:

1) **Atributos básicos:** se pueden utilizar prácticamente en 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) Atributos para internacionalización: los utilizan las páginas que muestran sus contenidos en varios idiomas o 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`.

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**).

3) Atributos de eventos: sólo se utilizan en las páginas web dinámicas creadas con JavaScript.

Atributo	Descripción
<code>onclick, ondblclick, onmousedown, onmouseup, onmouseover, onmousemove, onmouseout, onkeypress, onkeydown, onkeyup</code>	Permiten controlar los eventos producidos sobre cada elemento 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 atributos anteriores, es posible responder de forma adecuada a cada evento.

4) Atributos para los elementos que pueden obtener el foco:

Cuando el usuario selecciona un elemento de la interfaz de 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`.

En el resto de la documentación, se emplearán las palabras "*básicos*", "*i18n*", "*eventos*" y "*foco*" respectivamente para referirse a cada uno de los grupos de atributos comunes definidos anteriormente.

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:

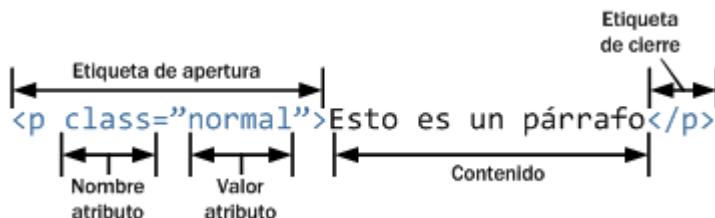


Figura 2.7. Esquema de las partes que componen un elemento HTML

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 (Este 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):

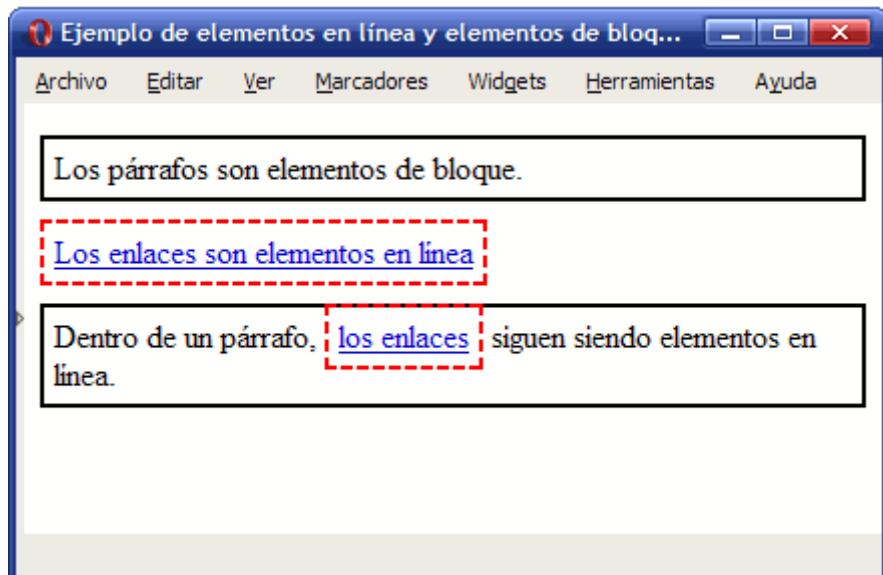


Figura 2.8. Diferencias entre elementos en línea y elementos de bloque

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, hr, isindex, menu, noframes, nos-cript, 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:

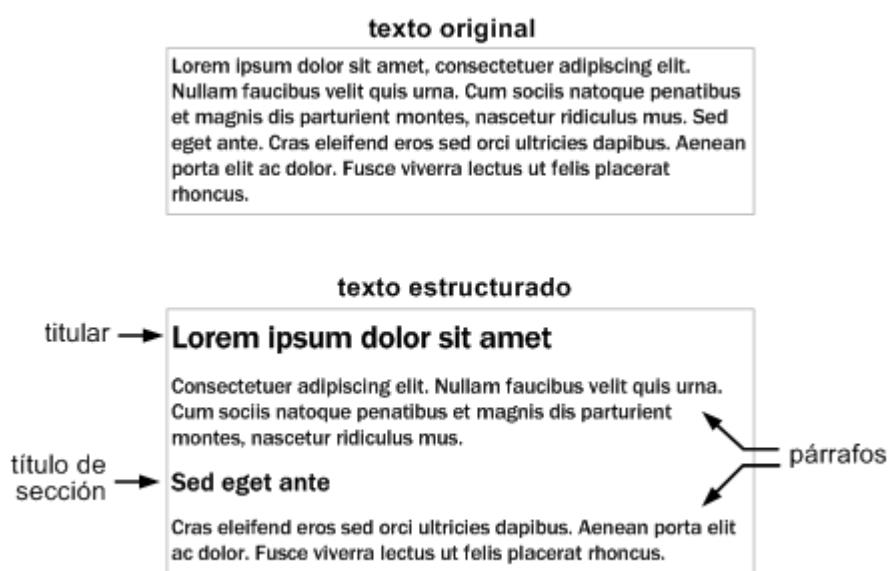


Figura 3.1. Resultado de estructurar un texto sencillo

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.

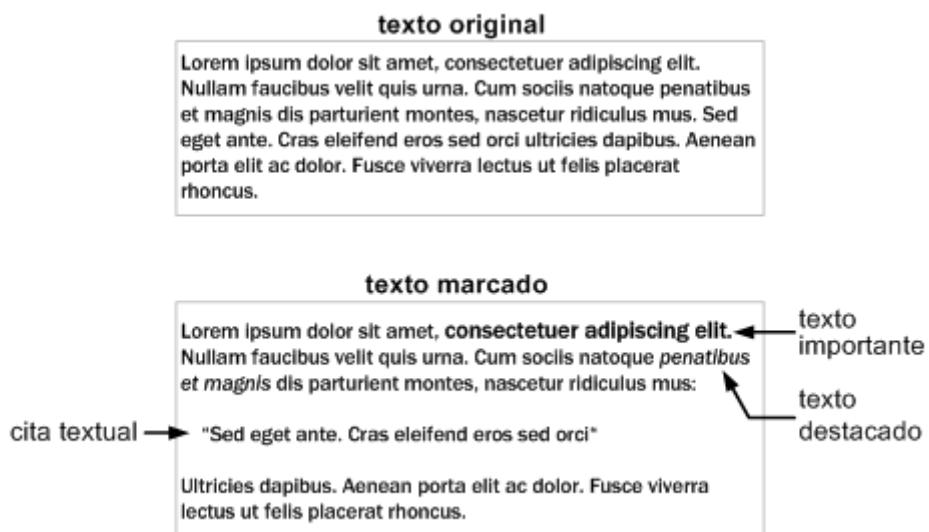


Figura 3.2. Resultado de marcar un texto sencillo

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:

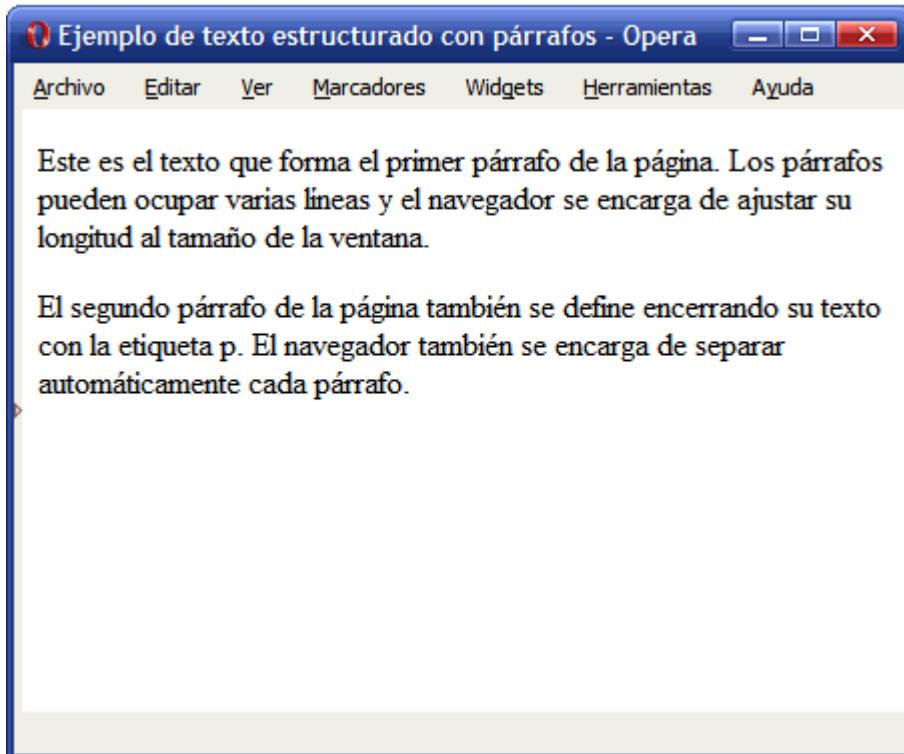


Figura 3.3. Ejemplo de texto HTML estructurado con párrafos

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

<p>	Párrafos
Atributos comunes	básicos, i18n y eventos
Atributos específicos	-
Tipo de elemento	Bloque
Descripción	Delimita el contenido de un párrafo de texto

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:

<h1>	Sección (titular) de nivel 1
Atributos comunes	básicos, i18n y eventos
Atributos específicos	-
Tipo de elemento	Bloque
Descripción	Define los títulos de las secciones de mayor importancia de la página.

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:

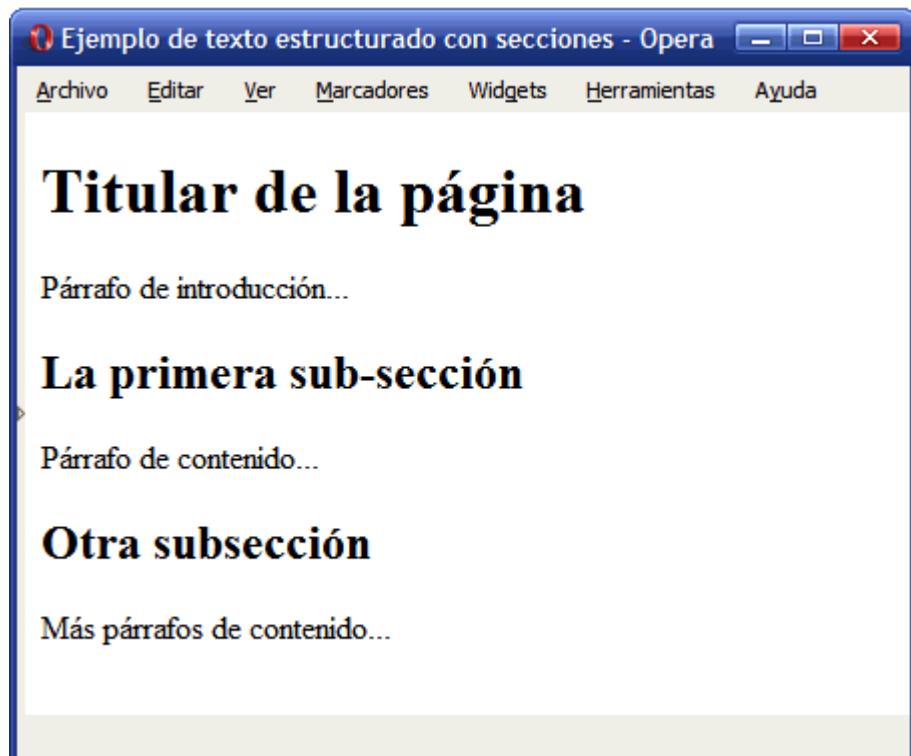


Figura 3.4. Ejemplo de texto HTML estructurado con párrafos y secciones

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:

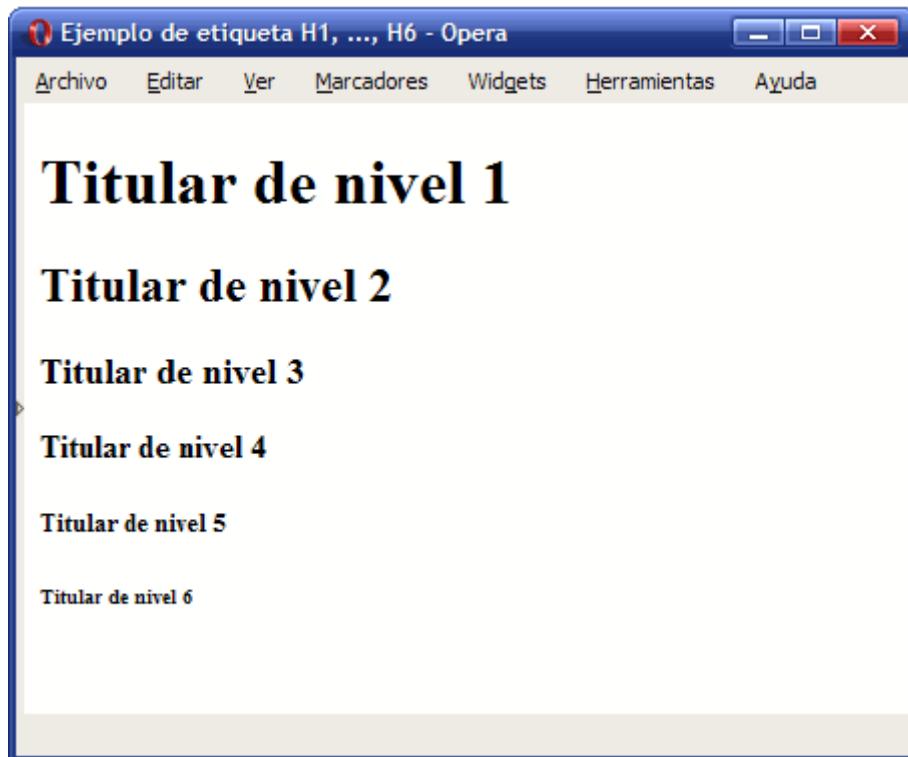


Figura 3.5. Ejemplo de uso de las etiquetas h1, h2, h3, h4, h5 y h6

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>	Énfasis
Atributos comunes	básicos, i18n y eventos
Atributos específicos	-
Tipo de elemento	En línea
Descripción	Realza la importancia del texto que encierra

	Énfasis más acentuado
Atributos comunes	básicos, i18n y eventos
Atributos específicos	-
Tipo de elemento	En línea
Descripción	Realza con la máxima importancia el texto que encierra

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:

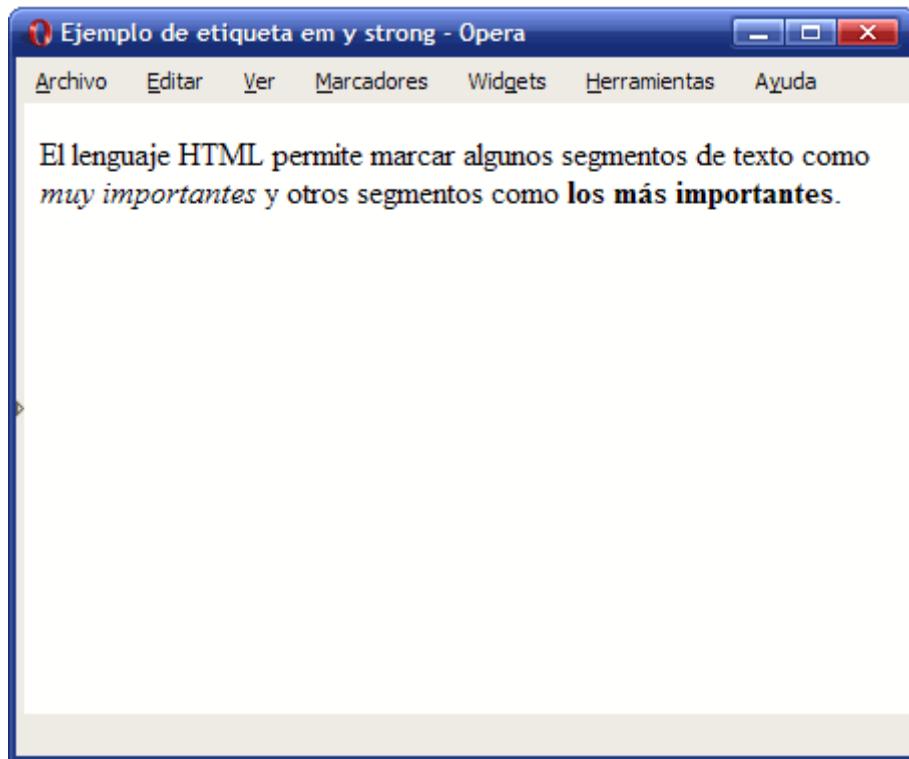


Figura 3.6. Ejemplo de uso de las etiquetas em y strong

Ejercicio 2

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

The screenshot shows a web browser window with the title bar 'La exploración espacial'. The menu bar includes 'Archivo', 'Edición', 'Ver', 'Marcadores', 'Widgets', 'Titulares', 'Herramientas', and 'Ayuda'. The main content area contains the following text:

La exploración espacial

La exploración espacial 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).

Las personas que pilotan naves espaciales, o son pasajeros en ellas, se llaman astronautas (en Rusia: *cosmonautas*; en China: *taikonautas*). 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).

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, *Somnium*, de Johannes Kepler, que narra un hipotético viaje a la Luna. Más tarde, en 1865, en una famosa obra de ficción titulada "*De la Terre à la Lune*", Julio Verne escribe sobre un grupo de hombres que viajó hasta la Luna usando un gigantesco cañón.

En Francia, Georges Méliès, uno de los pioneros del cine, tomaba la novela de Verne para crear "*Le voyage dans la Lune*" (1902), una de las primeras películas de ciencia ficción en la que describía un increíble viaje a la Luna. En obras como "*The War of the Worlds*" (1898) y "*The First Men in The Moon*" (1901), Herbert George Wells también se concibieron ideas de exploración del espacio y de contacto con civilizaciones extraterrestres.

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>	Inserción
Atributos comunes	básicos, i18n y eventos
Atributos específicos	<ul style="list-style-type: none"> ▪ cite = "url" - Indica la URL de la página en la que se puede obtener más información sobre el motivo por el que se realizó la modificación. ▪ datetime = "fecha" - Especifica la fecha y hora en la que se realizó el cambio
Tipo de elemento	Bloque y en línea
Descripción	Se emplea para marcar una modificación en los contenidos originales consistente en la inserción de un nuevo contenido

	Borrado
Atributos comunes	básicos, i18n y eventos
Atributos específicos	<ul style="list-style-type: none"> ▪ cite = "url" - Indica la URL de la página en la que se puede obtener más información sobre el motivo por el que se realizó la modificación. ▪ datetime = "fecha" - Especifica la fecha y hora en la que se realizó el cambio
Tipo de elemento	Bloque y en línea
Descripción	Se emplea para marcar una modificación en los contenidos originales consistente en el borrado de cierto contenido

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://www.librosweb.es/mas_informacion.html">marcado de
hipertexto</del> <ins datetime="20091026" cite="http://www.librosweb.es/
mas_informacion.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:

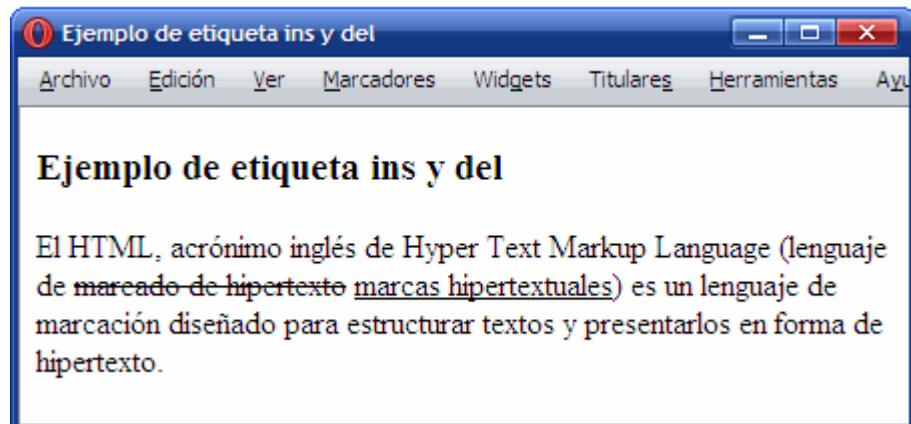


Figura 3.8. Ejemplo de uso de las etiquetas ins y del

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>	Citas
Atributos comunes	básicos, i18n y eventos
Atributos específicos	<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
Tipo de elemento	Bloque
Descripción	Se emplea para indicar que el texto que encierra es una cita textual de otro texto externo

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:

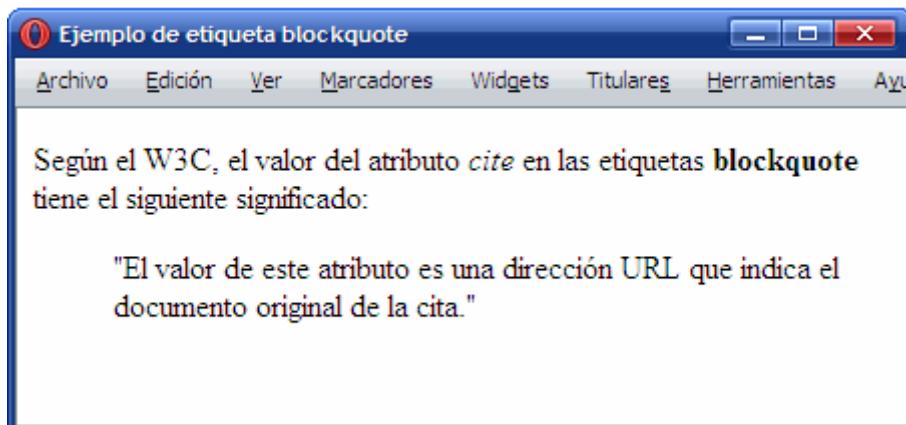


Figura 3.9. Ejemplo de uso de la etiqueta blockquote

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>	Abreviaturas
Atributos comunes	básicos, i18n y eventos
Atributos específicos	<ul style="list-style-type: none"> ▪ title = "texto" - Indica el significado completo de la abreviatura
Tipo de elemento	En línea
Descripción	Se emplea para marcar las abreviaturas del texto y proporcionar el significado de esas abreviaturas

<acronym>	Acrónimos o siglas
Atributos comunes	básicos, i18n y eventos
Atributos específicos	<ul style="list-style-type: none"> ▪ title = "texto" - Indica el significado completo del acrónimo o sigla
Tipo de elemento	En línea
Descripción	Se emplea para marcar las siglas o acrónimos del texto y proporcionar el significado de esas siglas

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`:

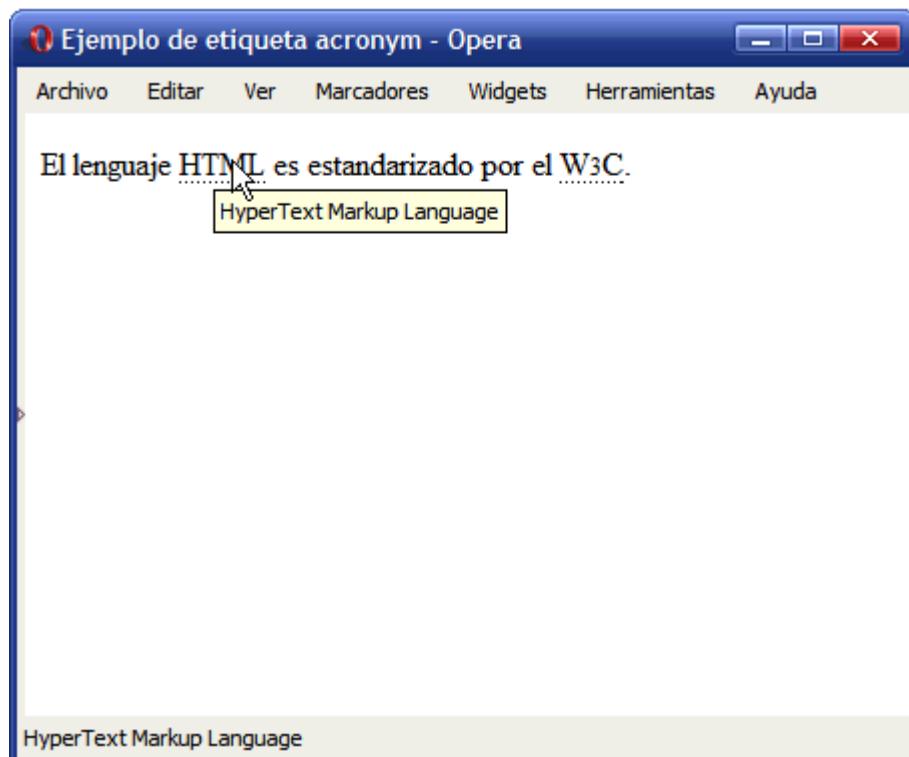


Figura 3.10. Ejemplo de uso de la etiqueta acronym

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>	Definición
Atributos comunes	básicos, i18n y eventos
Atributos específicos	<ul style="list-style-type: none"> ▪ title = "texto" - Indica el significado completo del término
Tipo de elemento	En línea
Descripción	Se emplea para marcar las definiciones de ciertos términos y proporcionar el significado de esos términos

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>	Cita
Atributos comunes	básicos, i18n y eventos
Atributos específicos	-
Tipo de elemento	En línea
Descripción	Se emplea para marcar una cita o una referencia a otras fuentes

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:

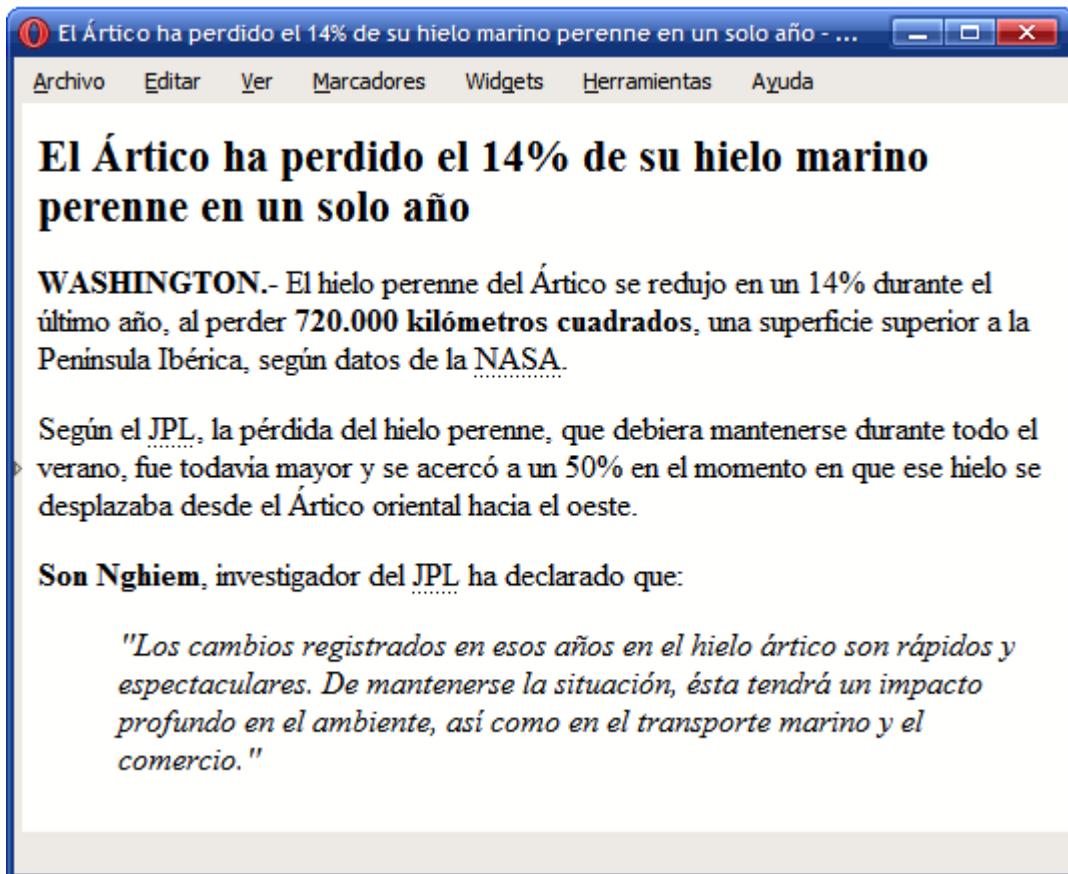


Figura 3.11. Texto HTML correctamente estructurado y marcado

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:

`Importante: 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:

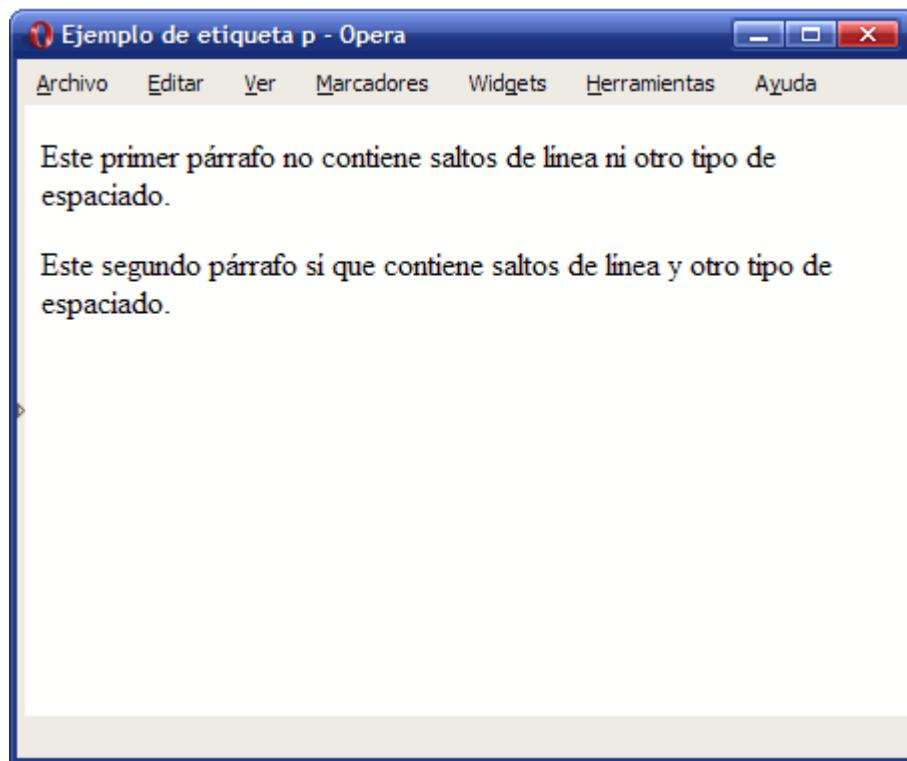


Figura 3.12. Ejemplo de comportamiento de HTML con los espacios en blanco

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>	Nueva línea
Atributos comunes	básicos
Atributos específicos	-
Tipo de elemento	En línea y etiqueta vacía
Descripción	Fuerza al navegador a insertar una nueva línea

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:

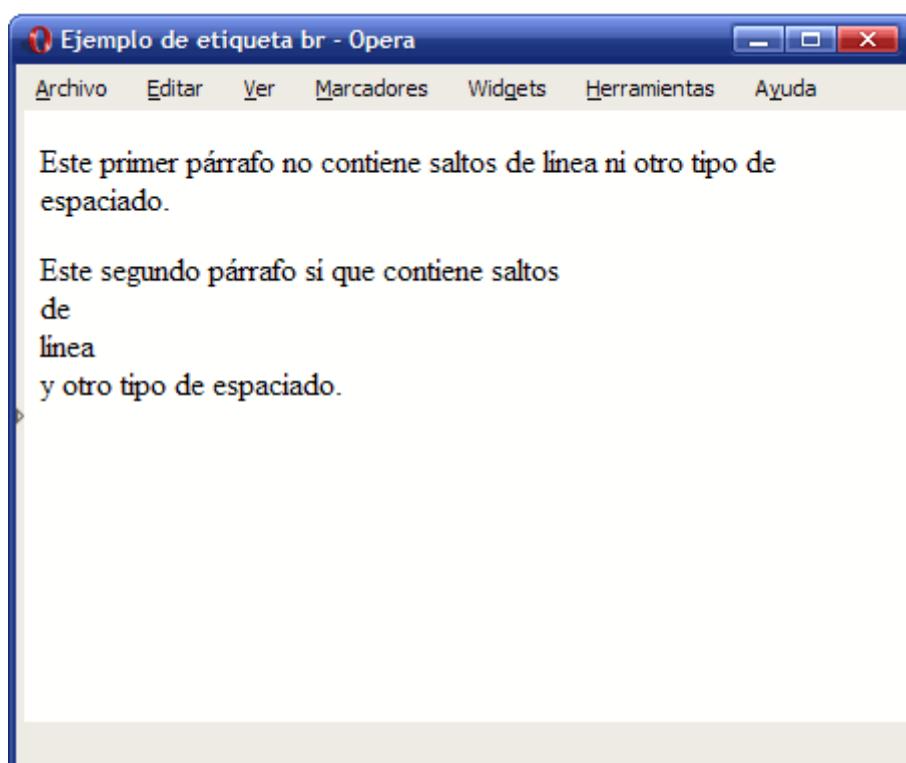


Figura 3.13. Ejemplo de uso de la etiqueta br

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; otro &nbsp; tipo &nbsp; de &nbsp; espaciado.</p>
</body>

</html>
```

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

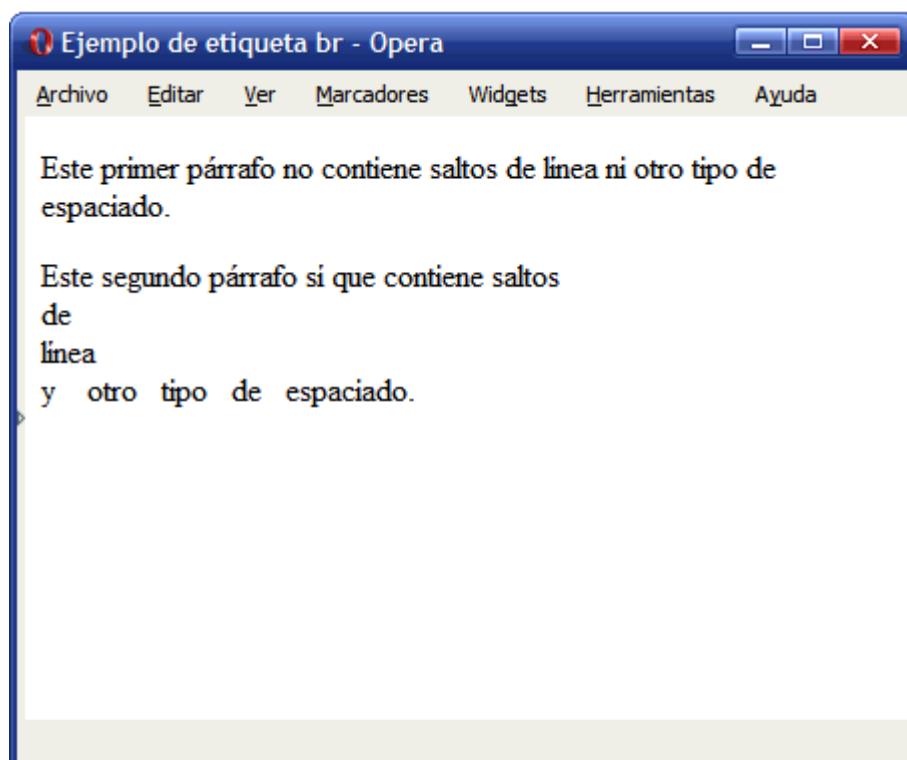


Figura 3.14. Ejemplo de uso de espacios en blanco en HTML

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:

The screenshot shows a Windows application window titled "Espacios en blanco arbitrarios". The menu bar includes Archivo, Edición, Ver, Marcadores, Widgets, Titulares, Herramientas, and Ayuda. The main content area displays a table with the following data:

Nombre	Diametro relativo	Periodo orbital	Número de lunas
<hr/>			
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 preformatado

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:

<pre>	Texto preformatado
Atributos comunes	básicos, i18n y eventos
Atributos específicos	-
Tipo de elemento	Bloque
Descripción	Muestra el texto que encierra tal y como está escrito (respetando los espacios en blanco)

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:

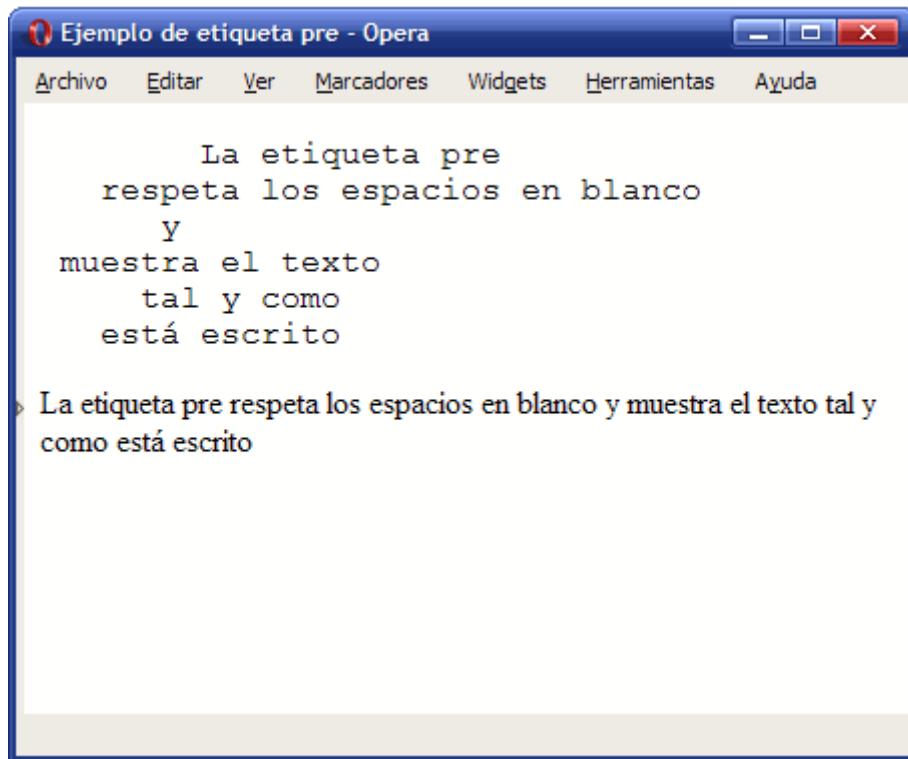


Figura 3.16. Ejemplo de uso de la etiqueta pre

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.

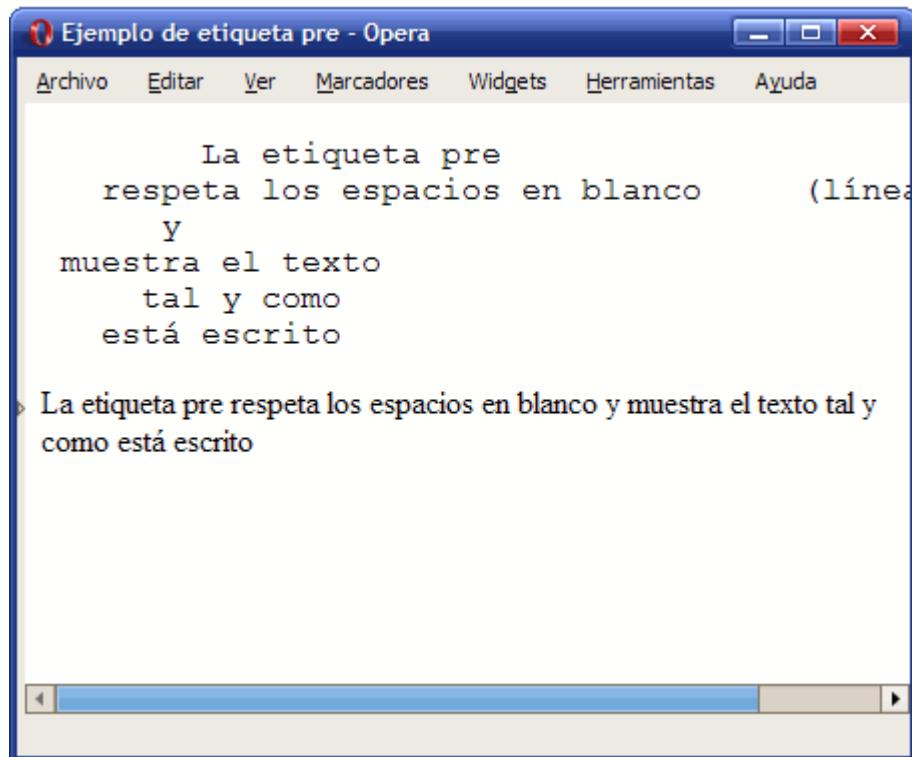


Figura 3.17. Ejemplo de aparición de scroll horizontal con la etiqueta pre

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>	Código fuente
Atributos comunes	básicos, i18n y eventos
Atributos específicos	-
Tipo de elemento	En línea
Descripción	Delimita el texto considerado un fragmento de código fuente

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 es similar a la  
etiqueta pre, sobre todo en el formato  
del texto.</p>  
  
</body>  
</html>
```

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

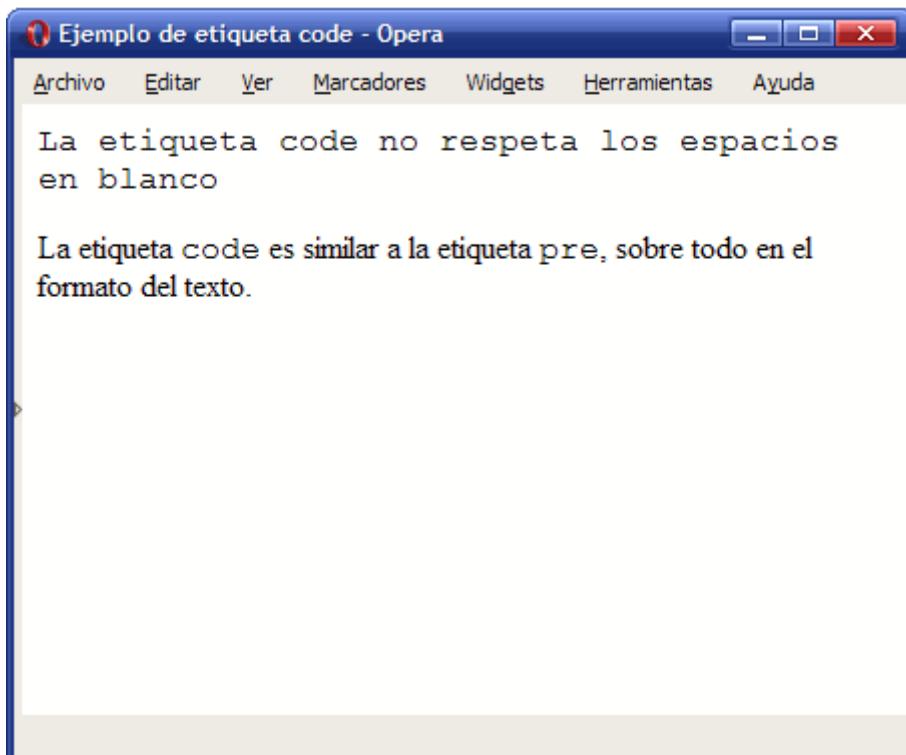


Figura 3.18. Ejemplo de uso de la etiqueta code

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, ñ, 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 <, >, " y & pueden dar problemas con los textos en HTML</p>

Ejercicio 5

Determinar el código HTML que corresponde al siguiente documento:

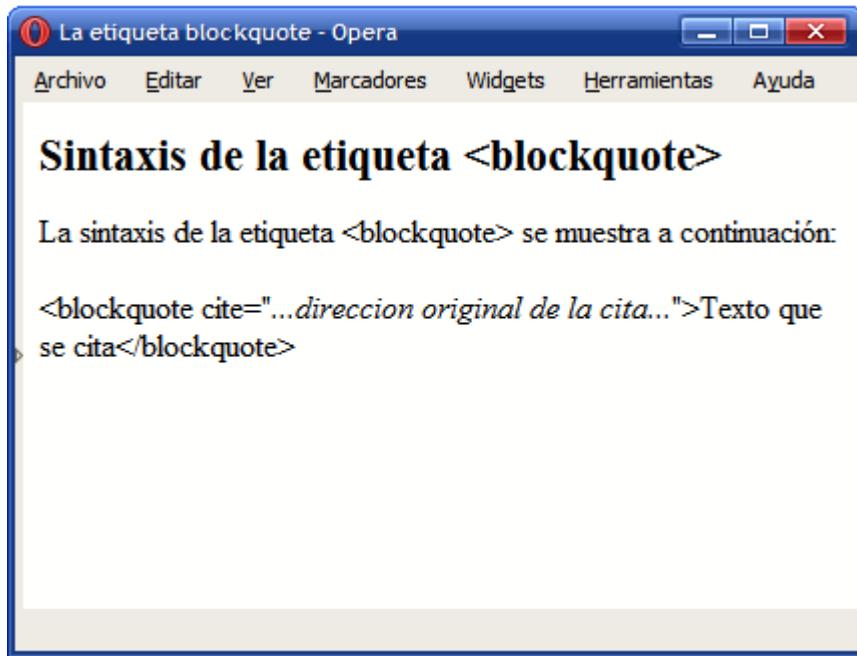


Figura 3.19. Texto HTML que incluye caracteres especiales

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
ñ	ñ	latin letter n with tilde
Ñ	Ñ	latin capital n letter with tilde
á	á	a acute
é	é	e acute
í	í	i acute
ó	ó	o acute
ú	ú	u acute
Á	Á	A acute
É	É	E acute
Í	Í	I acute
Ó	Ó	O acute
Ú	Ú	U acute
€	€	euro

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

```
| <p>Este p&aacute;rrafo 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`): *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`)
- Consulta (`?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`): 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		%7C

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

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. 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.

Origen	http://www.ejemplo.com/ruta1/ruta2/ruta3/pagina1.html
Recurso enlazado	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.

Origen	http://www.ejemplo.com/ruta1/ruta2/ruta3/pagina1.html
Recurso enlazado	Página web llamada pagina2.html y que se encuentra en el directorio superior llamado ruta2
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:

Origen	http://www.ejemplo.com/ruta1/ruta2/ruta3/pagina1.html
Recurso enlazado	Página web llamada pagina2.html y que se encuentra en el directorio superior llamado ruta1
URL absoluta	http://www.ejemplo.com/ruta1/pagina2.html
URL relativa	.. / .. / pagina2.html

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

Origen	http://www.ejemplo.com/ruta1/ruta2/ruta3/pagina1.html
Recurso enlazado	Página web llamada pagina2.html y que se encuentra en un directorio superior llamado ruta4 que se encuentra en la raíz del servidor
URL absoluta	http://www.ejemplo.com/ruta4/pagina2.html

URL relativa	.../../ruta4/pagina2.html
--------------	---------------------------

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.

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.

Origen	http://www.ejemplo.com/ruta1/ruta2/ruta3/pagina1.html
Recurso enlazado	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:

Origen	http://www.ejemplo.com/ruta1/ruta2/ruta3/pagina1.html
Recurso enlazado	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.

Origen	http://www.ejemplo.com/ruta1/ruta2/ruta3/pagina1.html
--------	---

Recurso enlazado	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
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>	Enlaces
Atributos comunes	básicos, i18n, eventos y foco
Atributos específicos	<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
Tipo de elemento	En línea
Descripción	Se emplea para enlazar todo tipo de recursos

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, consectetuer 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:

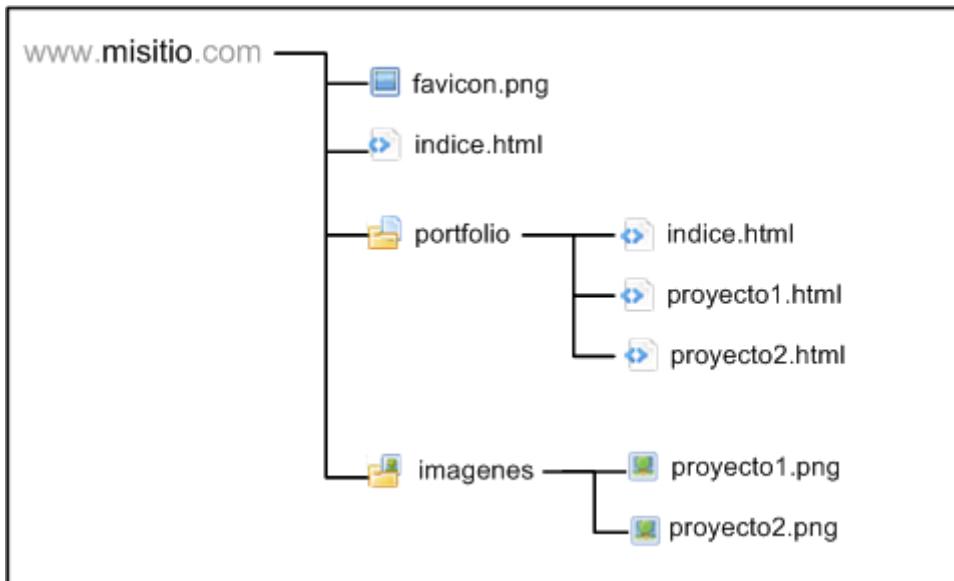


Figura 4.2. Estructura de archivos y directorios de un sitio web de ejemplo

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

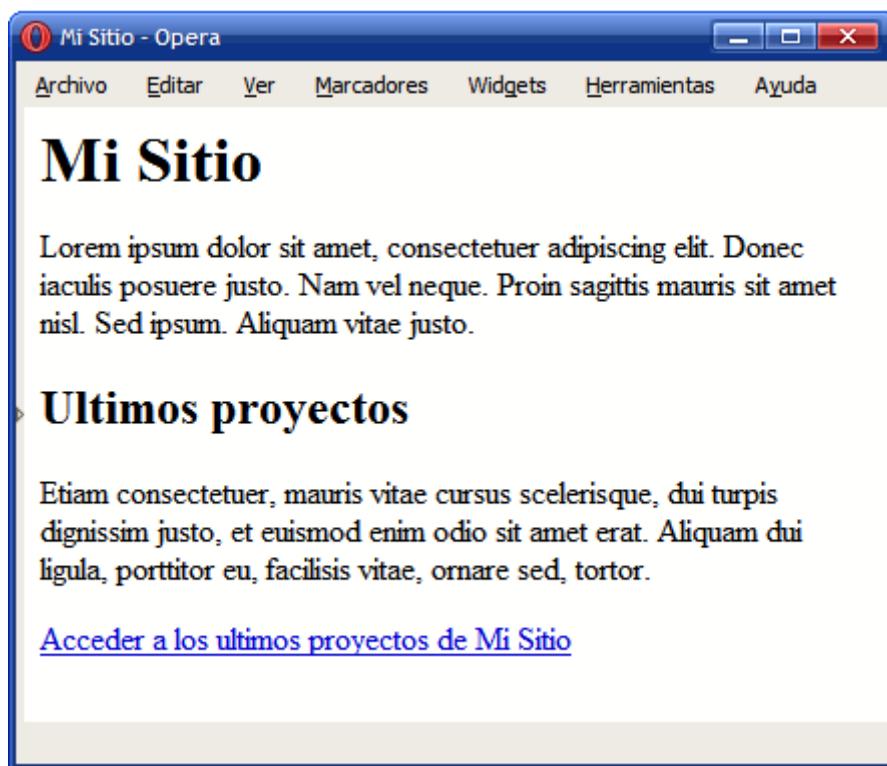


Figura 4.3. Página principal del sitio web de ejemplo

2) Crear la página de índice del portfolio:

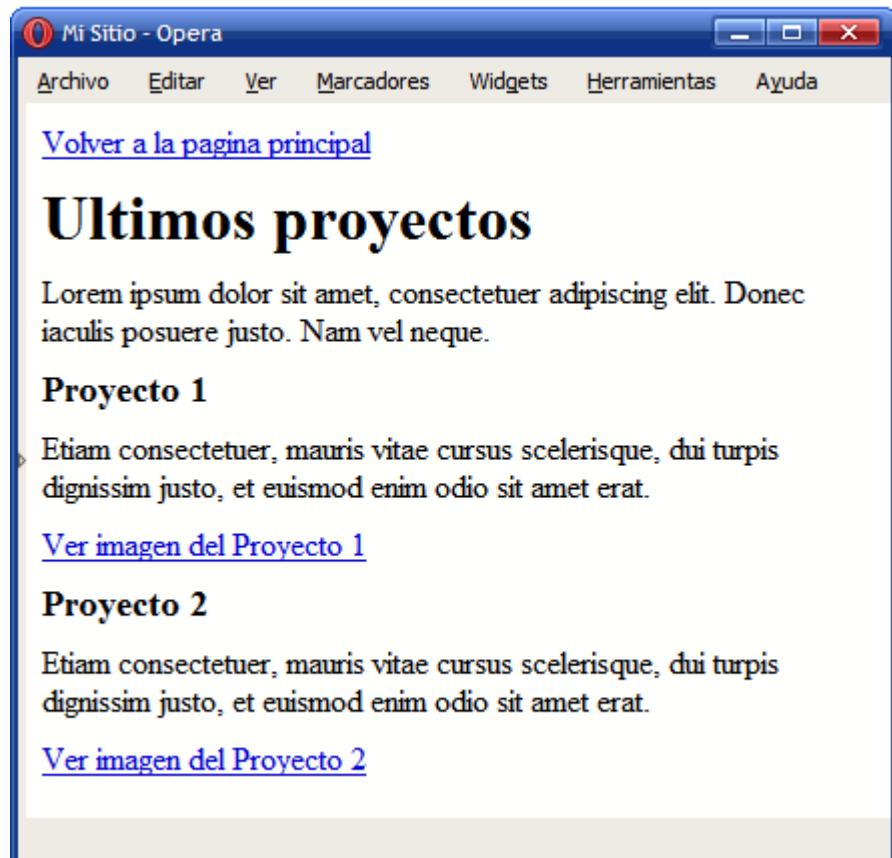


Figura 4.4. Página con la información sobre los proyectos realizados

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>	Enlaces
Atributos comunes	básicos, i18n, eventos y foco
Atributos específicos	<ul style="list-style-type: none"> ▪ name = "texto" - Permite nombrar al enlace para que se pueda acceder desde otros enlaces ▪ href = "url" - Indica la URL del recurso que se quiere enlazar ▪ hreflang = "codigo_idioma" - Idioma del recurso enlazado ▪ type = "tipo_de_contenido" - 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 ▪ rel = "tipo_de_relacion" - Describe la relación del documento actual con el recurso enlazado ▪ rev = "tipo_de_relacion" - Describe la relación del recurso enlazado con el documento actual ▪ charset = "tipo_de_charset" - Describe la codificación del recurso enlazado
Tipo de elemento	En línea
Descripción	Se emplea para enlazar todo tipo de recursos

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
en	Inglés	-
en-US	Inglés	Estados Unidos
es	Español	-
es-ES	Español	España
es-AR	Español	Argentina

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>	Código ejecutable
Atributos comunes	-
Atributos específicos	<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
Tipo de elemento	Bloque y en línea (también puede ser una etiqueta vacía)
Descripción	Se emplea para enlazar o definir un bloque de código (normalmente JavaScript)

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">
        //![CDATA[
            window.onload = function() { alert("La página se ha cargado completamente"); }
        //]]
    </script>
</head>
<body>
    ...
</body>
</html>
```

Cuando se incluye código JavaScript en la propia página XHTML, se debe insertar dentro de una sección especial llamada **CDATA**. Para ello, el código JavaScript se debe encerrar entre **<![CDATA[** y **]]>**. 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.

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 **//** al comienzo y al final de la sección **CDATA** son necesarios para los navegadores que no son capaces de procesar correctamente estas secciones.

La etiqueta **<script>** (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 (**<head>...</head>**).

La segunda etiqueta de XHTML para enlazar recursos es **<link>**, que permite enlazar y relacionar la página con otros recursos externos.

<link>	Enlazar recursos
Atributos comunes	básicos, i18n y eventos
Atributos específicos	<ul style="list-style-type: none"> ▪ Los siguientes con el mismo significado que para la etiqueta "a": charset, href, hreflang, type, rel y rev ▪ media = "tipo_de_medio" - Indica el medio para el que debe aplicarse la relación
Tipo de elemento	Etiqueta vacía
Descripción	Se emplea para enlazar y establecer relaciones entre el documento y otros recursos

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 ícono para favoritos es el pequeño ícono que muestran las páginas en varias partes del navegador. Dependiendo del navegador que se utilice, este ícono 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.

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 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>	Lista no ordenada
Atributos comunes	básicos, i18n y eventos
Atributos específicos	-
Tipo de elemento	Bloque
Descripción	Se emplea para definir listas no ordenadas

<code></code>	Elemento de una lista
Atributos comunes	básicos, i18n y eventos
Atributos específicos	-
Tipo de elemento	Bloque
Descripción	Se emplea para definir los elementos de las listas (ordenadas y no ordenadas)

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.

<code></code>	Lista ordenada
Atributos comunes	básicos, i18n y eventos
Atributos específicos	-
Tipo de elemento	Bloque
Descripción	Se emplea para definir listas ordenadas

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>

```

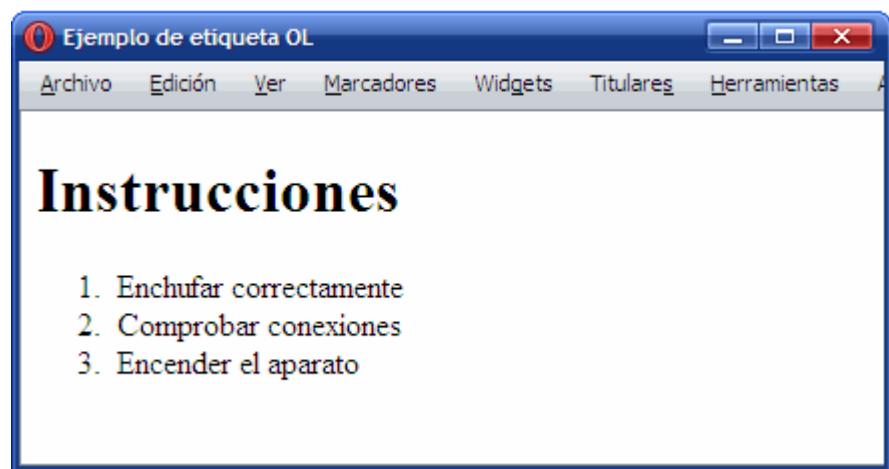


Figura 5.2. Ejemplo de uso de la etiqueta ol

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.

<code><dl></code>	Lista de definición
Atributos comunes	básicos, i18n y eventos
Atributos específicos	-
Tipo de elemento	Bloque
Descripción	Se emplea para definir listas de definición

<dt>	Término de una definición
Atributos comunes	básicos, i18n y eventos
Atributos específicos	-
Tipo de elemento	Bloque
Descripción	Se emplea para definir los términos de los elementos de una lista de definición

<dd>	Descripción de una definición
Atributos comunes	básicos, i18n y eventos
Atributos específicos	-
Tipo de elemento	Bloque
Descripción	Se emplea para indicar las definiciones de los elementos de una lista de definición

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>

```

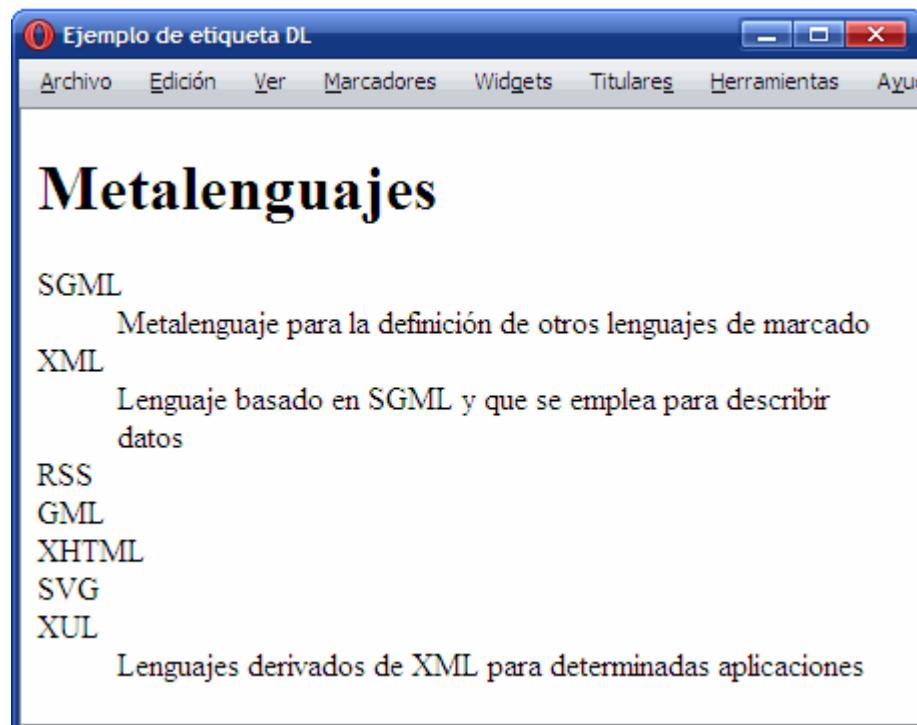


Figura 5.3. Ejemplo de uso de la etiqueta dl

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:

<code></code>	Imagen
Atributos comunes	básicos, i18n y eventos
Atributos específicos	<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)
Tipo de elemento	En línea y etiqueta vacía
Descripción	Se emplea para incluir imágenes en los documentos

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>	Mapa de imagen
Atributos comunes	básicos, i18n y eventos
Atributos específicos	<ul style="list-style-type: none"> ▪ name = "texto" - Nombre que identifica de forma única al mapa definido (es obligatorio indicar un nombre único)
Tipo de elemento	Bloque y en línea
Descripción	Se emplea para definir mapas de imagen

<area>	Area de un mapa de imagen
Atributos comunes	básicos, i18n, eventos y foco
Atributos específicos	<ul style="list-style-type: none"> ▪ href = "url" - URL a la que se accede al pinchar sobre el área ▪ nohref = "nohref" - Se emplea para las áreas que no son seleccionables ▪ shape = "default rect circle poly" - Indica el tipo de área que se define (toda la imagen, rectangular, circular o poligonal) ▪ coords = "lista de números" - 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,...,XnYn (coordenadas de los vértices del polígono. Si las últimas coordenadas no son iguales que las primeras, se cierra automáticamente el polígono uniendo ambos vértices)
Tipo de elemento	Etiqueta vacía
Descripción	Se emplea para definir las distintas áreas que forman un mapa de imagen

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:

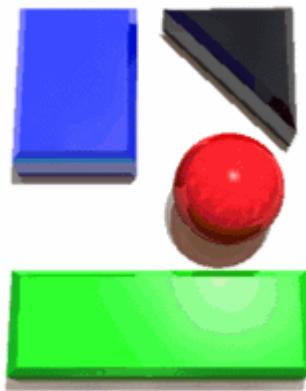


Figura 6.2. Ejemplo de imagen que incluye un mapa de imagen

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>	Objeto
Atributos comunes	básicos, i18n y eventos
Atributos específicos	<ul style="list-style-type: none"> ▪ data = "url" - Indica la URL de los datos que utiliza el objeto ▪ classid, codebase, codetype - Información específica que depende del tipo de objeto ▪ type - Indica el tipo de contenido de los datos ▪ height = "unidad_de_medida" - Indica la altura con la que se debe mostrar el objeto ▪ width = "unidad_de_medida" - Indica la anchura con la que se debe mostrar el objeto
Tipo de elemento	Bloque y en línea
Descripción	Se emplea para embeber objetos en los documentos

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>	Parámetros de un objeto
Atributos comunes	id
Atributos específicos	<ul style="list-style-type: none"> ▪ name = "texto" - Indica el nombre del parámetro ▪ value = "texto" - Indica el valor del parámetro
Tipo de elemento	Etiqueta vacía
Descripción	Se emplea para indicar el valor de los parámetros del objeto

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>	Embeber objetos
Atributos comunes	básicos, i18n y eventos
Atributos específicos	<ul style="list-style-type: none"> ▪ src = "url" - Indica la URL del archivo u objeto que se incluye en la página ▪ type = "tipo_de_contenido" - Indica el tipo de contenido del objeto (flash, quicktime, java, etc.) ▪ height = "unidad_de_medida" - Indica la altura con la que se debe mostrar el objeto ▪ width = "unidad_de_medida" - Indica la anchura con la que se debe mostrar el objeto
Tipo de elemento	Bloque
Descripción	Se emplea para embeber objetos en los documentos

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.

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:

Diagrama que muestra una tabla HTML con el siguiente contenido:

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:

The screenshot shows a window titled "Ejemplo de tabla sencilla - Opera". The menu bar includes Archivo, Editar, Ver, Marcadores, Widgets, Herramientas, and Ayuda. The main content area has a title "Listado de cursos" and a table:

Curso	Horas	Horario
CSS	20	16:00 - 20:00
HTML	20	16:00 - 20:00
Dreamweaver	60	16:00 - 20:00

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>`.

<table>	Tabla
Atributos comunes	básicos, i18n y eventos
Atributos específicos	<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)
Tipo de elemento	Bloque
Descripción	Se emplea para definir tablas de datos

<tr>	Fila de tabla
Atributos comunes	básicos, i18n y eventos
Atributos específicos	-
Tipo de elemento	Bloque
Descripción	Se emplea para definir cada fila de las tablas de datos

<td>	Celda de tabla
Atributos comunes	básicos, i18n y eventos
Atributos específicos	<ul style="list-style-type: none"> ▪ abbr = "texto" - Permite describir el contenido de la celda (se emplea sobre todo con los navegadores de voz 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
Tipo de elemento	Bloque
Descripción	Se emplea para definir cada una de las celdas que forman las filas de una tabla, es decir, las columnas de la tabla

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>	Celda cabecera de tabla
Atributos comunes	básicos, i18n y eventos
Atributos específicos	<ul style="list-style-type: none"> ▪ abbr = "texto" - Permite describir el contenido de la celda (se emplea sobre todo con los navegadores de voz 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
Tipo de elemento	Bloque
Descripción	Se emplea para definir las celdas que son cabecera de una fila o de una columna de la tabla

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>	Leyenda o título de tabla
Atributos comunes	básicos, i18n y eventos
Atributos específicos	-
Tipo de elemento	En línea
Descripción	Se emplea para definir la leyenda o título de una tabla

Ejercicio 11

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

The screenshot shows a web browser window with the title 'Ejemplo de tabla sencilla - Opera'. The menu bar includes 'Archivo', 'Editar', 'Ver', 'Marcadores', 'Widgets', 'Herramientas', and 'Ayuda'. Below the menu is a heading 'Su pedido'. A table is displayed with the following data:

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.

Ejemplo de tabla avanzada - Opera

Resultado de la búsqueda	
Imagen	Datos
	<u>Portátil - 3 GHz - 4 GB RAM</u> <u>Comprar: 2.990 € 2.599 €</u>
	<u>Videocámara - Alta definición 1080p - 60 GB</u> <u>Comprar: 1.099 € 999 €</u>
	<u>Televisor - 46" - Full HD</u> <u>Comprar: 1.999 € 1.799 €</u>
	<u>Móvil - 3G - Wi-Fi - 8 GB</u> <u>Comprar: 399 € 349 €</u>

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:

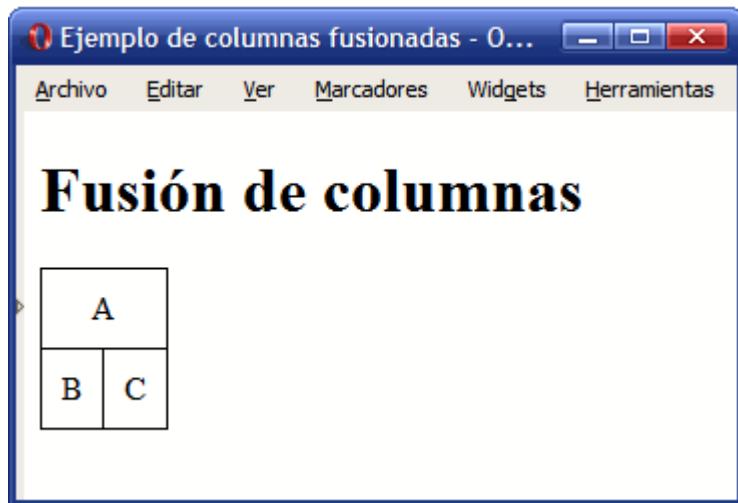


Figura 7.5. Ejemplo sencillo de fusión de columnas

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:

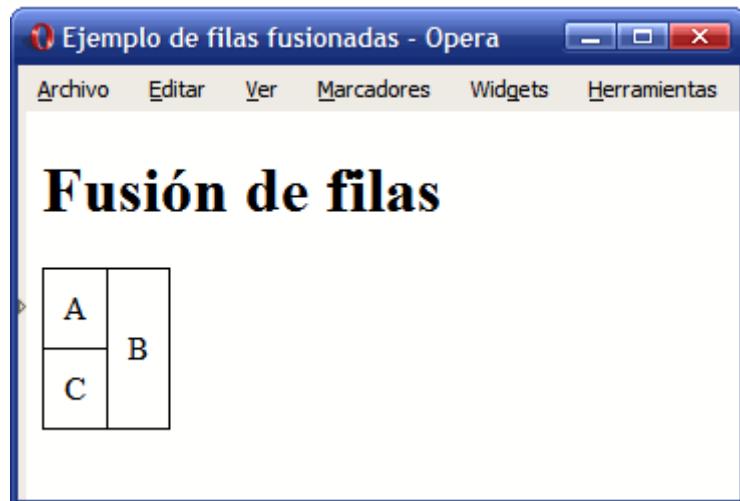


Figura 7.6. Ejemplo sencillo de fusión de filas

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:

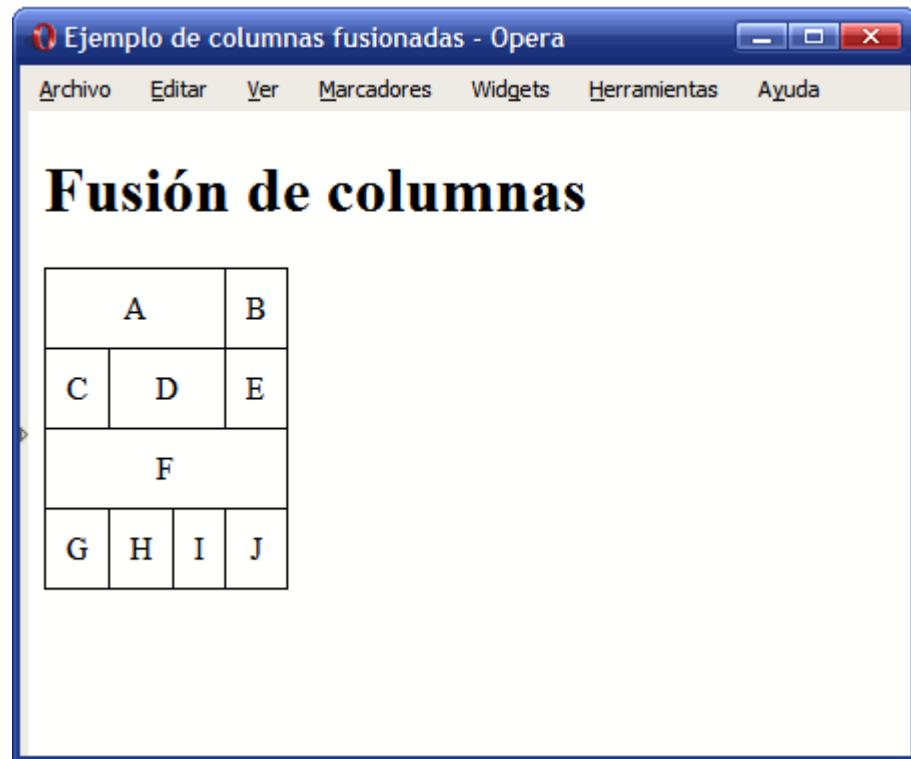


Figura 7.7. Ejemplo complejo de fusión de columnas

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>
```

```
</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>
<tr>
  <td>I</td>
  <td>J</td>
  <td>K</td>
  <td></td>
</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:

Ejemplo de tabla compleja - Opera

Archivo Editar Ver Marcadores Widgets Herramientas Ayuda

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	<input type="radio"/>	<input checked="" type="radio"/> 	<input type="radio"/>	<input type="radio"/> 	<input type="radio"/>
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:

	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>`.

<thead>	Cabecera de tabla
<tbody>	Sección de una tabla
<tfoot>	Pie de tabla
Atributos comunes	básicos, i18n y eventos
Atributos específicos	-
Tipo de elemento	Bloque
Descripción	Se emplean para agrupar varias filas en una cabecera (thead) un pie (tfoot) o una sección (tbody) de una tabla

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:

The screenshot shows a browser window titled "Ejemplo de tabla avanzada - Opera". The menu bar includes Archivo, Editar, Ver, Marcadores, Widgets, Herramientas, and Ayuda. The main content area has a title "Análisis de ventas" and a subtitle "Análisis de ventas anuales". Below this is a table with the following structure and data:

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	Producto A	Producto B	Producto C	Producto D
	Expansión de ventas			

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>`:

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>

```

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

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>	Formulario
Atributos comunes	básicos, i18n y eventos
Atributos específicos	<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: accept-charset, onsubmit, onreset
Tipo de elemento	Bloque
Descripción	Se emplea para insertar un formulario en la página

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 envian 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 información. 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 qué 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>	Control de un formulario
Atributos comunes	básicos, i18n, eventos y foco
Atributos específicos	<ul style="list-style-type: none"> ▪ type = "text password checkbox radio submit reset file hidden image button" - Indica el tipo de control que se incluye en el formulario ▪ name = "texto" - Asigna un nombre al control (es imprescindible para que el servidor pueda procesar el formulario) ▪ value = "texto" - Valor inicial del control ▪ size = "unidad_de_medida" - 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) ▪ maxlength = "numero" - Máximo número de caracteres para los controles de texto y de password ▪ checked = "checked" - Para los controles checkbox y radiobutton permite indicar qué opción aparece preseleccionada ▪ disabled = "disabled" - El control aparece deshabilitado y su valor no se envía al servidor junto con el resto de datos ▪ readonly = "readonly" - El contenido del control no se puede modificar ▪ src = "url" - Para el control que permite crear botones con imágenes, indica la URL de la imagen que se emplea como botón de formulario ▪ alt = "texto" - Descripción del control
Tipo de elemento	En línea y etiqueta vacía
Descripción	Se emplean para insertar un control en un formulario

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:

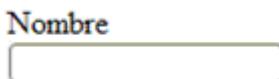


Figura 8.2. Ejemplo de etiqueta input (type=text)

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.

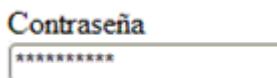


Figura 8.3. Ejemplo de etiqueta input (type=password)

```
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 deselecciona la otra opción que estaba seleccionada.

Sexo
<input checked="" type="radio"/> Hombre
<input type="radio"/> 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 es utilizado 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.

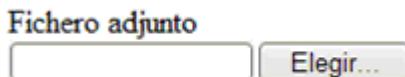


Figura 8.8. Ejemplo de etiqueta input (type=file)

```
| 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:

Guardar Cambios

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:

The screenshot shows a window titled "Rellena tu CV - Opera". The menu bar includes Archivo, Editar, Ver, Marcadores, Widgets, Herramientas, and Ayuda. The main content area has a title "Rellena tu CV". It contains several input fields and controls:

- Nombre: An input field.
- Apellidos: An input field.
- Contraseña: An input field.
- DNI: An input field.
- Sexo: A group of radio buttons labeled "Hombre" and "Mujer".
- Incluir mi foto: An input field with a browse button labeled "Elegir...".
- Suscribirme al boletín de novedades: A checked checkbox.
- Guardar cambios: A button.
- Borrar los datos introducidos: A button.

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>	Agrupación de campos
Atributos comunes	básicos, i18n y eventos
Atributos específicos	-
Tipo de elemento	Bloque
Descripción	Se emplea para agrupar de forma lógica varios campos de un formulario

<legend>	Título o leyenda de un fieldset
Atributos comunes	básicos, i18n y eventos
Atributos específicos	-
Tipo de elemento	En línea
Descripción	Se emplea para definir el título o leyenda de un conjunto de campos de formulario agrupados con la etiqueta fieldset

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>	Título o leyenda de un campo de formulario
Atributos comunes	básicos, i18n y eventos
Atributos específicos	<ul style="list-style-type: none"> ▪ for = "id_de_elemento" - Indica el ID del campo del formulario para el que este elemento es su título ▪ Otros: accesskey, onfocus y onblur
Tipo de elemento	En línea
Descripción	Se emplea para definir el título o leyenda de los campos definidos en un formulario

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 `radio`button 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:

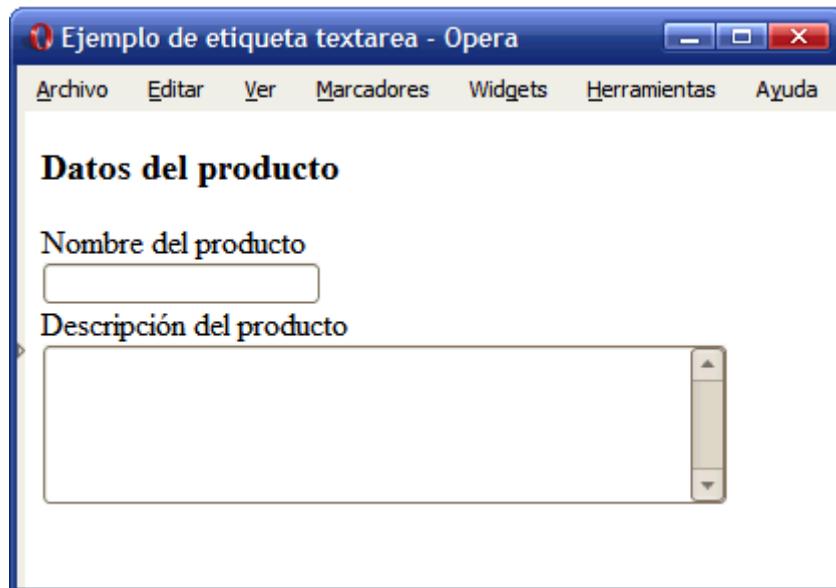


Figura 8.14. Ejemplo de uso de la etiqueta textarea

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>	Área de texto
Atributos comunes	básicos, i18n, eventos y foco
Atributos específicos	<ul style="list-style-type: none"> ▪ rows = "numero" - Número de filas de texto que mostrará el textarea ▪ cols = "numero" - Número de caracteres que se muestran en cada fila del textarea ▪ Otros: name, disabled, readonly, onselect, onchange, onfocus, onblur
Tipo de elemento	En línea
Descripción	Se emplea para incluir un área de texto en un formulario

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:

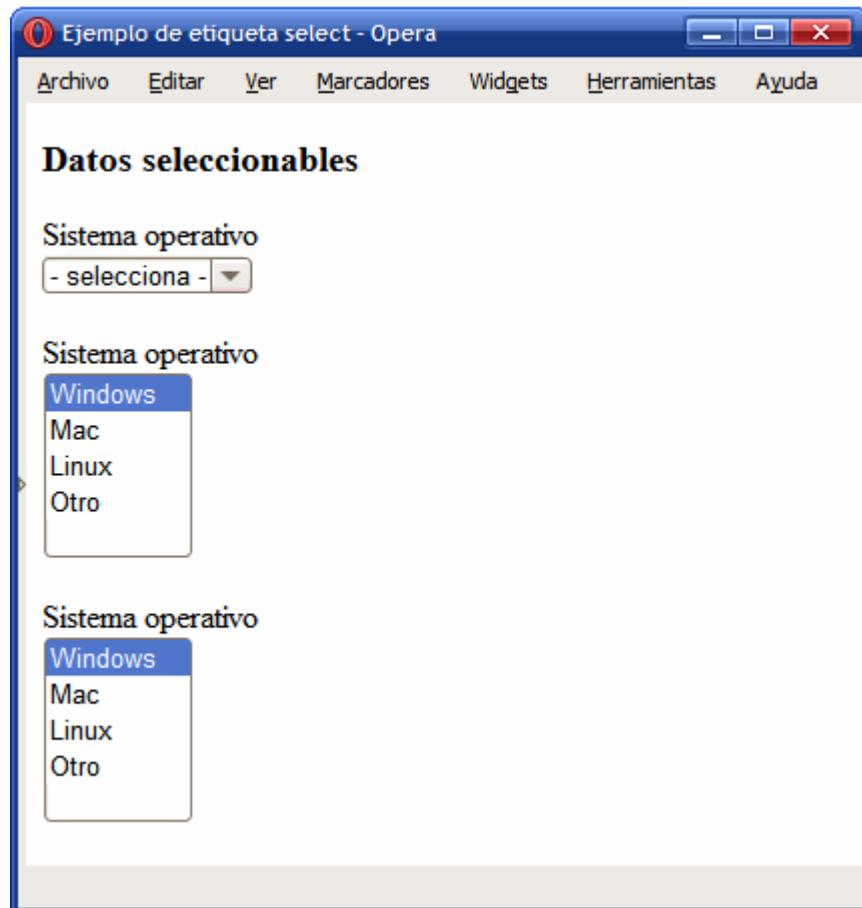


Figura 8.15. Ejemplo de uso de la etiqueta select

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>	Lista desplegable
Atributos comunes	básicos, i18n y eventos
Atributos específicos	<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
Tipo de elemento	En línea
Descripción	Se emplea para incluir una lista desplegable en un formulario

<option>	Elemento de una lista desplegable
Atributos comunes	básicos, i18n y eventos
Atributos específicos	<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
Tipo de elemento	-
Descripción	Se emplea para definir cada elemento de una lista desplegable

La inmensa mayoría de listas desplegables que utilizan las aplicaciones web son simples, por lo que el código HTML habitual de las listas desplegables 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 desplegables permiten agrupar sus opciones de forma que el usuario pueda encontrar fácilmente las opciones cuando la lista es muy larga:

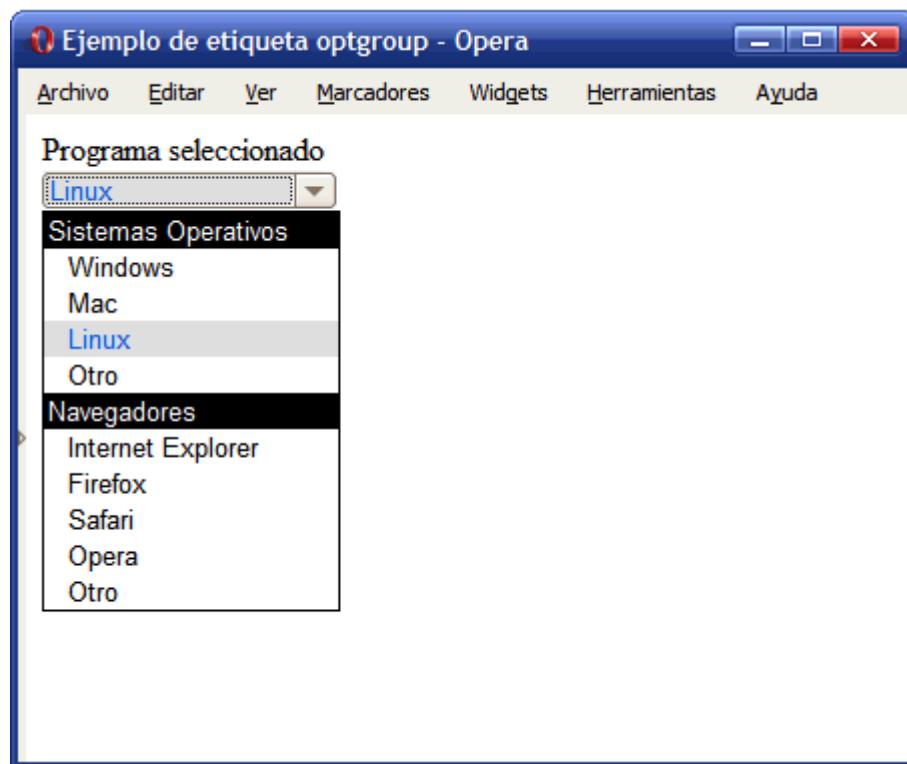


Figura 8.16. Ejemplo de uso de la etiqueta optgroup

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>	Agrupación de elementos de una lista desplegable
Atributos comunes	básicos, i18n y eventos
Atributos específicos	<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>
Tipo de elemento	-
Descripción	Se emplea para definir una agrupación lógica de opciones de una lista desplegable

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:

The screenshot shows a web browser window titled "Rellena tu CV - Opera". The menu bar includes Archivo, Editar, Ver, Marcadores, Widgets, Herramientas, and Ayuda. The main content area has a title "Rellena tu CV". A section titled "Datos personales" contains a "Provincia" field with a dropdown menu showing "- selecciona -". Below it is a "Fecha de nacimiento" field with three input boxes. A section titled "Temas de interés" contains a list of checkboxes: "Administración de bases de datos", "Análisis y programación", "Arquitectura", "Calidad", and "ERP, CRM, Business Intelligence".

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 complex product information form titled "Información sobre el producto" (Product Information) displayed in a web browser window. The browser's title bar reads "Información sobre el producto - Opera". The menu bar includes Archivo, Editar, Ver, Marcadores, Widgets, Herramientas, and Ayuda. The form is divided into two main sections: "Datos básicos" (Basic Data) and "Datos económicos" (Economic Data).

Datos básicos:

- Nombre:** A text input field.
- Descripción:** A large text area with a vertical scrollbar.
- Foto:** A text input field followed by a "Elegir..." button.
- Añadir contador de visitas:** A checkbox.

Datos económicos:

- Precio:** A text input field with a € symbol.
- Impuestos:** A dropdown menu showing options: 4% (selected), 7%, 16%, and 25%.
- Promoción:**
 - Ninguno
 - Transporte gratuito
 - Descuento 5%

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:

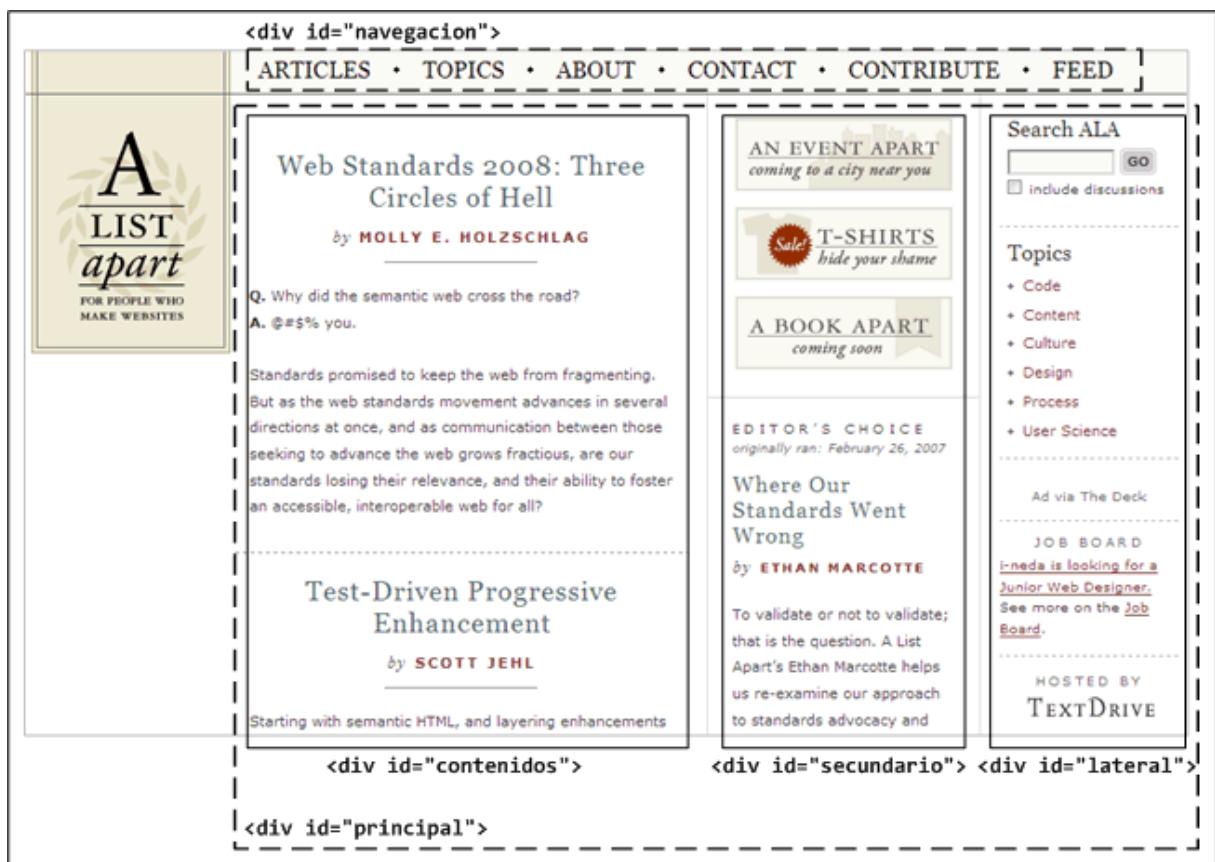


Figura 9.1. Ejemplo de página compleja estructurada con etiquetas div

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

<div>	Divisiones
Atributos comunes	básicos, i18n y eventos
Atributos específicos	-
Tipo de elemento	Bloque
Descripción	Agrupa elementos de bloque

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>	Cabecera
Atributos comunes	i18n
Atributos específicos	<ul style="list-style-type: none"> ▪ profile = "url" - Especifica la URL del perfil o perfiles que utilizan los metadatos ▪ lang = "codigo_de_idioma" - Especifica el idioma principal de los contenidos de la página
Tipo de elemento	-
Descripción	Define la cabecera del documento HTML

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>	Título del documento
Atributos comunes	i18n
Atributos específicos	<ul style="list-style-type: none"> ▪ lang = "codigo_de_idioma" - Especifica el idioma principal del título de la página
Tipo de elemento	-
Descripción	Define el título del documento HTML

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>.

<meta>	Metadatos
Atributos comunes	i18n
Atributos específicos	<ul style="list-style-type: none"> ▪ name = "texto" - El nombre de la propiedad que se define (no existe una lista oficial de propiedades) ▪ content = "texto" - El valor de la propiedad definida (no existe una lista de valores permitidos) ▪ http-equiv = "texto" - En ocasiones, reemplaza al atributo "name" y lo emplean los servidores para adaptar sus respuestas al documento ▪ scheme = "texto" - Indica el esquema que se debe emplear para interpretar el valor de la propiedad
Tipo de elemento	-
Descripción	Permite definir el valor de los metadatos que forman la metainformación del documento

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 etiqueta `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 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">
```

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

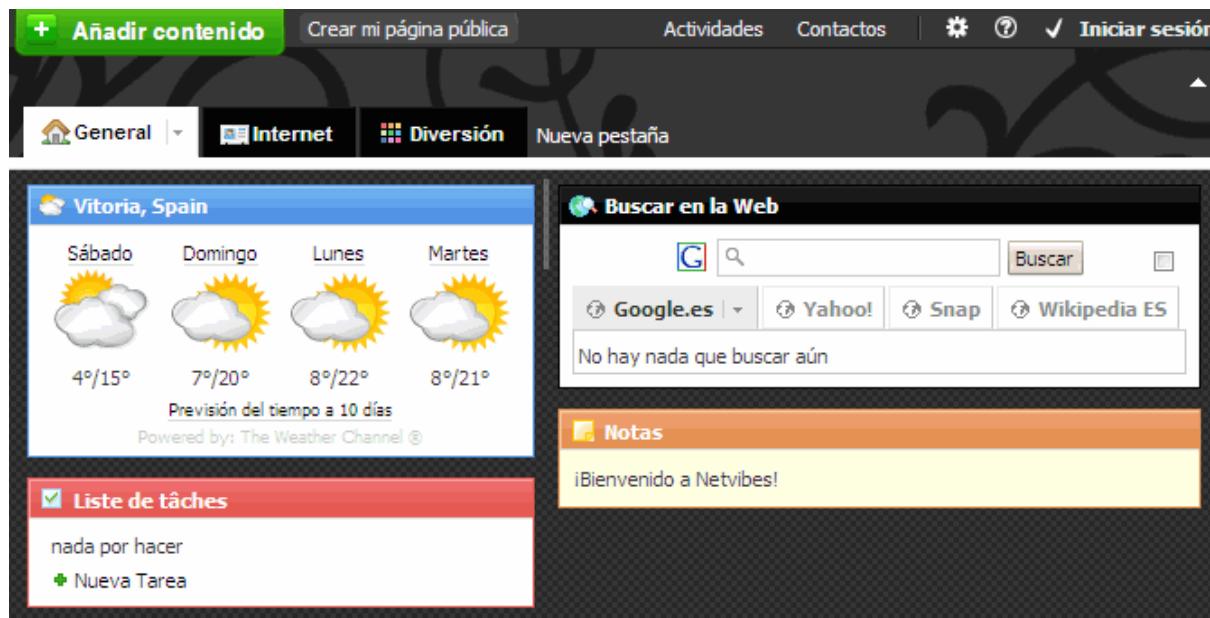


Figura 11.1. Ejemplo de página compleja con JavaScript activado

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.

<noscript>	Sin soporte de scripts
Atributos comunes	básicos, i18n y eventos
Atributos específicos	-
Tipo de elemento	Bloque
Descripción	Define un mensaje alternativo que se muestra al usuario cuando su navegador no soporta la ejecución de scripts

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, consectetuer 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 veces 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>	Marco (frame) en línea
Atributos comunes	básicos
Atributos específicos	<ul style="list-style-type: none"> ▪ src = "url" - URL del documento HTML que se visualiza en el iframe ▪ height = "longitud" - Altura que ocupará el iframe en el documento ▪ width = "longitud" - Anchura que ocupará el iframe en el documento ▪ name = "texto" - Nombre que identifica al iframe ▪ longdesc = "url" - Dirección en la que puede encontrarse una descripción larga del contenido del iframe ▪ scrolling = "yes no auto" - Indica si el iframe debe mostrar barras de scroll (horizontal y vertical) cuando el contenido incluido no cabe en el iframe
Tipo de elemento	Bloque y en línea
Descripción	Se emplea para incluir en la página un marco que muestra otro documento HTML

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>	Direcciones
Atributos comunes	básicos, i18n y eventos
Atributos específicos	-
Tipo de elemento	Bloque
Descripción	Define la información de contacto de un documento

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>	Línea horizontal
Atributos comunes	básicos, i18n y eventos
Atributos específicos	-
Tipo de elemento	Bloque
Descripción	Permite incluir una línea horizontal de separación

La siguiente imagen muestra el aspecto con el que los navegadores muestran por defecto las líneas horizontales creadas con `<hr>`:

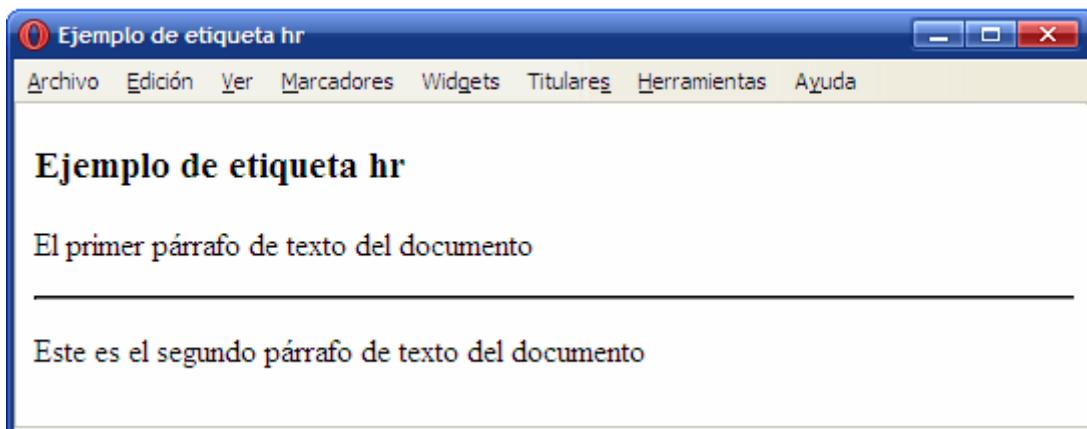


Figura 11.4. Ejemplo de uso de la etiqueta `hr`

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>
```

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) (<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 cumplen 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 equivalentes 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>

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:

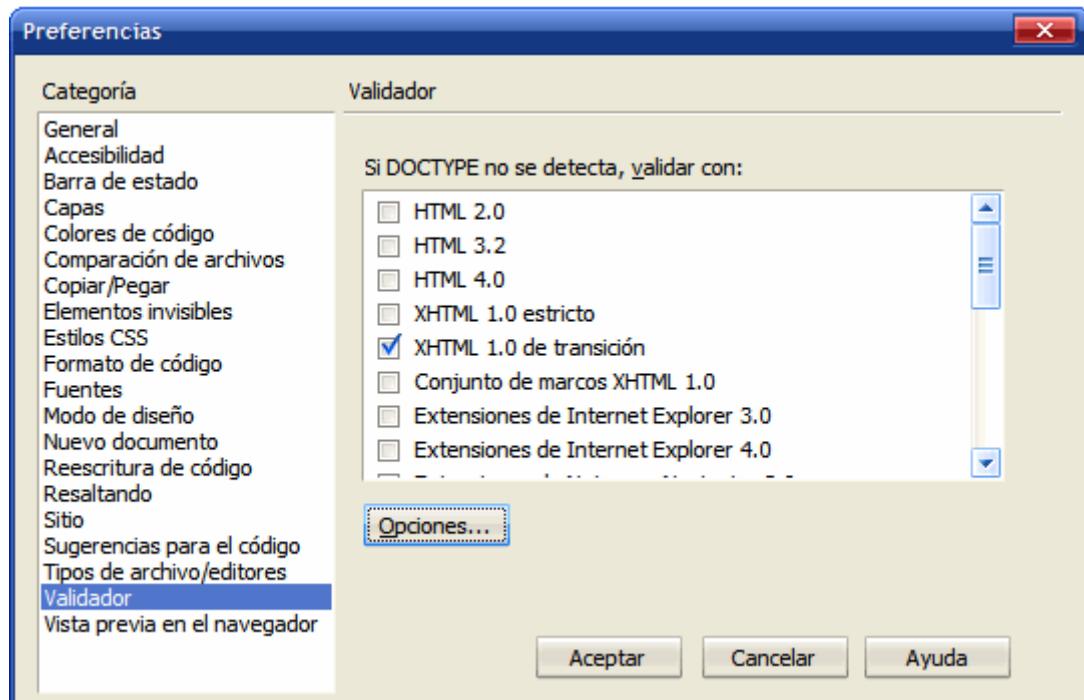


Figura 13.1. Configuración del validador de Dreamweaver

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:

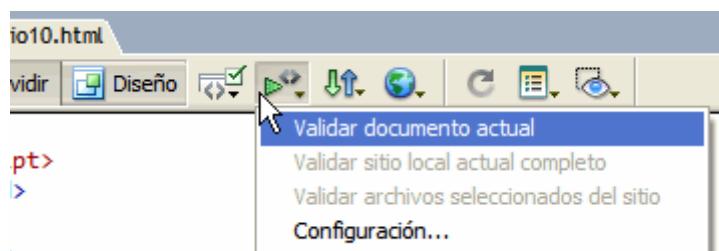


Figura 13.2. Icónico que permite acceder a la herramienta de validación de Dreamweaver

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:

The screenshot shows a code editor with an XHTML document. The code includes a doctype declaration, a head section with a meta tag and a title, and a body section starting with <body>. Below the code is a validation toolbar with tabs for 'Resultados', 'Buscar', 'Referencia', 'Validación' (which is selected), 'Revisión del navegador de destino', and 'Verificador de vínculos'. The validation results table has columns for 'Línea' (Line) and 'Descripción' (Description). It lists three errors: one warning about an entity reference, and two errors about quotes between tags.

Línea	Descripción
5	á encontrado dentro de las etiquetas. Considere la posibilidad de utilizar la entidad equivalente (&am
37	Se han encontrado unas comillas entre las etiquetas; los documentos HTML no deben contener estos
37	Se han encontrado unas comillas entre las etiquetas; los documentos HTML no deben contener estos

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/>

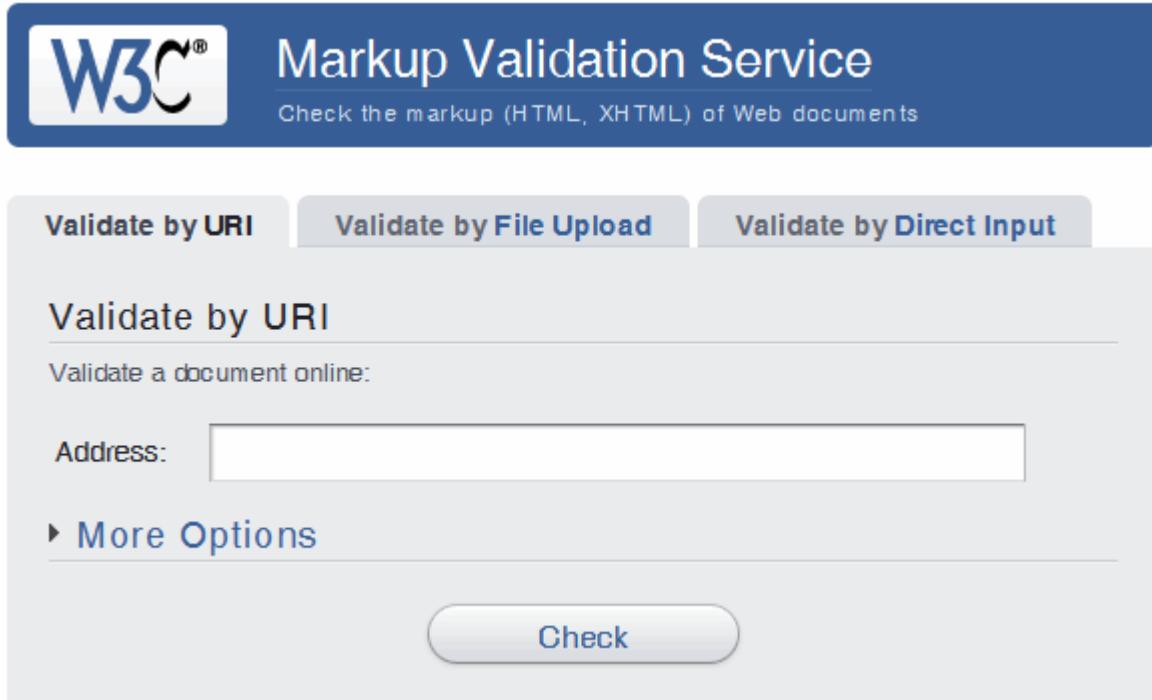


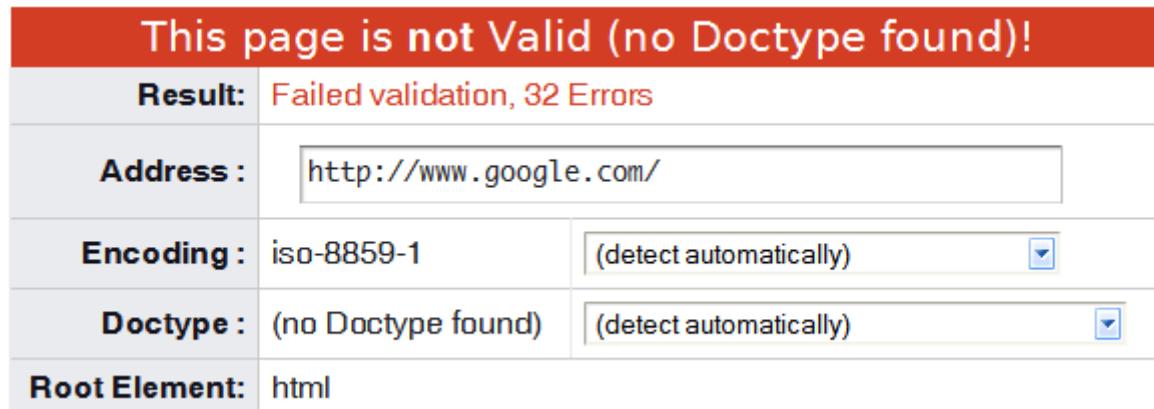
Figura 13.4. Página principal del validador del W3C

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**:



The screenshot shows the W3C Markup Validation Service interface. At the top, a red bar displays the message "This page is not Valid (no Doctype found!)". Below this, a table summarizes the validation results:

Result:	Failed validation, 32 Errors
Address :	<input type="text" value="http://www.google.com/"/>
Encoding :	iso-8859-1 <input type="button" value="(detect automatically)"/>
Doctype :	(no Doctype found) <input type="button" 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:

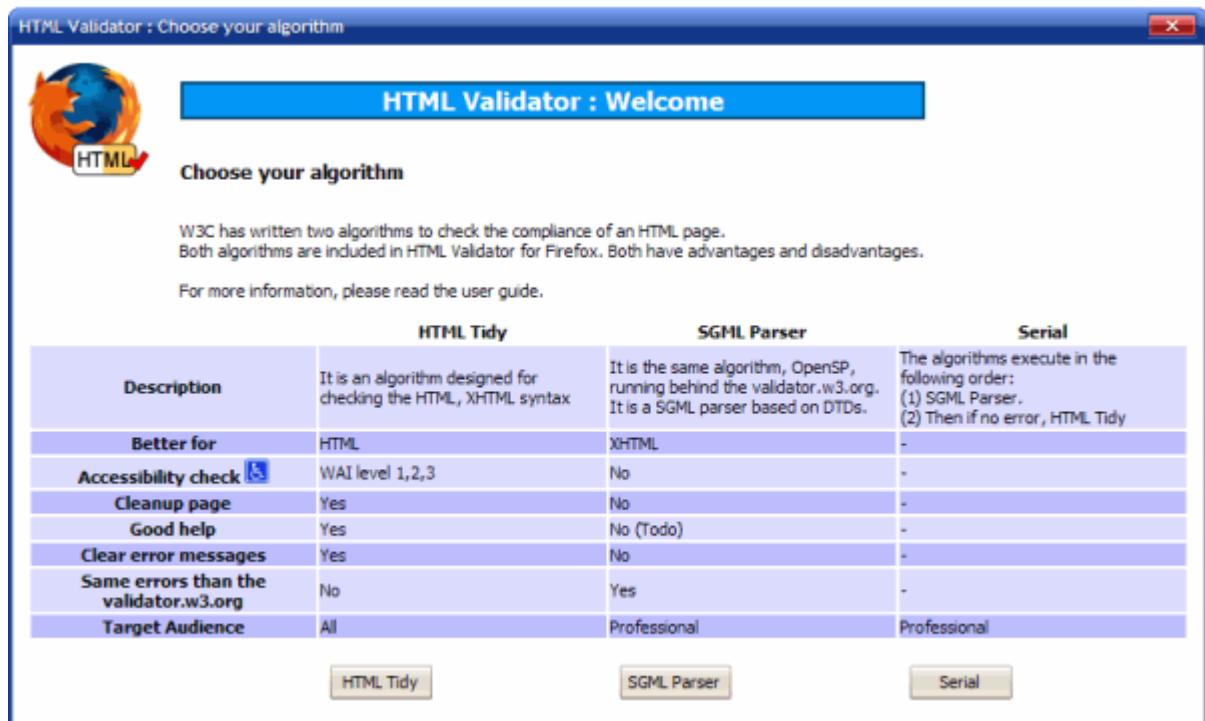


Figura 13.6. Configuración del validador Html Validator

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 ícono que indica si la página es válida o no. Cuando la página no es válida, aparece un ícono correspondiente a un error. Si colocas el puntero del ratón sobre el ícono, se muestra la información específica sobre los errores encontrados:

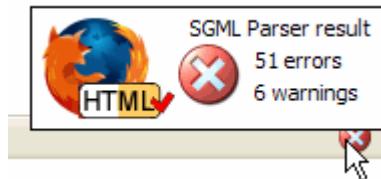


Figura 13.7. Información de error proporcionada por el validador Html Validator

Si pulsas dos veces sobre el ícono, 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 ícono 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 Mayús + F9. Una vez abierta esta ventana, el aspecto que muestra se ve en la parte inferior izquierda de la siguiente imagen:

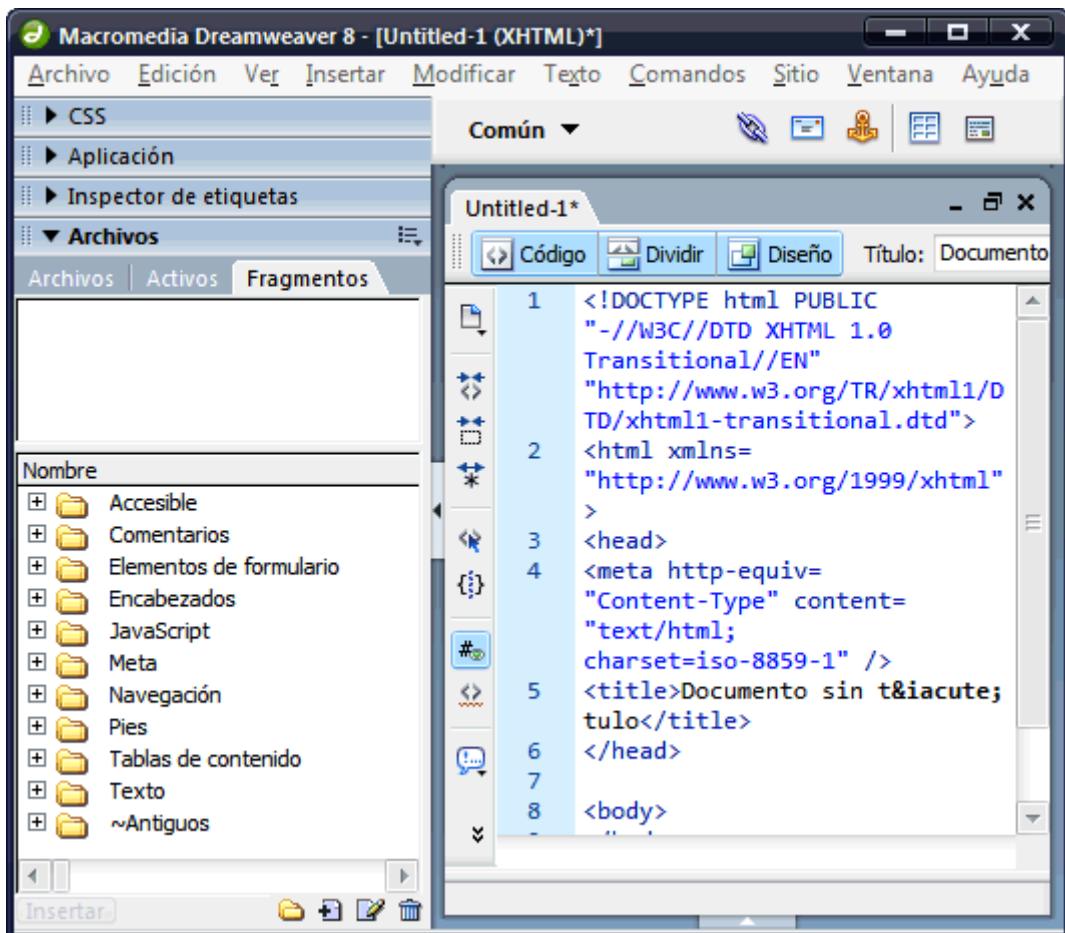


Figura 14.1. Visualizando la ventana de fragmentos de código en Dreamweaver

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:

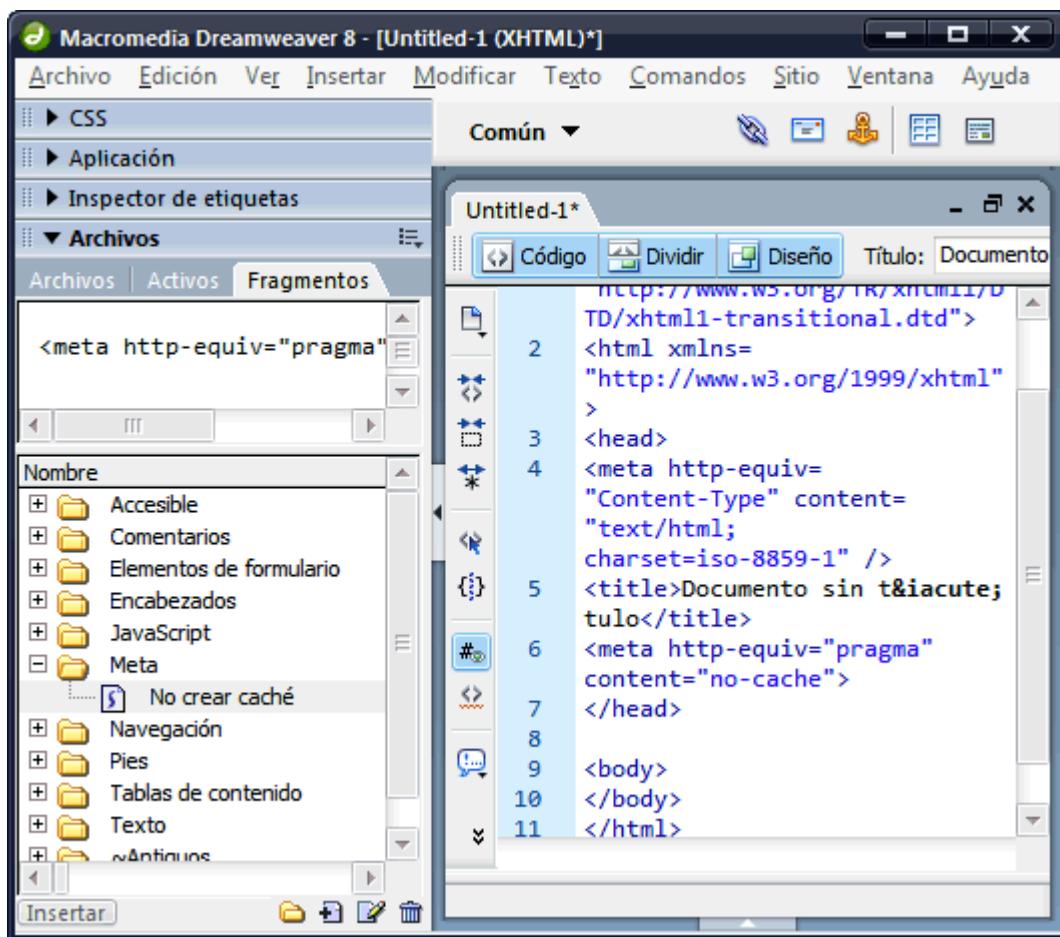


Figura 14.2. Insertando un fragmento de código de Dreamweaver en una página web

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:

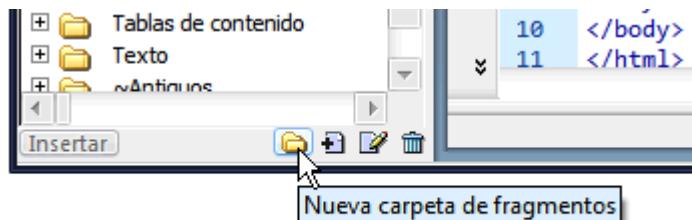


Figura 14.3. Creando una nueva carpeta para guardar los fragmentos de código propios

Una vez escrito el nombre de la nueva carpeta, el aspecto que muestra la ventana de *snippets* es el siguiente:

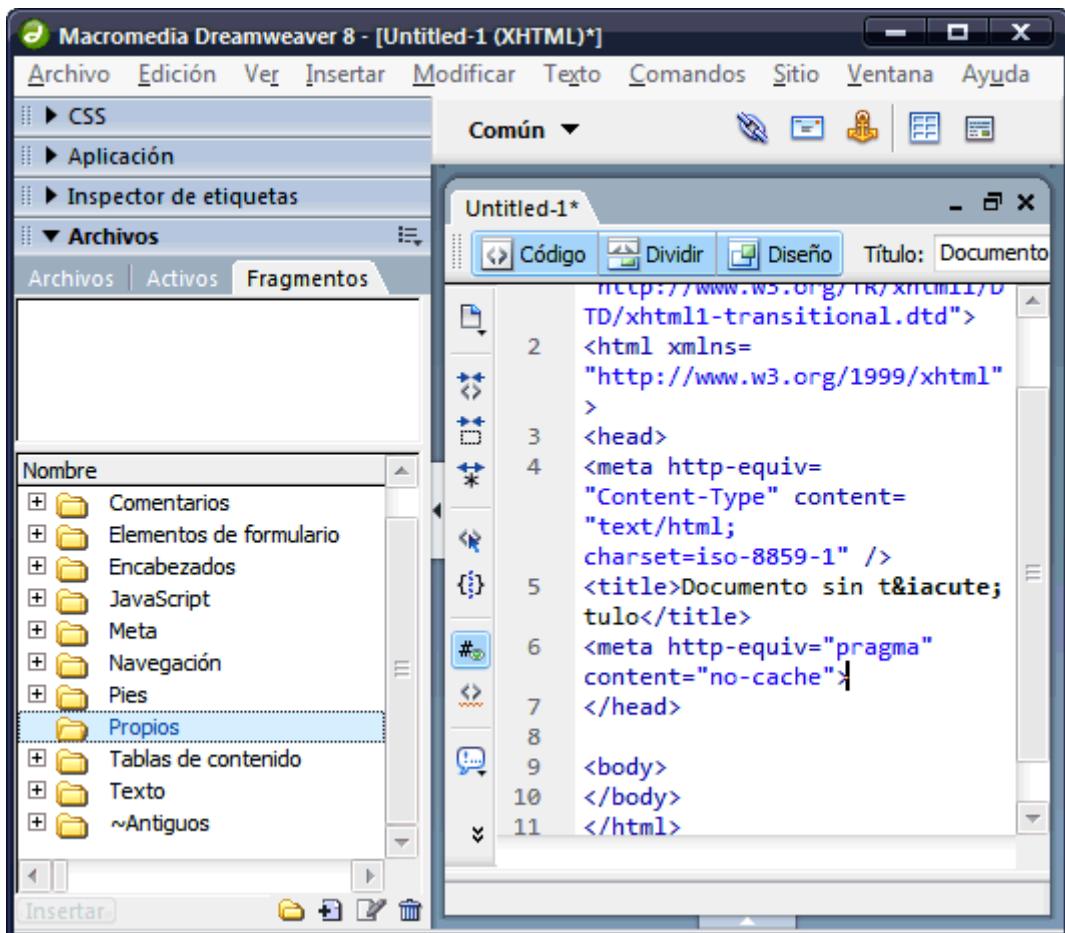


Figura 14.4. Nueva carpeta creada para guardar los fragmentos de código propios

- 4) Para crear un fragmento de código propio, se pulsa sobre el pequeño ícono que simboliza un nuevo fragmento:



Figura 14.5. Creando un nuevo fragmento de código

Después de pulsar sobre ese ícono, aparece la ventana en la que se puede escribir el nombre y el contenido del fragmento:

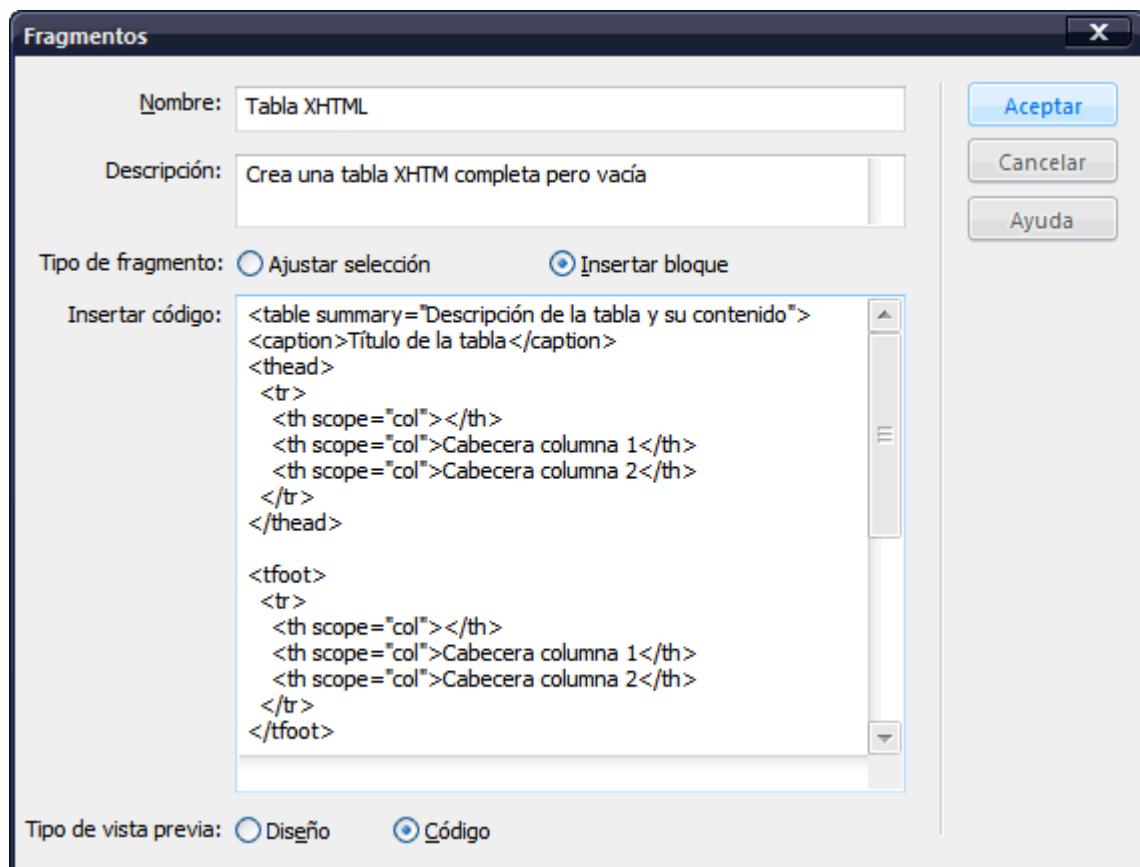


Figura 14.6. Propiedades (nombre, descripción y contenido) del fragmento de código propio

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:

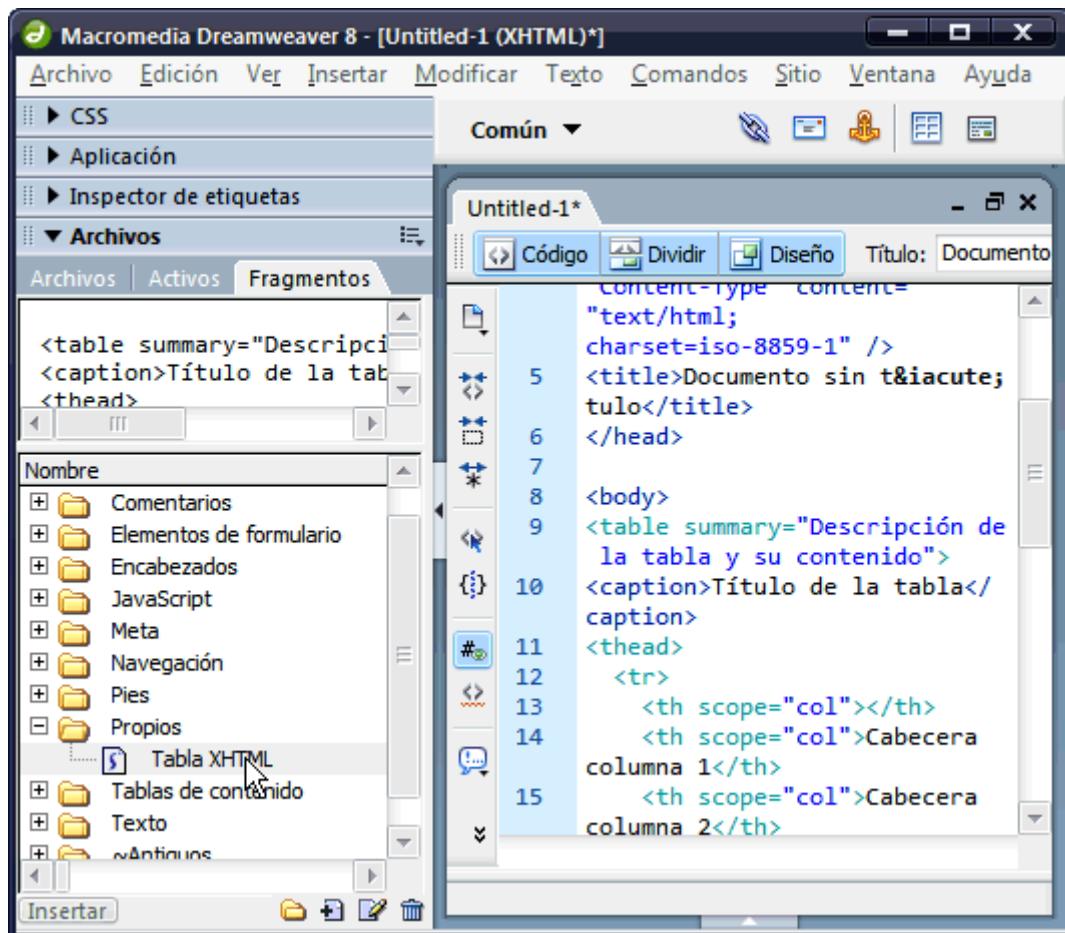


Figura 14.7. Insertando un fragmento de código propio en una página XHTML

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">
  //<![CDATA[
  // Código JavaScript
  //]]&gt;
&lt;/script&gt;
&lt;/head&gt;</pre>
```

14.3. Tabla

Las tablas XHTML complejas están formadas por cabecera, pie y uno o más cuerpos. Además, es necesario incluir atributos como `summary` y `scope` para mejorar la accesibilidad.

```
<table summary="Descripción de la tabla y su contenido">
<caption>Título de la tabla</caption>
<thead>
  <tr>
    <th scope="col"></th>
    <th scope="col">Cabecera columna 1</th>
    <th scope="col">Cabecera columna 2</th>
  </tr>
</thead>

<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">Área 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>
```

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/xhtml11/
DTD/xhtml11-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 blockquote</title>
</head>

<body>

<h1>Sintaxis de la etiqueta &lt;blockquote&gt;</h1>

<p>
La sintaxis de la etiqueta &lt;blockquote&gt; se muestra a continuación: <br/><br/>
&lt;blockquote cite="<em>...direccion original de la cita...</em>">&gt;Texto
que se cita&lt;/blockquote&gt;

</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, consectetuer 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 consectetuer, 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" />
<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, consectetuer adipiscing elit. Donec iaculis posuere justo. Nam vel neque.</p>
```

```
<h3>Proyecto 1</h3>

<p>Etiam consectetuer, mauris vitae cursus scelerisque, dui turpis dignissim justo, et
euismod 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 consectetuer, mauris vitae cursus scelerisque, dui turpis dignissim justo, et
euismod 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>
```

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, consectetuer 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 consectetuer, 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, consectetuer adipiscing elit. Donec iaculis posuere justo. Nam vel neque.</p>

<h3>Proyecto 1</h3>

<p>Etiam consectetuer, mauris vitae cursus scelerisque, dui turpis dignissim justo, et euismod 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 consectetuer, mauris vitae cursus scelerisque, dui turpis dignissim justo, et euismod 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>
```

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/xhtml11/
DTD/xhtml11-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>
</ul>
```

```
</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 consectetuer, mauris vitae cursus scelerisque, dui turpis dignissim justo, et euismod 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>
```

15.11. Ejercicio 11

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml11/  
DTD/xhtml11-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><br/><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><br/><strong>1.499 &euro;</strong></p>  
      </td>  
    </tr>  
  </table>
```

```

<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>

```

15.13. Ejercicio 13

```

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml11/
DTD/xhtml11-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>

<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>
```

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>
                </select>
            </fieldset>
        </form>
    </body>
</html>
```

```
<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>
<input type="text" size="5" id="precio" name="precio" /> &euro;

&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;

<label for="impuestos">Impuestos</label>
<select id="impuestos" name="impuestos">
  <option value="4">4%</option>
  <option value="7">7%</option>
  <option value="16">16%</option>
  <option value="25">25%</option>
</select>

<br/><br/>

<label>Promoción</label> <br/>
<input type="radio" name="promocion" value="ninguno" checked="checked" /> Ninguno
<br/>
<input type="radio" name="promocion" value="portes" /> Transporte gratuito <br/>
<input type="radio" name="promocion" value="descuento" /> Descuento 5%
</fieldset>
</form>

</body>

</html>
```

2.1 INTRODUCCIÓN

El lenguaje HTML está limitado a la hora de aplicarle forma a un documento. Esto es así porque fué concebido para otros usos (científicos sobre todo), distinto a los actuales, mucho más amplios.

Para solucionar estos problemas los diseñadores han utilizado técnicas tales como la utilización de tablas, imágenes transparentes para ajustarlas, utilización de etiquetas que no son estándares del HTML y otras. Estas "trampas" han causado a menudo problemas en las páginas a la hora de su visualización en distintas plataformas.

Además, los diseñadores se han visto frustrados por la dificultad con la que, aun utilizando estos trucos, se encontraban a la hora de maquetar las páginas, ya que muchos de ellos venían maquetando páginas sobre el papel, donde el control sobre la forma del documento es absoluto.

Finalmente, otro antecedente que ha hecho necesario el desarrollo de esta tecnología consiste en que las páginas web tienen mezclado en su código HTML el contenido del documento con las etiquetas necesarias para darle forma. Esto tiene sus inconvenientes ya que la lectura del código HTML se hace pesada y difícil a la hora de buscar errores o depurar las páginas. Aunque, desde el punto de vista de la riqueza de la información y la utilidad de las páginas a la hora de almacenar su contenido, es un gran problema que estos textos estén mezclados con etiquetas incrustadas para dar forma a estos: se degrada su utilidad.

En las siguientes páginas se pretende dar a conocer la tecnología CSS con un enfoque práctico para que en pocos capítulos se puedan utilizar de una manera depurada. No se pretende explorar todos los aspectos de la tecnología ya que para realizar sería necesario un curso entero.

Introducción a **CSS**

Javier Eguíluz Pérez

Sobre este libro...

- Los contenidos de este libro están bajo una licencia Creative Commons Reconocimiento - No Comercial - Sin Obra Derivada 3.0 (<http://creativecommons.org/licenses/by-nc-nd/3.0/deed.es>)
- **Esta versión impresa se creó el 6 de octubre de 2008 y todavía está incompleta.** La versión más actualizada de los contenidos de este libro se puede encontrar en <http://www.librosweb.es/css>
- Si quieras aportar sugerencias, comentarios, críticas o informar sobre errores, puedes enviarnos un mensaje a **contacto@librosweb.es**

Capítulo 1. Introducción	5
1.1. ¿Qué es CSS?	5
1.2. Breve historia de CSS	5
1.3. Soporte de CSS en los navegadores.....	6
1.4. Especificación oficial.....	7
1.5. Funcionamiento básico de CSS.....	7
1.6. Cómo incluir CSS en un documento XHTML.....	9
1.7. Glosario básico	11
1.8. Medios CSS	12
1.9. Comentarios	14
1.10. Sintaxis de la definición de cada propiedad CSS.....	15
Capítulo 2. Selectores.....	17
2.1. Selectores básicos.....	17
2.2. Selectores avanzados	25
2.3. Agrupación de reglas.....	27
2.4. Herencia.....	28
2.5. Colisiones de estilos.....	29
Capítulo 3. Unidades de medida y colores	31
3.1. Unidades de medida	31
3.2. Colores.....	37
Capítulo 4. Modelo de cajas (box model).....	42
4.1. Anchura y altura	44
4.2. Margen y relleno	45
4.3. Bordes.....	53
4.4. Margen, relleno, bordes y modelo de cajas	60
4.5. Fondos	62
Capítulo 5. Posicionamiento y visualización.....	71
5.1. Tipos de elementos	71
5.2. Posicionamiento	73
5.3. Posicionamiento normal.....	74
5.4. Posicionamiento relativo	76
5.5. Posicionamiento absoluto	78
5.6. Posicionamiento fijo	83
5.7. Posicionamiento flotante	84
5.8. Visualización	90
Capítulo 6. Texto	97
6.1. Tipografía.....	97
6.2. Texto	104
Capítulo 7. Enlaces	115
7.1. Estilos básicos	115
7.2. Estilos avanzados	117
Capítulo 8. Imágenes	121

8.1. Estilos básicos	121
8.2. Estilos avanzados	121
Capítulo 9. Listas	124
9.1. Estilos básicos	124
9.2. Estilos avanzados	131
Capítulo 10. Tablas	137
10.1. Estilos básicos	137
10.2. Estilos avanzados	140
Capítulo 11. Formularios	144
11.1. Estilos básicos	144
11.2. Estilos avanzados	149
Capítulo 12. Layout	152
12.1. Centrar una página horizontalmente	152
12.2. Centrar una página verticalmente	156
12.3. Estructura o layout	158
12.4. Alturas/anchuras máximas y mínimas	163
12.5. Estilos avanzados	165
Capítulo 13. Otros	167
13.1. Propiedades shorthand	167
13.2. Versión para imprimir	168
13.3. Personalizar el cursor	171
13.4. Hacks y filtros	174
13.5. Prioridad de las declaraciones CSS	176
13.6. Validador	177
13.7. Recomendaciones generales sobre CSS	177
Capítulo 14. Recursos útiles	183
14.1. Extensiones de Firefox	183
14.2. Aplicaciones web	186
14.3. Sitios web de inspiración	187
14.4. Referencias y colecciones de recursos	187
Capítulo 15. Ejercicios	189
Capítulo 16. Ejercicios resueltos	214

Capítulo 1. Introducción

1.1. ¿Qué es CSS?

CSS es un lenguaje de hojas de estilos creado para controlar el aspecto o presentación de los documentos electrónicos definidos con HTML y XHTML. CSS es la mejor forma de separar los contenidos y su presentación y es imprescindible para crear páginas web complejas.

Separar la definición de los contenidos y la definición de su aspecto presenta numerosas ventajas, ya que obliga a crear documentos HTML/XHTML bien definidos y con significado completo (también llamados "*documentos semánticos*"). Además, mejora la accesibilidad del documento, reduce la complejidad de su mantenimiento y permite visualizar el mismo documento en infinidad de dispositivos diferentes.

Al crear una página web, se utiliza en primer lugar el lenguaje HTML/XHTML para *marcar* los contenidos, es decir, para designar la función de cada elemento dentro de la página: párrafo, titular, texto destacado, tabla, lista de elementos, etc.

Una vez creados los contenidos, se utiliza el lenguaje CSS para definir el aspecto de cada elemento: color, tamaño y tipo de letra del texto, separación horizontal y vertical entre elementos, posición de cada elemento dentro de la página, etc.

1.2. Breve historia de CSS

Las hojas de estilos aparecieron poco después que el lenguaje de etiquetas SGML, alrededor del año 1970. Desde la creación de SGML, se observó la necesidad de definir un mecanismo que permitiera aplicar de forma consistente diferentes estilos a los documentos electrónicos.

El gran impulso de los lenguajes de hojas de estilos se produjo con el boom de Internet y el crecimiento exponencial del lenguaje HTML para la creación de documentos electrónicos. La guerra de navegadores y la falta de un estándar para la definición de los estilos dificultaban la creación de documentos con la misma apariencia en diferentes navegadores.

El organismo W3C (<http://www.w3.org/>) (World Wide Web Consortium), encargado de crear todos los estándares relacionados con la web, propuso la creación de un lenguaje de hojas de estilos específico para el lenguaje HTML y se presentaron nueve propuestas. Las dos propuestas que se tuvieron en cuenta fueron la CHSS (*Cascading HTML Style Sheets*) y la SSP (*Stream-based Style Sheet Proposal*).

La propuesta CHSS fue realizada por Håkon Wium Lie y SSP fue propuesto por Bert Bos. Entre finales de 1994 y 1995 Lie y Bos se unieron para definir un nuevo lenguaje que tomaba lo mejor de cada propuesta y lo llamaron CSS (*Cascading Style Sheets*).

En 1995, el W3C decidió apostar por el desarrollo y estandarización de CSS y lo añadió a su grupo de trabajo de HTML. A finales de 1996, el W3C publicó la primera recomendación oficial, conocida como "CSS nivel 1".

A principios de 1997, el W3C decide separar los trabajos del grupo de HTML en tres secciones: el grupo de trabajo de HTML, el grupo de trabajo de DOM y el grupo de trabajo de CSS.

El 12 de Mayo de 1998, el grupo de trabajo de CSS publica su segunda recomendación oficial, conocida como "CSS nivel 2". La versión de CSS que utilizan todos los navegadores de hoy en día es CSS 2.1, una revisión de CSS 2 que aún se está elaborando (la última actualización es del 19 de julio de 2007). Al mismo tiempo, la siguiente recomendación de CSS, conocida como "CSS nivel 3", continúa en desarrollo desde 1998 y hasta el momento sólo se han publicado borradores.

La adopción de CSS por parte de los navegadores ha requerido un largo periodo de tiempo. El mismo año que se publicó CSS 1, Microsoft lanzaba su navegador Internet Explorer 3.0, que disponía de un soporte bastante reducido de CSS. El primer navegador con soporte completo de CSS 1 fue la versión para Mac de Internet Explorer 5, que se publicó en el año 2000. Por el momento, ningún navegador tiene soporte completo de CSS 2.1.

1.3. Soporte de CSS en los navegadores

El trabajo del diseñador web siempre está limitado por las posibilidades de los navegadores que utilizan los usuarios para acceder a sus páginas. Por este motivo es imprescindible conocer el soporte de CSS en cada uno de los navegadores más utilizados del mercado.

Internamente los navegadores están divididos en varios componentes. La parte del navegador que se encarga de interpretar el código HTML y CSS para mostrar las páginas se denomina motor. Desde el punto de vista del diseñador CSS, la versión de un motor es mucho más importante que la versión del propio navegador.

La siguiente tabla muestra el soporte de CSS 1, CSS 2.1 y CSS 3 de los cuatro navegadores más utilizados por los usuarios:

Navegador	Motor	CSS 1	CSS 2.1	CSS 3
Internet Explorer	Trident	Completo desde la versión 6.0	Casi completo desde la versión 7.0	Prácticamente nulo
Firefox	Gecko	Completo	Casi completo	Selectores, pseudo-clases y algunas propiedades
Safari	WebKit	Completo	Casi completo	Todos los selectores, pseudo-clases y muchas propiedades
Opera	Presto	Completo	Casi completo	Todos los selectores, pseudo-clases y muchas propiedades
Google Chrome	WebKit	Completo	Casi completo	Todos los selectores, pseudo-clases y muchas propiedades

Los navegadores Safari y Opera son los más avanzados en el soporte de CSS, ya que incluyen muchos elementos de la futura versión CSS 3 y un soporte casi perfecto de la actual versión 2.1. El navegador Firefox no tiene un soporte tan avanzado de CSS 3 pero las últimas versiones están alcanzando rápidamente a Safari y Opera.

Por su parte, el navegador Internet Explorer sólo puede considerarse adecuado desde el punto de vista de CSS a partir de su versión 7. De hecho, la versión Internet Explorer 6 que aún utilizan muchos usuarios sufre carencias muy importantes y contiene decenas de errores en su soporte de CSS. Afortunadamente, Internet Explorer 8 asegura que su soporte de CSS 2.1 será tan completo como el del resto de navegadores.

La información de la tabla anterior ha sido elaborada a partir de la información que se puede encontrar en la página *Comparison of layout engines* ([http://en.wikipedia.org/wiki/Comparison_of_layout_engines_\(Cascading_Style_Sheets\)](http://en.wikipedia.org/wiki/Comparison_of_layout_engines_(Cascading_Style_Sheets))) de la Wikipedia, donde se muestra una comparación exhaustiva sobre el soporte de todas las características de CSS por parte de cada navegador.

1.4. Especificación oficial

La especificación o norma oficial que se utiliza actualmente para diseñar páginas web con CSS es la versión CSS 2.1, actualizada por última vez el 19 de julio de 2007 y que se puede consultar libremente en <http://www.w3.org/TR/CSS21/>

Desde hace varios años, el organismo W3C trabaja en la elaboración de la próxima versión de CSS, conocida como CSS 3. Esta nueva versión incluye miles de cambios importantes en todos los niveles y es mucho más avanzada y compleja que CSS 2.

No obstante, pasarán muchos años hasta que se publique la versión definitiva completa de CSS 3 y hasta que los principales navegadores del mercado incluyan la mayor parte del nuevo estándar.

El sitio web del organismo W3C dispone de una sección en la que se detalla el trabajo que el W3C está desarrollando actualmente en relación a CSS (<http://www.w3.org/Style/CSS/current-work>) y también dispone de un blog en el que se publican todas las novedades relacionadas con CSS (<http://www.w3.org/blog/CSS>).

1.5. Funcionamiento básico de CSS

Antes de la adopción de CSS, los diseñadores de páginas web debían definir el aspecto de cada elemento dentro de las etiquetas HTML de la página. El siguiente ejemplo muestra una página HTML con estilos definidos sin utilizar CSS:

```
<!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">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
<title>Ejemplo de estilos sin CSS</title>
</head>

<body>

<h1><font color="red" face="Arial" size="5">Titular de la página</font></h1>

<p><font color="gray" face="Verdana" size="2">Un párrafo de texto no muy
```

```
largo.</font></p>

</body>
</html>
```

El ejemplo anterior utiliza la etiqueta `` con sus atributos `color`, `face` y `size` para definir el color, la tipografía y el tamaño del texto de cada elemento del documento.

El principal problema de esta forma de definir el aspecto de los elementos se puede ver claramente con el siguiente ejemplo: si la página tuviera 50 elementos diferentes, habría que insertar 50 etiquetas ``. Si el sitio web entero se compone de 10.000 páginas diferentes, habría que definir 500.000 etiquetas ``. Como cada etiqueta `` tiene 3 atributos, habría que definir 1.5 millones de atributos.

Por otra parte, el diseño de los sitios web está en constante evolución y es habitual modificar cada cierto tiempo los colores principales de las páginas o la tipografía utilizada para el texto. Si se emplea la etiqueta ``, habría que modificar el valor de 1.5 millones de atributos para modificar el diseño general del sitio web.

La solución que propone CSS es mucho mejor, como se puede ver en el siguiente ejemplo:

```
<!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">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
<title>Ejemplo de estilos con CSS</title>
<style type="text/css">
  h1 { color: red; font-family: Arial; font-size: large; }
  p { color: gray; font-family: Verdana; font-size: medium; }
</style>
</head>

<body>
<h1>Titular de la página</h1>
<p>Un párrafo de texto no muy largo.</p>
</body>
</html>
```

CSS permite separar los contenidos de la página y su aspecto o presentación. En el ejemplo anterior, dentro de la propia página HTML se reserva una zona en la que se incluye toda la información relacionada con los estilos de la página.

Utilizando CSS, en esa zona reservada se indica que todas las etiquetas `<h1>` de la página se deben ver de color rojo, con un tipo de letra Arial y con un tamaño de letra grande. Además, las etiquetas `<p>` de la página se deben ver de color gris, con un tipo de letra Verdana y con un tamaño de letra medio.

Definiendo los estilos de esta forma, no importa el número de elementos `<p>` que existan en la página, ya que todos tendrán el mismo aspecto establecido mediante CSS. Como se verá a continuación, esta forma de trabajar con CSS no es ideal, ya que si el sitio web dispone de 10.000 páginas, habría que definir 10.000 veces el mismo estilo CSS.

1.6. Cómo incluir CSS en un documento XHTML

Una de las principales características de CSS es su flexibilidad y las diferentes opciones que ofrece para realizar una misma tarea. De hecho, existen tres opciones para incluir CSS en un documento HTML.

1.6.1. Incluir CSS en el mismo documento HTML

Los estilos se definen en una zona específica del propio documento HTML. Se emplea la etiqueta <style> de HTML y solamente se pueden incluir en la cabecera del documento (sólo dentro de la sección <head>).

Ejemplo:

```
<!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">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
<title>Ejemplo de estilos CSS en el propio documento</title>
<style type="text/css">
  p { color: black; font-family: Verdana; }
</style>
</head>

<body>
<p>Un párrafo de texto.</p>
</body>
</html>
```

Este método se emplea cuando se define un número pequeño de estilos o cuando se quieren incluir estilos específicos en una determinada página HTML que completen los estilos que se incluyen por defecto en todas las páginas del sitio web.

El principal inconveniente es que si se quiere hacer una modificación en los estilos definidos, es necesario modificar todas las páginas que incluyen el estilo que se va a modificar.

1.6.2. Definir CSS en un archivo externo

En este caso, todos los estilos CSS se incluyen en un archivo de tipo CSS que las páginas HTML enlazan mediante la etiqueta <link>. Un archivo de tipo CSS no es más que un archivo simple de texto cuya extensión es .css. Se pueden crear todos los archivos CSS que sean necesarios y cada página HTML puede enlazar tantos archivos CSS como necesite.

En el siguiente ejemplo, se crea un archivo de texto, se cambia su nombre a `estilos.css` y se incluye el siguiente contenido:

```
| p { color: black; font-family: Verdana; }
```

A continuación, en la página HTML se utiliza la etiqueta <link> para enlazar el archivo CSS externo que tiene los estilos que va a utilizar la página:

```
<!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">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
<title>Ejemplo de estilos CSS en un archivo externo</title>
<link rel="stylesheet" type="text/css" href="/css/estilos.css" media="screen" />
</head>

<body>
<p>Un párrafo de texto.</p>
</body>
</html>
```

Normalmente, la etiqueta `<link>` incluye cuatro atributos cuando se enlaza un archivo CSS:

- `rel`: indica el tipo de relación que tiene el recurso enlazado (en este caso, el archivo CSS) y la página HTML. Para los archivos CSS, siempre se utiliza el valor `stylesheet`
- `type`: indica el tipo de recurso enlazado. Sus valores están estandarizados y para los archivos CSS su valor siempre es `text/css`
- `href`: indica la URL del archivo CSS que contiene los estilos. La URL indicada puede ser relativa o absoluta y puede apuntar a un recurso interno o externo al sitio web.
- `media`: indica el medio en el que se van a aplicar los estilos del archivo CSS. Más adelante se explican en detalle los medios CSS y su funcionamiento.

De todas las formas de incluir CSS en las páginas HTML, esta es la más utilizada con mucha diferencia. La principal ventaja es que se puede incluir un mismo archivo CSS en multitud de páginas HTML, por lo que se garantiza la aplicación homogénea de los mismos estilos a todas las páginas que forman un sitio web.

Con este método, el mantenimiento del sitio web se simplifica al máximo, ya que un solo cambio en un solo archivo CSS permite variar de forma instantánea los estilos de todas las páginas HTML que enlazan ese archivo.

Aunque generalmente se emplea la etiqueta `<link>` para enlazar los archivos CSS externos, también se puede utilizar la etiqueta `<style>`. La forma alternativa de incluir un archivo CSS externo se muestra a continuación:

```
<!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">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
<title>Ejemplo de estilos CSS en un archivo externo</title>
<style type="text/css" media="screen">
  @import '/css/estilos.css';
</style>
</head>

<body>
<p>Un párrafo de texto.</p>
</body>
```

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

En este caso, para incluir en la página HTML los estilos definidos en archivos CSS externos se utiliza una regla especial de tipo `@import`. Las reglas de tipo `@import` siempre preceden a cualquier otra regla CSS (con la única excepción de la regla `@charset`).

La URL del archivo CSS externo se indica mediante una cadena de texto encerrada con comillas simples o dobles o mediante la palabra reservada `url()`. De esta forma, las siguientes reglas `@import` son equivalentes:

```
@import '/css/estilos.css';
@import "/css/estilos.css";
@import url('/css/estilos.css');
@import url("/css/estilos.css");
```

1.6.3. Incluir CSS en los elementos HTML

El último método para incluir estilos CSS en documentos HTML es el peor y el menos utilizado, ya que tiene los mismos problemas que la utilización de las etiquetas ``.

Ejemplo:

```
<!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">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
<title>Ejemplo de estilos CSS en el propio documento</title>
</head>

<body>
<p style="color: black; font-family: Verdana;">Un párrafo de texto.</p>
</body>
</html>
```

Esta forma de incluir CSS directamente en los elementos HTML solamente se utiliza en determinadas situaciones en las que se debe incluir un estilo muy específico para un solo elemento concreto.

1.7. Glosario básico

CSS define una serie de términos que permiten describir cada una de las partes que componen los estilos CSS. El siguiente esquema muestra las partes que forman un estilo CSS muy básico:

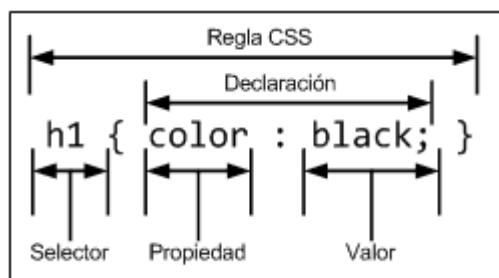


Figura 1.1. Componentes de un estilo CSS básico

Los diferentes términos se definen a continuación:

- Regla: cada uno de los estilos que componen una hoja de estilos CSS. Cada regla está compuesta de una parte de *"selectores"*, un símbolo de *"llave de apertura"* ({}, otra parte denominada *"declaraciones"* y por último, un símbolo de *"llave de cierre"* }).
- Selector: indica el elemento o elementos HTML a los que se aplica la regla CSS.
- Declaración: especifica los estilos que se aplican a los elementos. Está compuesta por una o más propiedades CSS.
- Propiedad: permite modificar el aspecto de una característica del elemento.
- Valor: indica el nuevo valor de la característica modificada en el elemento.

Un archivo CSS puede contener *infinitas* reglas CSS, cada regla puede contener *infinitos* selectores y cada declaración puede estar formada por un número *infinito* de pares propiedad/valor.

1.8. Medios CSS

Una de las características más importantes de las hojas de estilos CSS es que permiten definir diferentes estilos para diferentes medios o dispositivos: pantallas, impresoras, móviles, proyectores, etc.

Además, CSS define algunas propiedades específicamente para determinados medios, como por ejemplo la paginación y los saltos de página para los medios impresos o el volumen y tipo de voz para los medios de audio. La siguiente tabla muestra el nombre que CSS utiliza para identificar cada medio y su descripción:

Medio	Descripción
all	Todos los medios definidos
braille	Dispositivos táctiles que emplean el sistema braille
embossed	Impresoras braille
handheld	Dispositivos de mano: móviles, PDA, etc.
print	Impresoras y navegadores en el modo <i>"Vista Previa para Imprimir"</i>
projection	Proyectores y dispositivos para presentaciones
screen	Pantallas de ordenador
speech	Sintetizadores para navegadores de voz utilizados por personas discapacitadas
tty	Dispositivos textuales limitados como teletipos y terminales de texto
tv	Televisores y dispositivos con resolución baja

Los medios más utilizados actualmente son *screen* (para definir el aspecto de la página en pantalla) y *print* (para definir el aspecto de la página cuando se imprime), seguidos de *handheld* (que define el aspecto de la página cuando se visualiza mediante un dispositivo móvil).

Además, CSS clasifica a los medios en diferentes grupos según sus características. La siguiente tabla resume todos los grupos definidos en el estándar:

Medio	Continuo / Paginado	Visual / Auditivo / Táctil / Vocal	Mapa de bits / Caracteres	Interactivo / Estático
braille	continuo	táctil	caracteres	ambos
embossed	paginado	táctil	caracteres	estático
handheld	ambos	visual, auditivo, vocal	ambos	ambos
print	paginado	visual	mapa de bits	estático
projection	paginado	visual	mapa de bits	interactivo
screen	continuo	visual, auditivo	mapa de bits	ambos
speech	continuo	vocal	(no tiene sentido)	ambos
tty	continuo	visual	caracteres	ambos
tv	ambos	visual, auditivo	mapa de bits	ambos

La gran ventaja de CSS es que permite modificar los estilos de una página en función del medio en el que se visualiza. Existen cuatro formas diferentes de indicar el medio en el que se deben aplicar los estilos CSS.

1.8.1. Medios definidos con las reglas de tipo @media

Las reglas @media son un tipo especial de regla CSS que permiten indicar de forma directa el medio o medios en los que se aplicarán los estilos incluidos en la regla. Para especificar el medio en el que se aplican los estilos, se incluye su nombre después de @media. Si los estilos se aplican a varios medios, se incluyen los nombres de todos los medios separados por comas.

A continuación se muestra un ejemplo sencillo:

```
@media print {
    body { font-size: 10pt }
}
@media screen {
    body { font-size: 13px }
}
@media screen, print {
    body { line-height: 1.2 }
}
```

El ejemplo anterior establece que el tamaño de letra de la página cuando se visualiza en una pantalla debe ser 13 píxel. Sin embargo, cuando se imprimen los contenidos de la página, su tamaño de letra debe ser de 10 puntos. Por último, tanto cuando la página se visualiza en una pantalla como cuando se imprimen sus contenidos, el interlineado del texto debe ser de 1.2 veces el tamaño de letra del texto.

1.8.2. Medios definidos con las reglas de tipo @import

Cuando se utilizan reglas de tipo `@import` para enlazar archivos CSS externos, se puede especificar el medio en el que se aplican los estilos indicando el nombre del medio después de la URL del archivo CSS:

```
| @import url("estilos_basicos.css") screen;
| @import url("estilos_impresora.css") print;
```

Las reglas del ejemplo anterior establecen que cuando la página se visualiza por pantalla, se cargan los estilos definidos en el primer archivo CSS. Por otra parte, cuando la página se imprime, se tienen en cuenta los estilos que define el segundo archivo CSS.

Si los estilos del archivo CSS externo deben aplicarse en varios medios, se indican los nombres de todos los medios separados por comas. Si no se indica el medio en una regla de tipo `@import`, el navegador sobreentiende que el medio es `all`, es decir, que los estilos se aplican en todos los medios.

1.8.3. Medios definidos con la etiqueta <link>

Si se utiliza la etiqueta `<link>` para enlazar los archivos CSS externos, se puede utilizar el atributo `media` para indicar el medio o medios en los que se aplican los estilos de cada archivo:

```
| <link rel="stylesheet" type="text/css" media="screen" href="basico.css" />
| <link rel="stylesheet" type="text/css" media="print, handheld" href="especial.css" />
```

En este ejemplo, el primer archivo CSS se tiene en cuenta cuando la página se visualiza en la pantalla (`media="screen"`). Los estilos indicados en el segundo archivo CSS, se aplican al imprimir la página (`media="print"`) o al visualizarla en un dispositivo móvil (`media="handheld"`), como por ejemplo en un iPhone.

1.8.4. Medios definidos mezclando varios métodos

CSS también permite mezclar los tres métodos anteriores para indicar los medios en los que se aplica cada archivo CSS externo:

```
| <link rel="stylesheet" type="text/css" media="screen" href="basico.css" />
| @import url("estilos_seccion.css") screen;
| @media print {
|   /* Estilos específicos para impresora */
| }
```

Los estilos CSS que se aplican cuando se visualiza la página en una pantalla se obtienen mediante el recurso enlazado con la etiqueta `<link>` y mediante el archivo CSS externo incluido con la regla de tipo `@import`. Además, los estilos aplicados cuando se imprime la página se indican directamente en la página HTML mediante la regla de tipo `@media`.

1.9. Comentarios

CSS permite incluir comentarios entre sus reglas y estilos. Los comentarios son contenidos de texto que el diseñador incluye en el archivo CSS para su propia información y utilidad. Los

navegadores ignoran por completo cualquier comentario de los archivos CSS, por lo que es común utilizarlos para estructurar de forma clara los archivos CSS complejos.

El comienzo de un comentario se indica mediante los caracteres `/*` y el final del comentario se indica mediante `*/`, tal y como se muestra en el siguiente ejemplo:

```
/* Este es un comentario en CSS */
```

Los comentarios pueden ocupar tantas líneas como sea necesario, pero no se puede incluir un comentario dentro de otro comentario:

```
/* Este es un
comentario CSS de varias
lineas */
```

Aunque los navegadores ignoran los comentarios, su contenido se envía junto con el resto de estilos, por lo que no se debe incluir en ellos ninguna información sensible o confidencial.

La sintaxis de los comentarios CSS es muy diferente a la de los comentarios HTML, por lo que no deben confundirse:

```
<!-- Este es un comentario en HTML -->
<!-- Este es un
comentario HTML de varias
lineas -->
```

1.10. Sintaxis de la definición de cada propiedad CSS

A lo largo de los próximos capítulos, se incluyen las definiciones formales de la mayoría de propiedades de CSS. La definición formal se basa en la información recogida en el estándar oficial y se muestra en forma de tabla.

Una de las principales informaciones de cada definición es la lista de posibles valores que admite la propiedad. Para definir la lista de valores permitidos se sigue un formato que es necesario detallar.

Si el valor permitido se indica como una sucesión de palabras sin ningún carácter que las separe (paréntesis, comas, barras, etc.) el valor de la propiedad se debe indicar tal y como se muestra y con esas palabras en el mismo orden.

Si el valor permitido se indica como una sucesión de valores separados por una barra simple () el valor de la propiedad debe tomar uno y sólo uno de los valores indicados. Por ejemplo, la notación `<porcentaje> | <medida> | inherit` indica que la propiedad solamente puede tomar como valor la palabra reservada `inherit` o un porcentaje o una medida.

Si el valor permitido se indica como una sucesión de valores separados por una barra doble () el valor de la propiedad puede tomar uno o más valores de los indicados y en cualquier orden.

Por ejemplo, la notación `<color> || <estilo> || <medida>` indica que la propiedad puede tomar como valor cualquier combinación de los valores indicados y en cualquier orden. Se

podría establecer un color y un estilo, solamente una medida o una medida y un estilo. Además, el orden en el que se indican los valores es indiferente. Opcionalmente, se pueden utilizar paréntesis para agrupar diferentes valores.

Por último, en cada valor o agrupación de valores se puede indicar el tipo de valor: opcional, obligatorio, múltiple o restringido.

El carácter * indica que el valor ocurre cero o más veces; el carácter + indica que el valor ocurre una o más veces; el carácter ? indica que el valor es opcional y por último, el carácter {número_1, número_2} indica que el valor ocurre al menos tantas veces como el valor indicado en número_1 y como máximo tantas veces como el valor indicado en número_2.

Por ejemplo, el valor [<family-name> ,]* indica que el valor de tipo <family_name> seguido por una coma se puede incluir cero o más veces. El valor <url>? <color> significa que la URL es opcional y el color obligatorio y en el orden indicado. Por último, el valor [<medida> | thick | thin] {1,4} indica que se pueden escribir entre 1 y 4 veces un valor que sea o una medida o la palabra thick o la palabra thin.

No obstante, la mejor forma de entender la notación formal para las propiedades de CSS es observar la definición de cada propiedad y volver a esta sección siempre que sea necesario.

Capítulo 2. Selectores

Para crear diseños web profesionales, es imprescindible conocer y dominar los selectores de CSS. Como se vio en el capítulo anterior, una regla de CSS está formada por una parte llamada "selector" y otra parte llamada "declaración".

La declaración indica "*qué hay que hacer*" y el selector indica "*a quién hay que hacérselo*". Por lo tanto, los selectores son imprescindibles para aplicar de forma correcta los estilos CSS en una página.

A un mismo elemento HTML se le pueden asignar *infinitas* reglas CSS y cada regla CSS puede aplicarse a un número *infinito* de elementos. En otras palabras, una misma regla puede aplicarse sobre varios selectores y un mismo selector se puede utilizar en varias reglas.

El estándar de CSS 2.1 incluye una docena de tipos diferentes de selectores, que permiten seleccionar de forma muy precisa elementos individuales o conjuntos de elementos dentro de una página web.

No obstante, la mayoría de páginas de los sitios web se pueden diseñar utilizando solamente los cinco selectores básicos.

2.1. Selectores básicos

2.1.1. Selector universal

Se utiliza para seleccionar todos los elementos de la página. El siguiente ejemplo elimina el margen y el relleno de todos los elementos HTML (por ahora no es importante fijarse en la parte de la declaración de la regla CSS):

```
* {  
    margin: 0;  
    padding: 0;  
}
```

El selector universal se indica mediante un asterisco (*). A pesar de su sencillez, no se utiliza habitualmente, ya que es difícil que un mismo estilo se pueda aplicar a todos los elementos de una página.

No obstante, sí que se suele combinar con otros selectores y además, forma parte de algunos *hacks* muy utilizados (como se verá más adelante).

2.1.2. Selector de tipo o etiqueta

Selecciona todos los elementos de la página cuya etiqueta HTML coincide con el valor del selector. El siguiente ejemplo selecciona todos los párrafos de la página:

```
p {  
    ...  
}
```

Para utilizar este selector, solamente es necesario indicar el nombre de una etiqueta HTML (sin los caracteres < y >) correspondiente a los elementos que se quieren seleccionar.

El siguiente ejemplo aplica diferentes estilos a los titulares y a los párrafos de una página HTML:

```
h1 {  
    color: red;  
}  
  
h2 {  
    color: blue;  
}  
  
p {  
    color: black;  
}
```

Si se quiere aplicar los mismos estilos a dos etiquetas diferentes, se pueden encadenar los selectores. En el siguiente ejemplo, los títulos de sección h1, h2 y h3 comparten los mismos estilos:

```
h1 {  
    color: #8A8E27;  
    font-weight: normal;  
    font-family: Arial, Helvetica, sans-serif;  
}  
h2 {  
    color: #8A8E27;  
    font-weight: normal;  
    font-family: Arial, Helvetica, sans-serif;  
}  
h3 {  
    color: #8A8E27;  
    font-weight: normal;  
    font-family: Arial, Helvetica, sans-serif;  
}
```

En este caso, CSS permite agrupar todas las reglas individuales en una sola regla con un selector múltiple. Para ello, se incluyen todos los selectores separados por una coma (,) y el resultado es que la siguiente regla CSS es equivalente a las tres reglas anteriores:

```
h1, h2, h3 {  
    color: #8A8E27;  
    font-weight: normal;  
    font-family: Arial, Helvetica, sans-serif;  
}
```

En las hojas de estilo complejas, es habitual agrupar las propiedades comunes de varios elementos en una única regla CSS y posteriormente definir las propiedades específicas de esos mismos elementos. El siguiente ejemplo establece en primer lugar las propiedades comunes de los títulos de sección (color y tipo de letra) y a continuación, establece el tamaño de letra de cada uno de ellos:

```
h1, h2, h3 {  
    color: #8A8E27;
```

```
    font-weight: normal;
    font-family: Arial, Helvetica, sans-serif;
}

h1 { font-size: 2em; }
h2 { font-size: 1.5em; }
h3 { font-size: 1.2em; }
```

2.1.3. Selector descendente

Selecciona los elementos que se encuentra dentro de otros elementos. Un elemento es descendiente de otro cuando se encuentra entre las etiquetas de apertura y de cierre del otro elemento.

El selector del siguiente ejemplo selecciona todos los elementos `` de la página que se encuentren dentro de un elemento `<p>`:

```
| p span { color: red; }
```

Si el código HTML de la página es el siguiente:

```
<p>
  ...
<span>texto1</span>
  ...
<a href="">...<span>texto2</span></a>
  ...
</p>
```

El selector `p span` selecciona tanto `texto1` como `texto2`. El motivo es que en el selector descendente, un elemento no tiene que ser "*hijo directo*" de otro. La única condición es que un elemento debe estar dentro de otro elemento, sin importar lo profundo que se encuentre.

Al resto de elementos `` de la página que no están dentro de un elemento `<p>`, no se les aplica la regla CSS anterior.

Los selectores descendentes permiten mejorar la precisión del selector de tipo o etiqueta. Así, utilizando el selector descendente es posible aplicar diferentes estilos a los elementos del mismo tipo. El siguiente ejemplo amplía el anterior y muestra de color azul todo el texto de los `` contenidos dentro de un `<h1>`:

```
| p span { color: red; }
| h1 span { color: blue; }
```

Con las reglas CSS anteriores:

- Los elementos `` que se encuentran dentro de un elemento `<p>` se muestran de color rojo.
- Los elementos `` que se encuentran dentro de un elemento `<h1>` se muestran de color azul.
- El resto de elementos `` de la página, se muestran con el color por defecto aplicado por el navegador.

La sintaxis formal del selector descendente se muestra a continuación:

```
| elemento1 elemento2 elemento3 ... elementoN
```

Los selectores descendentes siempre están formados por dos o más partes separadas entre sí por espacios en blanco. La última parte indica el elemento sobre el que se aplican los estilos y todas las partes anteriores indican el lugar en el que se tiene que encontrar ese elemento para que los estilos se apliquen realmente.

En el siguiente ejemplo, el selector descendente se compone de cuatro partes:

```
| p a span em { text-decoration: underline; }
```

Los estilos de la regla anterior se aplican a los elementos de tipo `` que se encuentren dentro de elementos de tipo ``, que a su vez se encuentren dentro de elementos de tipo `<a>` que se encuentren dentro de elementos de tipo `<p>`.

No debe confundirse el selector descendente con la combinación de selectores:

```
/* El estilo se aplica a todos los elementos "p", "a", "span" y "em" */
p, a, span, em { text-decoration: underline; }

/* El estilo se aplica solo a los elementos "em" que se
   encuentran dentro de "p a span" */
p a span em { text-decoration: underline; }
```

Si se emplea el selector descendente combinado con el selector universal, se puede restringir el alcance de un selector descendente. El siguiente ejemplo, muestra los dos enlaces de color rojo:

```
| p a { color: red; }

<p><a href="#">Enlace</a></p>
<p><span><a href="#">Enlace</a></span></p>
```

Sin embargo, en el siguiente ejemplo solamente el segundo enlace se muestra de color rojo:

```
| p * a { color: red; }

<p><a href="#">Enlace</a></p>
<p><span><a href="#">Enlace</a></span></p>
```

La razón es que el selector `p * a` se traduce como *todos los elementos de tipo `<a>` que se encuentren dentro de cualquier elemento que, a su vez, se encuentre dentro de un elemento de tipo `<p>`*. Como el primer elemento `<a>` se encuentra directamente bajo un elemento `<p>`, no se cumple la condición del selector `p * a`.

2.1.4. Selector de clase

Si se considera el siguiente código HTML de ejemplo:

```
| <body>
|   <p>Lorem ipsum dolor sit amet...</p>
|   <p>Nunc sed lacus et est adipiscing accumsan...</p>
|   <p>Class aptent taciti sociosqu ad litora...</p>
| </body>
```

¿Cómo se pueden aplicar estilos CSS sólo al primer párrafo? El selector universal (*) no se puede utilizar porque selecciona todos los elementos de la página. El selector de tipo o etiqueta (p) tampoco se puede utilizar porque seleccionaría todos los párrafos. Por último, el selector descendente (body p) tampoco se puede utilizar porque todos los párrafos se encuentran en el mismo sitio.

Una de las soluciones más sencillas para aplicar estilos a un solo elemento de la página consiste en utilizar el atributo `class` de HTML sobre ese elemento para indicar directamente la regla CSS que se le debe aplicar:

```
<body>
  <p class="destacado">Lorem ipsum dolor sit amet...</p>
  <p>Nunc sed lacus et est adipiscing accumsan...</p>
  <p>Class aptent taciti sociosqu ad litora...</p>
</body>
```

A continuación, se crea en el archivo CSS una nueva regla llamada `destacado` con todos los estilos que se van a aplicar al elemento. Para que el navegador no confunda este selector con los otros tipos de selectores, se prefija el valor del atributo `class` con un punto (.) tal y como muestra el siguiente ejemplo:

```
.destacado { color: red; }
```

El selector `.destacado` se interpreta como "*cualquier elemento de la página cuyo atributo class sea igual a destacado*", por lo que solamente el primer párrafo cumple esa condición.

Este tipo de selectores se llaman selectores de clase y son los más utilizados junto con los selectores de ID que se verán a continuación. La principal característica de este selector es que en una misma página HTML varios elementos diferentes pueden incluir el mismo valor en el atributo `class`:

```
<body>
  <p class="destacado">Lorem ipsum dolor sit amet...</p>
  <p>Nunc sed lacus et <a href="#" class="destacado">est adipiscing</a> accumsan...</p>
  <p>Class aptent taciti <em class="destacado">sociosqu ad</em> litora...</p>
</body>
```

Los selectores de clase son imprescindibles para diseñar páginas web complejas, ya que permiten disponer de una precisión total al seleccionar los elementos. Además, estos selectores permiten reutilizar los mismos estilos para varios elementos diferentes.

A continuación se muestra otro ejemplo de selectores de clase:

```
.aviso {
  padding: 0.5em;
  border: 1px solid #98be10;
  background: #f6fed4;
}

.error {
  color: #930;
  font-weight: bold;
}
```

```
<span class="error">...</span>
<div class="aviso">...</div>
```

El elemento `` tiene un atributo `class="error"`, por lo que se le aplican las reglas CSS indicadas por el selector `.error`. Por su parte, el elemento `<div>` tiene un atributo `class="aviso"`, por lo que su estilo es el que definen las reglas CSS del selector `.aviso`.

En ocasiones, es necesario restringir el alcance del selector de clase. Si se considera de nuevo el ejemplo anterior:

```
<body>
  <p class="destacado">Lorem ipsum dolor sit amet...</p>
  <p>Nunc sed lacus et <a href="#" class="destacado">est adipiscing</a> accumsan...</p>
    <p>Class aptent taciti <em class="destacado">sociosqu ad</em> litora...</p>
</body>
```

¿Cómo es posible aplicar estilos solamente al párrafo cuyo atributo `class` sea igual a `destacado`? Combinando el selector de tipo y el selector de clase, se obtiene un selector mucho más específico:

```
p.destacado { color: red }
```

El selector `p.destacado` se interpreta como "*aquellos elementos de tipo `<p>` que dispongan de un atributo `class` con valor `destacado`*". De la misma forma, el selector `a.destacado` solamente selecciona los enlaces cuyo atributo `class` sea igual a `destacado`.

De lo anterior se deduce que el atributo `.destacado` es equivalente a `*.destacado`, por lo que todos los diseñadores obvian el símbolo `*` al escribir un selector de clase normal.

No debe confundirse el selector de clase con los selectores anteriores:

```
/* Todos los elementos de tipo "p" con atributo class="aviso" */
p.aviso { ... }

/* Todos los elementos con atributo class="aviso" que estén dentro
   de cualquier elemento de tipo "p" */
p .aviso { ... }

/* Todos los elementos "p" de la página y todos los elementos con
   atributo class="aviso" de la página */
p, .aviso { ... }
```

Por último, es posible aplicar los estilos de varias clases CSS sobre un mismo elemento. La sintaxis es similar, pero los diferentes valores del atributo `class` se separan con espacios en blanco. En el siguiente ejemplo:

```
<p class="especial destacado error">Párrafo de texto...</p>
```

Al párrafo anterior se le aplican los estilos definidos en las reglas `.especial`, `.destacado` y `.error`, por lo que en el siguiente ejemplo, el texto del párrafo se vería de color rojo, en negrita y con un tamaño de letra de 15 píxel:

```
.error { color: red; }
.destacado { font-size: 15px; }
```

```
.especial { font-weight: bold; }

<p class="especial destacado error">Párrafo de texto...</p>
```

Si un elemento dispone de un atributo `class` con más de un valor, es posible utilizar un selector más avanzado:

```
.error { color: red; }
.error.destacado { color: blue; }
.destacado { font-size: 15px; }
.especial { font-weight: bold; }

<p class="especial destacado error">Párrafo de texto...</p>
```

En el ejemplo anterior, el color de la letra del texto es azul y no rojo. El motivo es que se ha utilizado un selector de clase múltiple `.error.destacado`, que se interpreta como "*aquellos elementos de la página que dispongan de un atributo class con al menos los valores error y destacado*".

2.1.5. Selectores de ID

En ocasiones, es necesario aplicar estilos CSS a un único elemento de la página. Aunque puede utilizarse un selector de clase para aplicar estilos a un único elemento, existe otro selector más eficiente en este caso.

El selector de ID permite seleccionar un elemento de la página a través del valor de su atributo `id`. Este tipo de selectores sólo seleccionan un elemento de la página porque el valor del atributo `id` no se puede repetir en dos elementos diferentes de una misma página.

La sintaxis de los selectores de ID es muy parecida a la de los selectores de clase, salvo que se utiliza el símbolo de la almohadilla (#) en vez del punto (.) como prefijo del nombre de la regla CSS:

```
#destacado { color: red; }

<p>Primer párrafo</p>
<p id="destacado">Segundo párrafo</p>
<p>Tercer párrafo</p>
```

En el ejemplo anterior, el selector `#destacado` solamente selecciona el segundo párrafo (cuyo atributo `id` es igual a `destacado`).

La principal diferencia entre este tipo de selector y el selector de clase tiene que ver con HTML y no con CSS. Como se sabe, en una misma página, el valor del atributo `id` debe ser único, de forma que dos elementos diferentes no pueden tener el mismo valor de `id`. Sin embargo, el atributo `class` no es obligatorio que sea único, de forma que muchos elementos HTML diferentes pueden compartir el mismo valor para su atributo `class`.

De esta forma, la recomendación general es la de utilizar el selector de ID cuando se quiere aplicar un estilo a un solo elemento específico de la página y utilizar el selector de clase cuando se quiere aplicar un estilo a varios elementos diferentes de la página HTML.

Al igual que los selectores de clase, en este caso también se puede restringir el alcance del selector mediante la combinación con otros selectores. El siguiente ejemplo aplica la regla CSS solamente al elemento de tipo `<p>` que tenga un atributo `id` igual al indicado:

```
| p#aviso { color: blue; }
```

A primera vista, restringir el alcance de un selector de ID puede parecer absurdo. En realidad, un selector de tipo `p#aviso` sólo tiene sentido cuando el archivo CSS se aplica sobre muchas páginas HTML diferentes.

En este caso, algunas páginas pueden disponer de elementos con un atributo `id` igual a `aviso` y que no sean párrafos, por lo que la regla anterior no se aplica sobre esos elementos.

No debe confundirse el selector de ID con los selectores anteriores:

```
/* Todos los elementos de tipo "p" con atributo id="aviso" */
p#aviso { ... }

/* Todos los elementos con atributo id="aviso" que estén dentro
   de cualquier elemento de tipo "p" */
p #aviso { ... }

/* Todos los elementos "p" de La página y todos los elementos con
   atributo id="aviso" de La página */
p, #aviso { ... }
```

2.1.6. Combinación de selectores básicos

CSS permite la combinación de uno o más tipos de selectores para restringir el alcance de las reglas CSS. A continuación se muestran algunos ejemplos habituales de combinación de selectores.

```
| .aviso .especial { ... }
```

El anterior selector solamente selecciona aquellos elementos con un `class="especial"` que se encuentren dentro de cualquier elemento con un `class="aviso"`.

Si se modifica el anterior selector:

```
| div.aviso span.especial { ... }
```

Ahora, el selector solamente selecciona aquellos elementos de tipo `` con un atributo `class="especial"` que estén dentro de cualquier elemento de tipo `<div>` que tenga un atributo `class="aviso"`.

La combinación de selectores puede llegar a ser todo lo compleja que sea necesario:

```
| ul#menuPrincipal li.destacado a#inicio { ... }
```

El anterior selector hace referencia al enlace con un atributo `id` igual a `inicio` que se encuentra dentro de un elemento de tipo `` con un atributo `class` igual a `destacado`, que forma parte de una lista `` con un atributo `id` igual a `menuPrincipal`.

Ejercicio 1 Ver enunciado en la página 189

2.2. Selectores avanzados

Utilizando solamente los selectores básicos de la sección anterior, es posible diseñar prácticamente cualquier página web. No obstante, CSS define otros selectores más avanzados que permiten simplificar las hojas de estilos.

Desafortunadamente, el navegador Internet Explorer 6 y sus versiones anteriores no soportaban este tipo de selectores avanzados, por lo que su uso no era común hasta hace poco tiempo. Si quieras consultar el soporte de los selectores en los distintos navegadores, puedes utilizar las siguientes referencias:

- Lista completa de los selectores que soporta cada versión de cada navegador (<http://dev.l-c-n.com/CSS3-selectors/browser-support.php>)
- Test online en el que puedes comprobar los selectores que soporta el navegador con el que realizas el test (<http://www.css3.info/selectors-test/>)

2.2.1. Selector de hijos

Se trata de un selector similar al selector descendente, pero muy diferente en su funcionamiento. Se utiliza para seleccionar un elemento que es *hijo directo* de otro elemento y se indica mediante el "signo de mayor que" (>):

```
p > span { color: blue; }

<p><span>Texto1</span></p>
<p><a href="#"><span>Texto2</span></a></p>
```

En el ejemplo anterior, el selector p > span se interpreta como "*cualquier elemento que sea hijo directo de un elemento <p>*", por lo que el primer elemento cumple la condición del selector. Sin embargo, el segundo elemento no la cumple porque es descendiente pero no es hijo directo de un elemento <p>.

El siguiente ejemplo muestra las diferencias entre el selector descendente y el selector de hijos:

```
p a { color: red; }
p > a { color: red; }

<p><a href="#">Enlace1</a></p>
<p><span><a href="#">Enlace2</a></span></p>
```

El primer selector es de tipo descendente y por tanto se aplica a todos los elementos <a> que se encuentran dentro de elementos <p>. En este caso, los estilos de este selector se aplican a los dos enlaces.

Por otra parte, el selector de hijos obliga a que el elemento <a> sea hijo directo de un elemento <p>. Por lo tanto, los estilos del selector p > a no se aplican al segundo enlace del ejemplo anterior.

2.2.2. Selector adyacente

El selector adyacente utiliza el signo + y su sintaxis es:

```
| elemento1 + elemento2 { ... }
```

La explicación del comportamiento de este selector no es sencilla, ya que selecciona todos los elementos de tipo `elemento2` que cumplan las dos siguientes condiciones:

- `elemento1` y `elemento2` deben ser hermanos, por lo que su elemento padre debe ser el mismo.
- `elemento2` debe aparecer inmediatamente después de `elemento1` en el código HTML de la página.

En el siguiente ejemplo:

```
h1 + h2 { color: red }

<body>
<h1>Título1</h1>

<h2>Subtítulo</h2>
...
<h2>Otro subtítulo</h2>
...
</body>
```

Los estilos del selector `h1 + h2` se aplican al primer elemento `<h2>` de la página, pero no al segundo `<h2>`, ya que:

- El elemento padre de `<h1>` es `<body>`, el mismo padre que el de los dos elementos `<h2>`. Así, los dos elementos `<h2>` cumplen la primera condición del selector adyacente.
- El primer elemento `<h2>` aparece en el código HTML justo después del elemento `<h1>`, por lo que este elemento `<h2>` también cumple la segunda condición del selector adyacente.
- Por el contrario, el segundo elemento `<h2>` no aparece justo después del elemento `<h1>`, por lo que no cumple la segunda condición del selector adyacente y por tanto no se le aplican los estilos de `h1 + h2`.

El siguiente ejemplo es muy útil para los textos que se muestran como libros:

```
| p + p { text-indent: 1.5em; }
```

En muchos libros, suele ser habitual que la primera línea de todos los párrafos esté indentada, salvo la primera línea del primer párrafo. Con el selector `p + p`, se seleccionan todos los párrafos que estén dentro del mismo elemento padre que otros párrafos y que vayan justo después de otro párrafo. En otras palabras, el selector `p + p` selecciona todos los párrafos de un elemento salvo el primer párrafo.

2.2.3. Selector de atributos

El último tipo de selectores avanzados lo forman los selectores de atributos, que permiten seleccionar elementos HTML en función de sus atributos y/o valores de esos atributos.

Los cuatro tipos de selectores de atributos son:

- [nombre_atributo], selecciona los elementos que tienen establecido el atributo llamado nombre_atributo, independientemente de su valor.
- [nombre_atributo=valor], selecciona los elementos que tienen establecido un atributo llamado nombre_atributo con un valor igual a valor.
- [nombre_atributo~=valor], selecciona los elementos que tienen establecido un atributo llamado nombre_atributo y al menos uno de los valores del atributo es valor.
- [nombre_atributo|=valor], selecciona los elementos que tienen establecido un atributo llamado nombre_atributo y cuyo valor es una serie de palabras separadas con guiones, pero que comienza con valor. Este tipo de selector sólo es útil para los atributos de tipo lang que indican el idioma del contenido del elemento.

A continuación se muestran algunos ejemplos de estos tipos de selectores:

```

/* Se muestran de color azul todos los enlaces que tengan
   un atributo "class", independientemente de su valor */
a[class] { color: blue; }

/* Se muestran de color azul todos los enlaces que tengan
   un atributo "class" con el valor "externo" */
a[class="externo"] { color: blue; }

/* Se muestran de color azul todos los enlaces que apunten
   al sitio "http://www.ejemplo.com" */
a[href="http://www.ejemplo.com"] { color: blue; }

/* Se muestran de color azul todos los enlaces que tengan
   un atributo "class" en el que al menos uno de sus valores
   sea "externo" */
a[class~="externo"] { color: blue; }

/* Selecciona todos los elementos de la página cuyo atributo
   "Lang" sea igual a "en", es decir, todos los elementos en inglés */
*[lang=en] { ... }

/* Selecciona todos los elementos de la página cuyo atributo
   "Lang" empiece por "es", es decir, "es", "es-ES", "es-AR", etc. */
*[lang|= "es"] { color : red }

```

2.3. Agrupación de reglas

Cuando se crean archivos CSS complejos con decenas o cientos de reglas, es habitual que los estilos que se aplican a un mismo selector se definan en diferentes reglas:

```
| h1 { color: red; }  
| ...  
| h1 { font-size: 2em; }  
| ...  
| h1 { font-family: Verdana; }
```

Las tres reglas anteriores establecen el valor de tres propiedades diferentes de los elementos `<h1>`. Antes de que el navegador muestre la página, procesa todas las reglas CSS de la página para tener en cuenta todos los estilos definidos para cada elemento.

Cuando el selector de dos o más reglas CSS es idéntico, se pueden agrupar las declaraciones de las reglas para hacer las hojas de estilos más eficientes:

```
| h1 {  
|   color: red;  
|   font-size: 2em;  
|   font-family: Verdana;  
| }
```

El ejemplo anterior tiene el mismo efecto que las tres reglas anteriores, pero es más eficiente y es más fácil de modificar y mantener por parte de los diseñadores. Como CSS ignora los espacios en blanco y las nuevas líneas, también se pueden agrupar las reglas de la siguiente forma:

```
| h1 { color: red; font-size: 2em; font-family: Verdana; }
```

Si se quiere reducir al máximo el tamaño del archivo CSS para mejorar ligeramente el tiempo de carga de la página web, también es posible indicar la regla anterior de la siguiente forma:

```
| h1 {color:red;font-size:2em;font-family:Verdana;}
```

2.4. Herencia

Uno de los conceptos más característicos de CSS es la herencia de los estilos definidos para los elementos. Cuando se establece el valor de alguna propiedad en un elemento, todos sus descendientes heredan inicialmente ese mismo valor.

Si se indica por ejemplo un tipo de letra al elemento `<body>` de una página, todos los elementos de la página mostrarán ese tipo de letra, salvo que se indique lo contrario:

```
<!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">  
<head>  
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />  
<title>Ejemplo de herencia de estilos</title>  
<style type="text/css">  
  body { font-family: Arial; color: black; }  
  h1 { font-family: Verdana; }  
  p { color: red; }  
</style>  
</head>  
  
<body>  
<h1>Titular de la página</h1>
```

```
<p>Un párrafo de texto no muy largo.</p>
</body>
</html>
```

En el ejemplo anterior, se ha indicado que la etiqueta `<body>` tiene asignado un tipo de letra `Arial` y un color de letra negro. Así, todos los elementos de la página (salvo que se indique lo contrario) se muestran de color negro y con la fuente `Arial`.

La segunda regla indica que los elementos `<h1>` se muestran con otra tipografía diferente a la heredada. La tercera regla indica que los elementos `<p>` varían su color respecto del color que han heredado.

La herencia de estilos no funciona en todas las propiedades CSS, por lo que se debe estudiar cada propiedad de forma individual.

2.5. Colisiones de estilos

En las hojas de estilos complejas, es habitual que varias reglas CSS se apliquen a un mismo elemento HTML. El problema de estas reglas múltiples es que se pueden dar colisiones como la del siguiente ejemplo:

```
p { color: red; }
p { color: blue; }

<p>...</p>
```

¿De qué color se muestra el párrafo anterior? CSS tiene un mecanismo de resolución de colisiones muy complejo y que tiene en cuenta el tipo de hoja de estilo que se trate (de navegador, de usuario o de diseñador), la importancia de cada regla y lo específico que sea el selector.

Aunque los tipos de hojas de estilos y su importancia se verán más adelante, se describe a continuación el método genérico seguido por CSS para resolver las colisiones:

1. Determinar todas las declaraciones que se aplican al elemento para el medio CSS seleccionado.
2. Ordenar las declaraciones según su origen (CSS de navegador, de usuario o de diseñador) y su importancia (palabra clave `!important`).
3. Ordenar las declaraciones según lo específico que sea el selector. Cuanto más genérico es un selector, menos importancia tienen sus declaraciones.
4. Si después de aplicar las normas anteriores existen dos o más reglas con la misma prioridad, se aplica la que se indicó en último lugar.

Cuando se estudie cada uno de los conceptos del método anterior, se comprenderá completamente su funcionamiento. De momento, la norma que se puede seguir es la de la "*especificidad*" del selector:

1. Cuanto más específico sea un selector, más importancia tiene su regla asociada.
2. A igual *especificidad*, se considera la última regla indicada.

Como en el ejemplo anterior los dos selectores son idénticos, las dos reglas tienen la misma prioridad y prevalece la que se indicó en último lugar, por lo que el párrafo se muestra de color azul.

En el siguiente ejemplo, la regla CSS que prevalece se decide por lo específico que es cada selector:

```
p { color: red; }
p#especial { color: green; }
* { color: blue; }

<p id="especial">...</p>
```

Al elemento `<p>` se le aplican las tres declaraciones. Como su origen y su importancia es la misma, decide la especificidad del selector. El selector `*` es el menos específico, ya que se refiere a "*todos los elementos de la página*". El selector `p` es poco específico porque se refiere a "*todos los párrafos de la página*". Por último, el selector `p#especial` sólo hace referencia a "*el párrafo de la página cuyo atributo id sea igual a especial*". Como el selector `p#especial` es el más específico, su declaración es la que se tiene en cuenta y por tanto el párrafo se muestra de color verde.

Ejercicio 2 Ver enunciado en la página 190

Capítulo 3. Unidades de medida y colores

Muchas de las propiedades de CSS que se ven en los próximos capítulos permiten indicar unidades de medida y colores en sus valores. Además, CSS es tan flexible que permite indicar las medidas y colores de muchas formas diferentes. Por este motivo, se presentan a continuación todas las alternativas disponibles en CSS para indicar las medidas y los colores. En los siguientes capítulos, simplemente se indicará que el valor de una propiedad puede tomar el valor de una medida o de un color, sin detallar las diferentes alternativas disponibles para cada valor.

3.1. Unidades de medida

Las medidas en CSS se emplean, entre otras, para definir la altura, anchura y márgenes de los elementos y para establecer el tamaño de letra del texto. Todas las medidas se indican como un valor numérico entero o decimal seguido de una unidad de medida (sin ningún espacio en blanco entre el número y la unidad de medida).

CSS divide todas las unidades de medida en dos grupos: absolutas y relativas. Las medidas relativas definen su valor en relación con otra medida, por lo que para obtener su valor real, se debe realizar alguna operación con el valor indicado. Las unidades absolutas establecen de forma completa el valor de una medida, por lo que su valor real es directamente el valor indicado.

Si el valor es `0`, la unidad de medida es opcional. Si el valor es distinto a `0` y no se indica ninguna unidad, la medida se ignora completamente, lo que suele ser una fuente habitual de errores para los diseñadores que empiezan con CSS. Algunas propiedades permiten indicar medidas negativas, aunque habitualmente sus valores son positivos.

3.1.1. Unidades relativas

Las unidades relativas son más flexibles que las unidades absolutas porque se adaptan más fácilmente a los diferentes medios. A continuación se muestra la lista de unidades de medida relativas y la referencia que se toma para determinar su valor real:

- `em`, relativa respecto del tamaño de letra empleado. Aunque no es una definición exacta, el valor de `1 em` se puede aproximar por la anchura de la letra `M` (*"eme mayúscula"*) del tipo de letra que se esté utilizando
- `ex`, relativa respecto de la altura de la letra `x` (*"equis minúscula"*) del tipo de letra que se esté utilizando
- `px`, (píxel) relativa respecto de la pantalla del usuario

Las unidades `em` y `ex` no han sido definidas por CSS, sino que llevan décadas utilizándose en el campo de la tipografía. La unidad `em` hace referencia al tamaño en puntos de la letra que se está utilizando. Si se utiliza una tipografía de 12 puntos, `1 em` equivale a 12 puntos. El valor de `1 ex` se puede aproximar por `0.5 em`.

En el siguiente ejemplo, se indica que el tamaño de letra del texto de la página debe ser el 90% del tamaño por defecto (que depende de cada navegador, aunque es muy similar entre ellos):

```
| body { font-size: 0.9em; }
```

Como `em` es una unidad relativa, el valor `0.9` indicado sólo tiene sentido cuando se tiene en consideración su referencia. Para la unidad `em`, la referencia es el tamaño de letra por defecto del sistema (ordenador, dispositivo móvil, etc.) del usuario.

Por lo tanto, `0.9em` significa que se debe multiplicar `0.9` por el tamaño de letra por defecto, lo que en la práctica significa que la medida indicada es igual al 90% del tamaño de letra por defecto. Si este tamaño por defecto es 12, el valor `0.9em` sería igual a $0.9 \times 12 = 10.8$.

Cuando el valor decimal de una medida es inferior a 1, se puede omitir el `0` de la izquierda, por lo que el código anterior es equivalente al código siguiente:

```
| body { font-size: .9em; }
```

El siguiente ejemplo muestra el uso de la unidad `em` para establecer el tamaño de la letra de diferentes párrafos:

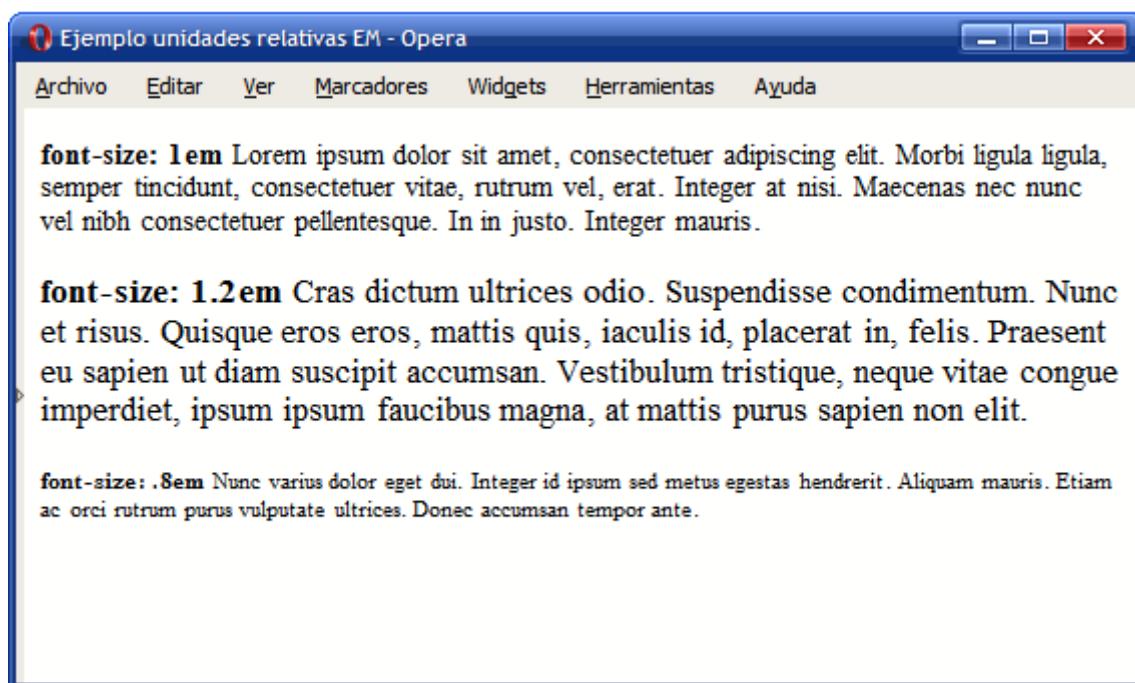


Figura 3.1. Ejemplo de tamaño de letra definido con la unidad relativa `em`

El primer párrafo muestra la letra con un tamaño de `1em`, es decir, el tamaño por defecto en el navegador del usuario. El segundo párrafo ha establecido el tamaño de letra en `1.2em`, es decir, un 20% más grande que el tamaño por defecto. Por último, el tercer párrafo ha indicado un tamaño de `.8em`, es decir, un 20% inferior al tamaño por defecto.

Cuando se estudian por primera vez, las unidades relativas parecen demasiado complicadas. Al fin y al cabo, siempre se debe tomar la referencia de la unidad para obtener su valor real. Sin embargo, sus ventajas son mucho mayores que sus inconvenientes.

El ejemplo anterior establece el tamaño de la letra mediante los valores `1em`, `1.2em` y `.8em`. En otras palabras, el código anterior está estableciendo los tamaños de letra a "normal", "grande" y "pequeño" respectivamente. Independientemente del tamaño de letra por defecto del dispositivo del usuario, el primer párrafo se verá con un tamaño de letra "normal" (`1em`), el segundo párrafo se verá más "grande" de lo normal (`1.2em`) y el último párrafo se verá "pequeño" (`.8em`).

De esta forma, si el usuario tiene problemas de visión y aumenta el tamaño de letra en su navegador, las proporciones se mantendrán. Si el tamaño de letra por defecto es `12`, el primer párrafo se verá con tamaño `12`, pero si el usuario aumenta el tamaño de letra por defecto a `20`, el primer párrafo se verá con tamaño `20`. Como se ve, las unidades relativas permiten mantener las proporciones del diseño independientemente del tamaño de letra por defecto del navegador del usuario.

Como se verá más adelante, la propiedad `font-size` permite establecer el tamaño de letra del texto de un elemento. En este caso, la referencia para el valor de `font-size` de un elemento siempre es el tamaño de letra de su elemento padre (es decir, del elemento en el que se encuentra). Si el elemento no se encuentra dentro de ningún otro elemento, la referencia es el tamaño de letra del elemento `<body>`. Si no se indica de forma explícita un valor para el tamaño de letra del elemento `<body>`, la referencia es el tamaño de letra por defecto del navegador.

Siguiendo esta norma, si en el ejemplo anterior se modifica el tamaño de letra del elemento `<body>` (que es el elemento padre de los tres párrafos) y se le asigna un valor de `0.8em`, el aspecto que muestran los párrafos en el navegador es el siguiente:

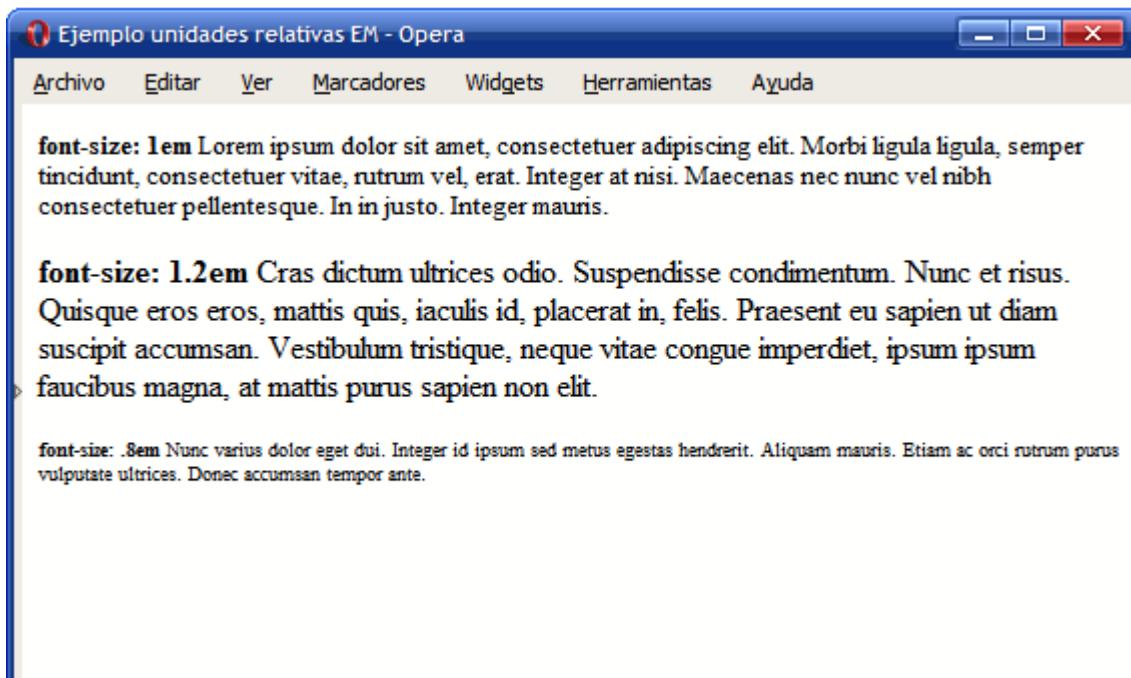


Figura 3.2. Ejemplo de tamaño de letra definido con la unidad relativa em

Al haber reducido el tamaño de letra que era la referencia del tamaño de letra de los tres párrafos, su texto se ve con una letra más pequeña, aunque manteniendo las proporciones: el

primer párrafo se ve con un tamaño de letra normal, el segundo se ve con un tamaño grande y el tercero se visualiza con un tamaño de letra más pequeño de lo normal.

El funcionamiento de la unidad `ex` es idéntico a `em`, salvo que en este caso, la referencia es la altura de la letra x minúscula.

Aunque puede resultar paradójico, las medidas indicadas en píxel también se consideran relativas, ya que el aspecto de los elementos dependerá de la resolución del dispositivo en el que se visualiza el documento HTML. Cuando se visualiza un documento en un dispositivo de alta resolución (por ejemplo una impresora de 1200 dpi) se reescalan los píxel del documento y cada uno de los píxel originales se visualizan como un conjunto de píxel del dispositivo de alta resolución.

Las distintas unidades se pueden mezclar entre los diferentes elementos de una misma página, como en el siguiente ejemplo:

```
body { font-size: 10px; }
h1 { font-size: 2.5em; }
```

En primer lugar, se establece un tamaño de letra base de 10 píxel para toda la página. A continuación, se asigna un tamaño de 2.5em al elemento `<h1>`, por lo que su tamaño de letra real será de $2.5 \times 10\text{px} = 25\text{px}$.

Como se vio en los capítulos anteriores, muchas propiedades CSS se heredan desde los elementos padre hasta los hijos. Así por ejemplo, si se establece el tamaño de letra al elemento `<body>`, todos los elementos de la página tendrán el mismo tamaño de letra, salvo que indiquen otro valor.

Sin embargo, las medidas relativas no se heredan directamente, sino que se heredan sus valores reales una vez calculados. El siguiente ejemplo muestra este comportamiento:

```
body {
  font-size: 12px;
  text-indent: 3em;
}
h1 { font-size: 15px }
```

La propiedad `text-indent`, como se verá en los próximos capítulos, se utiliza para tabular la primera línea de un texto. El elemento `<body>` define un valor para esta propiedad, pero el elemento `<h1>` no lo hace, por lo que heredará el valor de su elemento padre. Sin embargo, el valor heredado no es 3em, sino 36px.

Si se heredara el valor 3em, al multiplicarlo por el valor de `font-size` del elemento `<h1>` (que vale 15px) el resultado sería $3\text{em} \times 15\text{px} = 45\text{px}$. No obstante, como se ha comentado, los valores que se heredan no son los relativos, sino los valores ya calculados.

Por lo tanto, en primer lugar se calcula el valor real de 3em para el elemento `<body>`: $3\text{em} \times 12\text{px} = 36\text{px}$. Una vez calculado el valor real, este es el valor que se hereda para el resto de elementos.

3.1.2. Unidades absolutas

Las unidades absolutas definen las medidas de forma completa, ya que sus valores reales no se calculan a partir de otro valor de referencia, sino que son directamente los valores indicados. A continuación se muestra la lista completa de unidades absolutas definidas por CSS y su significado:

- **in**, del inglés "*inches*", pulgadas (1 pulgada son 2.54 centímetros)
- **cm**, centímetros
- **mm**, milímetros
- **pt**, puntos (1 punto equivale a 1 pulgada/72, es decir, unos 0.35 milímetros)
- **pc**, picas (1 pica equivale a 12 puntos, es decir, unos 4.23 milímetros)

A continuación se muestran ejemplos de utilización de unidades absolutas:

```
body { margin: 0.5in; }
h1 { line-height: 2cm; }
p { word-spacing: 4mm; }
a { font-size: 12pt; }
span { font-size: 1pc; }
```

Su uso es idéntico al de las unidades relativas, siendo su única diferencia que los valores indicados son directamente los valores que se utilizan, sin necesidad de calcular los valores reales en función de otras referencias.

De todas las unidades absolutas, la única que se utiliza con cierta frecuencia es la de los puntos (pt). El motivo es que se trata de la unidad preferida para indicar el tamaño de letra del texto para los documentos que se van a imprimir, es decir, para el medio `print` de CSS (como se verá más adelante).

3.1.3. Porcentajes

CSS define otra unidad de medida relativa basada en los porcentajes. Un porcentaje está formado por un valor numérico seguido del símbolo % y siempre está referenciado a otra medida. Cada una de las propiedades de CSS que permiten indicar como valor un porcentaje, define el valor al que hace referencia ese porcentaje.

Los porcentajes se pueden utilizar por ejemplo para establecer el valor del tamaño de letra de los elementos:

```
body { font-size: 1em; }
h1 { font-size: 200%; }
h2 { font-size: 150%; }
```

Los tamaños establecidos para los elementos `<h1>` y `<h2>` mediante las reglas anteriores, son equivalentes a `2em` y `1.5em` respectivamente, por lo que es más habitual definirlos mediante `em`.

Los porcentajes también se utilizan para establecer la anchura de los elementos:

```
div#contenido { width: 600px; }
div.principal { width: 80%; }

<div id="contenido">
  <div class="principal">
    ...
  </div>
</div>
```

En el ejemplo anterior, la referencia del valor 80% es la anchura de su elemento padre. Por tanto, el elemento `<div>` cuyo atributo `class` vale `principal` tiene una anchura de $80\% \times 600\text{px} = 480\text{px}$.

3.1.4. Recomendaciones

En general, se recomienda el uso de unidades relativas siempre que sea posible, ya que mejora la accesibilidad de la página y permite que los documentos se adapten fácilmente a cualquier medio y dispositivo.

El documento "*Recomendaciones sobre técnicas CSS para la mejora de la accesibilidad de los contenidos HTML*" (<http://www.w3.org/TR/WCAG10-CSS-TECHS/>) elaborado por el organismo W3C, recomienda el uso de la unidad `em` para indicar el tamaño del texto y para todas las medidas que sean posibles.

Normalmente se utilizan píxel y porcentajes para definir el layout del documento (básicamente, la anchura de las columnas y elementos de las páginas) y `em` y porcentajes para el tamaño de letra de los textos.

Por otra parte, uno de los problemas habituales cuando se utilizan unidades relativas es el problema de "*el texto cada vez se ve más pequeño*" o "*el texto cada vez se ve más grande*". El siguiente ejemplo muestra el primer caso:

```
div { font-size: 0.9em; }

<div>
  <p>Texto 1</p>
  <div>
    <p>Texto 2</p>
    <div>
      <p>Texto 3</p>
    </div>
  </div>
</div>
```

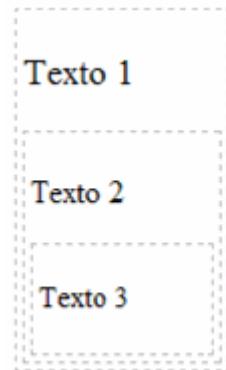


Figura 3.3. El texto cada vez se ve más pequeño

En el ejemplo anterior, el tamaño del texto de todos los elementos `<div>` se define mediante la medida relativa `0.9em`. Como se trata de una medida relativa, su valor real se calcula a partir del tamaño de letra de su elemento padre. De esta forma, el tamaño de letra del primer `<div>` es igual a `0.9em` respecto del tamaño de letra por defecto.

En el segundo elemento `<div>`, el tamaño de letra es `0.9em` respecto al tamaño de letra del primer `<div>`, es decir, $0.9em \times 0.9em = 0.81em$ respecto del tamaño de letra por defecto, por lo que su letra se ve más pequeña que la del primer `<div>`.

Por último, el tamaño de letra del tercer `<div>` será igual a `0.9em` respecto al tamaño de la letra del segundo elemento `<div>`, es decir, $0.9em \times 0.9em \times 0.9em = 0.729em$ respecto del tamaño de letra por defecto. De esta forma, el tamaño de letra de este tercer `<div>` es mucho más pequeño que el del primer `<div>`. Si se anidan varios elementos `<div>`, la letra se hará tan pequeña que no será posible leerla.

En el caso de que se indique un valor mayor que 1 para la medida relativa, el comportamiento es muy similar al descrito anteriormente, salvo que en este caso el tamaño de letra cada vez es mayor.

3.2. Colores

Los colores en CSS se pueden indicar de cinco formas diferentes: palabras clave, colores del sistema, RGB hexadecimal, RGB numérico y RGB porcentual. Aunque el método más habitual es el del RGB hexadecimal, a continuación se muestran todas las alternativas que ofrece CSS.

3.2.1. Palabras clave

CSS define 17 palabras clave para referirse a los colores básicos. Las palabras se corresponden con el nombre en inglés de cada color:

`aqua, black, blue, fuchsia, gray, green, lime, maroon, navy, olive, orange, purple, red, silver, teal, white, yellow`



Figura 3.4. Colores definidos mediante las palabras clave de CSS

La imagen anterior ha sido extraída de la sección sobre colores de la especificación oficial de CSS (<http://www.w3.org/TR/CSS21/syndata.html#value-def-color>) .

Aunque es una forma muy sencilla de referirse a los colores básicos, este método prácticamente no se utiliza en las hojas de estilos de los sitios web reales, ya que se trata de una gama de colores muy limitada.

Además de la lista básica, los navegadores modernos soportan muchos otros nombres de colores. La lista completa se puede ver en <http://en.wikipedia.org/wiki/Websafe>.

3.2.2. RGB decimal

En el campo del diseño gráfico, se han definido varios modelos diferentes para referirse a los colores. Los dos modelos más conocidos son RGB y CMYK. Aunque no es una definición exacta, el modelo RGB consiste en definir un color indicando que cantidad de color rojo, verde y azul se debe *mezclar* para obtener el color. Técnicamente, el modelo RGB es un modelo de tipo "aditivo", ya que se suman colores para obtener el color deseado.

Por lo tanto, en el modelo RGB un color se define indicando sus tres componentes R (rojo), G (verde) y B (azul). Cada una de las componentes puede tomar un valor entre cero y un valor máximo. De esta forma, el color rojo puro en RGB se crea mediante el máximo valor de la componente R y un valor de 0 para las componentes G y B.

Si todas las componentes valen 0, el color creado es el negro; si todas las componentes toman su valor máximo, el color obtenido es el blanco. En CSS, las componentes de los colores definidos mediante RGB decimal pueden tomar valores entre 0 y 255. El siguiente ejemplo establece el color del texto de un párrafo:

```
| p { color: rgb(71, 98, 176); }
```

La sintaxis que se utiliza para indicar los colores es `rgb()` y entre paréntesis se indican las tres componentes RGB, en ese mismo orden y separadas por comas. El color del ejemplo anterior se obtendría mezclando las componentes R=71, G=98, B=176, que se corresponde con un color azul claro.

Si se indica un valor menor que 0 para una componente, automáticamente se transforma su valor en 0. Igualmente, si se indica un valor mayor que 255, se transforma automáticamente su valor a 255.

3.2.3. RGB porcentual

Otra forma de indicar las componentes RGB de un color es mediante un porcentaje. El funcionamiento y la sintaxis de este método es el mismo que el del RGB decimal. La única diferencia en este caso es que el valor de las componentes RGB puede tomar valores entre 0% y 100%. El mismo color del ejemplo anterior se puede representar de forma porcentual:

```
| p { color: rgb(27%, 38%, 69%); }
```

Al igual que sucede con el RGB decimal, si se indica un valor inferior a 0%, se transforma automáticamente en 0% y si se indica un valor superior a 100%, se trunca su valor a 100%.

3.2.4. RGB hexadecimal

Aunque es el método más complicado de indicar los colores, se trata del método más utilizado con mucha diferencia. De hecho, prácticamente todos los sitios web reales utilizan exclusivamente este método.

En este método, se hace uso del sistema numérico hexadecimal. En el sistema decimal, se utilizan 10 símbolos para representar los números: del 0 al 9. En el sistema hexadecimal se utilizan 16 símbolos (de ahí su nombre): del 0 al 9 y de la A a la F. Así, en el sistema hexadecimal, después del 9 no va el 10, sino la A. La letra B sería 11, la C sería 12, etc.

Para definir un color con RGB hexadecimal se siguen los siguientes pasos:

1. Se toman las componentes RGB del color original, por ejemplo R = 71, G = 98, B = 176
2. El valor numérico de cada componente se transforma al sistema numérico hexadecimal.
Esta operación es exclusivamente matemática. En este ejemplo, las nuevas componentes son: R = 47, G = 62, B = B0.
3. Para obtener el color completo, se concatenan los valores de las componentes RGB y se añade el prefijo #.

Con esta nueva notación, el color del mismo ejemplo anterior se indica de la siguiente forma:

```
| p { color: #4762B0; }
```

Recuerda que aunque es el método más complicado para definir un color, se trata del método que utilizan la inmensa mayoría de sitios web, por lo que es imprescindible dominarlo. Afortunadamente, todos los programas de diseño gráfico convierten de forma automática los valores RGB decimales a sus valores RGB hexadecimales:

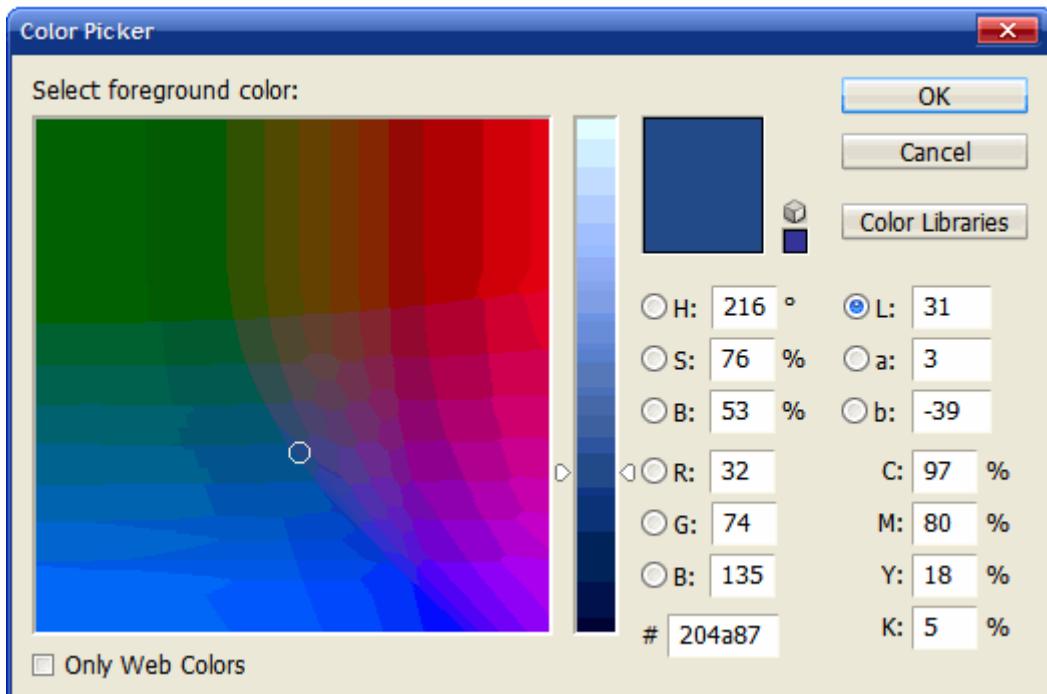


Figura 3.5. Herramienta de color de Photoshop para definir los colores según los modelos RGB, CMYK, Lab, HSB y RGB hexadecimal

Una de las ventajas del formato RGB hexadecimal es que se pueden comprimir sus valores cuando todas sus componentes son iguales dos a dos:

```
#AAA = #AAAAAA
#FFF = #FFFFFF
#A0F = #AA00FF
#369 = #336699
```

En el siguiente ejemplo se establece el color de fondo de la página a blanco, el color del texto a negro y el color de la letra de los titulares se define de color rojo:

```
body { background-color: #FFF; color: #000; }
h1, h2, h3, h4, h5, h6 { color: #C00; }
```

3.2.5. Colores del sistema

Los colores del sistema son similares a los colores indicados mediante su nombre, pero en este caso hacen referencia al color que muestran algunos elementos del sistema operativo del usuario.

Existen varios colores definidos, como por ejemplo ActiveBorder, que hace referencia al color del borde de las ventanas activas. La lista completa de colores definidos se puede ver en <http://www.w3.org/TR/CSS21/ui.html#system-colors>.

Aunque es posible definir los colores en CSS utilizando estos nombres, se trata de un método que nunca se utiliza, por lo que se puede considerar prácticamente como una rareza de CSS.

3.2.6. Colores web safe

Como cada componente RGB de los colores puede tomar un valor entre 0 y 255, el número total de colores que se pueden representar con este formato es de $256 \times 256 \times 256 = 16.777.216$ colores. Sin embargo, en la década de los 90 los monitores de los usuarios no eran capaces de mostrar más de 256 colores diferentes.

A partir de todos los colores disponibles, se eligieron 216 colores que formaron la paleta de colores "*web safe*". Esta paleta de colores podía ser utilizada por los diseñadores con la seguridad de que se verían correctamente en cualquier navegador de cualquier sistema operativo de cualquier usuario.

Hoy en día, su importancia ha descendido notablemente, ya que prácticamente todos los usuarios utilizan dispositivos con una profundidad de color de 16 y 32 bits. No obstante, el auge en el uso de los dispositivos móviles hace que siga siendo un tema a considerar, ya que las pantallas de muchos móviles sólo pueden representar un número reducido de colores.

La lista completa de colores web safe y sus valores hexadecimales se pueden consultar en http://en.wikipedia.org/wiki/Web_colors#Web-safe_colors.

Capítulo 4. Modelo de cajas (box model)

El modelo de cajas o "*box model*" es seguramente la característica más importante del lenguaje de hojas de estilos CSS, ya que condiciona el diseño de todas las páginas web. El "*box model*" es el comportamiento de CSS que hace que todos los elementos incluidos en una página HTML se representen mediante cajas rectangulares. CSS permite controlar el aspecto de todas las cajas.

El diseño de cualquier página XHTML está compuesto por cajas rectangulares. CSS permite definir la altura y anchura de cada caja, el margen existente entre cajas y el espacio de relleno interior que muestra cada caja. Además, CSS permite controlar la forma en la que se visualizan las cajas: se pueden ocultar, desplazar respecto de su posición original y fijarlas en una posición específica dentro del documento.

Como la mayoría de cajas de las páginas web no muestran ningún color de fondo ni ningún borde, no son visibles a simple vista. La siguiente imagen muestra las cajas que forman la página web de <http://www.alistapart.com/> después de forzar a que todas las cajas muestren un borde punteado:

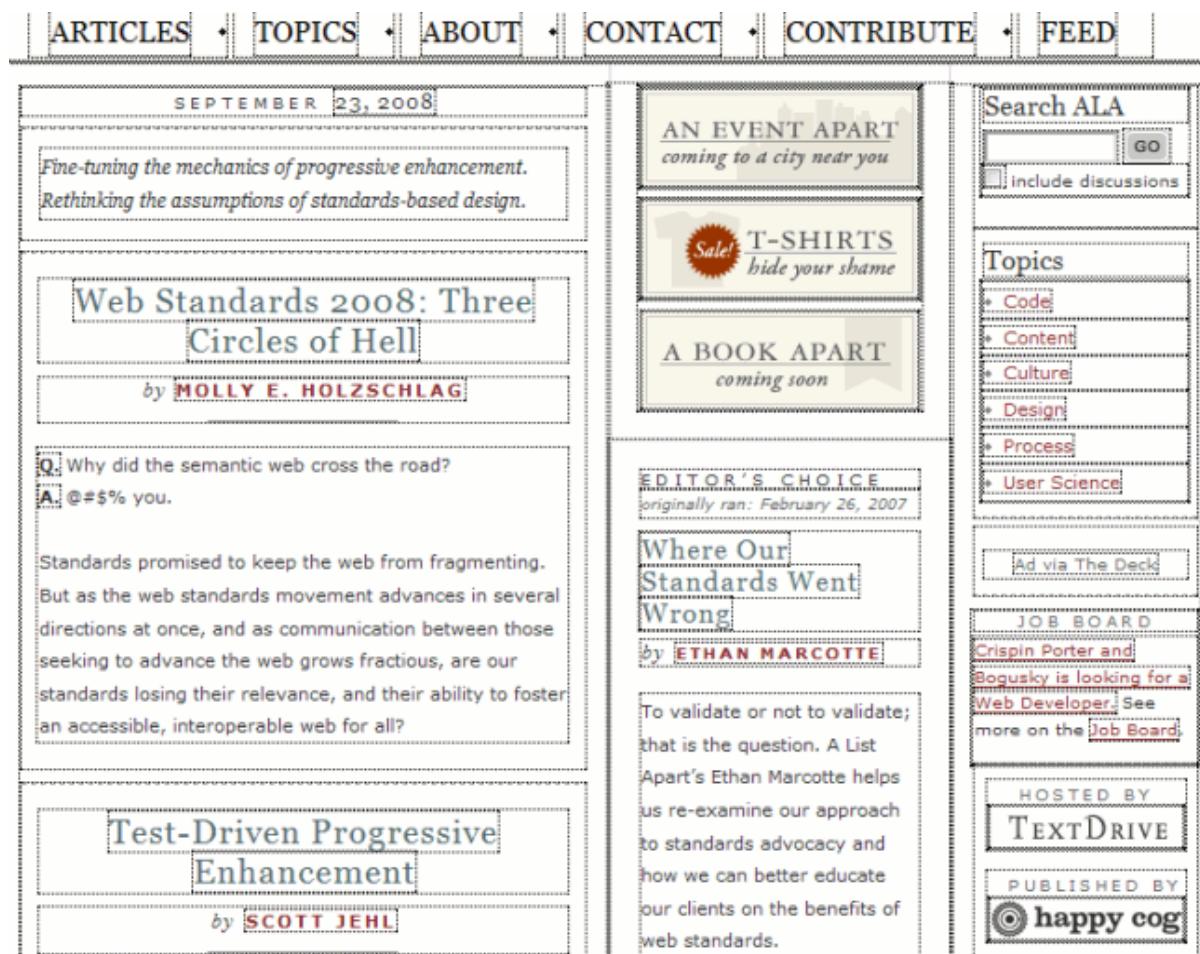


Figura 4.1. Cajas que forman el box model de la página [alistapart.com](http://www.alistapart.com/)

Las cajas de una página se crean automáticamente. Cada vez que se inserta una etiqueta o elemento en la página, se crea una nueva caja rectangular que encierra los contenidos del

elemento. El siguiente esquema muestra la creación automática de cajas por parte de HTML para cada elemento definido en el código HTML de la página:

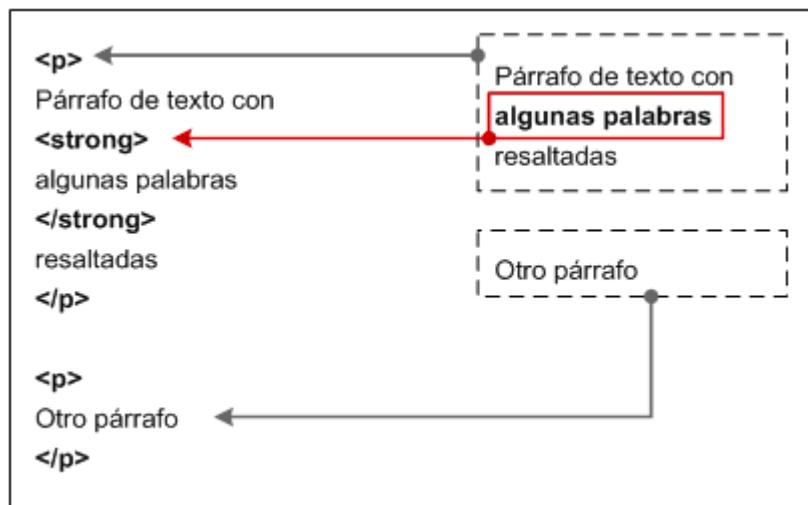


Figura 4.2. Las cajas se crean automáticamente al definir cada elemento HTML

Cada una de las cajas está formada por seis partes, tal y como se muestra en la siguiente imagen tridimensional:

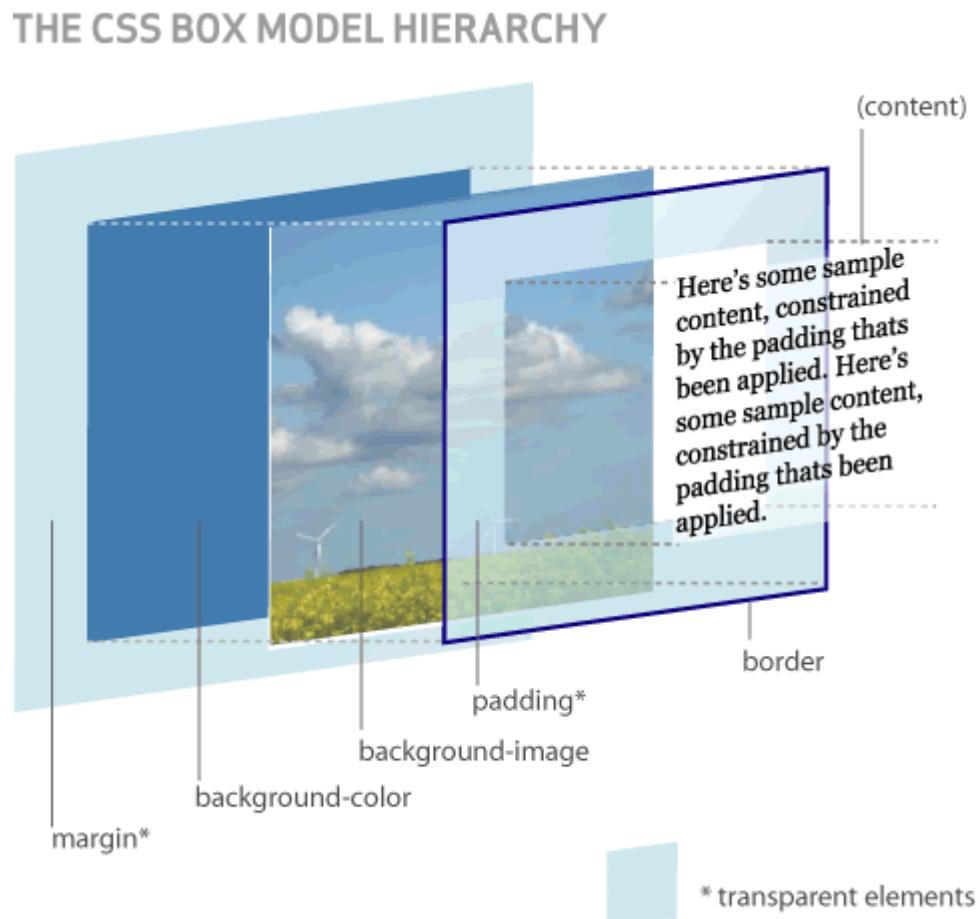


Figura 4.3. Representación tridimensional del box model de CSS

(Esquema utilizado con permiso de <http://www.hicksdesign.co.uk/boxmodel/>)

Las partes que componen cada caja y su orden de visualización desde el punto de vista del usuario son las siguientes:

- Contenido (*content*): se trata del contenido HTML del elemento (las palabras de un párrafo, una imagen, el texto de una lista de elementos, etc.)
- Relleno (*padding*): espacio libre opcional entre el contenido y el borde que lo encierra.
- Borde (*border*): línea que encierra completamente el contenido y su relleno.
- Imagen de fondo (*background image*): imagen que se muestra por detrás del contenido y el espacio de relleno.
- Color de fondo (*background color*): color que se muestra por detrás del contenido y el espacio de relleno.
- Margen (*margin*): espacio libre entre la caja y las posibles cajas adyacentes.

El relleno y el margen son transparentes, por lo que en el espacio ocupado por el relleno se muestra el color o imagen de fondo (si están definidos) y en el espacio ocupado por el margen se muestra el color o imagen de fondo de su elemento padre (si están definidos). Si ningún elemento padre tiene definido un color o imagen de fondo, se muestra el color o imagen de fondo de la propia página (si están definidos).

Si una caja define tanto un color como una imagen de fondo, la imagen tiene más prioridad y es la que se visualiza.

4.1. Anchura y altura

4.1.1. Anchura

La propiedad CSS que controla la anchura de los elementos se denomina `width`.

width	Anchura
Valores	<code><medida></code> <code><porcentaje></code> <code>auto</code> <code>inherit</code>
Se aplica a	Todos los elementos, salvo los elementos en línea que no sean imágenes, las filas de tabla y los grupos de filas de tabla
Valor inicial	<code>auto</code>
Descripción	Establece la anchura de un elemento

La propiedad `width` no admite valores negativos y los valores en porcentaje se calculan a partir de la anchura de su elemento padre. El valor `inherit` indica que la anchura del elemento se hereda de su elemento padre. El valor `auto`, que es el que se utiliza si no se establece de forma explícita un valor a esta propiedad, indica que el navegador debe calcular automáticamente la anchura del elemento, teniendo en cuenta sus contenidos y el sitio disponible en la página.

El siguiente ejemplo establece el valor de la anchura del elemento `<div>` lateral:

```
#lateral { width: 200px; }

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

CSS define otras dos propiedades relacionadas con la anchura de los elementos: `min-width` y `max-width`, que se verán más adelante.

4.1.2. Altura

La propiedad CSS que controla la altura de los elementos se denomina `height`.

height	Altura
Valores	<code><medida> <porcentaje> auto inherit</code>
Se aplica a	Todos los elementos, salvo los elementos en línea que no sean imágenes, las columnas de tabla y los grupos de columnas de tabla
Valor inicial	<code>auto</code>
Descripción	Establece la altura de un elemento

Al igual que sucede con `width`, la propiedad `height` no admite valores negativos. Si se indica un porcentaje, se toma como referencia la altura del elemento padre. Si el elemento padre no tiene una altura definida explícitamente, se asigna el valor `auto` a la altura.

El valor `inherit` indica que la altura del elemento se hereda de su elemento padre. El valor `auto`, que es el que se utiliza si no se establece de forma explícita un valor a esta propiedad, indica que el navegador debe calcular automáticamente la altura del elemento, teniendo en cuenta sus contenidos y el sitio disponible en la página.

El siguiente ejemplo establece el valor de la altura del elemento `<div>` de cabecera:

```
#cabecera { height: 60px; }

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

CSS define otras dos propiedades relacionadas con la altura de los elementos: `min-height` y `max-height`, que se verán más adelante.

4.2. Margen y relleno

4.2.1. Margen

CSS define cuatro propiedades para controlar cada uno de los márgenes horizontales y verticales de un elemento.

margin-top	Margen superior
margin-right	Margen derecho
margin-bottom	Margen inferior
margin-left	Margen izquierdo
Valores	(<medida> <porcentaje> auto) inherit
Se aplica a	Todos los elementos, salvo margin-top y margin-bottom que sólo se aplican a los elementos de bloque y a las imágenes
Valor inicial	0
Descripción	Establece cada uno de los márgenes horizontales y verticales de un elemento

Cada una de las propiedades establece la separación entre el borde lateral de la caja y el resto de cajas adyacentes:

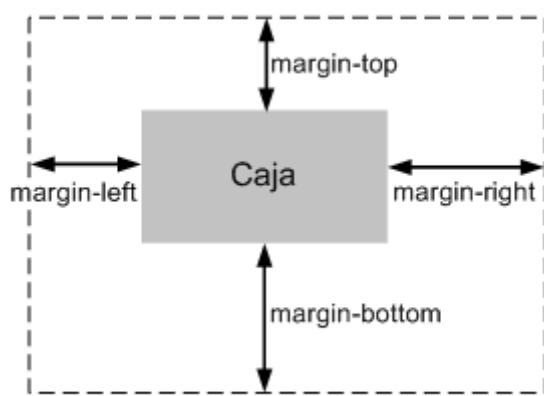


Figura 4.4. Las cuatro propiedades relacionadas con los márgenes

Las unidades más utilizadas para indicar los márgenes de un elemento son los píxeles (cuando se requiere una precisión total), los em (para hacer diseños que mantengan las proporciones) y los porcentajes (para hacer diseños líquidos o fluidos).

El siguiente ejemplo añade un margen izquierdo al segundo párrafo:

```
.destacado {
    margin-left: 2em;
}

<p>Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Nam et elit.
Vivamus placerat lorem. Maecenas sapien. Integer ut massa. Cras diam ipsum,
laoreet non, tincidunt a, viverra sed, tortor.</p>

<p class="destacado">Vestibulum lectus diam, luctus vel, venenatis ultrices,
cursus vel, tellus. Etiam placerat erat non sem. Nulla molestie odio non
nisl tincidunt faucibus.</p>

<p>Aliquam euismod sapien eu libero. Ut tempor orci at nulla. Nam in eros
egestas massa vehicula nonummy. Morbi posuere, nibh ultricies consectetuer tincidunt,
risus turpis laoreet elit, ut tincidunt risus sem et nunc.</p>
```

A continuación se muestra el aspecto del ejemplo anterior en cualquier navegador:

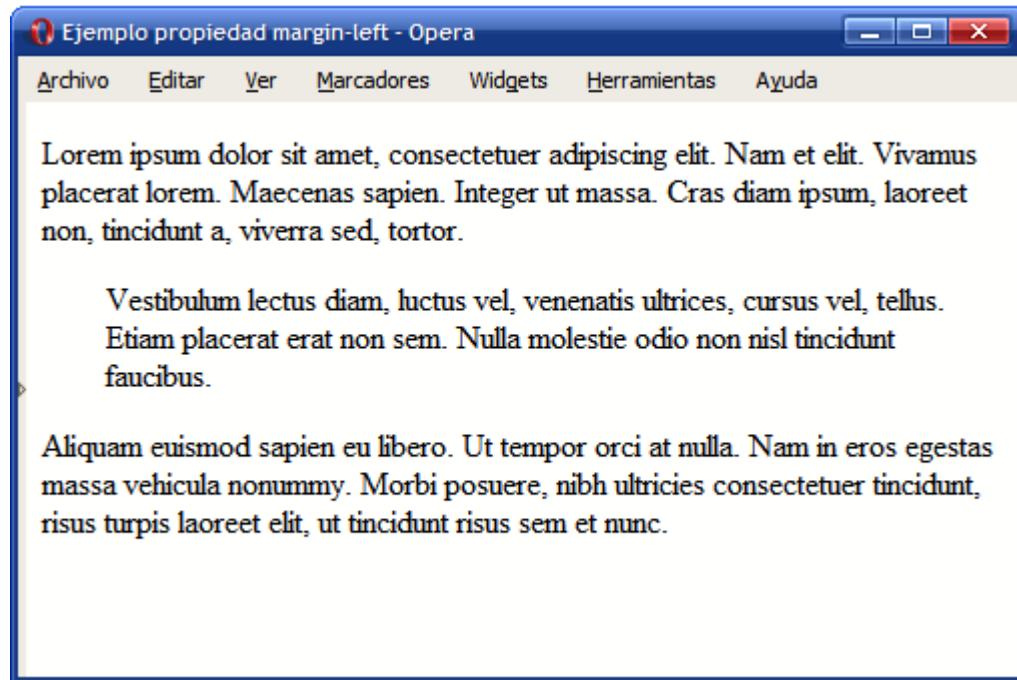


Figura 4.5. Ejemplo de propiedad margin-left

Algunos diseñadores web utilizan la etiqueta <blockquote> para encerrar los contenidos de un párrafo que se quiere mostrar tabulado respecto al resto de contenidos, como en el ejemplo anterior. Se trata de un error grave porque utiliza código XHTML erróneo para modificar el aspecto de los contenidos. Como ya se sabe, CSS es el único responsable del aspecto de los contenidos y dispone de propiedades como `margin-left` que permite conseguir los mismos resultados de forma correcta.

Los márgenes verticales (`margin-top` y `margin-bottom`) sólo se pueden aplicar a los elementos de bloque y las imágenes, mientras que los márgenes laterales (`margin-left` y `margin-right`) se pueden aplicar a cualquier elemento, tal y como muestra la siguiente imagen:

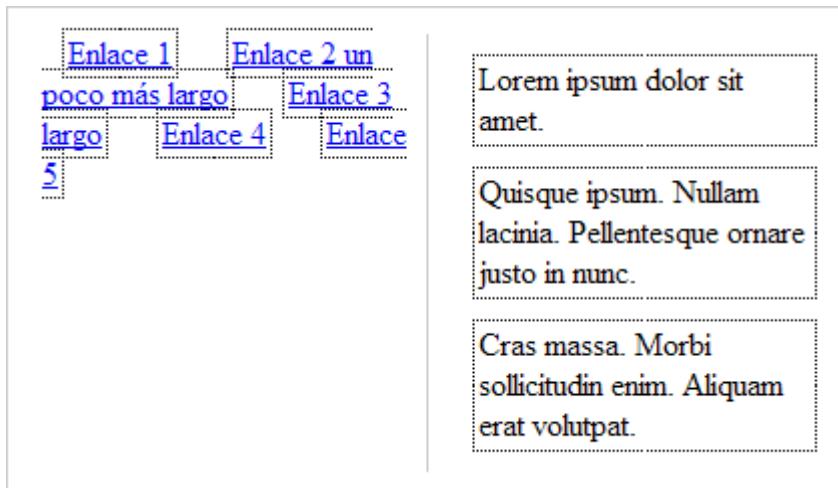


Figura 4.6. Los márgenes verticales sólo se aplican a los elementos de bloque e imágenes

La imagen anterior muestra el resultado de aplicar los mismos márgenes a varios enlaces (elementos en línea) y varios párrafos (elementos de bloque). En los elementos en línea los márgenes verticales no tienen ningún efecto, por lo que los enlaces no muestran ninguna separación vertical, al contrario de lo que sucede con los párrafos. Sin embargo, los márgenes laterales funcionan sobre cualquier tipo de elemento, por lo que los enlaces se muestran separados entre sí y los párrafos aumentan su separación con los bordes laterales de su elemento contenedor.

El siguiente ejemplo utiliza el mismo valor en los cuatro márgenes de cada imagen para facilitar su identificación y mejorar el diseño general de la página:

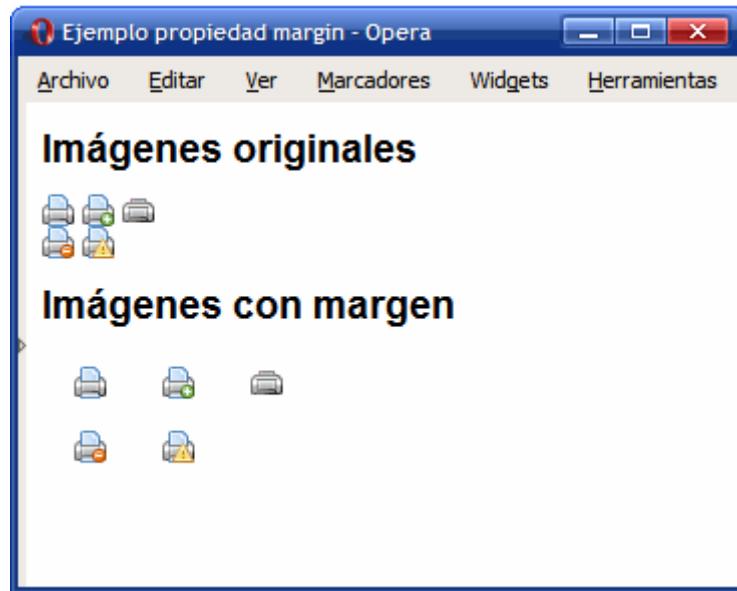


Figura 4.7. Ejemplo de propiedad margin

El código CSS del ejemplo anterior se muestra a continuación:

```
div img {  
    margin-top: .5em;  
    margin-bottom: .5em;  
    margin-left: 1em;  
    margin-right: .5em;  
}
```

Además de las cuatro propiedades que controlan cada uno de los márgenes del elemento, CSS define una propiedad que permite establecer los cuatro márgenes de forma directa empleando una única propiedad. Este tipo de propiedades resumidas se denominan propiedades de tipo "*shorthand*" y CSS define varias propiedades de este tipo, como se verá más adelante.

La propiedad que permite definir de forma simultanea los cuatro márgenes se denomina `margin`.

margin	Margen
Valores	(<medida> <porcentaje> auto) {1, 4} inherit
Se aplica a	Todos los elementos salvo algunos casos especiales de elementos mostrados como tablas
Valor inicial	-
Descripción	Establece de forma directa todos los márgenes de un elemento

La notación {1, 4} de la definición anterior significa que la propiedad `margin` admite entre uno y cuatro valores, con el siguiente significado:

- Si solo se indica un valor, todos los márgenes tienen ese valor.
- Si se indican dos valores, el primero se asigna al margen superior e inferior y el segundo se asigna a los márgenes izquierdo y derecho.
- Si se indican tres valores, el primero se asigna al margen superior, el tercero se asigna al margen inferior y el segundo valor se asigna los márgenes izquierdo y derecho.
- Si se indican los cuatro valores, el orden de asignación es: margen superior, margen derecho, margen inferior y margen izquierdo.

El ejemplo anterior de márgenes se puede reescribir utilizando la propiedad `margin`:

Código CSS original:

```
div img {
    margin-top: .5em;
    margin-bottom: .5em;
    margin-left: 1em;
    margin-right: .5em;
}
```

Alternativa directa:

```
div img {
    margin: .5em .5em .5em 1em;
}
```

Otra alternativa:

```
div img {
    margin: .5em;
    margin-left: 1em;
}
```

El comportamiento de los márgenes verticales es más complejo de lo que se puede imaginar. Cuando se juntan dos o más márgenes verticales, se fusionan de forma automática y la altura del nuevo margen será igual a la altura del margen más alto de los que se han fusionado.

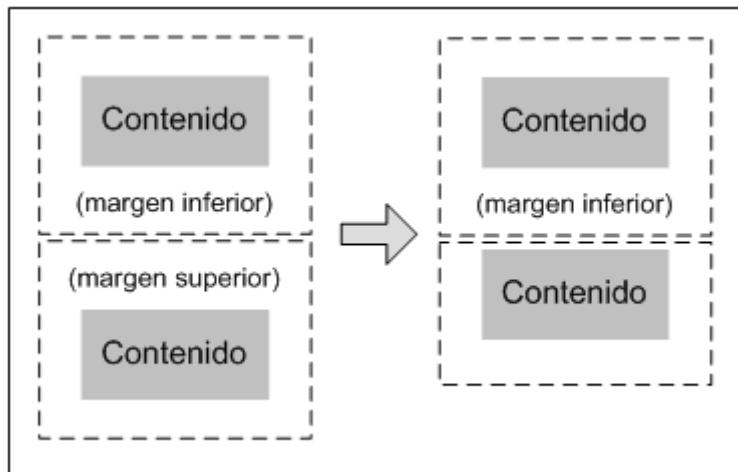


Figura 4.8. Fusión automática de los márgenes verticales

De la misma forma, si un elemento está contenido dentro de otro elemento, sus márgenes verticales se fusionan y resultan en un nuevo margen de la misma altura que el mayor margen de los que se han fusionado:

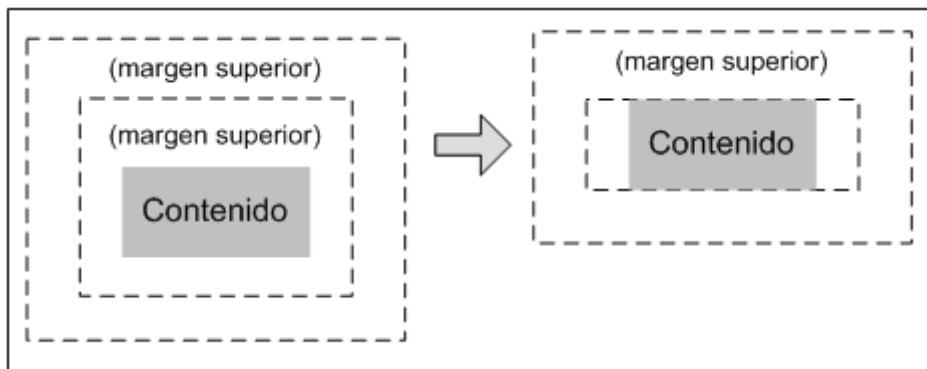


Figura 4.9. Fusión de los márgenes de los elementos interiores

Aunque en principio puede parecer un comportamiento extraño, la razón por la que se propuso este mecanismo de fusión automática de márgenes verticales es el de dar uniformidad a las páginas web habituales. En una página con varios párrafos, si no se diera este comportamiento y se estableciera un determinado margen a todos los párrafos, el primer párrafo no mostraría un aspecto homogéneo respecto de los demás.

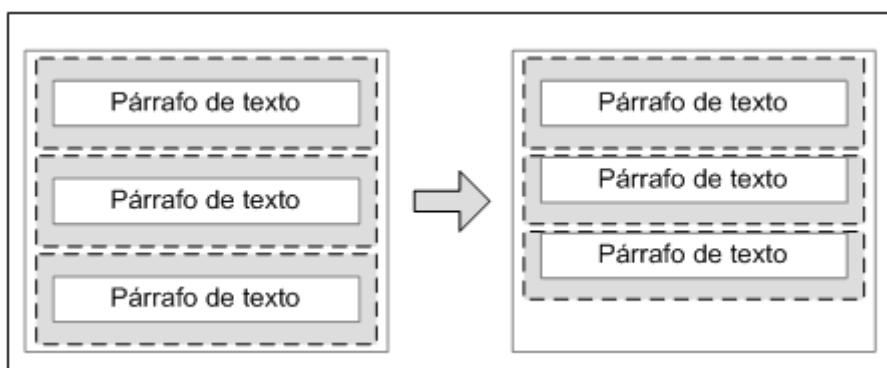


Figura 4.10. Motivo por el que se fusionan automáticamente los márgenes verticales

En el caso de un elemento que se encuentra en el interior de otro y sus márgenes se fusionan de forma automática, se puede evitar este comportamiento añadiendo un pequeño relleno (`padding: 1px`) o un borde (`border: 1px solid transparent`) al elemento contenedor.

4.2.2. Relleno

CSS define cuatro propiedades para controlar cada uno de los espacios de relleno horizontales y verticales de un elemento.

padding-top padding-right padding-bottom padding-left	Relleno superior Relleno derecho Relleno inferior Relleno izquierdo
Valores	(<code><medida></code> <code><porcentaje></code>) <code>inherit</code>
Se aplica a	Todos los elementos excepto algunos elementos de tablas como grupos de cabeceras y grupos de pies de tabla
Valor inicial	0
Descripción	Establece cada uno de los rellenos horizontales y verticales de un elemento

Cada una de las propiedades establece la separación entre el lateral de los contenidos y el borde lateral de la caja:

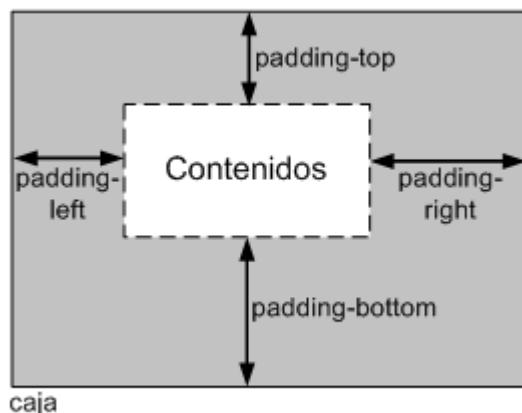


Figura 4.11. Las cuatro propiedades relacionadas con los rellenos

La siguiente imagen muestra la diferencia entre el margen y el relleno de los elementos:

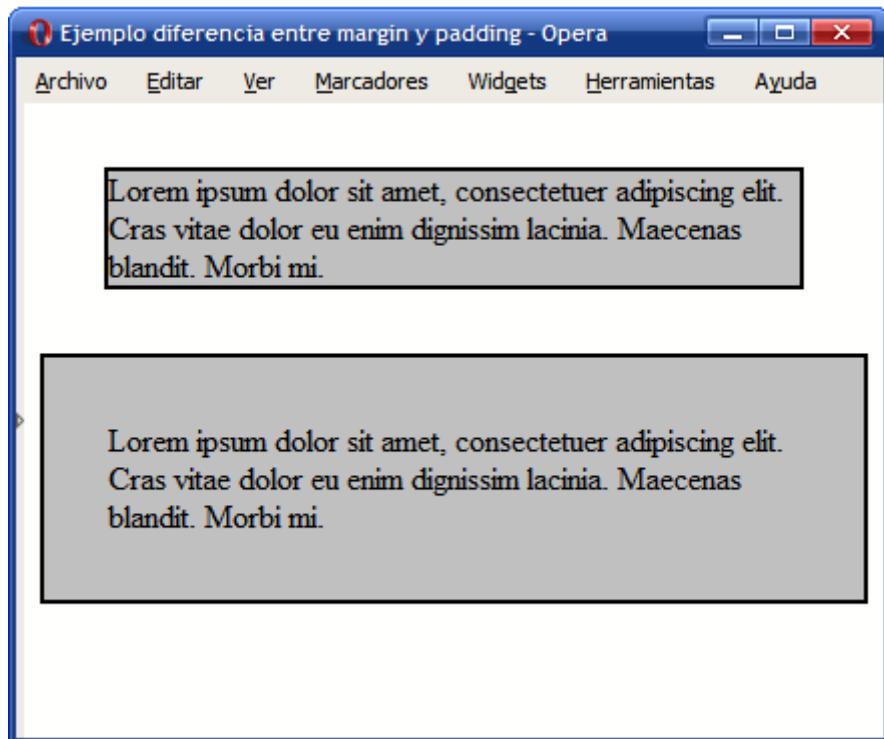


Figura 4.12. Diferencia entre relleno (padding) y margen (margin)

El código HTML y CSS del ejemplo se muestra a continuación:

```
.margen {
    margin-top: 2em; margin-right: 2em; margin-bottom: 2em; margin-left: 2em;
}
.relleno {
    padding-top: 2em; padding-right: 2em; padding-bottom: 2em; padding-left: 2em;
}

<p class="margen">Lorem ipsum dolor sit amet, consectetuer adipiscing elit.
Cras vitae dolor eu enim dignissim lacinia. Maecenas blandit. Morbi mi.</p>

<p class="relleno">Lorem ipsum dolor sit amet, consectetuer adipiscing elit.
Cras vitae dolor eu enim dignissim lacinia. Maecenas blandit. Morbi mi.</p>
```

Como sucede con la propiedad margin, CSS también define una propiedad de tipo "*shorthand*" para establecer los cuatro rellenos de un elemento de forma directa. La propiedad que permite definir de forma simultanea los cuatro márgenes se denomina padding.

padding	Relleno
Valores	(<medida> <porcentaje>) {1, 4} inherit
Se aplica a	Todos los elementos excepto algunos elementos de tablas como grupos de cabeceras y grupos de pies de tabla
Valor inicial	-
Descripción	Establece de forma directa todos los rellenos de los elementos

La notación {1, 4} de la definición anterior significa que la propiedad `padding` admite entre uno y cuatro valores, con el mismo significado que el de la propiedad `margin`. Ejemplo:

```
body {padding: 2em}      /* Todos los rellenos valen 2em */
body {padding: 1em 2em} /* Superior e inferior = 1em, Izquierdo y derecho = 2em */
body {padding: 1em 2em 3em} /* Superior = 1em, derecho = 2em, inferior = 3em, izquierdo = 2em */
body {padding: 1em 2em 3em 4em} /* Superior = 1em, derecho = 2em, inferior = 3em, izquierdo = 4em */
```

Ejercicio 3 Ver enunciado en la página 192

4.3. Bordes

CSS permite definir el aspecto de cada uno de los cuatro bordes horizontales y verticales de los elementos. Para cada borde se puede establecer su anchura, su color y su estilo.

4.3.1. Anchura

La anchura de los bordes se controla con las cuatro propiedades siguientes:

border-top-width	Anchura del borde superior
border-right-width	Anchura del borde derecho
border-bottom-width	Anchura del borde inferior
border-left-width	Anchura del borde izquierdo
Valores	(<medida> thin medium thick) inherit
Se aplica a	Todos los elementos
Valor inicial	Medium
Descripción	Establece la anchura de cada uno de los cuatro bordes de los elementos

La anchura de los bordes se puede indicar mediante una medida (absoluta o relativa y en cualquier unidad de medida de las definidas) o mediante las palabras clave `thin` (borde delgado), `medium` (borde normal) y `thick` (borde ancho).

Normalmente se utilizan medidas expresadas en píxel o en `em`, ya que las palabras clave definidas no son muy comunes.

El siguiente ejemplo muestra un elemento con cuatro anchuras diferentes de borde:

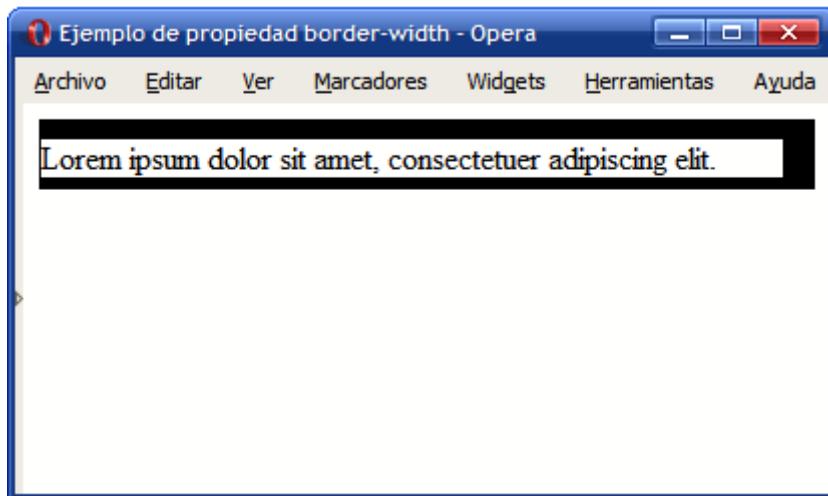


Figura 4.13. Ejemplo de propiedad border-width

Las reglas CSS utilizadas se muestran a continuación:

```
div {
    border-top-width: 10px;
    border-right-width: 1em;
    border-bottom-width: thick;
    border-left-width: thin;
}
```

Si se quiere establecer la misma anchura a todos los bordes, CSS permite la utilización de un atajo mediante una propiedad de tipo "*shorthand*", que permiten indicar varias propiedades de forma resumida:

border-width	Anchura del borde
Valores	(<medida> thin medium thick) {1, 4} inherit
Se aplica a	Todos los elementos
Valor inicial	Medium
Descripción	Establece la anchura de todos los bordes del elemento

La propiedad **border-width** permite indicar entre uno y cuatro valores. El significado de cada caso es el habitual de las propiedades "*shorthand*":

```
p { border-width: thin }           /* thin thin thin thin */
p { border-width: thin thick }     /* thin thick thin thick */
p { border-width: thin thick medium } /* thin thick medium thick */
p { border-width: thin thick medium thin } /* thin thick medium thin */
```

Si se indica un solo valor, se aplica a los cuatro bordes. Si se indican dos valores, el primero se aplica al borde superior e inferior y el segundo valor se aplica al borde izquierdo y derecho.

Si se indican tres valores, el primero se aplica al borde superior, el segundo se aplica al borde izquierdo y derecho y el tercer valor se aplica al borde inferior. Si se indican los cuatro valores, el orden de aplicación es superior, derecho, inferior e izquierdo.

4.3.2. Color

El color de los bordes se controla con las cuatro propiedades siguientes:

border-top-color	Color del borde superior
border-right-color	Color del borde derecho
border-bottom-color	Color del borde inferior
border-left-color	Color del borde izquierdo
Valores	<color> transparent inherit
Se aplica a	Todos los elementos
Valor inicial	-
Descripción	Establece el color de cada uno de los cuatro bordes de los elementos

El ejemplo anterior se puede modificar para mostrar cada uno de los bordes de un color diferente:

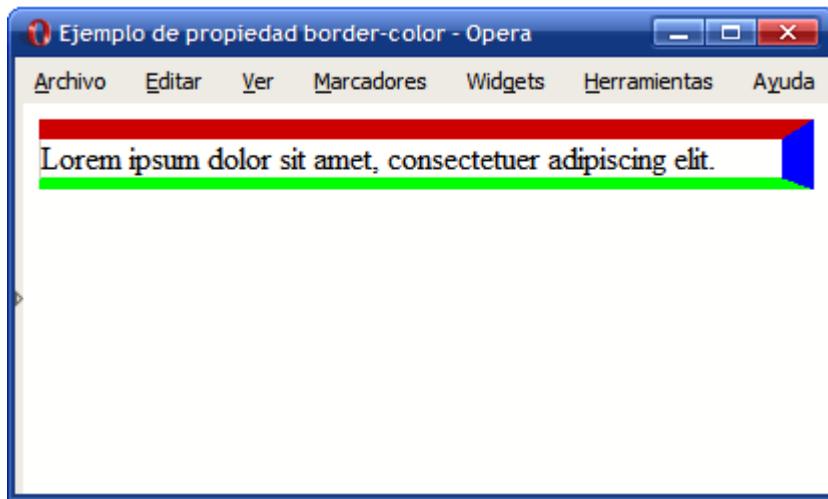


Figura 4.14. Ejemplo de propiedad border-color

Las reglas CSS necesarias para mostrar los colores anteriores son las siguientes:

```
div {
    border-top-color: #CC0000;
    border-right-color: blue;
    border-bottom-color: #00FF00;
    border-left-color: #CCC;
}
```

Si se quiere establecer el mismo color para todos los bordes, CSS permite la utilización de un atajo mediante una propiedad de tipo "*shorthand*", que permiten indicar varias propiedades de forma resumida:

border-color	Color del borde
Valores	(<color> transparent) {1, 4} inherit
Se aplica a	Todos los elementos
Valor inicial	-
Descripción	Establece el color de todos los bordes del elemento

En este caso, al igual que sucede con la propiedad `border-width`, es posible indicar de uno a cuatro valores y las reglas de aplicación son idénticas a la propiedad `border-width`.

4.3.3. Estilo

Por último, CSS permite establecer el estilo de cada uno de los bordes mediante las siguientes propiedades:

border-top-style border-right-style border-bottom-style border-left-style	Estilo del borde superior Estilo del borde derecho Estilo del borde inferior Estilo del borde izquierdo
Valores	none hidden dotted dashed solid double groove ridge inset outset inherit
Se aplica a	Todos los elementos
Valor inicial	none
Descripción	Establece el estilo de cada uno de los cuatro bordes de los elementos

El estilo de los bordes sólo se puede indicar mediante alguna de las palabras reservadas definidas por CSS. Como el valor por defecto de esta propiedad es `none`, los elementos no muestran ningún borde visible a menos que se establezca explícitamente un estilo de borde.

Siguiendo el ejemplo anterior, se puede modificar el estilo de cada uno de los bordes:

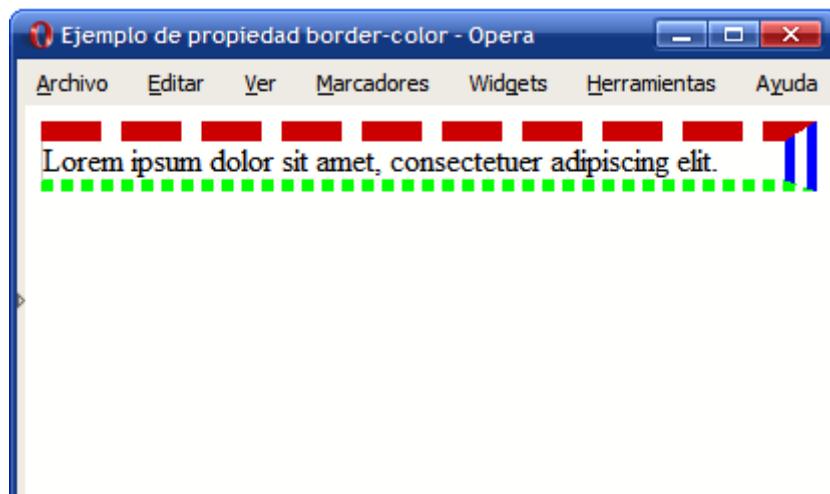


Figura 4.15. Ejemplo de propiedad border-style

Las reglas CSS necesarias para mostrar los estilos anteriores son las siguientes:

```
div {  
    border-top-style: dashed;  
    border-right-style: double;  
    border-bottom-style: dotted;  
    border-left-style: solid;  
}
```

El aspecto con el que los navegadores muestran los diferentes tipos de borde se muestra a continuación:

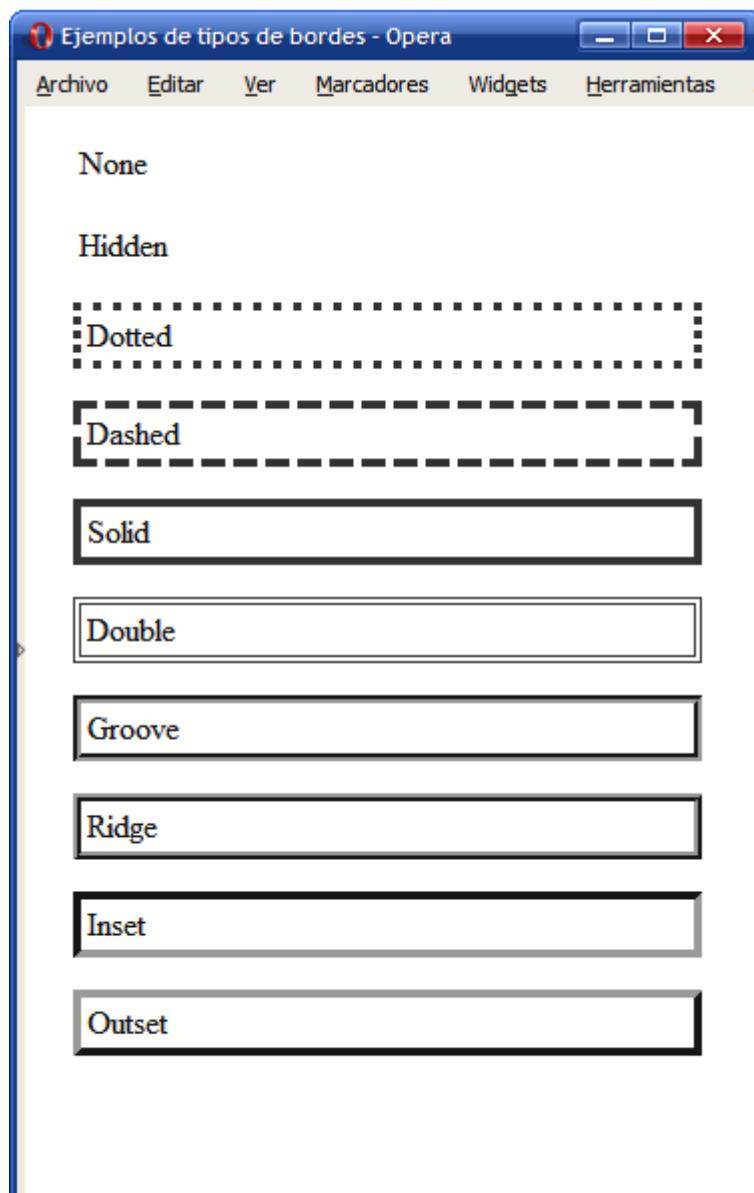


Figura 4.16. Tipos de bordes definidos por CSS

Los bordes más utilizados en los diseños habituales son `solid` y `dashed`, seguidos de `double` y `dotted`. Los estilos `none` y `hidden` son idénticos visualmente, pero se diferencian en la forma que los navegadores resuelven los conflictos entre los bordes de las celdas adyacentes en las tablas.

Si se quiere establecer el mismo estilo para todos los bordes, CSS define una propiedad de tipo "shorthand":

border-style	Estilo del borde
Valores	(none hidden dotted dashed solid double groove ridge inset outset) {1, 4} inherit
Se aplica a	Todos los elementos
Valor inicial	-
Descripción	Establece el estilo de todos los bordes del elemento

Como es habitual, la propiedad permite indicar de uno a cuatro valores diferentes y las reglas de aplicación son las habituales de las propiedades "shorthand".

4.3.4. Propiedades shorthand

Como sucede con los márgenes y los rellenos, CSS define una serie de propiedades de tipo "shorthand" que permiten establecer todos los atributos de los bordes de forma directa. CSS ha definido una propiedad "shorthand" para cada uno de los cuatro bordes y una propiedad "shorthand" global.

Antes de presentar las propiedades, es conveniente definir los tres siguientes tipos de valores:

```

<medida_borde> = <medida> | thin | medium | thick
<color_borde> = <color> | transparent
<estilo_borde> = none | hidden | dotted | dashed | solid | double | groove | ridge |
inset | outset
  
```

border-top border-right border-bottom border-left	Estilo completo del borde superior Estilo completo del borde derecho Estilo completo del borde inferior Estilo completo del borde izquierdo
Valores	(<medida_borde> <color_borde> <estilo_borde>) inherit
Se aplica a	Todos los elementos
Valor inicial	-
Descripción	Establece el estilo completo de cada uno de los cuatro bordes de los elementos

Las propiedades "shorthand" permiten establecer alguno o todos los atributos de cada borde. El siguiente ejemplo establece el color y el tipo del borde inferior, pero no su anchura:

```

h1 {
  border-bottom: solid red;
}
  
```

En el ejemplo anterior, la anchura del borde será la correspondiente al valor por defecto (`medium`). Este otro ejemplo muestra la forma habitual utilizada para establecer el estilo de cada borde:

```
div {
    border-top: 1px solid #369;
    border-bottom: 3px double #369;
}
```

Por ultimo, CSS define una propiedad de tipo "*shorthand*" global para establecer el valor de todos los atributos de todos los bordes de forma directa:

border	Estilo completo de todos los bordes
Valores	(<code><medida_borde></code> <code><color_borde></code> <code><estilo_borde></code>) <code>inherit</code>
Se aplica a	Todos los elementos
Valor inicial	-
Descripción	Establece el estilo completo de todos los bordes de los elementos

Las siguientes reglas CSS son equivalentes:

```
div {
    border-top: 1px solid red;
    border-right: 1px solid red;
    border-bottom: 1px solid red;
    border-left: 1px solid red;
}

div { border: 1px solid red; }
```

Como el valor por defecto de la propiedad `border-style` es `none`, si una propiedad *shorthand* no establece explícitamente el estilo de un borde, el elemento no muestra ese borde:

```
/* Sólo se establece el color, por lo que el estilo es
   "none" y el borde no se muestra */
div { border: red; }

/* Se establece el grosor y el color del borde, pero no
   su estilo, por lo que es "none" y el borde no se muestra */
div { border-bottom: 5px blue; }
```

Cuando los cuatro bordes no son idénticos pero sí muy parecidos, se puede utilizar la propiedad `border` para establecer de forma directa los atributos comunes de todos los bordes y posteriormente especificar para cada uno de los cuatro bordes sus propiedades particulares:

```
h1 {
    border: solid #000;
    border-top-width: 6px;
    border-left-width: 8px;
}
```

Ejercicio 4 Ver enunciado en la página 194

4.4. Margen, relleno, bordes y modelo de cajas

La anchura y altura de un elemento no solamente se calculan teniendo en cuenta sus propiedades `width` y `height`. El margen, el relleno y los bordes establecidos a un elemento determinan la anchura y altura final del elemento. En el siguiente ejemplo se muestran los estilos CSS de un elemento:

```
div {
    width: 300px;
    padding-left: 50px; padding-right: 50px;
    margin-left: 30px; margin-right: 30px;
    border: 10px solid black;
}
```

La anchura total con la que se muestra el elemento no son los 300 píxel indicados en la propiedad `width`, sino que se tienen en cuenta todos sus márgenes, rellenos y bordes:

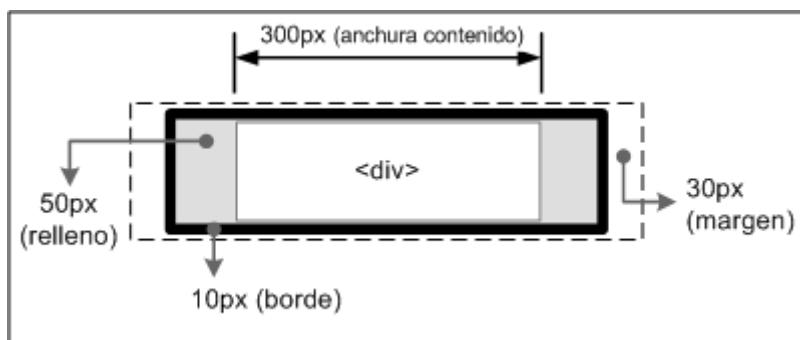


Figura 4.17. La anchura total de un elemento tiene en cuenta los márgenes, rellenos y bordes

De esta forma, la anchura del elemento en pantalla sería igual a la suma de la anchura original, los márgenes, los bordes y los rellenos:

$$30\text{px} + 10\text{px} + 50\text{px} + 300\text{px} + 50\text{px} + 10\text{px} + 30\text{px} = 480 \text{ píxel}$$

Así, la anchura/altura establecida con CSS siempre hace referencia a la anchura/altura del contenido. La anchura/altura total del elemento debe tener en cuenta además los valores del resto de partes que componen la caja del *box model*.

Por otra parte, la guerra de navegadores que se produjo en los años 90 provocó que cada fabricante (Microsoft y Netscape) añadiera sus propias extensiones y mejoras en sus productos. Posteriormente, aparecieron los estándares publicados por el W3C y los fabricantes se encontraron con el problema de la incompatibilidad entre sus implementaciones anteriores de HTML y CSS y las implementaciones que requerían los estándares.

La solución que aplicaron fue la de incluir en el navegador dos modos diferentes de funcionamiento: modo compatible con las páginas antiguas (denominado "*modo quirks*" y que se podría traducir como "*modo raro*") y modo compatible con los nuevos estándares (denominado "*modo estándar*"). El modo *quirks* es equivalente a la forma en la que se visualizaban las páginas en los navegadores Internet Explorer 4 y Netscape Navigator 4.

La diferencia más notable entre los dos modos es el tratamiento del "*box model*", lo que puede afectar gravemente al diseño de las páginas HTML. Los navegadores seleccionan automáticamente el modo en el que muestran las páginas en función del DOCTYPE definido por el

documento. En general, los siguientes tipos de DOCTYPE activan el modo *quirks* en los navegadores:

- No utilizar ningún DOCTYPE
- DOCTYPE anterior a HTML 4.0 (`<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2 Final//EN">`) * *DOCTYPE de HTML 4.01 sin URL* (`<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">`)

En el caso concreto de Internet Explorer, también activan el modo quirks los modos XHTML 1.0 que incluyen la declaración de XML (por ejemplo `<?xml version="1.0" encoding="UTF-8"?>`) al principio de la página web:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
```

Se pueden consultar todos los casos concretos que activan el modo *quirks* para cada navegador en la página <http://hsivonen.iki.fi/doctype/>

La versión 5.5 y anteriores de Internet Explorer y las versiones 6 y 7 en modo *quirks* siguen su propio modelo de cálculo de anchuras y alturas que es muy diferente al método definido por el estándar.

La siguiente imagen muestra el elemento del ejemplo anterior en la versión 6 de Internet Explorer en modo estándar:

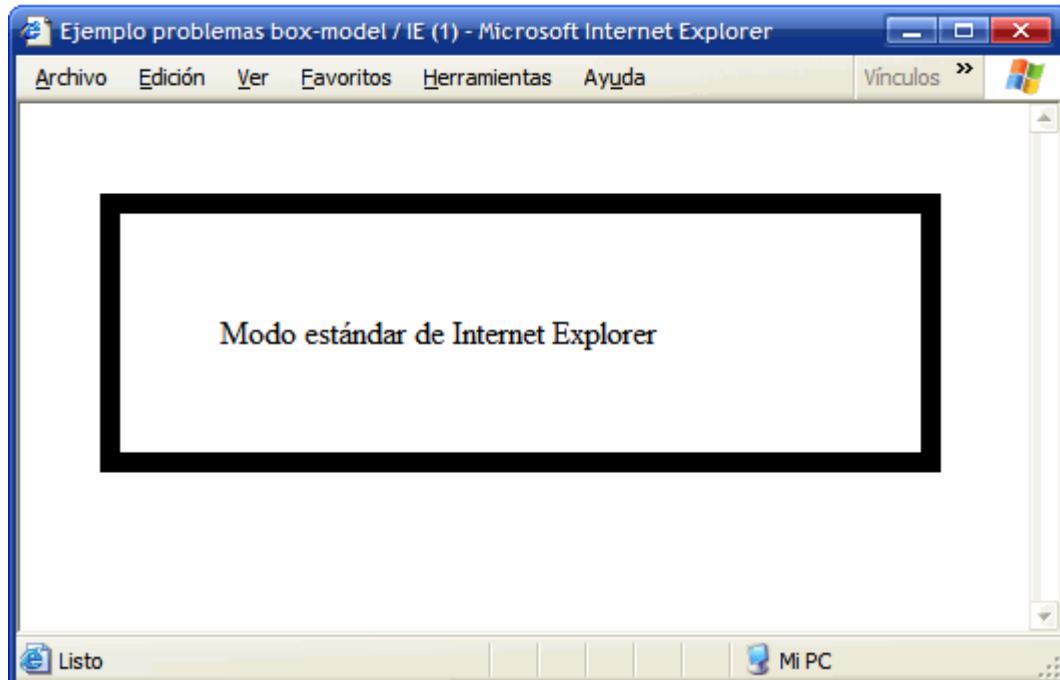


Figura 4.18. Internet Explorer 6 en modo estándar

La anchura del elemento es la que se obtiene de sumar la anchura de su contenido (300), sus bordes (2 x 10) y sus rellenos (2 x 50). Por lo tanto, la anchura del elemento son 420 píxel, a los que se suman los 30 píxel de margen lateral a cada lado.

Sin embargo, el mismo ejemplo en el modo *quirks* de la versión 6 de Internet Explorer muestra el siguiente aspecto:

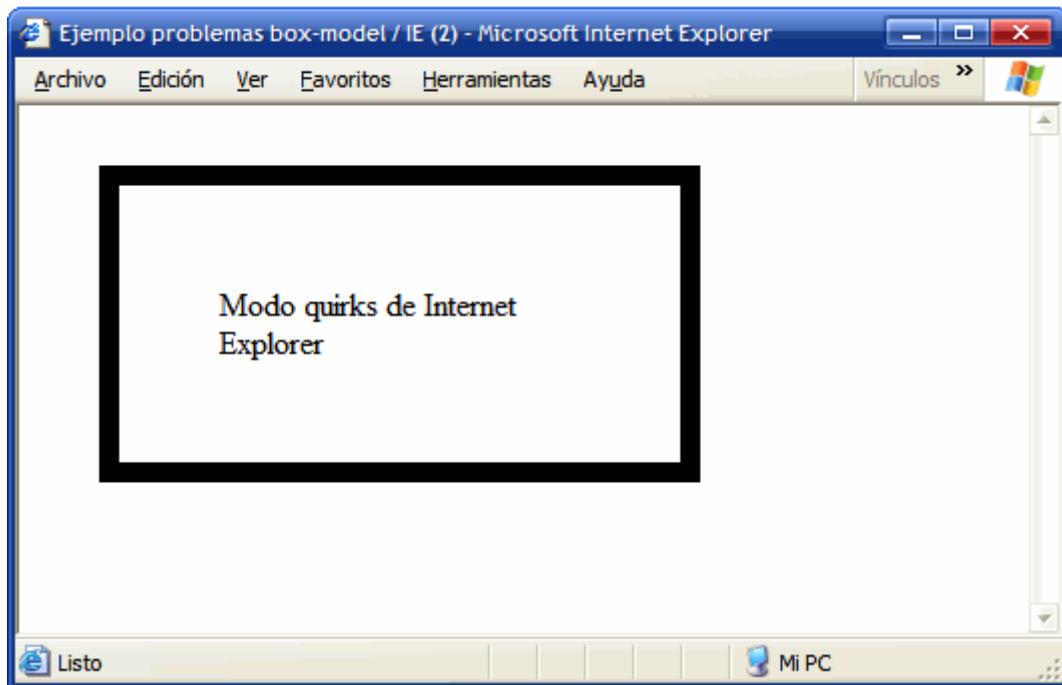


Figura 4.19. Internet Explorer 6 en modo quirks

Las versiones anteriores de Internet Explorer y las versiones 6 y 7 en modo *quirks* consideran que la anchura establecida por CSS no sólo es la anchura del contenido, sino que también incluye los bordes y el relleno.

Por lo tanto, en este caso la anchura total del elemento (sin contar los márgenes laterales) es de 300 píxel, el mismo valor que se indica en la propiedad `width`. El espacio ocupado por los bordes del elemento (2×10) y sus rellenos (2×50) se resta de la anchura de su contenido.

Para evitar este problema y crear diseños con el mismo aspecto en cualquier navegador, es necesario evitar el modo *quirks* de Internet Explorer. Por tanto, todas las páginas deberían incluir la declaración apropiada de DOCTYPE.

4.5. Fondos

El último elemento que forma el *box model* es el fondo de la caja del elemento. El fondo puede ser un color simple o una imagen. El fondo solamente se visualiza en el área ocupada por el contenido y su relleno, ya que el color de los bordes se controla directamente desde los bordes y las zonas de los márgenes siempre son transparentes.

Para establecer un color o imagen de fondo en la página entera, se debe establecer un fondo al elemento `<body>`. Si se establece un fondo a la página, como el valor inicial del fondo de los elementos es transparente, todos los elementos de la página se visualizan con el mismo fondo a menos que algún elemento especifique su propio fondo.

CSS define cinco propiedades para establecer el fondo de cada elemento (`background-color`, `background-image`, `background-repeat`, `background-attachment`, `background-position`) y otra propiedad de tipo "shorthand" (`background`).

background-color	Color de fondo
Valores	<code><color></code> <code>transparent</code> <code>inherit</code>
Se aplica a	Todos los elementos
Valor inicial	<code>transparent</code>
Descripción	Establece un color de fondo para los elementos

El siguiente ejemplo muestra una página web con un color gris claro de fondo:

```
body {
    background-color: #F5F5F5;
}
```

En ocasiones, es necesario crear un fondo más complejo que un simple color. CSS permite mostrar una imagen como fondo de cualquier elemento:

background-image	Imagen de fondo
Valores	<code><url></code> <code>none</code> <code>inherit</code>
Se aplica a	Todos los elementos
Valor inicial	<code>none</code>
Descripción	Establece una imagen como fondo para los elementos

CSS permite establecer de forma simultánea un color y una imagen de fondo. En este caso, la imagen se muestra delante del color, por lo que solamente si la imagen contiene zonas transparentes es posible ver el color de fondo.

El siguiente ejemplo muestra una imagen como fondo de toda la página:

```
body { background-image: url("imagenes/fondo.png") }
```

Las imágenes de fondo se indican a través de su URL, que puede ser absoluta o relativa. Suele ser recomendable crear una carpeta de imágenes que se encuentre en el mismo directorio que los archivos CSS y que almacene todas las imágenes utilizadas en el diseño de las páginas.

Así, las imágenes correspondientes al diseño de la página se mantienen separadas del resto de imágenes del sitio y el código CSS es más sencillo (por utilizar URL relativas) y más fácil de mantener (por no tener que actualizar URL absolutas en caso de que se cambie la estructura del sitio web).

Por otra parte, suele ser habitual indicar un color de fondo siempre que se muestra una imagen de fondo. En caso de que la imagen no se pueda mostrar o contenga errores, el navegador mostrará el color indicado (que debería ser, en lo posible, similar a la imagen) y la página no parecerá que contiene errores.

Si la imagen que se quiere mostrar es demasiado grande para el fondo del elemento, solamente se muestra la parte de imagen comprendida en el tamaño del elemento. Si la imagen es más pequeña que el elemento, CSS la repite horizontal y verticalmente hasta llenar el fondo del elemento.

Este comportamiento es útil para establecer un fondo complejo a una página web entera. El siguiente ejemplo utiliza una imagen muy pequeña para establecer un fondo complejo a toda una página:

Imagen original



Figura 4.20. Imagen original utilizada para el fondo de la página

Reglas CSS

```
body {  
    background-image:url(imagenes/fondo.gif);  
}
```

Resultado

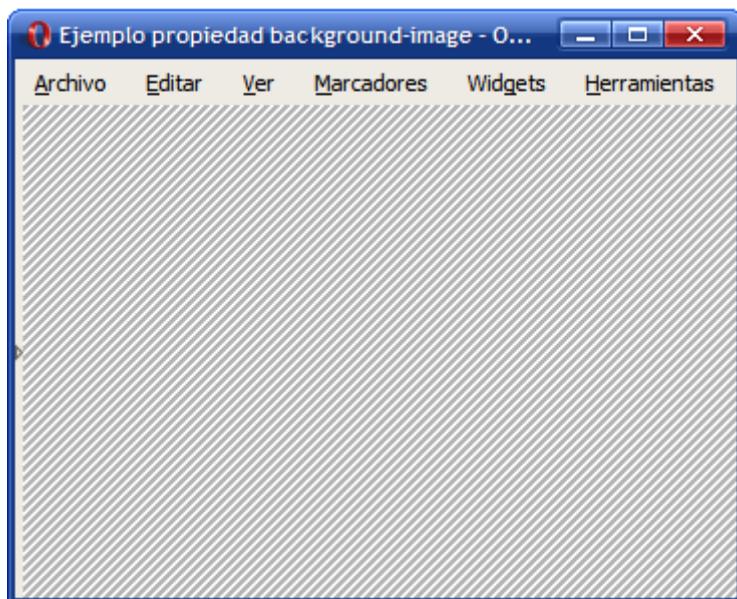


Figura 4.21. Página con una imagen de fondo

Con una imagen muy pequeña (y que por tanto, se puede descargar en muy poco tiempo) se consigue cubrir completamente el fondo de la página, con lo que se consigue un gran ahorro de ancho de banda.

En ocasiones, no es conveniente que la imagen de fondo se repita horizontal y verticalmente. Para ello, CSS introduce la propiedad `background-repeat` que permite controlar la forma de repetición de las imágenes de fondo.

background-repeat	Repetición de la imagen de fondo
Valores	repeat repeat-x repeat-y no-repeat inherit
Se aplica a	Todos los elementos
Valor inicial	repeat
Descripción	Controla la forma en la que se repiten las imágenes de fondo

El valor **repeat** indica que la imagen se debe repetir en todas direcciones y por tanto, es el comportamiento por defecto. El valor **no-repeat** muestra una sola vez la imagen y no se repite en ninguna dirección. El valor **repeat-x** repite la imagen sólo horizontalmente y el valor **repeat-y** repite la imagen solamente de forma vertical.

El sitio web <http://www.kottke.org/> utiliza el valor **repeat-x** para mostrar una imagen de fondo en la cabecera de la página:



Figura 4.22. Uso de repeat-x en la página de Kottke.org

Las reglas CSS definidas para la cabecera son:

```
#hdr {
    background: url(/images/ds.gif) repeat-x;
    width: 100%;
    text-align: center;
}
```

Por otra parte, el sitio web <http://veerle.duoh.com/> utiliza el valor `repeat-y` para mostrar el fondo de una columna de contenidos:

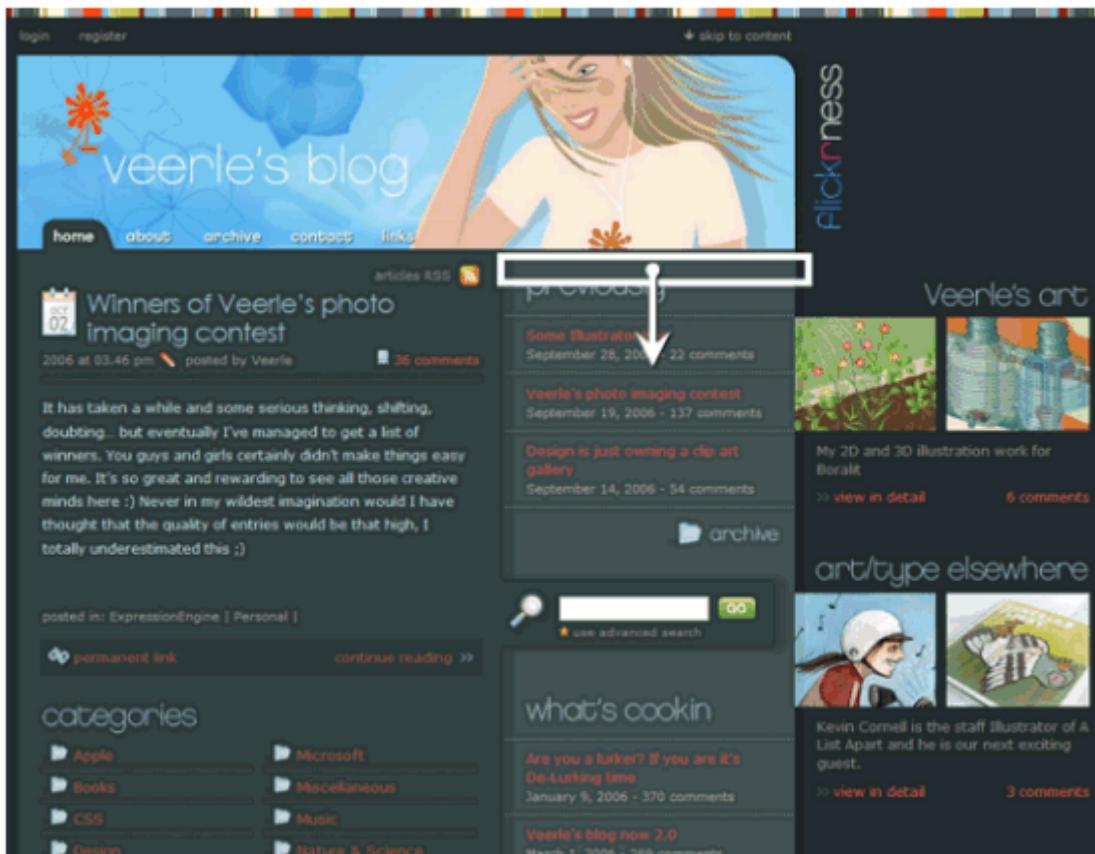


Figura 4.23. Uso de `repeat-y` en la página de [Veerle.duoh.com](http://veerle.duoh.com/)

Las reglas CSS definidas para esa columna de contenidos son:

```
.wide #content-secondary {
    width: 272px;
    margin: 13px 0 0 0;
    position: relative;
    margin-left: -8px;
    background: url("./graphics/wide/bg-content-secondary.gif) repeat-y;
}
```

Además de seleccionar el tipo de repetición de las imágenes de fondo, CSS permite controlar la posición de la imagen dentro del fondo del elemento mediante la propiedad `background-position`.

background-position	Posición de la imagen de fondo
Valores	((<porcentaje> <medida> left center right) (<porcentaje> <medida> top center bottom)?) ((left center right) (top center bottom)) inherit
Se aplica a	Todos los elementos
Valor inicial	0% 0%
Descripción	Controla la posición en la que se muestra la imagen en el fondo del elemento

La propiedad `background-position` permite indicar la distancia que se desplaza la imagen de fondo respecto de su posición original situada en la esquina superior izquierda.

Si se indican dos porcentajes o dos medidas, el primero indica el desplazamiento horizontal y el segundo el desplazamiento vertical respecto del origen (situado en la esquina superior izquierda). Si solamente se indica un porcentaje o una medida, se considera que es el desplazamiento horizontal y al desplazamiento vertical se le asigna automáticamente el valor de 50%.

Cuando se utilizan porcentajes, su interpretación no es intuitiva. Si el valor de la propiedad `background-position` se indica mediante dos porcentajes `x%` `y%`, el navegador coloca el punto (`x%`, `y%`) de la imagen de fondo en el punto (`x%`, `y%`) del elemento.

Las palabras clave permitidas son equivalentes a algunos porcentajes significativos: `top = 0%`, `left = 0%`, `center = 50%`, `bottom = 100%`, `right = 100%`.

CSS permite mezclar porcentajes y palabras clave, como por ejemplo `50% 2cm`, `center 2cm`, `center 10%`.

Si se utilizan solamente palabras clave, el orden es indiferente y por tanto, es equivalente indicar `top left` y `left top`.

El siguiente ejemplo muestra una misma imagen de fondo posicionada de tres formas diferentes:

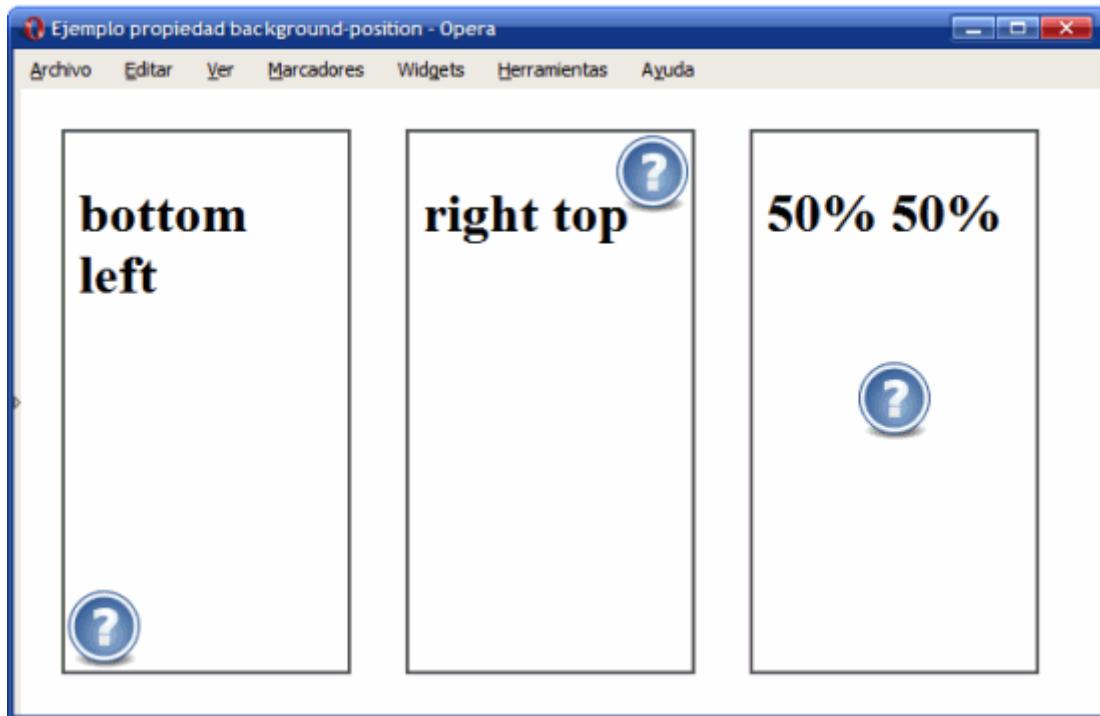


Figura 4.24. Ejemplo de propiedad background-position

Las reglas CSS del ejemplo anterior se muestran a continuación:

```
#caja1 {  
    background-image: url(images/help.png);  
    background-repeat: no-repeat;  
    background-position: bottom left;  
}  
#caja2 {  
    background-image: url(images/help.png);  
    background-repeat: no-repeat;  
    background-position: right top;  
}  
#caja3 {  
    background-image: url(images/help.png);  
    background-repeat: no-repeat;  
    background-position: 50% 50%;  
}  
  
<div id="caja1"><h1>bottom left</h1></div>  
<div id="caja2"><h1>right top</h1></div>  
<div id="caja3"><h1>50% 50%</h1></div>
```

Opcionalmente, se puede indicar que el fondo permanezca fijo cuando la ventana del navegador se desplaza mediante las barras de scroll. Se trata de un comportamiento que en general no es deseable y que algunos navegadores no soportan correctamente. La propiedad que controla este comportamiento es `background-attachment`.

background-attachment	Comportamiento de la imagen de fondo
Valores	scroll fixed inherit
Se aplica a	Todos los elementos
Valor inicial	scroll
Descripción	Controla la forma en la que se visualiza la imagen de fondo: permanece fija cuando se hace scroll en la ventana del navegador o se desplaza junto con la ventana

Para hacer que una imagen de fondo se muestre fija al desplazar la ventana del navegador, se debe añadir la propiedad `background-attachment: fixed`.

Por último, CSS define una propiedad de tipo "shorthand" para indicar todas las propiedades de los colores e imágenes de fondo de forma directa. La propiedad se denomina `background` y es la que generalmente se utiliza para establecer las propiedades del fondo de los elementos.

background	Fondo de un elemento
Valores	(<background-color> <background-image> <background-repeat> <background-attachment> <background-position>) inherit
Se aplica a	Todos los elementos
Valor inicial	-
Descripción	Establece todas las propiedades del fondo de un elemento

El orden en el que se indican las propiedades es indiferente, aunque en general se sigue el formato indicado de color, url de imagen, repetición y posición.

El siguiente ejemplo muestra la ventaja de utilizar la propiedad `background`:

```
/* Color e imagen de fondo de la página mediante una propiedad shorthand */
body { background: #222d2d url(./graphics/colorstrip.gif) repeat-x 0 0; }

/* La propiedad shorthand anterior es equivalente a las siguientes propiedades */
body {
    background-color: #222d2d;
    background-image: url(./graphics/colorstrip.gif);
    background-repeat: repeat-x;
    background-position: 0 0;
}
```

La propiedad `background` permite asignar todos o sólo algunos de todos los valores que se pueden definir para los fondos de los elementos:

```
background: url(./graphics/wide/bg-content-secondary.gif) repeat-y;

background: url(./graphics/wide/footer-content-secondary.gif) no-repeat bottom left;

background: transparent url(./graphics/navigation.gif) no-repeat 0 -27px;
```

```
background: none;  
background: #293838 url(./graphics/icons/icon-permalink-big.gif) no-repeat center left;
```

Ejercicio 5 | Ver enunciado en la página 196

Capítulo 5. Posicionamiento y visualización

Cuando los navegadores descargan el contenido HTML y CSS de las páginas web, aplican un procesamiento muy complejo antes de mostrar las páginas en la pantalla del usuario.

Para cumplir con el modelo de cajas presentado en el capítulo anterior, los navegadores crean una caja para representar a cada elemento de la página HTML. Los factores que se tienen en cuenta para generar cada caja son:

- Las propiedades `width` y `height` de la caja (si están establecidas).
- El tipo de cada elemento HTML (elemento de bloque o elemento en línea).
- Posicionamiento de la caja (normal, relativo, absoluto, fijo o flotante).
- Las relaciones entre elementos (dónde se encuentra cada elemento, elementos descendientes, etc.)
- Otro tipo de información, como por ejemplo el tamaño de las imágenes y el tamaño de la ventana del navegador.

En este capítulo se muestran los cinco tipos de posicionamientos definidos para las cajas y se presentan otras propiedades que afectan a la forma en la que se visualizan las cajas.

5.1. Tipos de elementos

El estándar HTML clasifica a todos sus elementos en dos grandes grupos: elementos en línea y elementos de bloque.

Los elementos de bloque ("*block elements*" en inglés) siempre empiezan en una nueva línea y ocupan todo el espacio disponible hasta el final de la línea. Por su parte, los elementos en línea ("*inline elements*" en inglés) no empiezan necesariamente en nueva línea y sólo ocupan el espacio necesario para mostrar sus contenidos.

Debido a este comportamiento, el tipo de un elemento influye de forma decisiva en la caja que el navegador crea para mostrarlo. La siguiente imagen muestra las cajas que crea el navegador para representar los diferentes elementos que forman una página HTML:

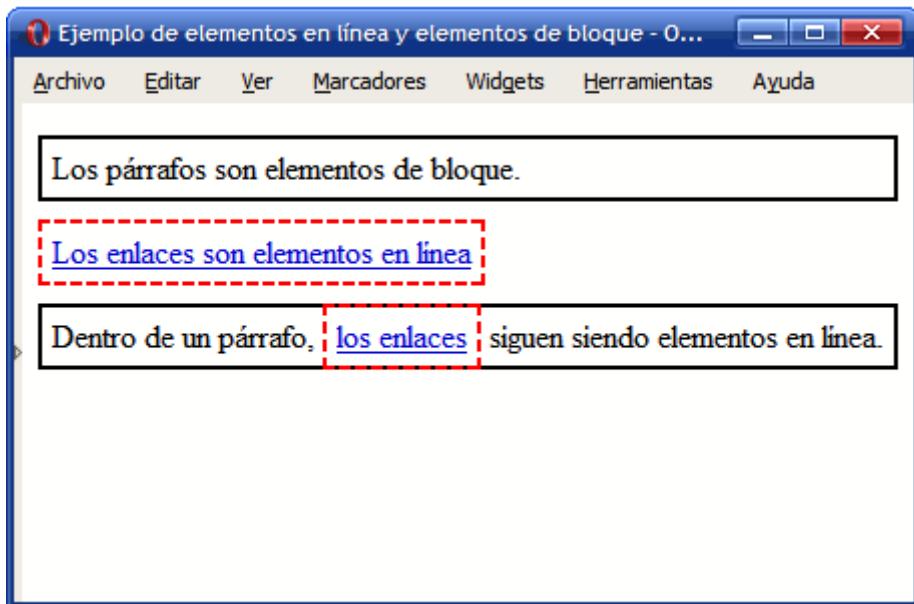


Figura 5.1. Cajas creadas por los elementos de línea y los elementos de bloque

El primer elemento de la página anterior es un párrafo. Los párrafos son elementos de bloque y por ese motivo su caja empieza en una nueva línea y llega hasta el final de esa misma línea. Aunque los contenidos de texto del párrafo no son suficientes para ocupar toda la línea, el navegador reserva todo el espacio disponible en la primera línea.

El segundo elemento de la página es un enlace. Los enlaces son elementos en línea, por lo que su caja sólo ocupa el espacio necesario para mostrar sus contenidos. Si después de este elemento se incluye otro elemento en línea (por ejemplo otro enlace o una imagen) el navegador mostraría los dos elementos en la misma línea, ya que existe espacio suficiente.

Por último, el tercer elemento de la página es un párrafo que se comporta de la misma forma que el primer párrafo. En su interior, se encuentra un enlace que también se comporta de la misma forma que el enlace anterior. Así, el segundo párrafo ocupa toda una línea y el segundo enlace sólo ocupa el espacio necesario para mostrar sus contenidos.

Por sus características, los elementos de bloque no pueden insertarse dentro de elementos en línea y tan sólo pueden aparecer dentro de otros elementos de bloque. En cambio, un elemento en línea puede aparecer tanto dentro de un elemento de bloque como 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, noscript, ol, p, pre, table, ul.

Los siguientes elementos también se considera que son de bloque: dd, dt, frameset, 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`.

5.2. Posicionamiento

Los navegadores crean y posicionan de forma automática todas las cajas que forman cada página HTML. No obstante, CSS permite al diseñador modificar la posición en la que se muestra cada caja.

Utilizando las propiedades que proporciona CSS para alterar la posición de las cajas es posible realizar efectos muy avanzados y diseñar estructuras de páginas que de otra forma no serían posibles.

El estándar de CSS define cinco modelos diferentes para posicionar una caja:

- Posicionamiento normal o estático: se trata del posicionamiento que utilizan los navegadores si no se indica lo contrario.
- Posicionamiento relativo: variante del posicionamiento normal que consiste en posicionar una caja según el posicionamiento normal y después desplazarla respecto de su posición original.
- Posicionamiento absoluto: la posición de una caja se establece de forma absoluta respecto de su elemento contenedor y el resto de elementos de la página ignoran la nueva posición del elemento.
- Posicionamiento fijo: variante del posicionamiento absoluto que convierte una caja en un elemento inamovible, de forma que su posición en la pantalla siempre es la misma independientemente del resto de elementos e independientemente de si el usuario sube o baja la página en la ventana del navegador.
- Posicionamiento flotante: se trata del modelo más especial de posicionamiento, ya que desplaza las cajas todo lo posible hacia la izquierda o hacia la derecha de la línea en la que se encuentran.

El posicionamiento de una caja se establece mediante la propiedad `position`:

position	Posicionamiento
Valores	<code>static</code> <code>relative</code> <code>absolute</code> <code>fixed</code> <code>inherit</code>
Se aplica a	Todos los elementos
Valor inicial	<code>static</code>
Descripción	Selecciona el posicionamiento con el que se mostrará el elemento

El significado de cada uno de los posibles valores de la propiedad `position` es el siguiente:

- `static`: corresponde al posicionamiento normal o estático. Si se utiliza este valor, se ignoran los valores de las propiedades `top`, `right`, `bottom` y `left` que se verán a continuación.

- **relative**: corresponde al posicionamiento relativo. El desplazamiento de la caja se controla con las propiedades `top`, `right`, `bottom` y `left`.
- **absolute**: corresponde al posicionamiento absoluto. El desplazamiento de la caja también se controla con las propiedades `top`, `right`, `bottom` y `left`, pero su interpretación es mucho más compleja, ya que el origen de coordenadas del desplazamiento depende del posicionamiento de su elemento contenedor.
- **fixed**: corresponde al posicionamiento fijo. El desplazamiento se establece de la misma forma que en el posicionamiento absoluto, pero en este caso el elemento permanece inamovible en la pantalla.

La propiedad `position` no permite controlar el posicionamiento flotante, que se establece con otra propiedad llamada `float` y que se explica más adelante. Además, la propiedad `position` sólo indica cómo se posiciona una caja, pero no la desplaza.

Normalmente, cuando se posiciona una caja también es necesario desplazarla respecto de su posición original o respecto de otro origen de coordenadas. CSS define cuatro propiedades llamadas `top`, `right`, `bottom` y `left` para controlar el desplazamiento de las cajas posicionadas:

top right bottom left	Desplazamiento superior Desplazamiento lateral derecho Desplazamiento inferior Desplazamiento lateral izquierdo
Valores	<code><medida></code> <code><porcentaje></code> <code>auto</code> <code>inherit</code>
Se aplica a	Todos los elementos posicionados
Valor inicial	<code>auto</code>
Descripción	Indican el desplazamiento horizontal y vertical del elemento respecto de su posición original

En el caso del posicionamiento relativo, cada una de estas propiedades indica el desplazamiento del elemento desde la posición original de su borde superior/derecho/inferior/izquierdo. Si el posicionamiento es absoluto, las propiedades indican el desplazamiento del elemento respecto del borde superior/derecho/inferior/izquierdo de su primer elemento padre posicionado.

En cualquiera de los dos casos, si el desplazamiento se indica en forma de porcentaje, se refiere al porcentaje sobre la anchura (propiedades `right` y `left`) o altura (propiedades `top` y `bottom`) del elemento.

5.3. Posicionamiento normal

El posicionamiento normal o estático es el modelo que utilizan por defecto los navegadores para mostrar los elementos de las páginas. En este modelo, ninguna caja se desplaza respecto de su posición original, por lo que sólo se tiene en cuenta si el elemento es de bloque o en línea.

Los elementos de bloque forman lo que CSS denomina "*contextos de formato de bloque*". En este tipo de contextos, las cajas se muestran una debajo de otra comenzando desde el principio del elemento contenedor. La distancia entre las cajas se controla mediante los márgenes verticales.

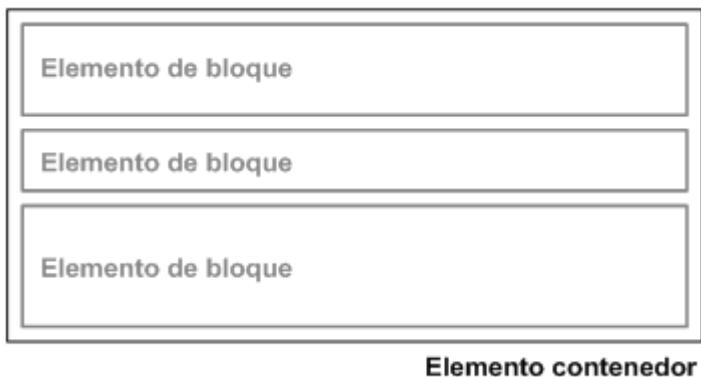


Figura 5.2. Posicionamiento normal de los elementos de bloque

Si un elemento se encuentra dentro de otro, el elemento padre se llama "*elemento contenedor*" y determina tanto la posición como el tamaño de todas sus cajas interiores.

Si un elemento no se encuentra dentro de un elemento contenedor, entonces su elemento contenedor es el elemento <body> de la página. Normalmente, la anchura de los elementos de bloque está limitada a la anchura de su elemento contenedor, aunque en algunos casos sus contenidos pueden desbordar el espacio disponible.

Los elementos en línea forman los "*contextos de formato en línea*". En este tipo de contextos, las cajas se muestran una detrás de otra de forma horizontal comenzando desde la posición más a la izquierda de su elemento contenedor. La distancia entre las cajas se controla mediante los márgenes laterales.

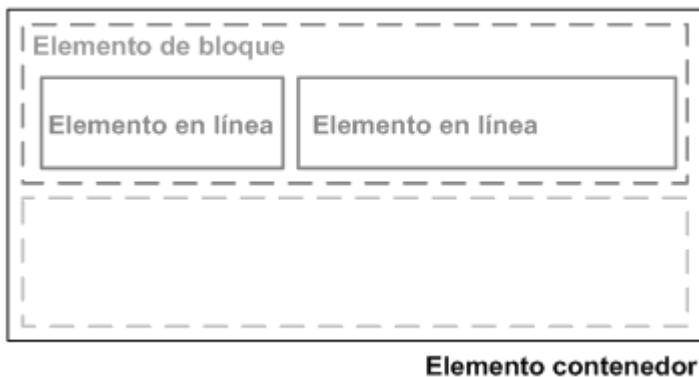


Figura 5.3. Posicionamiento normal de los elementos en línea

Si las cajas en línea ocupan más espacio del disponible en su propia línea, el resto de cajas se muestran en las líneas inferiores. Si las cajas en línea ocupan un espacio menor que su propia línea, se puede controlar la distribución de las cajas mediante la propiedad `text-align` para centrarlas, alinearlas a la derecha o justificarlas.

5.4. Posicionamiento relativo

El estándar CSS considera que el posicionamiento relativo es un caso particular del posicionamiento normal, aunque en realidad presenta muchas diferencias.

El posicionamiento relativo permite desplazar una caja respecto de su posición original establecida mediante el posicionamiento normal. El desplazamiento de la caja se controla con las propiedades `top`, `right`, `bottom` y `left`.

El desplazamiento de una caja no afecta al resto de cajas adyacentes, que se muestran en la misma posición que si la caja desplazada no se hubiera movido de su posición original.

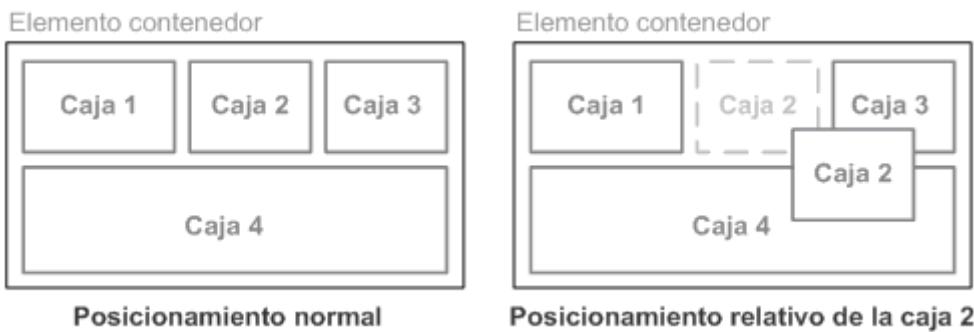


Figura 5.4. Ejemplo de posicionamiento relativo de un elemento

En la imagen anterior, la caja 2 se ha desplazado lateralmente hacia la derecha y verticalmente de forma descendente. Como el resto de cajas de la página no modifican su posición, se producen solapamientos entre los contenidos de las cajas.

La propiedad `left` desplaza la caja hacia su derecha, la propiedad `right` la desplaza hacia su izquierda, la posición `top` desplaza la caja de forma descendente y la propiedad `bottom` desplaza la caja de forma ascendente. Si se utilizan valores negativos en estas propiedades, su efecto es justamente el inverso.

Las cajas desplazadas de forma relativa no modifican su tamaño, por lo que los valores de las propiedades `left` y `right` siempre cumplen que `left = -right`.

Si tanto `left` como `right` tienen un valor de `auto` (que es su valor por defecto) la caja no se mueve de su posición original. Si sólo el valor de `left` es `auto`, su valor real es `-right`. Igualmente, si sólo el valor de `right` es `auto`, su valor real es `-left`.

Si tanto `left` como `right` tienen valores distintos de `auto`, uno de los dos valores se tiene que ignorar porque son mutuamente excluyentes. Para determinar la propiedad que se tiene en cuenta, se considera el valor de la propiedad `direction`.

La propiedad `direction` permite establecer la dirección del texto de un contenido. Si el valor de `direction` es `ltr`, el texto se muestra de izquierda a derecha, que es el método de escritura habitual en la mayoría de países. Si el valor de `direction` es `rtl`, el método de escritura es de derecha a izquierda, como el utilizado por los idiomas árabe y hebreo.

Si el valor de `direction` es `ltr`, y las propiedades `left` y `right` tienen valores distintos de `auto`, se ignora la propiedad `right` y sólo se tiene en cuenta el valor de la propiedad `left`. De la misma forma, si el valor de `direction` es `rtl`, se ignora el valor de `left` y sólo se tiene en cuenta el valor de `right`.

El siguiente ejemplo muestra tres imágenes posicionadas de forma normal:



Figura 5.5. Elementos posicionados de forma normal

Aplicando el posicionamiento relativo, se desplaza la primera imagen de forma descendente:

```
img.desplazada {  
    position: relative;  
    top: 8em;  
}  
  
  
  

```

El aspecto que muestran ahora las imágenes es el siguiente:

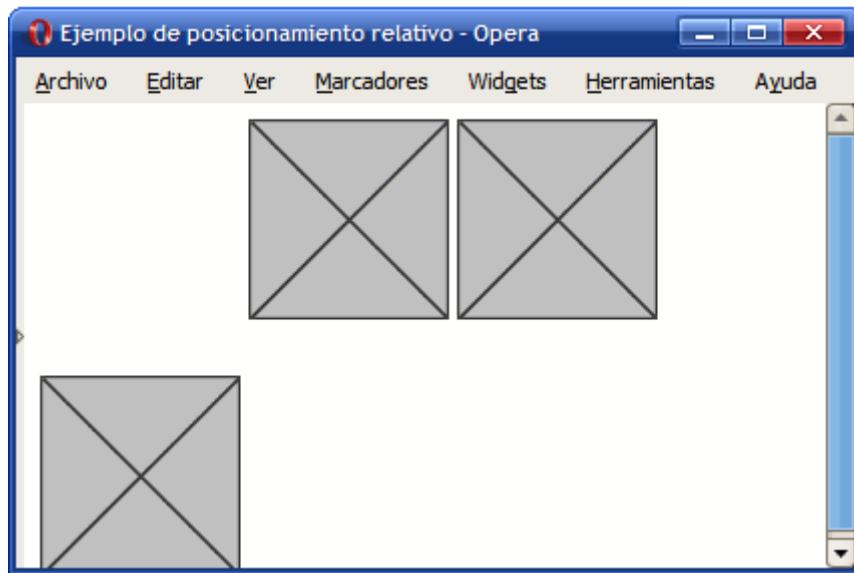


Figura 5.6. Elemento posicionado de forma relativa

El resto de imágenes no varían su posición y por tanto no ocupan el hueco dejado por la primera imagen, ya que el posicionamiento relativo no influye en el resto de elementos de la página. El único problema de posicionar elementos de forma relativa es que se pueden producir solapamientos con otros elementos de la página.

5.5. Posicionamiento absoluto

El posicionamiento absoluto se emplea para establecer de forma precisa la posición en la que se muestra la caja de un elemento. La nueva posición de la caja se indica mediante las propiedades `top`, `right`, `bottom` y `left`. La interpretación de los valores de estas propiedades es mucho más compleja que en el posicionamiento relativo, ya que en este caso dependen del posicionamiento del elemento contenedor.

Cuando una caja se posiciona de forma absoluta, el resto de elementos de la página la ignoran y ocupan el lugar original ocupado por la caja posicionada. Al igual que en el posicionamiento relativo, cuando se posiciona de forma absoluta una caja es probable que se produzcan solapamientos con otras cajas.

En el siguiente ejemplo, se posiciona de forma absoluta la caja 2:

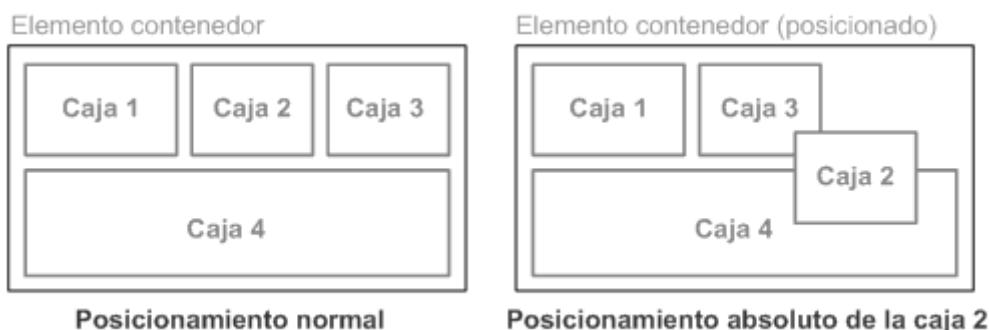


Figura 5.7. Ejemplo de posicionamiento absoluto de un elemento

La caja 2 está posicionada de forma absoluta, lo que implica que el resto de elementos ignoran que esa caja exista. Por este motivo, la caja 3 deja su lugar original y pasa a ocupar el hueco dejado por la caja 2.

En el estándar de CSS, esta característica de las cajas posicionadas de forma absoluta se explica como que la caja *sale por completo del flujo normal del documento*. De hecho, las cajas posicionadas de forma absoluta parecen que están en un nivel diferente al resto de elementos de la página.

Por otra parte, el desplazamiento de una caja posicionada de forma absoluta se indica mediante las propiedades `top`, `right`, `bottom` y `left`. A diferencia de posicionamiento relativo, en este caso la referencia de los valores de esas propiedades es el origen de coordenadas de su primer elemento contenedor posicionado.

Determinar el origen de coordenadas a partir del cual se desplaza una caja posicionada de forma absoluta es un proceso complejo que se compone de los siguientes pasos:

- Se buscan todos los elementos contenedores de la caja hasta llegar al elemento `<body>` de la página.
- Se recorren todos los elementos contenedores empezando por el más cercano a la caja y llegando hasta el `<body>`
- De todos ellos, el navegador se queda con el primer elemento contenedor que esté posicionado de cualquier forma diferente a `position: static`
- La esquina superior izquierda de ese elemento contenedor posicionado es el origen de coordenadas.

Una vez obtenido el origen de coordenadas, se interpretan los valores de las propiedades `top`, `right`, `bottom` y `left` respecto a ese origen y se desplaza la caja hasta su nueva posición.

En los siguientes ejemplos, se va a utilizar la página HTML que muestra la siguiente imagen:

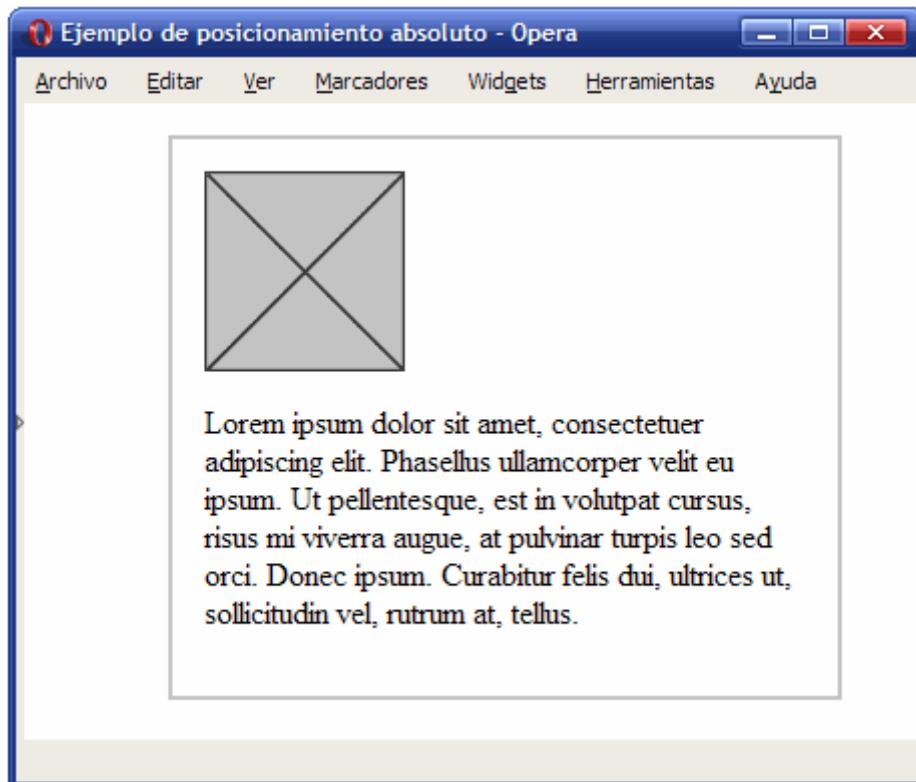


Figura 5.8. Situación original antes de modificar el posicionamiento

A continuación, se muestra el código HTML y CSS de la página original:

```
div {  
    border: 2px solid #CCC;  
    padding: 1em;  
    margin: 1em 0 1em 4em;  
    width: 300px;  
}  
  
<div>  
      
    <p>Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Phasellus ullamcorper velit eu ipsum. Ut pellentesque, est in volutpat cursus, risus mi viverra augue, at pulvinar turpis leo sed orci. Donec ipsum. Curabitur felis dui, ultrices ut, sollicitudin vel, rutrum at, tellus.</p>  
</div>
```

En primer lugar, se posiciona de forma absoluta la imagen mediante la propiedad `position` y se indica su nueva posición mediante las propiedades `top` y `left`:

```
div img {  
    position: absolute;  
    top: 3em;  
    left: 3em;  
}
```

El resultado visual se muestra en la siguiente imagen:

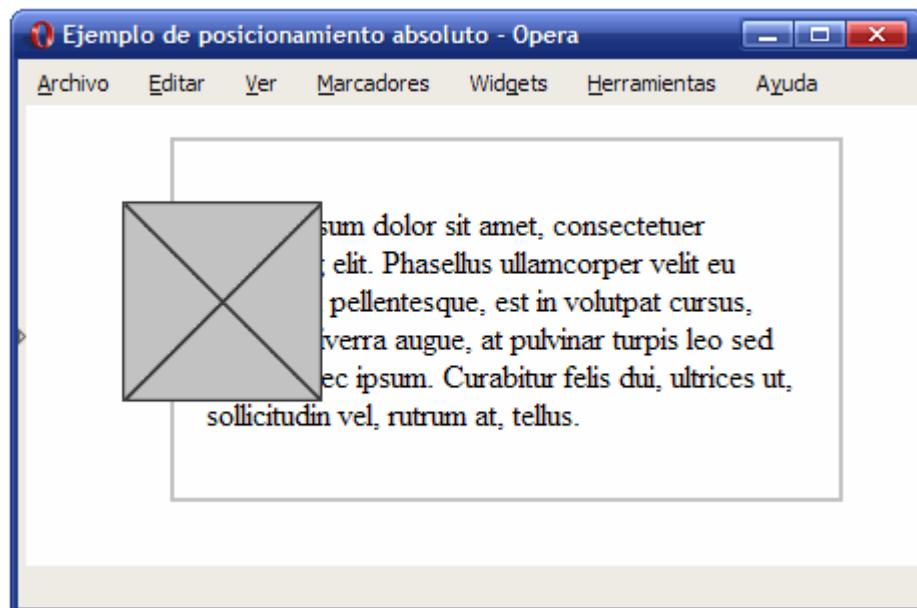


Figura 5.9. Imagen posicionada de forma absoluta

La imagen posicionada de forma absoluta no toma como origen de coordenadas la esquina superior izquierda de su elemento contenedor `<div>`, sino que su referencia es la esquina superior izquierda de la página:

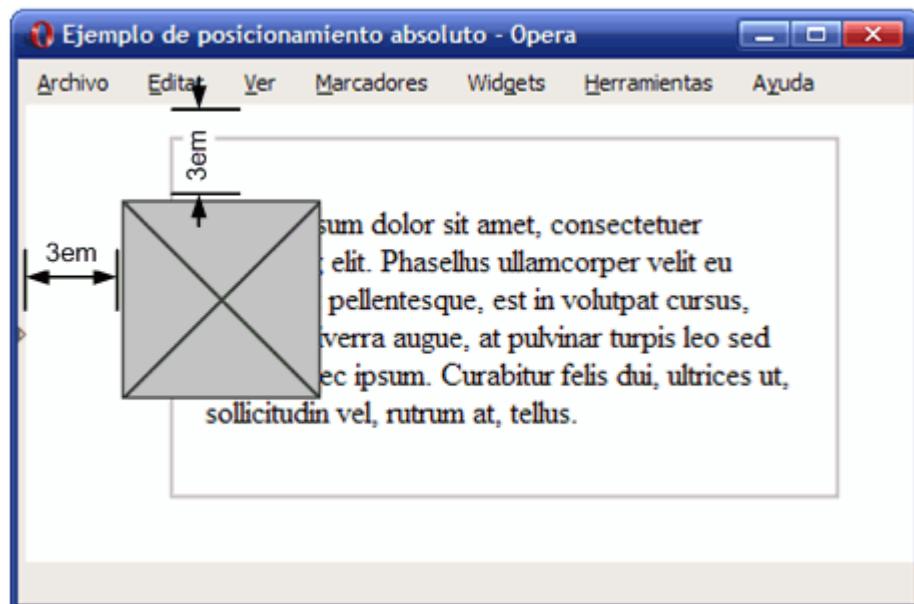


Figura 5.10. La referencia del posicionamiento absoluto es la página entera

Para posicionar la imagen de forma absoluta, el navegador realiza los siguientes pasos:

1. Obtiene la lista de elementos contenedores de la imagen: `<div>` y `<body>`.
2. Recorre la lista de elementos desde el más cercano a la imagen (el `<div>`) hasta terminar en el `<body>` buscando el primer elemento contenedor que esté posicionado.
3. El primer elemento contenedor es el `<div>`, pero su posicionamiento es el normal o estático, ya que ni siquiera tiene establecida la propiedad `position`.

4. Como el siguiente elemento contenedor es el <body>, el navegador establece directamente el origen de coordenadas en la esquina superior izquierda de la página.
5. A partir de ese origen de coordenadas, la caja se desplaza 3em a la derecha (left: 3em) y otros 3em de forma descendente (top: 3em).

Además, el párrafo que se mostraba debajo de la imagen sube y ocupa el lugar dejado por la imagen. El resultado es que el elemento <div> ahora sólo contiene el párrafo y la imagen se muestra en un nivel superior y cubre parcialmente los contenidos del párrafo.

A continuación, se posiciona de forma relativa el elemento <div> que contiene la imagen y el resto de reglas CSS no se modifican. La única propiedad añadida al <div> es position: relative por lo que el elemento contenedor se posiciona pero no se desplaza respecto de su posición original:

```
div {  
    border: 2px solid #CCC;  
    padding: 1em;  
    margin: 1em 0 1em 4em;  
    width: 300px;  
    position: relative;  
}  
  
div img {  
    position: absolute;  
    top: 3em;  
    left: 3em;  
}
```

La siguiente imagen muestra el resultado obtenido:

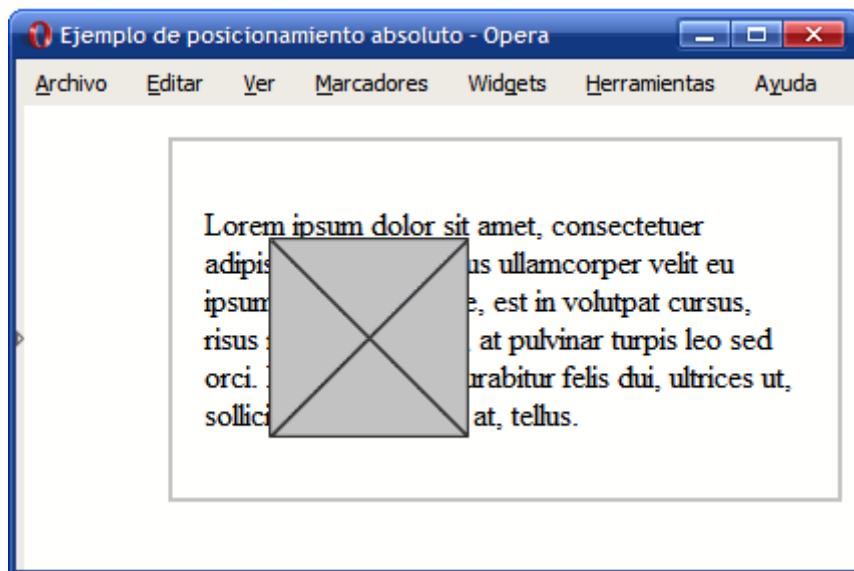


Figura 5.11. Imagen posicionada de forma absoluta

En este caso, el origen de coordenadas para determinar la nueva posición de la imagen corresponde a la esquina superior izquierda del elemento <div>:

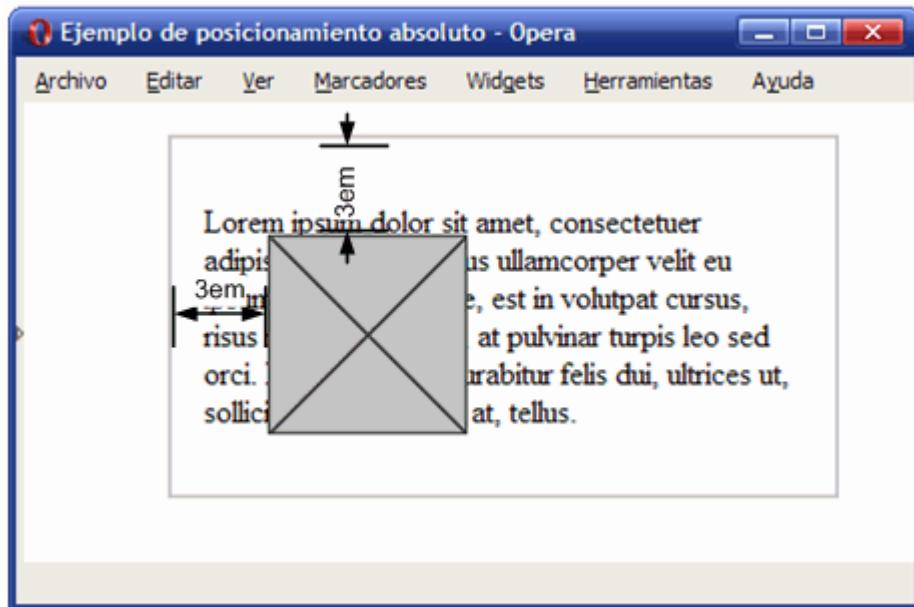


Figura 5.12. La referencia del posicionamiento absoluto es el elemento contenedor de la imagen

Si se quiere posicionar un elemento de forma absoluta respecto de su elemento contenedor, es imprescindible posicionar el elemento contenedor. Para ello, sólo es necesario añadir la propiedad `position: relative`, por lo que no es obligatorio desplazar el elemento contenedor respecto de su posición original.

5.6. Posicionamiento fijo

El estándar CSS considera que el posicionamiento fijo es un caso particular del posicionamiento absoluto, ya que sólo se diferencian en el comportamiento de las cajas posicionadas.

Cuando una caja se posiciona de forma fija, la forma de obtener el origen de coordenadas para interpretar su desplazamiento es idéntica al posicionamiento absoluto. De hecho, si el usuario no mueve la página HTML en la ventana del navegador, no existe ninguna diferencia entre estos dos modelos de posicionamiento.

La principal característica de una caja posicionada de forma fija es que su posición es inamovible dentro de la ventana del navegador. El posicionamiento fijo hace que las cajas no modifiquen su posición ni aunque el usuario suba o baje la página en la ventana de su navegador.

Si la página se visualiza en un medio paginado (por ejemplo en una impresora) las cajas posicionadas de forma fija se repiten en todas las páginas. Esta característica puede ser útil para crear encabezados o pies de página en páginas HTML preparadas para imprimir.

El posicionamiento fijo apenas se ha utilizado en el diseño de páginas web hasta hace poco tiempo porque el navegador Internet Explorer 6 y las versiones anteriores no lo soportan.

5.7. Posicionamiento flotante

El posicionamiento flotante es el más difícil de comprender pero al mismo tiempo es el más utilizado. La mayoría de estructuras de las páginas web complejas están diseñadas con el posicionamiento flotante, como se verá más adelante.

Cuando una caja se posiciona con el modelo de posicionamiento flotante, automáticamente se convierte en una *caja flotante*, lo que significa que se desplaza hasta la zona más a la izquierda o más a la derecha de la posición en la que originalmente se encontraba.

La siguiente imagen muestra el resultado de posicionar de forma flotante hacia la derecha la caja 1:

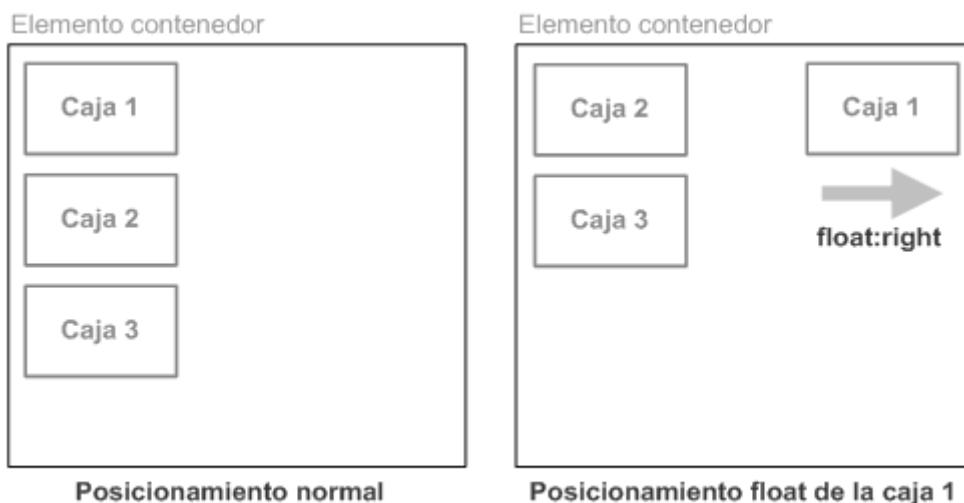


Figura 5.13. Ejemplo de posicionamiento float de una caja

Cuando se posiciona una caja de forma flotante:

- La caja deja de pertenecer al flujo normal de la página, lo que significa que el resto de cajas ocupan el lugar dejado por la caja flotante.
- La caja flotante se posiciona lo más a la izquierda o lo más a la derecha posible de la posición en la que se encontraba originalmente.

Si en el anterior ejemplo la caja 1 se posiciona de forma flotante hacia la izquierda, el resultado es el que muestra la siguiente imagen:

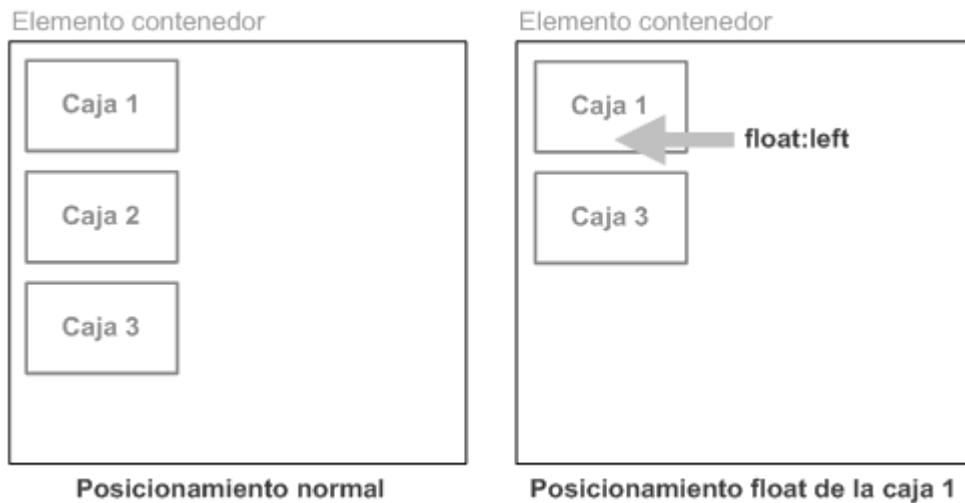


Figura 5.14. Ejemplo de posicionamiento float de una caja

La caja 1 es de tipo flotante, por lo que *desaparece del flujo normal* de la página y el resto de cajas ocupan su lugar. El resultado es que la caja 2 ahora se muestra donde estaba la caja 1 y la caja 3 se muestra donde estaba la caja 2.

Al mismo tiempo, la caja 1 se desplaza todo lo posible hacia la izquierda de la posición en la que se encontraba. El resultado es que la caja 1 se muestra encima de la nueva posición de la caja 2 y tapa todos sus contenidos.

Si existen otras cajas flotantes, al posicionar de forma flotante otra caja, se tiene en cuenta el sitio disponible. En el siguiente ejemplo se posicionan de forma flotante hacia la izquierda las tres cajas:

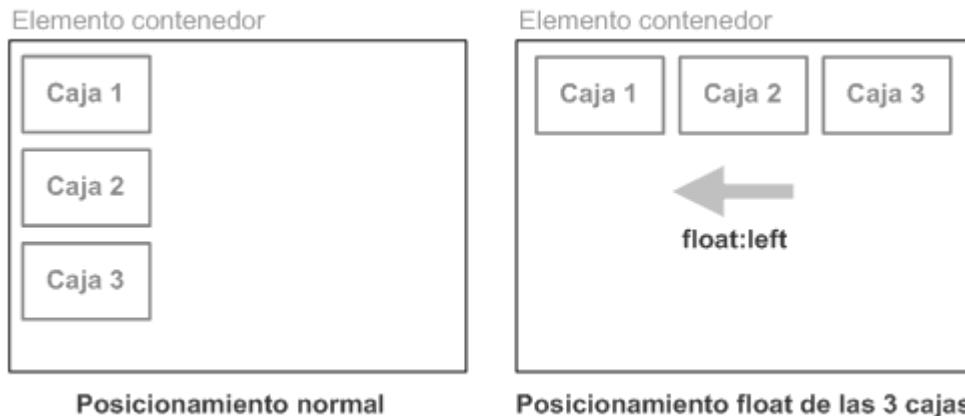


Figura 5.15. Ejemplo de posicionamiento float de varias cajas

En el ejemplo anterior, las cajas no se superponen entre sí porque las cajas flotantes tienen en cuenta las otras cajas flotantes existentes. Como la caja 1 ya estaba posicionada lo más a la izquierda posible, la caja 2 sólo puede colocarse al lado del borde derecho de la caja 1, que es el sitio más a la izquierda posible respecto de la zona en la que se encontraba.

Si no existe sitio en la línea actual, la caja flotante baja a la línea inferior hasta que encuentra el sitio necesario para mostrarse lo más a la izquierda o lo más a la derecha posible en esa nueva línea:

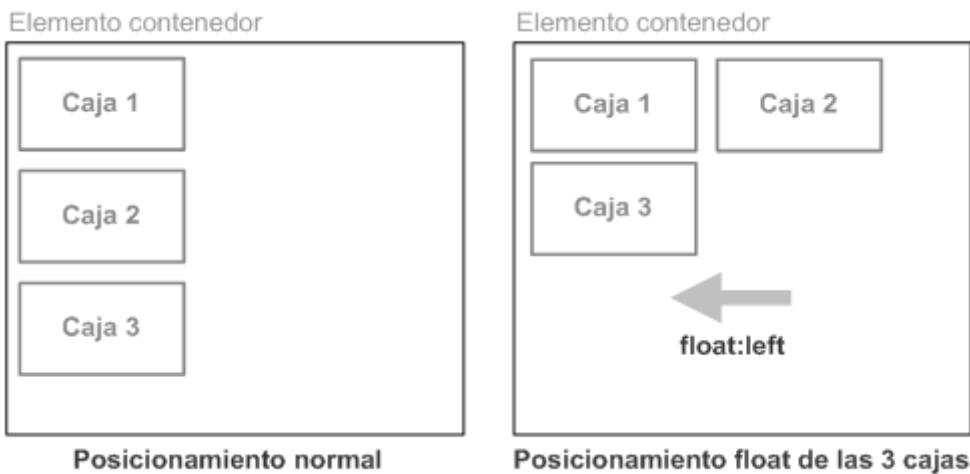


Figura 5.16. Ejemplo de posicionamiento float cuando no existe sitio suficiente

Las cajas flotantes influyen en la disposición de todas las demás cajas. Los elementos en línea *hacén sitio* a las cajas flotantes adaptando su anchura al espacio libre dejado por la caja desplazada. Los elementos de bloque no les hacen sitio, pero sí que adaptan sus contenidos para que no se solapen con las cajas flotantes.

La propiedad CSS que permite posicionar de forma flotante una caja se denomina **float**:

float	Posicionamiento float
Valores	<code>left right none inherit</code>
Se aplica a	Todos los elementos
Valor inicial	<code>none</code>
Descripción	Establece el tipo de posicionamiento flotante del elemento

Si se indica un valor `left`, la caja se desplaza hasta el punto más a la izquierda posible en esa misma línea (si no existe sitio en esa línea, la caja baja una línea y se muestra lo más a la izquierda posible en esa nueva línea). El resto de elementos adyacentes se adaptan y *fluyen* alrededor de la caja flotante.

El valor `right` tiene un funcionamiento idéntico, salvo que en este caso, la caja se desplaza hacia la derecha. El valor `none` permite anular el posicionamiento flotante de forma que el elemento se muestre en su posición original.

Ejercicio 6 Ver enunciado en la página 198

Los elementos que se encuentran alrededor de una caja flotante adaptan sus contenidos para que fluyan alrededor del elemento posicionado:

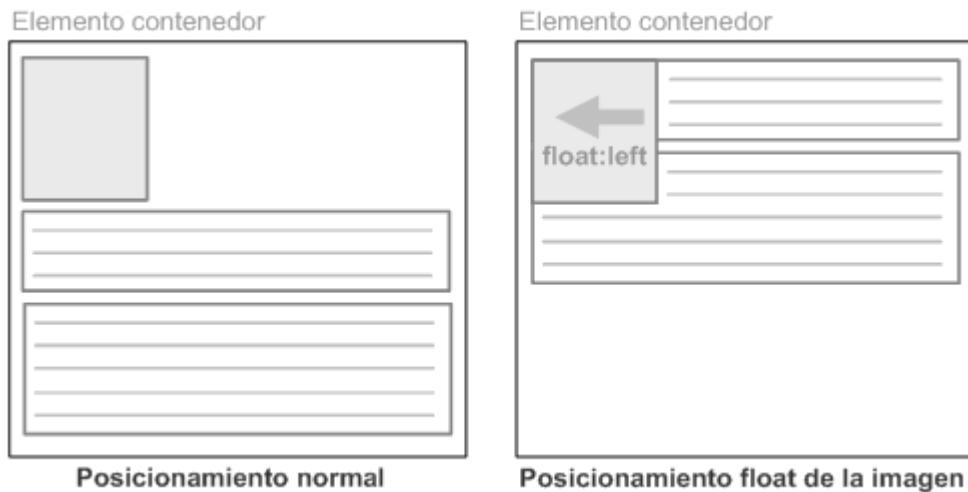


Figura 5.17. Elementos que fluyen alrededor de un elemento posicionado mediante float

La regla CSS que se aplica en la imagen del ejemplo anterior es:

```
img {
    float: left;
}
```

Uno de los principales motivos para la creación del posicionamiento `float` fue precisamente la posibilidad de colocar imágenes alrededor de las cuales fluye el texto.

CSS permite controlar la forma en la que los contenidos fluyen alrededor de los contenidos posicionados mediante `float`. De hecho, en muchas ocasiones es admisible que algunos contenidos fluyan alrededor de una imagen, pero el resto de contenidos deben mostrarse en su totalidad sin fluir alrededor de la imagen:

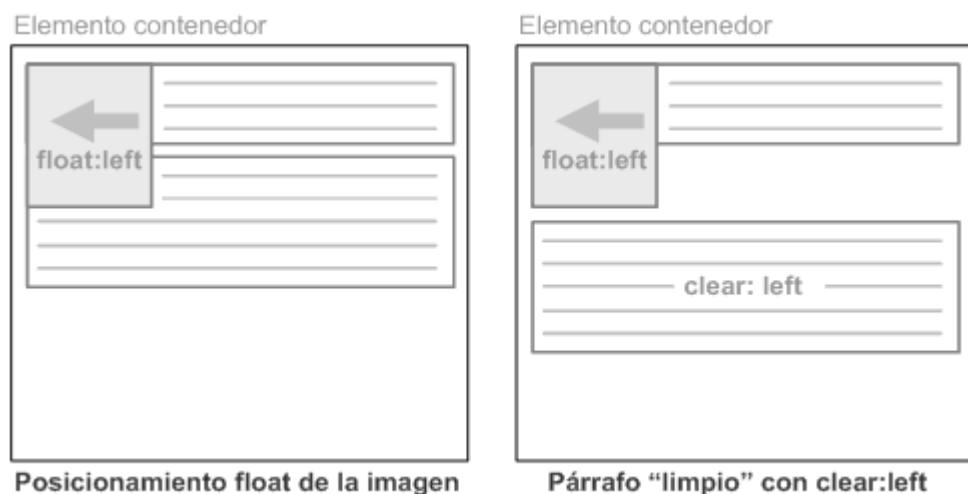


Figura 5.18. Forzando a que un elemento no fluya alrededor de otro elemento posicionado mediante float

La propiedad `clear` permite modificar el comportamiento por defecto del posicionamiento flotante para forzar a un elemento a mostrarse debajo de cualquier caja flotante. La regla CSS que se aplica al segundo párrafo del ejemplo anterior es la siguiente:

```
| <p style="clear: left;">...</p>
```

La definición formal de la propiedad `clear` se muestra a continuación:

clear	Despejar los elementos flotantes adyacentes
Valores	none left right both inherit
Se aplica a	Todos los elementos de bloque
Valor inicial	none
Descripción	Indica el lado del elemento que no debe ser adyacente a ninguna caja flotante

La propiedad `clear` indica el lado del elemento HTML que no debe ser adyacente a ninguna caja posicionada de forma flotante. Si se indica el valor `left`, el elemento se desplaza de forma descendente hasta que pueda colocarse en una línea en la que no haya ninguna caja flotante en el lado izquierdo.

La especificación oficial de CSS explica este comportamiento como "*un desplazamiento descendente hasta que el borde superior del elemento esté por debajo del borde inferior de cualquier elemento flotante hacia la izquierda*".

Si se indica el valor `right`, el comportamiento es análogo, salvo que en este caso se tienen en cuenta los elementos desplazados hacia la derecha.

El valor `both` despeja los lados izquierdo y derecho del elemento, ya que desplaza el elemento de forma descendente hasta que el borde superior se encuentre por debajo del borde inferior de cualquier elemento flotante hacia la izquierda o hacia la derecha.

Como se verá más adelante, la propiedad `clear` es imprescindible cuando se crean las estructuras de las páginas web complejas.

En el ejercicio anterior, se utiliza la propiedad `float` para posicionar de forma flotante los dos elementos:

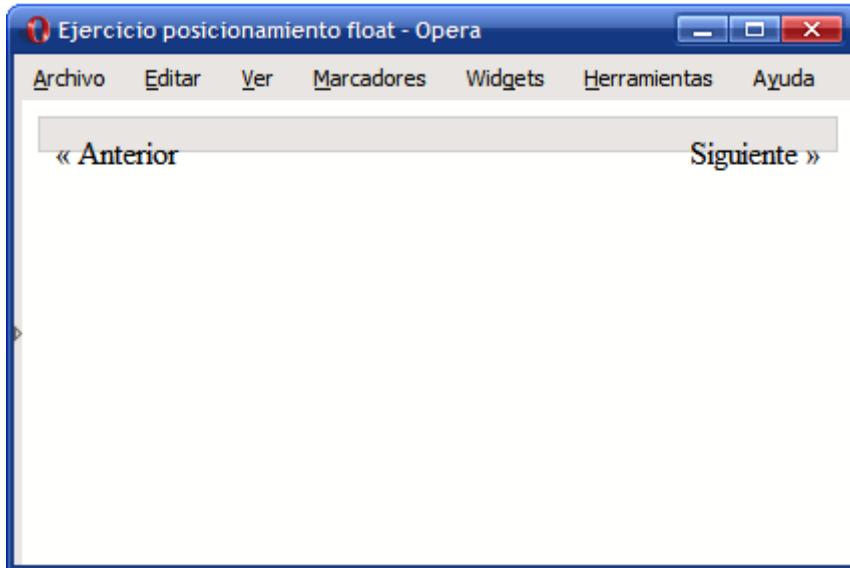


Figura 5.19. Visualización incorrecta de dos elementos posicionados mediante `float`

Como los dos elementos `` creados dentro del elemento `<div>` se han posicionado mediante `float`, los dos han salido del flujo normal del documento. Así, el elemento `<div>` no tiene contenidos y por eso no llega a cubrir el texto de los dos elementos ``:



Los elementos posicionados vacían de contenidos al <div>



El <div> añadido “limpia” el float y fuerza la altura del <div> original

Figura 5.20. Esquema del problema y solución de la visualización incorrecta de dos elementos posicionados mediante float

La solución consiste en añadir un elemento adicional invisible que *limpie* el `float` forzando a que el `<div>` original cubra completamente los dos elementos ``. El código HTML y CSS final se muestra a continuación:

```

<!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">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
<title>Ejercicio posicionamiento float</title>
<style type="text/css">
div#paginacion {
    border: 1px solid #CCC;
    background-color: #E0E0E0;
    padding: .5em;
}

.derecha { float: right; }
.izquierda { float: left; }

div.clear { clear: both; }
</style>
</head>

<body>
<div id="paginacion">
    <span class="izquierda">&laquo; Anterior</span>
    <span class="derecha">Siguiente &raquo;</span>
    <div class="clear"></div>
</div>
</body>
</html>

```

Al añadir un `<div>` con la propiedad `clear: both`, se tiene la seguridad de que el `<div>` añadido se va a mostrar debajo de cualquier elemento posicionado con `float` y por tanto, se asegura que el `<div>` original tenga la altura necesaria como para encerrar a todos sus contenidos posicionados con `float`.

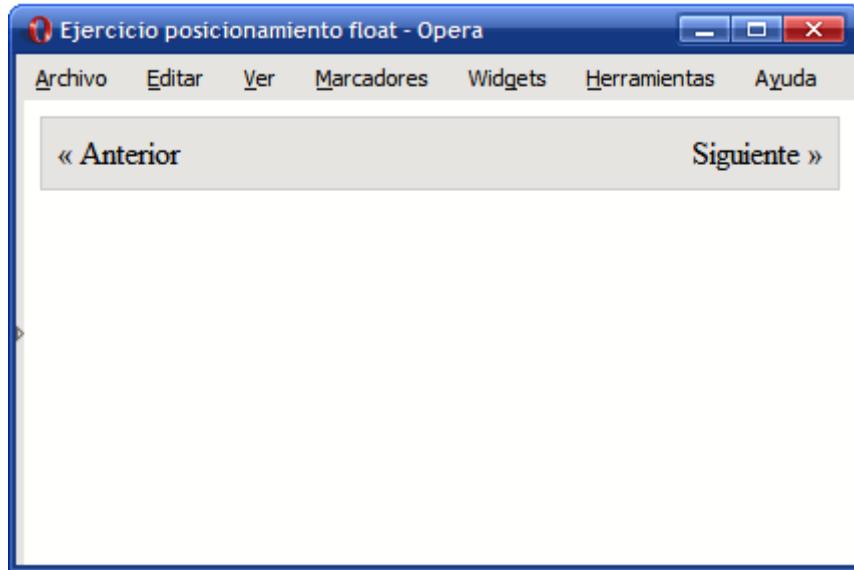


Figura 5.21. Visualización correcta de dos elementos posicionados mediante float

Además de un elemento `<div>` invisible, también se puede utilizar un `<p>` invisible o un `<hr/>` invisible.

5.8. Visualización

Además de las propiedades que controlan el posicionamiento de los elementos, CSS define otras cuatro propiedades para controlar su visualización: `display`, `visibility`, `overflow` y `z-index`.

Utilizando algunas de estas propiedades es posible ocultar y/o hacer invisibles las cajas de los elementos, por lo que son imprescindibles para realizar efectos avanzados y animaciones.

5.8.1. Propiedades `display` y `visibility`

Las propiedades `display` y `visibility` controlan la visualización de los elementos. Las dos propiedades permiten ocultar un elemento de la página. Habitualmente se utilizan junto con JavaScript para crear efectos y aplicaciones dinámicas (mostrar y ocultar determinados textos, hacer aparecer imágenes, etc.)

La siguiente imagen muestra las diferencias entre la propiedad `display` y la propiedad `visibility` al ocultar una caja (la número 5):

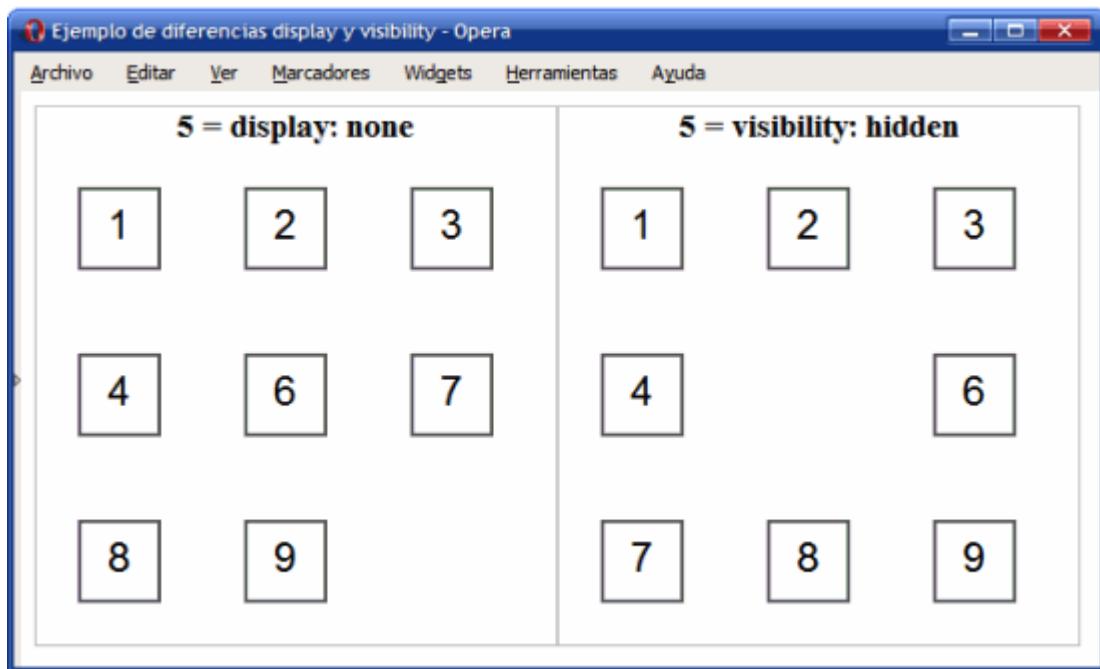


Figura 5.22. Diferencias visuales entre las propiedades display y visibility

La propiedad `display` permite ocultar un elemento haciendo que desaparezca de la página. Como el elemento oculto no existe, el resto de elementos de la página se muestran como si no existiera y ocupan el hueco dejado por la caja oculta.

Por otra parte, la propiedad `visibility` permite hacer invisible un elemento creando su caja pero no mostrando sus contenidos. En este caso, la posición del resto de elementos sí que se mantiene igual que si el elemento no fuera invisible.

En general, cuando se oculta un elemento no es deseable que siga ocupando sitio en la página, por lo que la propiedad `display` se utiliza mucho más que la propiedad `visibility`.

A continuación se muestra la definición completa de la propiedad `display`:

display	Visualización de un elemento
Valores	inline block none list-item run-in inline-block table inline-table table-row-group table-header-group table-footer-group table-row table-column-group table-column table-cell table-caption inherit
Se aplica a	Todos los elementos
Valor inicial	inline
Descripción	Permite controlar la forma de visualizar un elemento e incluso ocultarlo

Las posibilidades de la propiedad `display` son mucho mayores que ocultar un elemento. En realidad, la propiedad `display` modifica la forma en la que se visualiza un elemento.

Los valores más utilizados son `inline`, `block` y `none`. El valor `block` permite mostrar un elemento como si fuera un elemento de bloque, independientemente del tipo de elemento que se

trate. El valor `inline` permite visualizar un elemento en forma de elemento en línea, independientemente del tipo de elemento que se trate.

El valor `none` permite ocultar un elemento y hacer que desaparezca de la página. El resto de elementos se visualizan como si no existiera el elemento oculto, es decir, pueden ocupar el espacio en el que se debería visualizar el elemento.

El siguiente ejemplo muestra el uso de la propiedad `display` para mostrar un elemento de bloque como si fuera un elemento en línea y para mostrar un elemento en línea como si fuera un elemento de bloque:

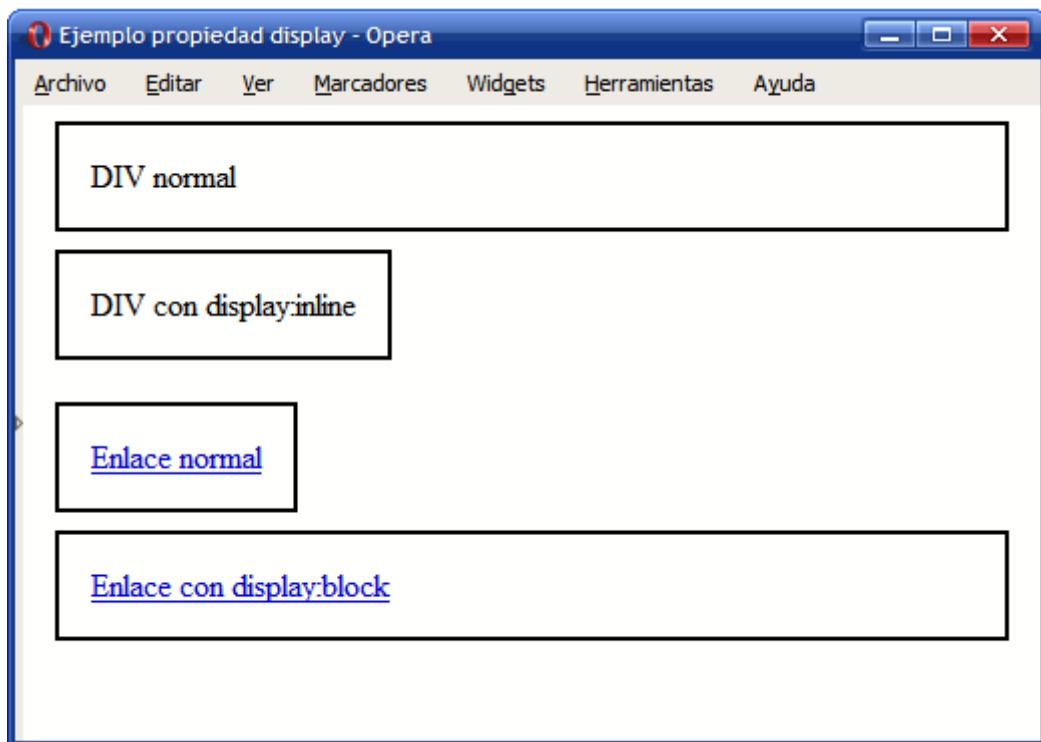


Figura 5.23. Ejemplo de propiedad display

Las reglas CSS del ejemplo anterior son las siguientes:

```
<div>DIV normal</div>
<div style="display:inline">DIV con display:inline</div>

<a href="#">Enlace normal</a>
<a href="#" style="display:block">Enlace con display:block</a>
```

Como se verá más adelante, la propiedad `display:inline` se utiliza habitualmente con las listas (``, ``) que se quieren mostrar horizontalmente y la propiedad `display:block` se emplea frecuentemente para los enlaces que forman el menú de navegación.

Por otra parte, la definición completa de la propiedad `visibility` es mucho más sencilla:

visibility	Visibilidad de un elemento
Valores	<code>visible</code> <code>hidden</code> <code>collapse</code> <code>inherit</code>
Se aplica a	Todos los elementos
Valor inicial	<code>visible</code>
Descripción	Permite hacer visibles e invisibles a los elementos

Las posibilidades de la propiedad `visibility` son mucho más limitadas que las de la propiedad `display`, ya que sólo permite hacer visibles o invisibles a los elementos de la página.

Por defecto, todas las cajas que componen la página son visibles. Haciendo uso del valor `hidden` es posible hacer que una caja sea invisible y que no muestre sus contenidos. El resto de elementos se muestran como si la caja fuera visible, por lo que el lugar donde originalmente se mostraba la caja, ahora se muestra un hueco vacío.

Por último, el valor `collapse` de la propiedad `visibility` sólo se puede utilizar en las filas, grupos de filas, columnas y grupos de columnas de una tabla. Su efecto es similar al de la propiedad `display`, ya que oculta completamente la fila y/o columna y se pueden mostrar otros contenidos en ese lugar. Si se utiliza el valor `collapse` sobre cualquier otro tipo de elemento, su efecto es idéntico al valor `hidden`.

5.8.2. Relación entre `display`, `float` y `position`

Cuando se establecen las propiedades `display`, `float` y `position` sobre una misma caja, su interpretación es la siguiente:

1. Si `display` vale `none`, se ignoran las propiedades `float` y `position` y la caja no se muestra en la página.
2. Si `position` vale `absolute` o `fixed`, la caja se posiciona de forma absoluta, se considera que `float` vale `none` y la propiedad `display` vale `block` tanto para los elementos en línea como para los elementos de bloque. La posición de la caja se determina mediante el valor de las propiedades `top`, `right`, `bottom` y `left`.
3. En cualquier otro caso, si `float` tiene un valor distinto de `none`, la caja se posiciona de forma flotante y la propiedad `display` vale `block` tanto para los elementos en línea como para los elementos de bloque.

5.8.3. Propiedad `overflow`

Normalmente, los contenidos de un elemento se pueden mostrar en el espacio reservado para ese elemento. Sin embargo, en algunas ocasiones el contenido de un elemento no cabe en el espacio reservado para ese elemento y se desborda.

La situación más habitual en la que el contenido sobresale de su espacio reservado es cuando se establece la anchura y/o altura de un elemento mediante la propiedad `width` y/o `height`. Otra situación habitual es la de las líneas muy largas contenidas dentro de un elemento `<pre>`, que hacen que la página entera sea demasiado ancha.

CSS define la propiedad `overflow` para controlar la forma en la que se visualizan los contenidos que sobresalen de sus elementos.

overflow	Parte sobrante de un elemento
Valores	<code>visible</code> <code>hidden</code> <code>scroll</code> <code>auto</code> <code>inherit</code>
Se aplica a	Elementos de bloque y celdas de tablas
Valor inicial	<code>visible</code>
Descripción	Permite controlar los contenidos sobrantes de un elemento

Los valores de la propiedad `overflow` tienen el siguiente significado:

- `visible`: el contenido no se corta y se muestra sobresaliendo la zona reservada para visualizar el elemento. Este es el comportamiento por defecto.
- `hidden`: el contenido sobrante se oculta y sólo se visualiza la parte del contenido que cabe dentro de la zona reservada para el elemento.
- `scroll`: solamente se visualiza el contenido que cabe dentro de la zona reservada para el elemento, pero también se muestran barras de `scroll` que permiten visualizar el resto del contenido.
- `auto`: el comportamiento depende del navegador, aunque normalmente es el mismo que la propiedad `scroll`.

La siguiente imagen muestra un ejemplo de los tres valores típicos de la propiedad `overflow`:

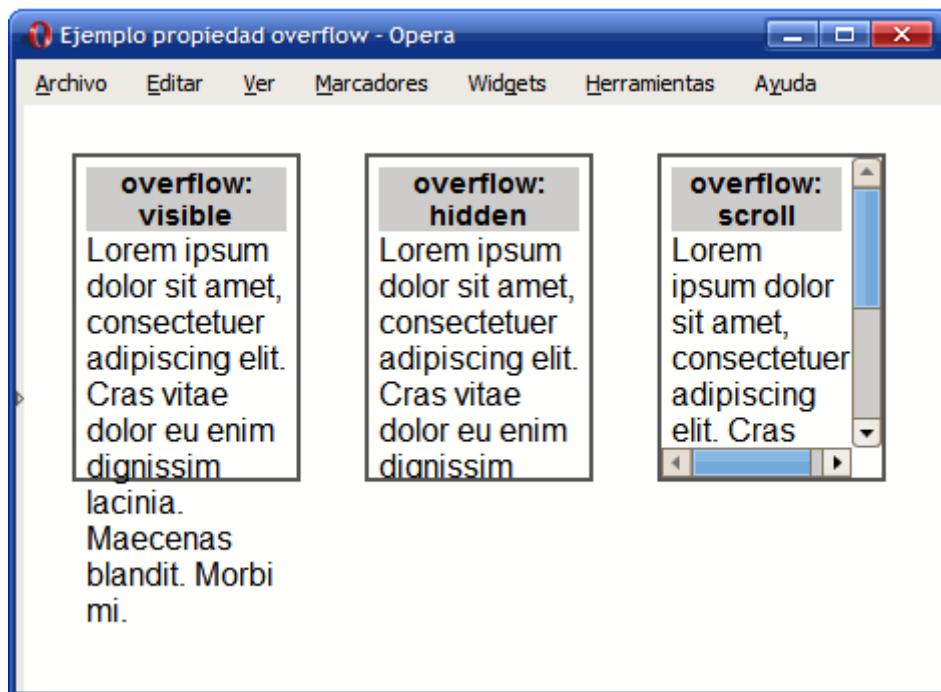


Figura 5.24. Ejemplo de propiedad `overflow`

El código HTML y CSS del ejemplo anterior se muestran a continuación:

```

div {
    display: inline;
    float: left;
    margin: 1em;
    padding: .3em;
    border: 2px solid #555;
    width: 100px;
    height: 150px;
    font: 1em Arial, Helvetica, sans-serif;
}

<div><h1>overflow: visible</h1> Lorem ipsum dolor sit amet, consectetur
adipiscing elit. Cras vitae dolor eu enim dignissim lacinia. Maecenas
blandit. Morbi mi.</div>

<div style="overflow:hidden"><h1>overflow: hidden</h1> Lorem ipsum dolor
sit amet, consectetur adipiscing elit. Cras vitae dolor eu enim dignissim
lacinia. Maecenas blandit. Morbi mi.</div>

<div style="overflow:scroll"><h1>overflow: scroll</h1> Lorem ipsum dolor sit
amet, consectetur adipiscing elit. Cras vitae dolor eu enim dignissim lacinia.
Maecenas blandit. Morbi mi.</div>

```

5.8.4. Propiedad z-index

Además de posicionar una caja de forma horizontal y vertical, CSS permite controlar la posición tridimensional de las cajas posicionadas. De esta forma, es posible controlar las cajas que se muestran delante o detrás de otras cajas cuando se producen solapamientos.

La posición tridimensional se establece sobre un tercer eje llamado Z y se controla mediante la propiedad z-index. Utilizando esta propiedad es posible crear páginas complejas con varios niveles o capas.

A continuación se muestra la definición formal de la propiedad z-index:

z-index	Orden tridimensional
Valores	auto <numero> inherit
Se aplica a	Elementos que han sido posicionados explícitamente
Valor inicial	auto
Descripción	Establece el nivel tridimensional en el que se muestra el elemento

El valor más común de la propiedad z-index es un número entero. Aunque la especificación oficial permite los números negativos, en general se considera el número 0 como el nivel más bajo.

Cuanto más alto sea el valor numérico, más cerca del usuario se muestra la caja. Un elemento con z-index: 10 se muestra por encima de los elementos con z-index: 8 o z-index: 9, pero por debajo de elementos con z-index: 20 o z-index: 50.

La siguiente imagen muestra un ejemplo de uso de la propiedad z-index:

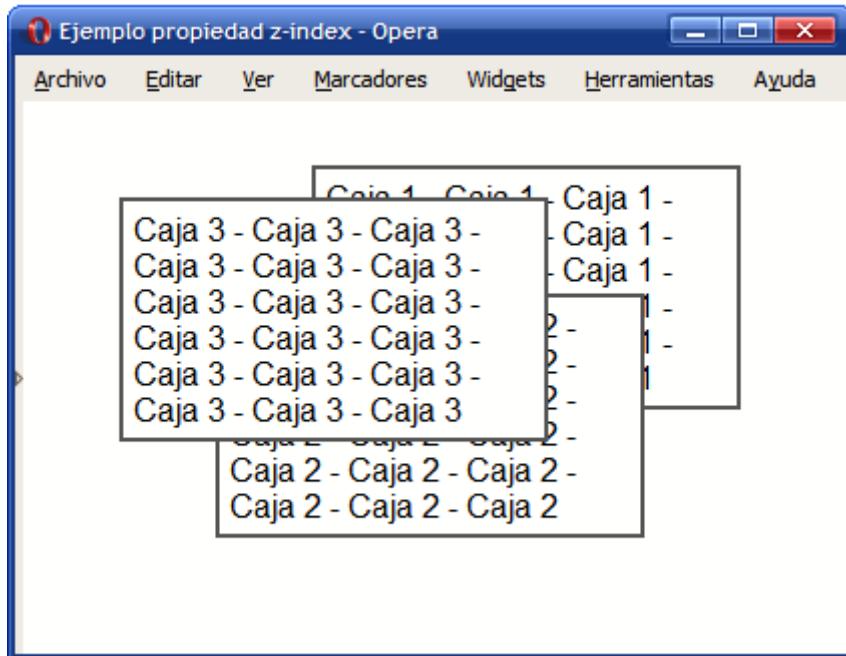


Figura 5.25. Ejemplo de propiedad z-index

El código HTML y CSS del ejemplo anterior es el siguiente:

Capítulo 6. Texto

6.1. Tipografía

CSS define numerosas propiedades para modificar la apariencia del texto. A pesar de que no dispone de tantas posibilidades como los lenguajes y programas específicos para crear documentos impresos, CSS permite aplicar estilos complejos y muy variados al texto de las páginas web.

La propiedad básica que define CSS relacionada con la tipografía se denomina `color` y se utiliza para establecer el color de la letra.

color	Color del texto
Valores	<code><color> inherit</code>
Se aplica a	Todos los elementos
Valor inicial	Depende del navegador
Descripción	Establece el color de letra utilizado para el texto

Aunque el color por defecto del texto depende del navegador, todos los navegadores principales utilizan el color negro. Para establecer el color de letra de un texto, se puede utilizar cualquiera de las cinco formas que incluye CSS para definir un color.

A continuación se muestran varias reglas CSS que establecen el color del texto de diferentes formas:

```

h1 { color: #369; }
p { color: black; }
a, span { color: #B1251E; }
div { color: rgb(71, 98, 176); }
```

Como el valor de la propiedad `color` se hereda, normalmente se establece la propiedad `color` en el elemento `body` para establecer el color de letra de todos los elementos de la página:

```
body { color: #777; }
```

En el ejemplo anterior, todos los elementos de la página muestran el mismo color de letra salvo que establezcan de forma explícita otro color. La única excepción de este comportamiento son los enlaces que se crean con la etiqueta `<a>`. Aunque el color de la letra se hereda de los elementos padre a los elementos hijo, con los enlaces no sucede lo mismo, por lo que es necesario indicar su color de forma explícita:

```

/* Establece el mismo color a todos los elementos de
la página salvo los enlaces */
body { color: #777; }

/* Establece el mismo color a todos los elementos de
```

```
La página, incluyendo Los enlaces */
body, a { color: #777; }
```

La otra propiedad básica que define CSS relacionada con la tipografía se denomina **font-family** y se utiliza para indicar el tipo de letra con el que se muestra el texto.

font-family	Tipo de letra
Valores	((<nombre_familia> <familia_generica>) (,<nombre_familia> <familia_generica>)*) inherit
Se aplica a	Todos los elementos
Valor inicial	Depende del navegador
Descripción	Establece el tipo de letra utilizado para el texto

El tipo de letra del texto se puede indicar de dos formas diferentes:

- Mediante el nombre de una *familia* tipográfica: en otras palabras, mediante el nombre del tipo de letra, como por ejemplo "Arial", "Verdana", "Garamond", etc.
- Mediante el nombre genérico de una *familia* tipográfica: los nombres genéricos no se refieren a ninguna fuente en concreto, sino que hacen referencia al estilo del tipo de letra. Las familias genéricas definidas son **serif** (tipo de letra similar a *Times New Roman*), **sans-serif** (tipo *Arial*), **cursive** (tipo *Comic Sans*), **fantasy** (tipo *Impact*) y **monospace** (tipo *Courier New*).

Los navegadores muestran el texto de las páginas web utilizando los tipos de letra instalados en el ordenador del propio usuario. De esta forma, si el diseñador indica en la propiedad **font-family** que el texto debe mostrarse con un tipo de letra especialmente raro o rebuscado, casi ningún usuario dispondrá de ese tipo de letra.

Para evitar el problema común de que el usuario no tenga instalada la fuente que quiere utilizar el diseñador, CSS permite indicar en la propiedad **font-family** más de un tipo de letra. El navegador probará en primer lugar con el primer tipo de letra indicado. Si el usuario la tiene instalada, el texto se muestra con ese tipo de letra.

Si el usuario no dispone del primer tipo de letra indicado, el navegador irá probando con el resto de tipos de letra hasta que encuentre alguna fuente que esté instalada en el ordenador del usuario. Evidentemente, el diseñador no puede indicar para cada propiedad **font-family** tantos tipos de letra como posibles fuentes parecidas existan.

Para solucionar este problema se utilizan las familias tipográficas genéricas. Cuando la propiedad **font-family** toma un valor igual a **sans-serif**, el diseñador no indica al navegador que debe utilizar la fuente Arial, sino que debe utilizar "*la fuente que más se parezca a Arial de todas las que tiene instaladas el usuario*".

Por todo ello, el valor de **font-family** suele definirse como una lista de tipos de letra alternativos separados por comas. El último valor de la lista es el nombre de la familia tipográfica genérica que más se parece al tipo de letra que se quiere utilizar.

Las listas de tipos de letra más utilizadas son las siguientes:

```
font-family: Arial, Helvetica, sans-serif;
font-family: "Times New Roman", Times, serif;
font-family: "Courier New", Courier, monospace;
font-family: Georgia, "Times New Roman", Times, serif;
font-family: Verdana, Arial, Helvetica, sans-serif;
```

Ya que las fuentes que se utilizan en la página deben estar instaladas en el ordenador del usuario, cuando se quiere disponer de un diseño complejo con fuentes muy especiales, se debe recurrir a soluciones alternativas.

La solución más sencilla consiste en crear imágenes en las que se muestra el texto con la fuente deseada. Esta técnica solamente es viable para textos cortos (por ejemplo los titulares de una página) y puede ser manual (creando las imágenes una por una) o automática, utilizando JavaScript, PHP y/o CSS.

Otra alternativa es la de la sustitución automática de texto basada en Flash. La técnica más conocida es la de sIFR, de la que se puede encontrar más información en <http://wiki.novemberborn.net/sifr>

Una vez seleccionado el tipo de letra, se puede modificar su tamaño mediante la propiedad **font-size**.

font-size	Tamaño de letra
Valores	<tamaño_absoluto> <tamaño_relativo> <medida> <porcentaje> inherit
Se aplica a	Todos los elementos
Valor inicial	medium
Descripción	Establece el tamaño de letra utilizado para el texto

Además de todas las unidades de medida relativas y absolutas y el uso de porcentajes, CSS permite utilizar una serie de palabras clave para indicar el tamaño de letra del texto:

- **tamaño_absoluto**: indica el tamaño de letra de forma absoluta mediante alguna de las siguientes palabras clave: `xx-small`, `x-small`, `small`, `medium`, `large`, `x-large`, `xx-large`.
- **tamaño_relativo**: indica de forma relativa el tamaño de letra del texto mediante dos palabras clave (`larger`, `smaller`) que toman como referencia el tamaño de letra del elemento padre.

La siguiente imagen muestra una comparación entre los tamaños típicos del texto y las unidades que más se utilizan:

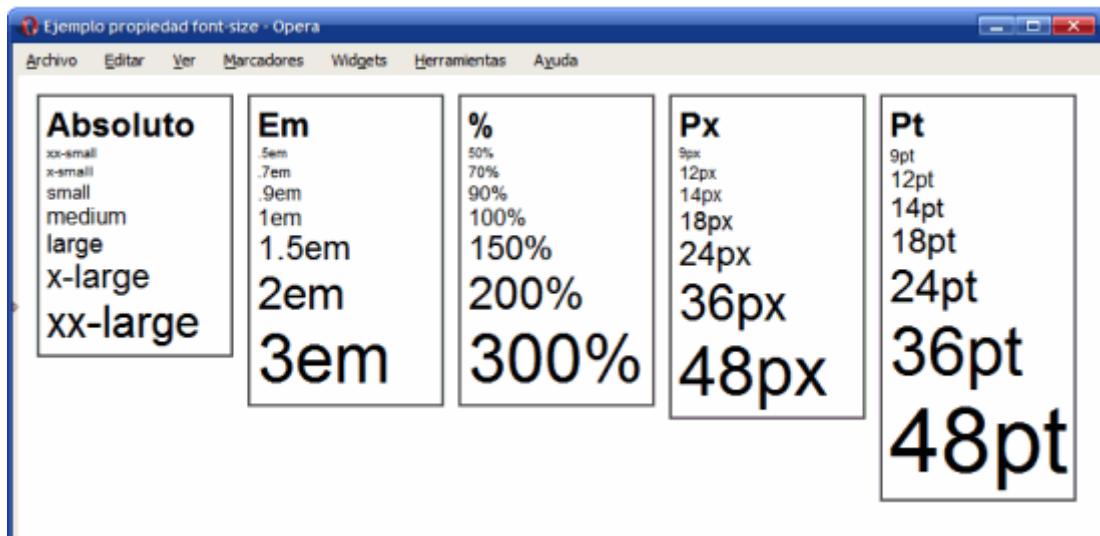


Figura 6.1. Comparación visual de las distintas unidades para indicar el tamaño del texto

CSS recomienda indicar el tamaño del texto en la unidad `em` o en porcentaje (%). Además, es habitual indicar el tamaño del texto en puntos (pt) cuando el documento está específicamente diseñado para imprimirla.

Por defecto los navegadores asignan los siguientes tamaños a los títulos de sección: `<h1> = xx-large`, `<h2> = x-large`, `<h3> = large`, `<h4> = medium`, `<h5> = small`, `<h6> = xx-small`.

Una vez indicado el tipo y el tamaño de letra, es habitual modificar otras características como su anchura (texto en negrita) y su estilo (texto en cursiva). La propiedad que controla la anchura de la letra es `font-weight`.

font-weight	Anchura de la letra
Valores	normal bold bolder lighter 100 200 300 400 500 600 700 800 900 inherit
Se aplica a	Todos los elementos
Valor inicial	normal
Descripción	Establece la anchura de la letra utilizada para el texto

Los valores que normalmente se utilizan son `normal` (el valor por defecto) y `bold` para los textos en negrita. El valor `normal` equivale al valor numérico `400` y el valor `bold` al valor numérico `700`.

El siguiente ejemplo muestra una aplicación práctica de la propiedad `font-weight`:

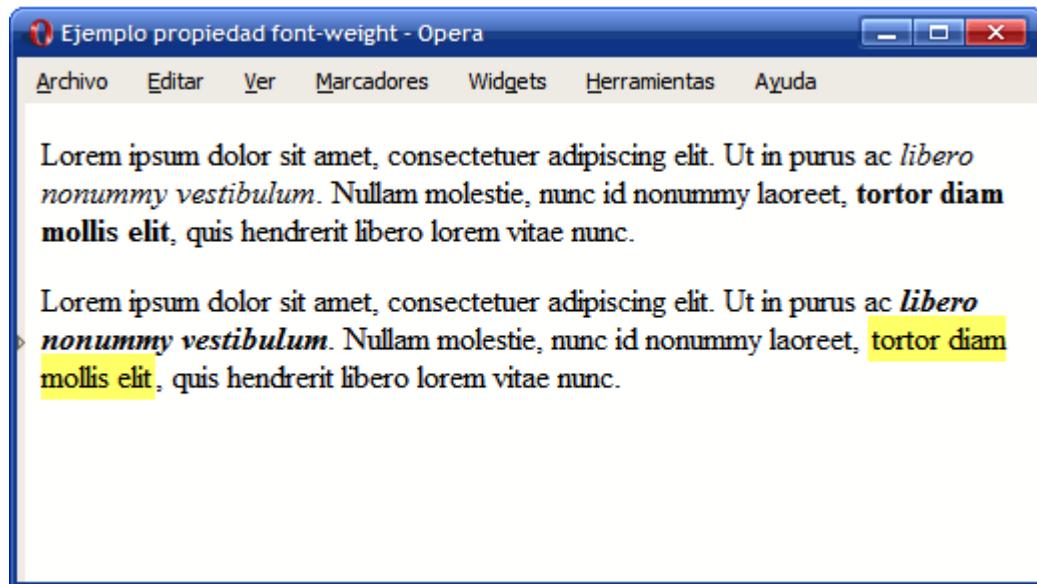


Figura 6.2. Ejemplo de propiedad font-weight

Por defecto, los navegadores muestran el texto de los elementos `` en cursiva y el texto de los elementos `` en negrita. La propiedad `font-weight` permite alterar ese aspecto por defecto y mostrar por ejemplo los elementos `` como cursiva y negrita y los elementos `` destacados mediante un color de fondo y sin negrita.

Las reglas CSS del ejemplo anterior se muestran a continuación:

```
#especial em {  
    font-weight: bold;  
}  
#especial strong {  
    font-weight: normal;  
    background-color: #FFFF66;  
    padding: 2px;  
}  
  
<p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Ut in  
purus ac <em>libero nonummy vestibulum</em>. Nullam molestie, nunc id  
nonummy laoreet, <strong>tortor diam mollis elit</strong>, quis hendrerit  
libero lorem vitae nunc.</p>  
  
<p id="especial">Lorem ipsum dolor sit amet, consectetur adipiscing elit.  
Ut in purus ac <em>libero nonummy vestibulum</em>. Nullam molestie, nunc id  
nonummy laoreet, <strong>tortor diam mollis elit</strong>, quis hendrerit  
libero lorem vitae nunc.</p>
```

Además de la anchura de la letra, CSS permite variar su estilo mediante la propiedad `font-style`.

font-style	Estilo de la letra
Valores	normal italic oblique inherit
Se aplica a	Todos los elementos
Valor inicial	normal
Descripción	Establece el estilo de la letra utilizada para el texto

Normalmente la propiedad `font-style` se emplea para mostrar un texto en cursiva mediante el valor `italic`.

El ejemplo anterior se puede modificar para personalizar aun más el aspecto por defecto de los elementos `` y ``:

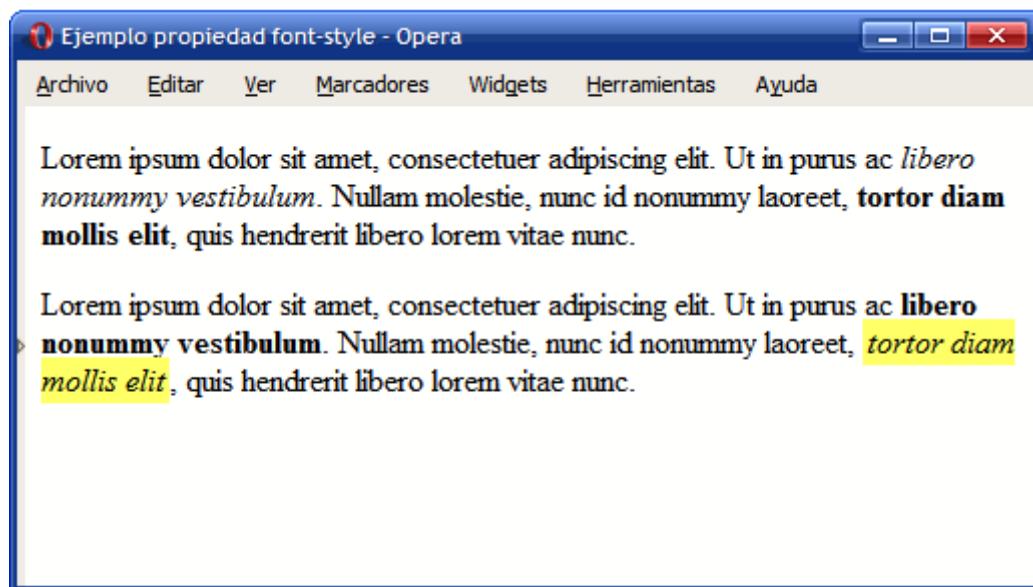


Figura 6.3. Ejemplo de propiedad `font-style`

Ahora, el texto del elemento `` se muestra como un texto en negrita y el texto del elemento `` se muestra como un texto en cursiva con un color de fondo muy destacado.

El único cambio necesario en las reglas CSS anteriores es el de añadir la propiedad `font-style`:

```
#especial em {
    font-weight: bold;
    font-style: normal;
}
#especial strong {
    font-weight: normal;
    font-style: italic;
    background-color:#FFFF66;
    padding: 2px;
}
```

Por último, CSS permite otra variación en el estilo del tipo de letra, controlado mediante la propiedad `font-variant`.

font-variant	Estilo alternativo de la letra
Valores	normal small-caps inherit
Se aplica a	Todos los elementos
Valor inicial	normal
Descripción	Establece el estilo alternativo de la letra utilizada para el texto

La propiedad `font-variant` no se suele emplear habitualmente, ya que sólo permite mostrar el texto con *letra versal* (mayúsculas pequeñas).

Siguiendo con el ejemplo anterior, se ha aplicado la propiedad `font-variant: small-caps` al segundo párrafo de texto:

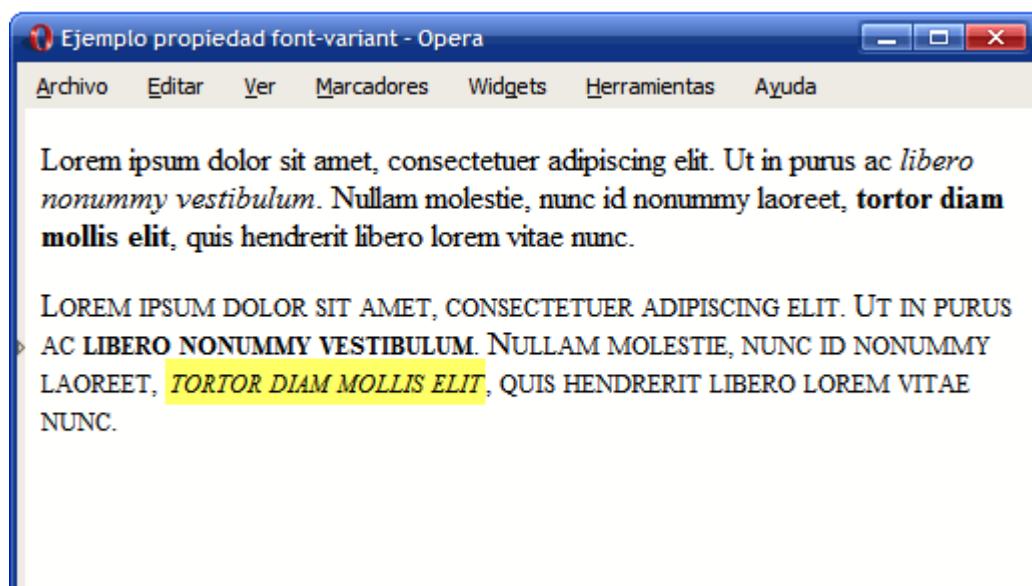


Figura 6.4. Ejemplo de propiedad `font-variant`

Para este último ejemplo, solamente es necesario añadir una regla a los estilos CSS:

```
#especial {
    font-variant: small-caps;
}
```

Por otra parte, CSS proporciona una propiedad tipo "shorthand" denominada `font` y que permite indicar de forma directa algunas o todas las propiedades de la tipografía de un texto.

font	Tipografía
Valores	((<font-style> <font-variant> <font-weight>)? <font-size> (/ <line-height>)? <font-family>) caption icon menu message-box small-caption status-bar inherit
Se aplica a	Todos los elementos
Valor inicial	-
Descripción	Permite indicar de forma directa todas las propiedades de la tipografía de un texto

El orden en el que se deben indicar las propiedades del texto es el siguiente:

- En primer lugar y de forma opcional se indican el `font-style`, `font-variant` y `font-weight` en cualquier orden.
- A continuación, se indica obligatoriamente el valor de `font-size` seguido opcionalmente por el valor de `line-height`.
- Por último, se indica obligatoriamente el tipo de letra a utilizar.

Ejemplos de uso de la propiedad `font`:

```
font: 76%/140% Verdana,Arial,Helvetica,sans-serif;
font: normal 24px/26px "Century Gothic","Trebuchet MS",Arial,Helvetica,sans-serif;
font: normal .94em "Trebuchet MS",Arial,Helvetica,sans-serif;
font: bold 1em "Trebuchet MS",Arial,Sans-Serif;
font: normal 0.9em "Lucida Grande", Verdana, Arial, Helvetica, sans-serif;
font: normal 1.2em/1em helvetica, arial, sans-serif;
font: 11px verdana, sans-serif;
font: normal 1.4em/1.6em "helvetica", arial, sans-serif;
font: bold 14px georgia, times, serif;
```

Aunque su uso no es muy común, la propiedad `font` también permite indicar el tipo de letra a utilizar mediante una serie de palabras clave: `caption`, `icon`, `menu`, `message-box`, `small-caption`, `status-bar`.

Si por ejemplo se utiliza la palabra `status-bar`, el navegador muestra el texto con el mismo tipo de letra que la que utiliza el sistema operativo para mostrar los textos de la barra de estado de las ventanas. La palabra `icon` se puede utilizar para mostrar el texto con el mismo tipo de letra que utiliza el sistema operativo para mostrar el nombre de los iconos y así sucesivamente.

Ejercicio 7 Ver enunciado en la página 199

6.2. Texto

Además de las propiedades relativas a la tipografía del texto, CSS define numerosas propiedades que determinan la apariencia del texto en su conjunto. Estas propiedades adicionales permiten controlar al alineación del texto, el interlineado, la separación entre palabras, etc.

La propiedad que define la alineación del texto se denomina `text-align`.

text-align	Alineación del texto
Valores	<code>left</code> <code>right</code> <code>center</code> <code>justify</code> <code>inherit</code>
Se aplica a	Elementos de bloque y celdas de tabla
Valor inicial	<code>left</code>
Descripción	Establece la alineación del contenido del elemento

Los valores definidos por CSS permiten alinear el texto según los valores tradicionales: a la izquierda (`left`), a la derecha (`right`), centrado (`center`) y justificado (`justify`).

La siguiente imagen muestra el efecto de establecer el valor `left`, `right`, `center` y `justify` respectivamente a cada uno de los párrafos de la página.



Figura 6.5. Ejemplo de propiedad `text-align`

La propiedad `text-align` no sólo alinea el texto que contiene un elemento, sino que también alinea todos sus contenidos, como por ejemplo las imágenes.

El interlineado de un texto se controla mediante la propiedad `line-height`, que permite controlar la altura ocupada por cada línea de texto:

line-height	Interlineado
Valores	<code>normal</code> <code><numero></code> <code><medida></code> <code><porcentaje></code> <code>inherit</code>
Se aplica a	Todos los elementos
Valor inicial	<code>normal</code>
Descripción	Permite establecer la altura de línea de los elementos

Además de todas las unidades de medida y el uso de porcentajes, la propiedad `line-height` permite indicar un número sin unidades que se interpreta como el múltiplo del tamaño normal de la letra. Por tanto, estas tres reglas CSS son equivalentes:

```
p { line-height: 1.2; font-size: 1em }
p { line-height: 1.2em; font-size: 1em }
p { line-height: 120%; font-size: 1em }
```

Siempre que se utilice de forma moderada, el interlineado mejora notablemente la legibilidad de un texto, como se puede observar en la siguiente imagen:

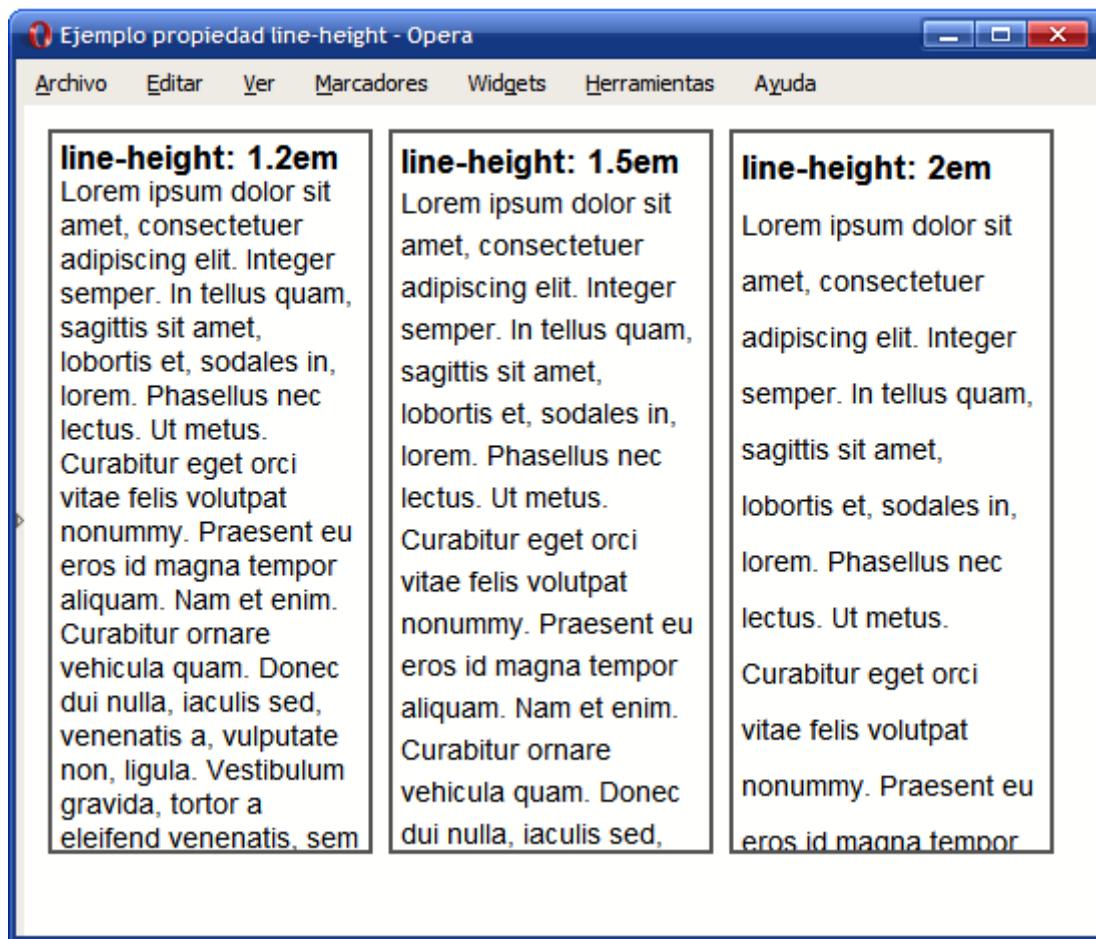


Figura 6.6. Ejemplo de propiedad line-height

Además de la decoración que se puede aplicar a la tipografía que utilizan los textos, CSS define otros estilos y decoraciones para el texto en su conjunto. La propiedad que decora el texto se denomina `text-decoration`.

text-decoration	Decoración del texto
Valores	none (underline overline line-through blink) inherit
Se aplica a	Todos los elementos
Valor inicial	none
Descripción	Establece la decoración del texto (subrayado, tachado, parpadeante, etc.)

El valor `underline` subraya el texto, por lo que puede confundir a los usuarios haciéndoles creer que se trata de un enlace. El valor `overline` añade una línea en la parte superior del texto, un aspecto que raramente es deseable. El valor `line-through` muestra el texto tachado con una línea continua, por lo que su uso tampoco es muy habitual. Por último, el valor `blink` muestra el texto parpadeante y se recomienda evitar su uso por las molestias que genera a la mayoría de usuarios.

Una de las propiedades de CSS más desconocidas y que puede ser de gran utilidad en algunas circunstancias es la propiedad `text-transform`, que puede variar de forma sustancial el aspecto del texto.

text-transform	Transformación del texto
Valores	<code>capitalize</code> <code>uppercase</code> <code>lowercase</code> <code>none</code> <code>inherit</code>
Se aplica a	Todos los elementos
Valor inicial	<code>none</code>
Descripción	Transforma el texto original (lo transforma a mayúsculas, a minúsculas, etc.)

La propiedad `text-transform` permite mostrar el texto original transformado en un texto completamente en mayúsculas (`uppercase`), en minúsculas (`lowercase`) o con la primera letra de cada palabra en mayúscula (`capitalize`).

La siguiente imagen muestra cada uno de los posibles valores:

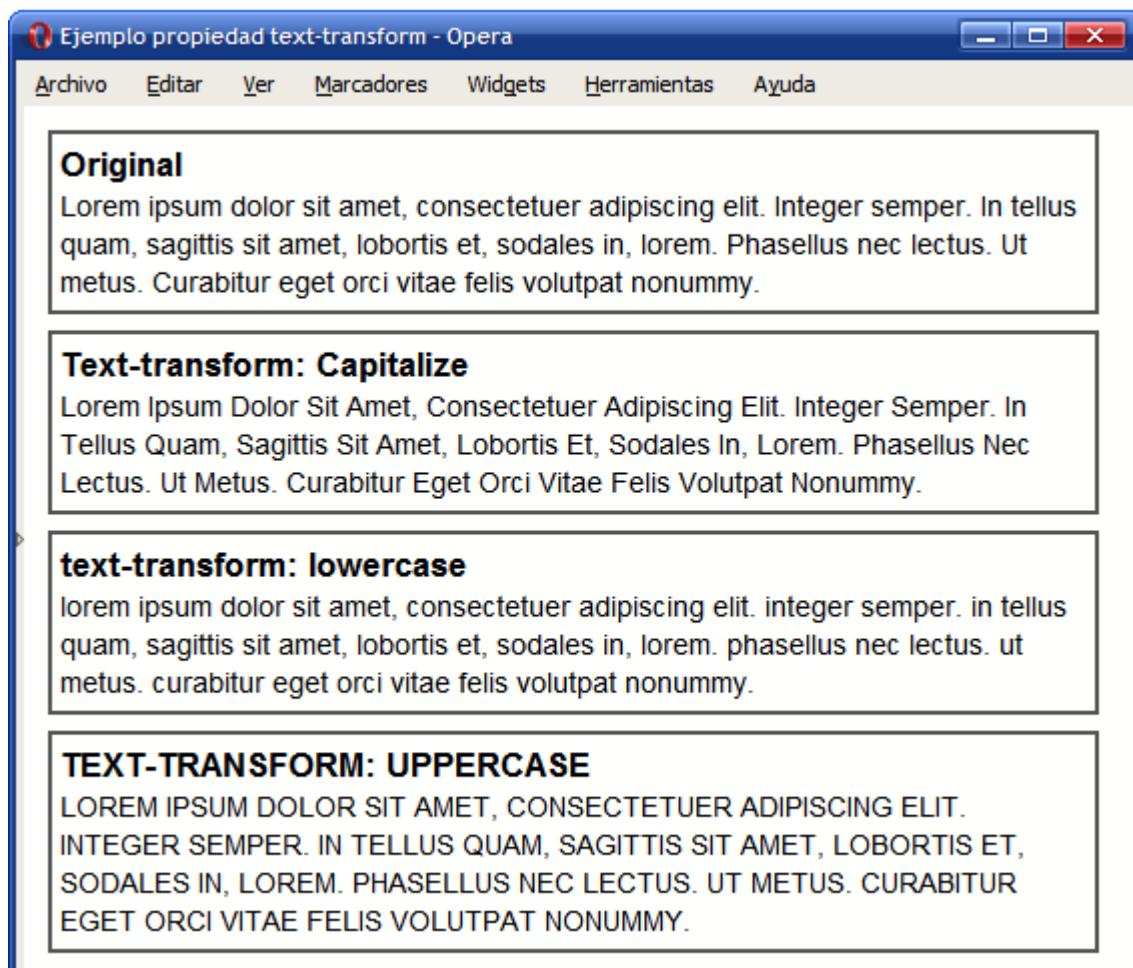


Figura 6.7. Ejemplo de propiedad `text-transform`

Las reglas CSS del ejemplo anterior se muestran a continuación:

```
<div style="text-transform: none"><h1>Original</h1>Lorem ipsum dolor  
sit amet...</div>
```

```
<div style="text-transform: capitalize"><h1>text-transform: capitalize</h1>
Lorem ipsum dolor sit amet...</div>

<div style="text-transform: lowercase"><h1>text-transform: lowercase</h1>
Lorem ipsum dolor sit amet...</div>

<div style="text-transform: uppercase"><h1>text-transform: uppercase</h1>
Lorem ipsum dolor sit amet...</div>
```

Uno de los principales problemas del diseño de documentos y páginas mediante CSS consiste en la alineación vertical en una misma línea de varios elementos diferentes como imágenes y texto. Para controlar esta alineación, CSS define la propiedad `vertical-align`.

vertical-align	Alineación vertical
Valores	baseline sub super top text-top middle bottom text-bottom <porcentaje> <medida> inherit
Se aplica a	Elementos en línea y celdas de tabla
Valor inicial	baseline
Descripción	Determina la alineación vertical de los contenidos de un elemento

A continuación se muestra una imagen con el aspecto que muestran los navegadores para cada uno de los posibles valores de la propiedad `vertical-align`:



Figura 6.8. Ejemplo de propiedad `vertical-align`

El valor por defecto es `baseline` y el valor más utilizado cuando se establece la propiedad `vertical-align` es `middle`.

En muchas publicaciones impresas suele ser habitual tabular la primera línea de cada párrafo para facilitar su lectura. CSS permite controlar esta tabulación mediante la propiedad `text-indent`.

text-indent	Tabulación del texto
Valores	<medida> <porcentaje> inherit
Se aplica a	Todos los elementos
Valor inicial	0
Descripción	Tabula desde la izquierda la primera línea del texto original

La siguiente imagen muestra la comparación entre un texto largo formado por varios párrafos sin tabular y el mismo texto con la primera línea de cada párrafo tabulada:

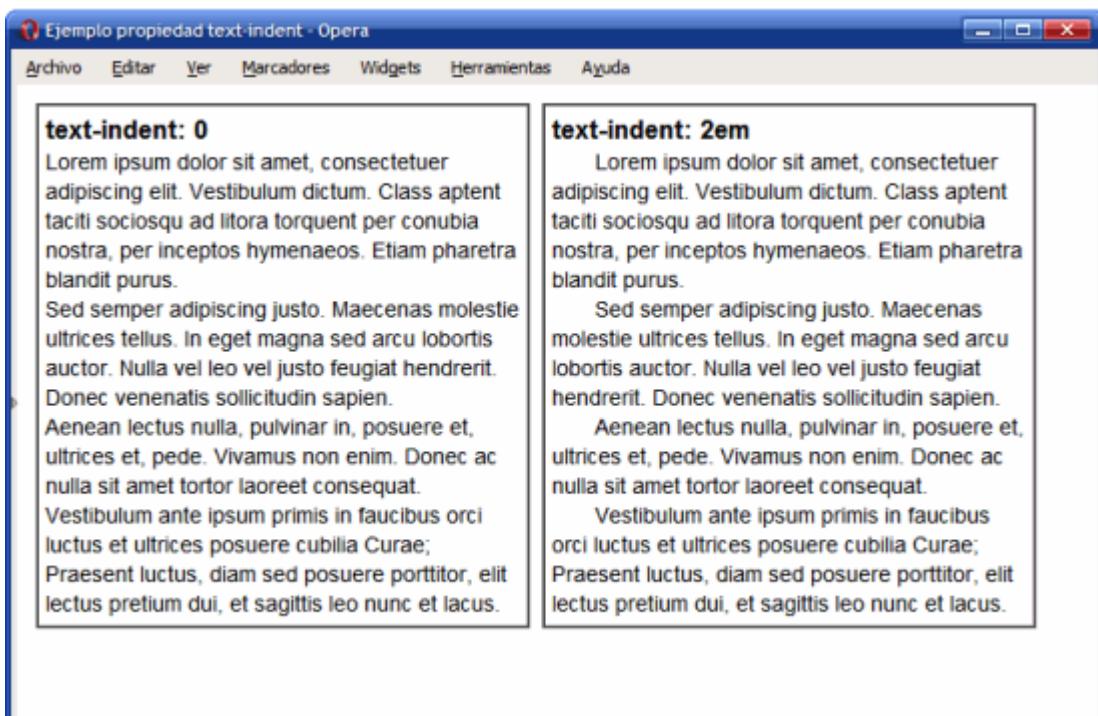


Figura 6.9. Ejemplo de propiedad `text-indent`

CSS también permite controlar la separación entre las letras que forman las palabras y la separación entre las palabras que forman los textos. La propiedad que controla la separación entre letras se llama `letter-spacing` y la separación entre palabras se controla mediante `word-spacing`.

letter-spacing	Espaciado entre letras
Valores	normal <medida> inherit
Se aplica a	Todos los elementos
Valor inicial	normal
Descripción	Permite establecer el espacio entre las letras que forman las palabras del texto

word-spacing	Espaciado entre palabras
Valores	normal <medida> inherit
Se aplica a	Todos los elementos
Valor inicial	normal
Descripción	Permite establecer el espacio entre las palabras que forman el texto

La siguiente imagen muestra la comparación entre un texto normal y otro con las propiedades letter-spacing y word-spacing aplicadas:

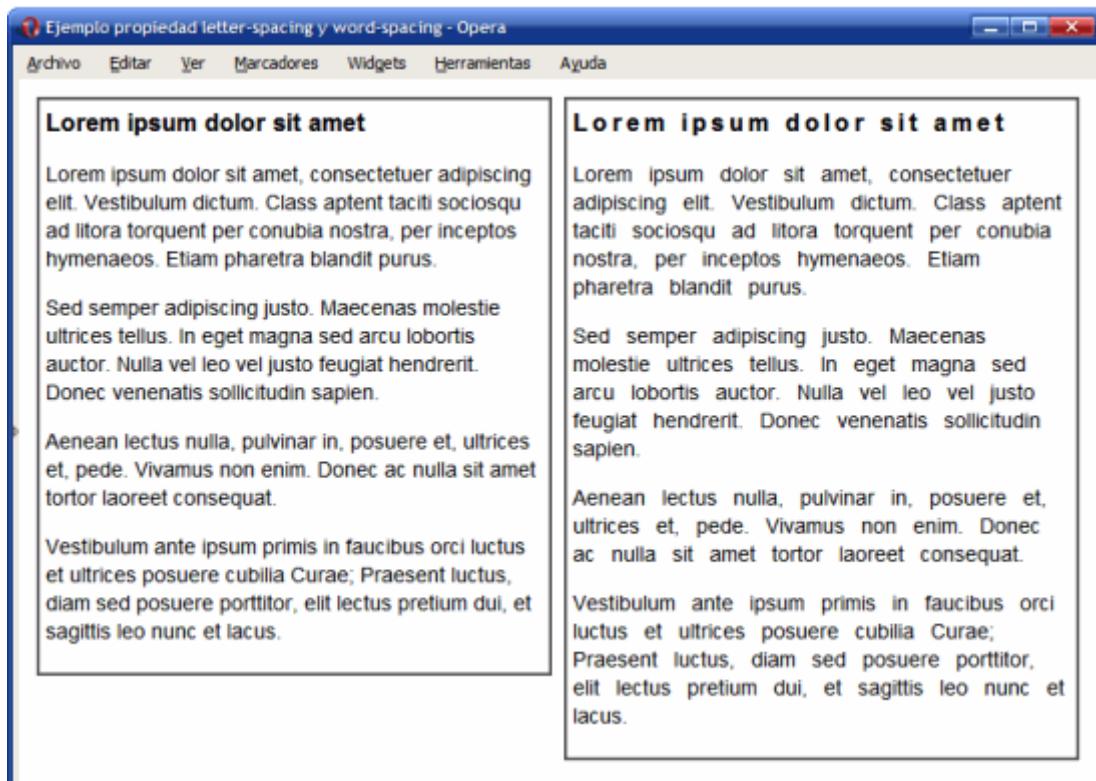


Figura 6.10. Ejemplo de propiedades letter-spacing y word-spacing

Las reglas CSS del ejemplo anterior se muestran a continuación:

```
.especial h1 { letter-spacing: .2em; }
.especial p { word-spacing: .5em; }

<div><h1>Lorem ipsum dolor sit amet</h1>
<p>Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Vestibulum
dictum. Class aptent taciti sociosqu ad litora torquent per conubia nostra,
per inceptos hymenaeos. Etiam pharetra blandit purus.</p>
...
</div>

<div class="especial"><h1>Lorem ipsum dolor sit amet</h1>
<p>Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Vestibulum
dictum. Class aptent taciti sociosqu ad litora torquent per conubia nostra,
per inceptos hymenaeos. Etiam pharetra blandit purus.</p>
...
</div>
```

Cuando se utiliza un valor numérico en las propiedades `letter-spacing` y `word-spacing`, se interpreta como la separación adicional que se añade (si el valor es positivo) o se quita (si el valor es negativo) a la separación por defecto entre letras y palabras respectivamente.

Como ya se sabe, el tratamiento que hace HTML de los espacios en blanco es uno de los aspectos más difíciles de comprender cuando se empiezan a crear las primeras páginas web. Básicamente, HTML elimina todos los espacios en blanco sobrantes, es decir, todos salvo un espacio en blanco entre cada palabra.

Para forzar los espacios en blanco adicionales se debe utilizar la entidad HTML ; y para forzar nuevas líneas, se utiliza el elemento `
`. Además, HTML proporciona el elemento `<pre>` que muestra el contenido tal y como se escribe, respetando todos los espacios en blanco y todas las nuevas líneas.

CSS también permite controlar el tratamiento de los espacios en blanco de los textos mediante la propiedad `white-space`.

white-space	Tratamiento de los espacios en blanco
Valores	<code>normal</code> <code>pre</code> <code>nowrap</code> <code>pre-wrap</code> <code>pre-line</code> <code>inherit</code>
Se aplica a	Todos los elementos
Valor inicial	<code>normal</code>
Descripción	Establece el tratamiento de los espacios en blanco del texto

El significado de cada uno de los valores es el siguiente:

- `normal`: comportamiento por defecto de HTML.
- `pre`: se respetan los espacios en blanco y las nuevas líneas (exactamente igual que la etiqueta `<pre>`). Si la línea es muy larga, *se sale* del espacio asignado para ese contenido.
- `nowrap`: elimina los espacios en blanco y las nuevas líneas. Si la línea es muy larga, *se sale* del espacio asignado para ese contenido.
- `pre-wrap`: se respetan los espacios en blanco y las nuevas líneas, pero ajustando cada línea al espacio asignado para ese contenido.
- `pre-line`: elimina los espacios en blanco y respeta las nuevas líneas, pero ajustando cada línea al espacio asignado para ese contenido.

En la siguiente tabla se resumen las características de cada valor:

Valor	Respetá espacios en blanco	Respetá saltos de línea	Ajustá las líneas
<code>normal</code>	no	no	si
<code>pre</code>	si	si	no
<code>nowrap</code>	no	no	no
<code>pre-wrap</code>	si	si	si
<code>pre-line</code>	no	si	si

La siguiente imagen muestra las diferencias entre los valores permitidos para `white-space`. El párrafo original contiene espacios en blanco y nuevas líneas y se ha limitado la anchura de su elemento contenedor:

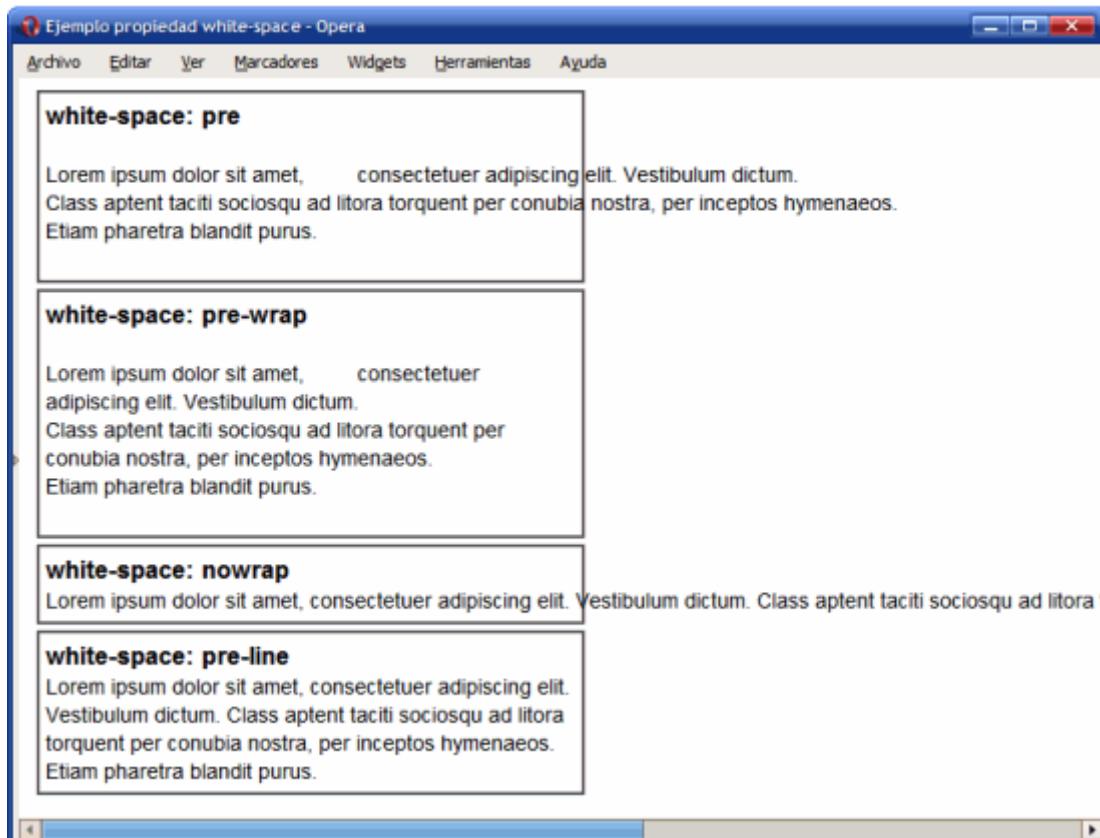


Figura 6.11. Ejemplo de propiedad `white-space`

Por último, CSS define unos elementos especiales llamados "*pseudo-elementos*" que permiten aplicar estilos a ciertas partes de un texto. En concreto, CSS permite definir estilos especiales a la primera frase de un texto y a la primera letra de un texto.

El pseudo-elemento `:first-line` permite aplicar estilos a la primera línea de un texto. Las palabras que forman la primera línea de un texto dependen del espacio reservado para mostrar el texto o del tamaño de la ventana del navegador, por lo que CSS calcula de forma automática las palabras que forman la primera línea de texto en cada momento.

La siguiente imagen muestra cómo aplica CSS los estilos indicados a la primera línea calculando para cada anchura las palabras que forman la primera línea:

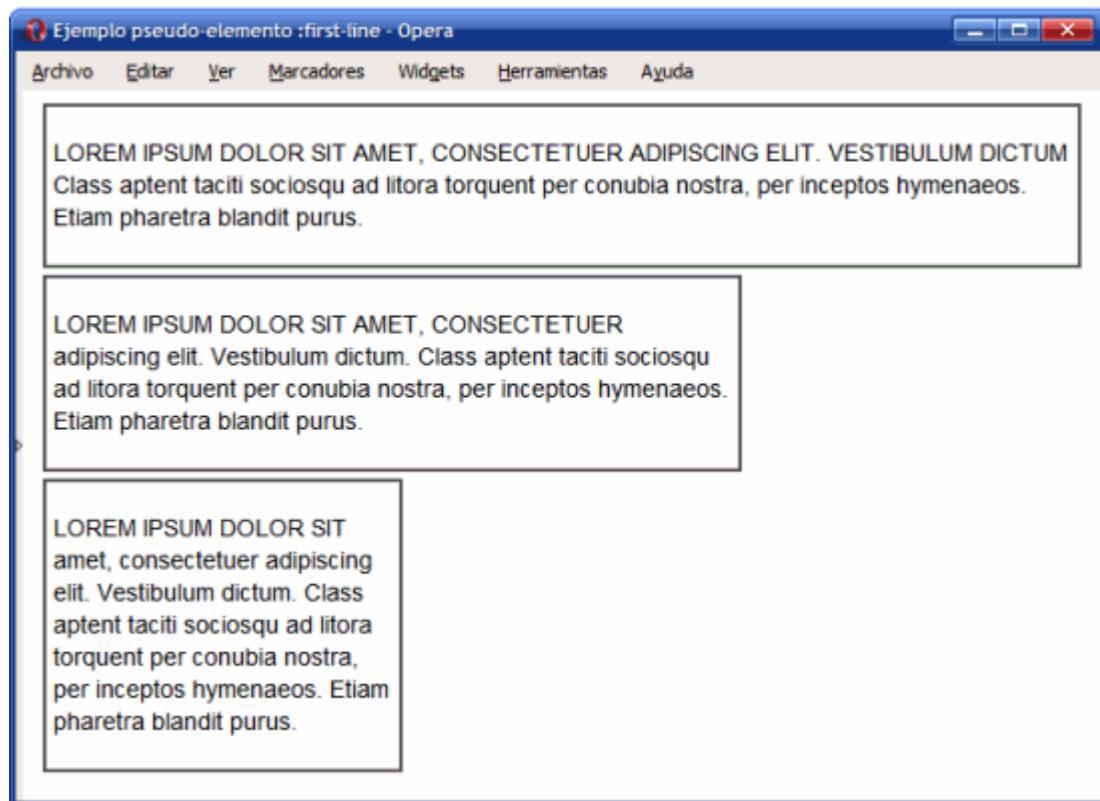


Figura 6.12. Ejemplo de pseudo-elemento first-line

La regla CSS utilizada para los párrafos del ejemplo se muestra a continuación:

```
p:first-line {  
    text-transform: uppercase;  
}
```

De la misma forma, CSS permite aplicar estilos a la primera letra del texto mediante el pseudo-elemento :first-letter. La siguiente imagen muestra el uso del pseudo-elemento :first-letter para crear una letra capital:

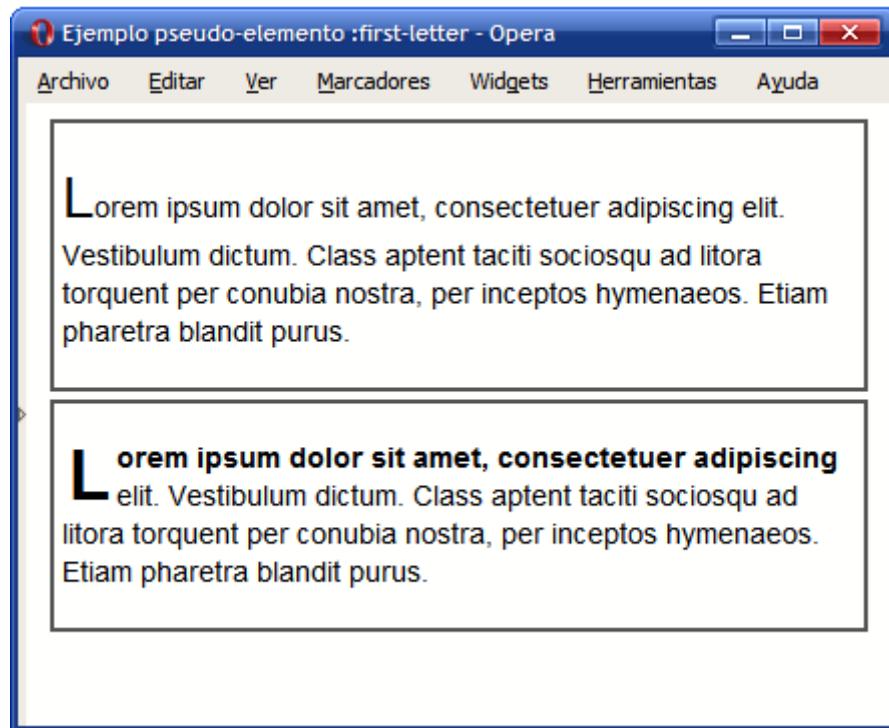


Figura 6.13. Ejemplo de pseudo-elemento first-letter

El código HTML y CSS se muestra a continuación:

```
#normal p:first-letter {
    font-size: 2em;
}
#avanzado p:first-letter {
    font-size: 2.5em;
    font-weight: bold;
    line-height: .9em;
    float: left;
    margin: .1em;
}
#avanzado p:first-line {
    font-weight: bold;
}

<div id="normal">
<p>Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Vestibulum
dictum. Class aptent taciti sociosqu ad litora torquent per conubia nostra,
per inceptos hymenaeos. Etiam pharetra blandit purus.</p>
</div>
<div id="avanzado">
<p>Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Vestibulum
dictum. Class aptent taciti sociosqu ad litora torquent per conubia nostra,
per inceptos hymenaeos. Etiam pharetra blandit purus.</p>
</div>
```

Capítulo 7. Enlaces

7.1. Estilos básicos

7.1.1. Tamaño, color y decoración

Los estilos más sencillos que se pueden aplicar a los enlaces son los que modifican su tamaño de letra, su color y la decoración del texto del enlace. Utilizando las propiedades `text-decoration` y `font-weight` se pueden conseguir estilos como los que se muestran en la siguiente imagen:



Figura 7.1. Ejemplo de enlaces con estilos diferentes

A continuación se muestran las reglas CSS del ejemplo anterior:

```
a { margin: 1em 0; display: block; }

.alternativo {color: #CC0000;}
.simple {text-decoration: none;}
.importante {font-weight: bold; font-size: 1.3em;}
.raro {text-decoration: overline;}

<a href="#">Enlace con el estilo por defecto</a>
<a class="alternativo" href="#">Enlace de color rojo</a>
<a class="simple" href="#">Enlace sin subrayado</a>
<a class="importante" href="#">Enlace muy importante</a>
<a class="raro" href="#">Enlace con un estilo raro</a>
```

7.1.2. Pseudo-clases

CSS define cuatro *pseudo-clases* que permiten aplicar estilos avanzados para los enlaces de los documentos. Las *pseudo-clases* permiten aplicar diferentes estilos a un mismo enlace en función de su estado: enlace no visitado, enlace visitado, enlace en el que se pasa el puntero del ratón por encima y enlace activo en ese momento.

Cada una de las pseudo-clases definidas se muestra a continuación:

- `:link`, permite aplicar estilos para los enlaces que aún no han sido visitados.
- `:visited`, aplica estilos a los enlaces que han sido pinchados anteriormente (el navegador elimina automáticamente el historial de enlaces visitados cada cierto tiempo).
- `:hover`, estilos que muestra el enlace cuando el usuario posiciona el puntero del ratón sobre el enlace.
- `:active`, estilos que se aplican al enlace cuando el usuario está pinchando sobre el enlace (el tiempo durante el que se aplica este estilo es muy breve).

El orden recomendado para la definición de las pseudo-clases de los enlaces es `:link`, `:visited`, `:hover`, `:active`.

Las pseudo-clases `:link` y `:visited` solamente están definidas para los enlaces, pero las pseudo-clases `:hover` y `:active` se definen para todos los elementos HTML. Desafortunadamente, Internet Explorer 6 y sus versiones anteriores solamente las soportan para los enlaces.

Las pseudo-clases de los enlaces permiten variar el comportamiento por defecto que los navegadores aplican a los enlaces. El siguiente ejemplo oculta el subrayado a los enlaces cuando el usuario pasa el ratón por encima de un enlace:



Figura 7.2. Ocultando el subrayado de los enlaces al pasar el ratón por encima

El código CSS necesario se muestra a continuación:

```
a { }  
a:hover { text-decoration: none; }
```

También es posible combinar en un mismo elemento las pseudo-clases que son compatibles entre sí:

```
/* Los estilos se aplican cuando el usuario pasa el ratón por encima de un  
enlace que todavía no ha visitado */  
a:link:hover { ... }  
  
/* Los estilos se aplican cuando el usuario pasa el ratón por encima de un  
enlace que ha visitado previamente */  
a:visited:hover { ... }
```

Ejercicio 8 Ver enunciado en la página 201

7.2. Estilos avanzados

7.2.1. Decoración personalizada

La propiedad `text-decoration` permite añadir o eliminar el subrayado de los enlaces. Sin embargo, el aspecto del subrayado lo controla automáticamente el navegador, por lo que su color siempre es el mismo que el del texto del enlace y su anchura es proporcional al tamaño de letra.

No obstante, utilizando la propiedad `border-bottom` es posible añadir cualquier tipo de subrayado para los enlaces de las páginas. El siguiente ejemplo muestra algunos enlaces con el subrayado personalizado:

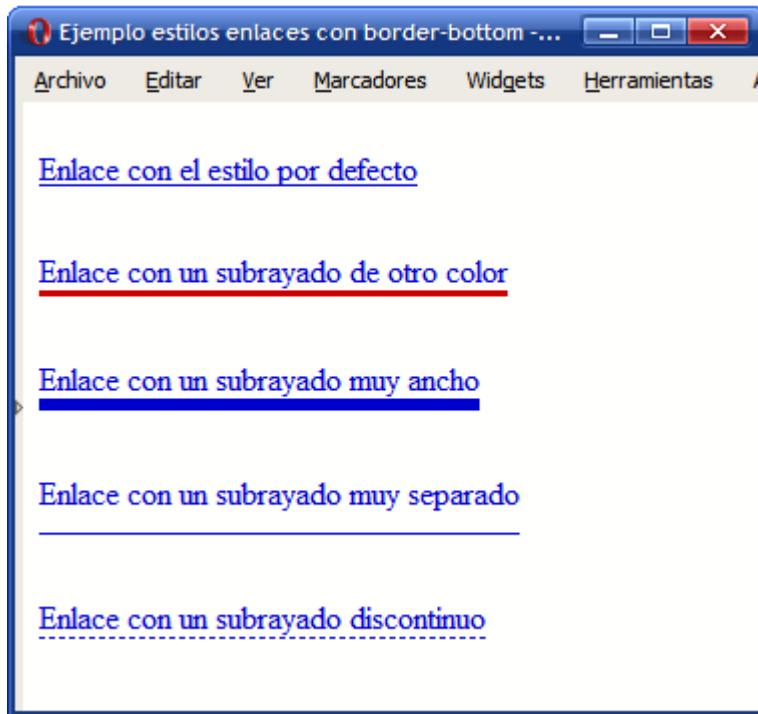


Figura 7.3. Enlaces con subrayado personalizado mediante la propiedad border

Las reglas CSS definidas en el ejemplo anterior se muestran a continuación:

```
a { margin: 1em 0; float: left; clear: left; text-decoration: none; }  
.simple {text-decoration: underline;}  
.color { border-bottom: medium solid #CC0000;}  
.ancho {border-bottom: thick solid;}  
.separado {border-bottom: 1px solid; padding-bottom: .6em;}
```

```
.discontinuo {border-bottom: thin dashed;}

<a class="simple" href="#">Enlace con el estilo por defecto</a>

<a class="color" href="#">Enlace un subrayado de otro color</a>

<a class="ancho" href="#">Enlace con un subrayado muy ancho</a>

<a class="separado" href="#">Enlace con un subrayado muy separado</a>

<a class="discontinuo" href="#">Enlace con un subrayado discontinuo</a>
```

7.2.2. Imágenes según el tipo de enlace

En ocasiones, puede resultar útil incluir un pequeño ícono al lado de un enlace para indicar el tipo de contenido que enlaza, como por ejemplo un archivo comprimido o un documento con formato PDF.

Este tipo de imágenes son puramente decorativas en vez de imágenes de contenido, por lo que se deberían añadir con CSS y no con elementos de tipo ``. Utilizando la propiedad `background` (y `background-image`) se puede personalizar el aspecto de los enlaces para que incluyan un pequeño ícono a su lado.

La técnica consiste en mostrar una imagen de fondo sin repetición en el enlace y añadir el `padding` necesario al texto del enlace para que no se solape con la imagen de fondo.

El siguiente ejemplo personaliza el aspecto de dos enlaces añadiéndoles una imagen de fondo:



Figura 7.4. Personalizando el aspecto de los enlaces en función de su tipo

Las reglas CSS necesarias se muestran a continuación:

```
a { margin: 1em 0; float: left; clear: left; }

.rss {
  color: #E37529;
  padding: 0 0 0 18px;
  background: #FFF url(imagenes/rss.gif) no-repeat left center;
}
```

```
.pdf {  
    padding: 0 0 0 22px;  
    background: #FFF url(imagenes/pdf.png) no-repeat left center;  
}  
  
<a href="#">Enlace con el estilo por defecto</a>  
  
<a class="rss" href="#">Enlace a un archivo RSS</a>  
  
<a class="pdf" href="#">Enlace a un documento PDF</a>
```

7.2.3. Mostrar los enlaces como si fueran botones

Las propiedades definidas por CSS permiten mostrar los enlaces con el aspecto de un botón, lo que puede ser útil en formularios en los que no se utilizan elementos específicos para crear botones.

El siguiente ejemplo muestra un enlace simple formateado como un botón:

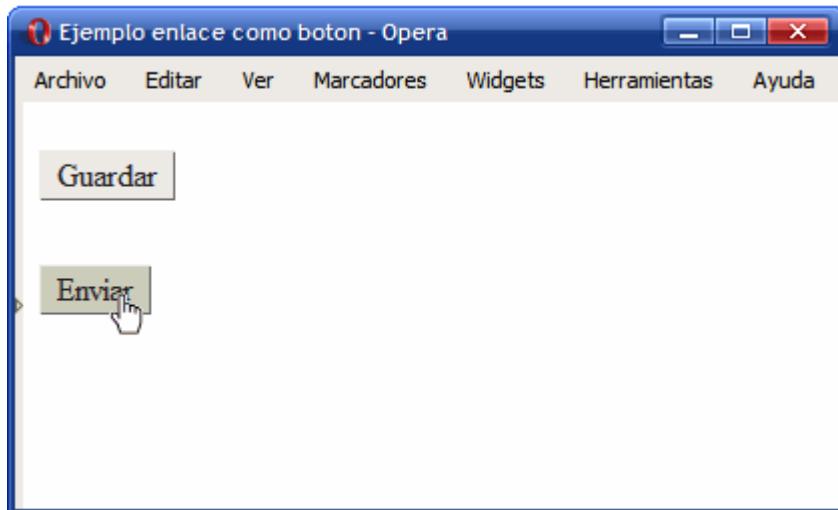


Figura 7.5. Mostrando un enlace como si fuera un botón

Las reglas CSS utilizadas en el ejemplo anterior son las siguientes:

```
a { margin: 1em 0; float: left; clear: left; }  
a.boton {  
    text-decoration: none;  
    background: #EEE;  
    color: #222;  
    border: 1px outset #CCC;  
    padding: .1em .5em;  
}  
a.boton:hover {  
    background: #CCB;  
}  
a.boton:active {  
    border: 1px inset #000;  
}
```

```
| <a class="boton" href="#">Guardar</a>
| <a class="boton" href="#">Enviar</a>
```

Capítulo 8. Imágenes

8.1. Estilos básicos

8.1.1. Establecer la anchura y altura de las imágenes

Utilizando las propiedades `width` y `height`, es posible mostrar las imágenes con cualquier altura/anchura, independientemente de su altura/anchura real:

```
#destacada {  
    width: 120px;  
    height: 250px;  
}  
  

```

No obstante, si se utilizan alturas/anchuras diferentes de las reales, el navegador deforma las imágenes y el resultado estético es muy desagradable.

Por otra parte, establecer la altura/anchura de todas las imágenes mediante CSS es una práctica poco recomendable. El motivo es que normalmente dos imágenes diferentes no comparten la misma altura/anchura. Por lo tanto, se debe crear una nueva regla CSS (y un nuevo selector) para cada una de las diferentes imágenes del sitio web.

En la práctica, esta forma de trabajo produce una sobrecarga de estilos que la hace inviable. Por este motivo, aunque es una solución que no respeta la separación completa entre contenidos (XHTML) y presentación (CSS), se recomienda establecer la altura/anchura de las imágenes mediante los atributos `width` y `height` de la etiqueta ``:

```

```

8.1.2. Eliminar el borde de las imágenes con enlaces

Cuando una imagen forma parte de un enlace, los navegadores muestran por defecto un borde azul grueso alrededor de las imágenes. Por tanto, una de las reglas más utilizadas en los archivos CSS es la que elimina los bordes de las imágenes con enlaces:

```
img {  
    border: none;  
}
```

Ejercicio 9 Ver enunciado en la página 201

8.2. Estilos avanzados

8.2.1. Sombra (drop shadow)

La mayoría de aplicaciones de diseño gráfico permiten añadir una sombra (llamada *drop shadow* en inglés) a las imágenes. CSS no incluye propiedades que permitan mostrar de forma sencilla sombras en los elementos.

No obstante, existen varias técnicas sencillas y otras más avanzadas que permiten crear sombras que se adapten a cualquier imagen o elemento. A continuación se muestra una técnica sencilla para añadir sombra a las imágenes.

El estilo final de las sombras creadas con esta técnica se muestra a continuación:

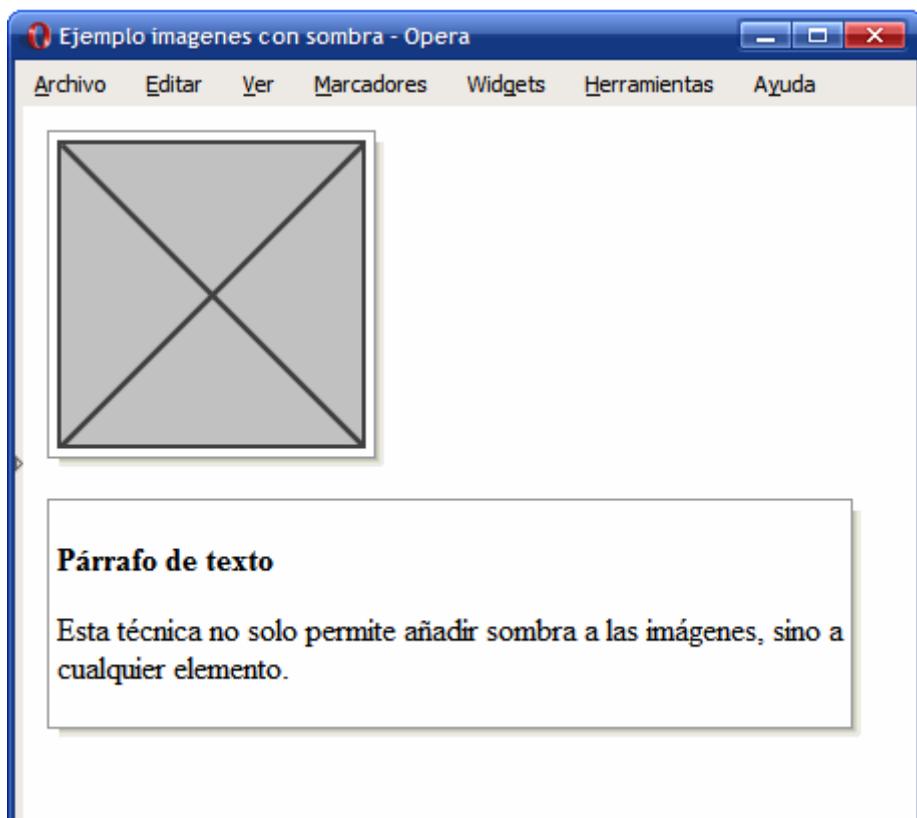


Figura 8.1. Sombra aplicada a las imágenes y otros elementos mediante CSS

La técnica mostrada se presentó por primera vez en la página web <http://wubbleyew.com/tests/dropshadows.htm> y consiste en la utilización de un par de elementos <div> alrededor del elemento que va a mostrar una sombra. La sombra se consigue mediante una imagen muy grande que contiene una sombra orientada hacia el lado derecho e inferior.

La ventaja de este método es que es sencillo y solamente requiere añadir un par de <div> por cada elemento. Además, como la sombra es una imagen muy grande, el efecto funciona con elementos de cualquier tamaño.

El mayor inconveniente de este método es que la sombra siempre se muestra en la misma dirección (derecha inferior) y que en Internet Explorer 6 y versiones anteriores sólo muestran la sombra correctamente cuando el color de fondo de la página es blanco (ya que Internet Explorer 6 y versiones anteriores no soportan imágenes en formato PNG con transparencias).

El código HTML y CSS del ejemplo anterior es bastante sencillo:

```
.dropshadow {  
    float:left;  
    clear:left;  
    background: url(imagenes/shadowAlpha.png) no-repeat bottom right !important;  
    background: url(imagenes/shadow.gif) no-repeat bottom right;  
    margin: 10px 0 10px 10px !important;  
    margin: 10px 0 10px 5px;  
    padding: 0px;  
}  
.innerbox {  
    position:relative;  
    bottom:6px;  
    right: 6px;  
    border: 1px solid #999999;  
    padding:4px;  
    margin: 0px 0px 0px 0px;  
}  
  
<div class="dropshadow">  
<div class="innerbox">  
      
</div>  
</div>  
  
<div class="dropshadow">  
<div class="innerbox">  
    <h4>Párrafo de texto</h4>  
    <p>Esta técnica no sólo permite añadir sombra a las imágenes, sino a cualquier  
elemento.</p>  
</div>  
</div>
```

Capítulo 9. Listas

9.1. Estilos básicos

9.1.1. Viñetas personalizadas

Por defecto, los navegadores muestran los elementos de las listas no ordenadas con una viñeta formada por un pequeño círculo de color negro. Los elementos de las listas ordenadas se muestran por defecto con la numeración decimal utilizada en la mayoría de países.

No obstante, CSS define varias propiedades para controlar el tipo de viñeta que muestran las listas, además de poder controlar la posición de la propia viñeta. La propiedad básica es la que controla el tipo de viñeta que se muestra y que se denomina `list-style-type`.

list-style-type	Tipo de viñeta
Valores	disc circle square decimal decimal-leading-zero lower-roman upper-roman lower-greek lower-latin upper-latin armenian georgian lower-alpha upper-alpha none inherit
Se aplica a	Elementos de una lista
Valor inicial	disc
Descripción	Permite establecer el tipo de viñeta mostrada para una lista

En primer lugar, el valor `none` permite mostrar una lista en la que sus elementos no contienen viñetas, números o letras. Se trata de un valor muy utilizado, ya que es imprescindible para los menús de navegación creados con listas, como se verá más adelante.

El resto de valores de la propiedad `list-style-type` se dividen en tres tipos: gráficos, numéricos y alfabéticos.

- Los valores gráficos son `disc`, `circle` y `square` y muestran como viñeta un círculo relleno, un círculo vacío y un cuadrado relleno respectivamente.
- Los valores numéricos están formados por `decimal`, `decimal-leading-zero`, `lower-roman`, `upper-roman`, `armenian` y `georgian`.
- Por último, los valores alfanuméricos se controlan mediante `lower-latin`, `lower-alpha`, `upper-latin`, `upper-alpha` y `lower-greek`.

La siguiente imagen muestra algunos de los valores definidos por la propiedad `list-style-type`:

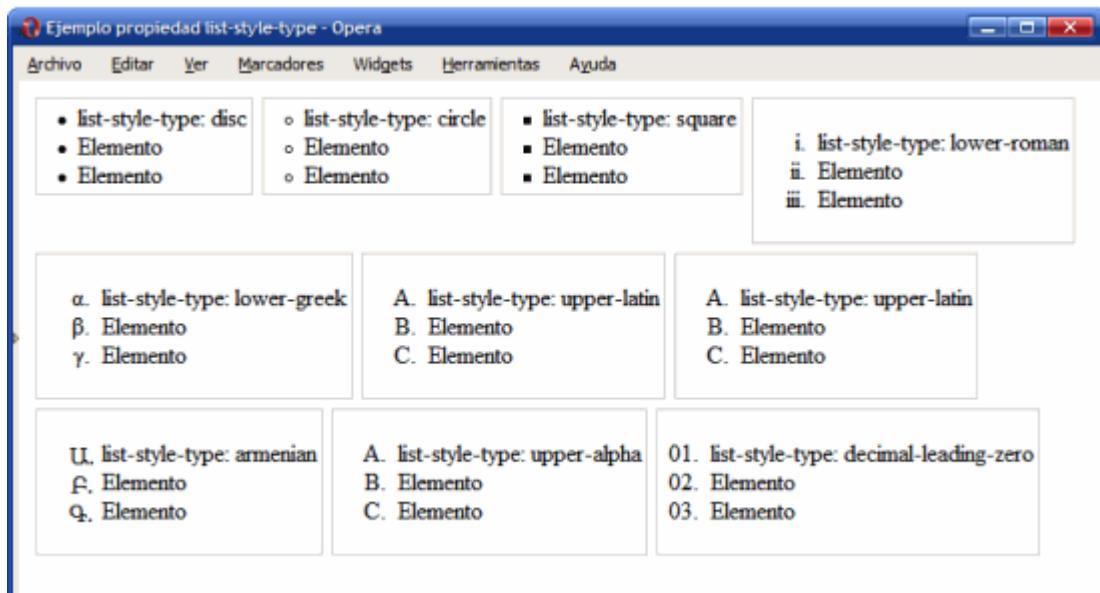


Figura 9.1. Ejemplo de propiedad list-style-type

El código CSS de algunas de las listas del ejemplo anterior se muestra a continuación:

```
<ul style="list-style-type: square">
    <li>list-style-type: square</li>
    <li>Elemento</li>
    <li>Elemento</li>
</ul>
<ol style="list-style-type: lower-roman">
    <li>list-style-type: lower-roman</li>
    <li>Elemento</li>
    <li>Elemento</li>
</ol>
<ol style="list-style-type: decimal-leading-zero; padding-left: 2em;">
    <li>list-style-type: decimal-leading-zero</li>
    <li>Elemento</li>
    <li>Elemento</li>
</ol>
```

La propiedad `list-style-position` permite controlar la colocación de las viñetas.

list-style-position	Posición de la viñeta
Valores	inside outside inherit
Se aplica a	Elementos de una lista
Valor inicial	outside
Descripción	Permite establecer la posición de la viñeta de cada elemento de una lista

La diferencia entre los valores `outside` y `inside` se hace evidente cuando los elementos contienen mucho texto, como en la siguiente imagen:



Figura 9.2. Ejemplo de propiedad list-style-position

Utilizando las propiedades anteriores (`list-style-type` y `list-style-position`), se puede seleccionar el tipo de viñeta y su posición, pero no es posible personalizar algunas de sus características básicas como su color y tamaño.

Cuando se requiere personalizar el aspecto de las viñetas, se debe emplear la propiedad `list-style-image`, que permite mostrar una imagen propia en vez de una viñeta automática.

list-style-image	Imagen de la viñeta
Valores	<url> none inherit
Se aplica a	Elementos de una lista
Valor inicial	none
Descripción	Permite reemplazar las viñetas automáticas por una imagen personalizada

Las imágenes personalizadas se indican mediante la URL de la imagen. Si no se encuentra la imagen o no se puede cargar, se muestra la viñeta automática correspondiente (salvo que explícitamente se haya eliminado mediante la propiedad `list-style-type`).

La siguiente imagen muestra el uso de la propiedad `list-style-image` mediante tres ejemplos sencillos de listas con viñetas personalizadas:

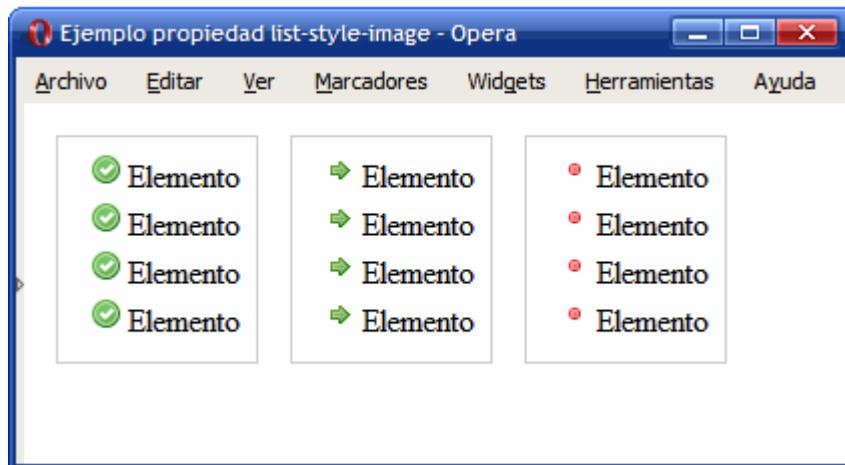


Figura 9.3. Ejemplo de propiedad list-style-image

Las reglas CSS correspondientes al ejemplo anterior se muestran a continuación:

```
ul {
    margin:0;
    padding-left: 1.5em;
    line-height: 1.5em;
}
ul li { padding-left: .2em; }
ul.ok { list-style-image: url(imagenes/ok.png); }
ul.go { list-style-image: url(imagenes/bullet_go.png); }
ul.redondo { list-style-image: url(imagenes/bullet_red.png); }
```

Como es habitual, CSS define una propiedad de tipo "*shorthand*" que permite establecer todas las propiedades de una lista de forma directa. La propiedad se denomina *list-style*.

list-style	Estilo de una lista
Valores	(<list-style-type> <list-style-position> <list-style-image>) inherit
Se aplica a	Elementos de una lista
Valor inicial	-
Descripción	Propiedad que permite establecer de forma simultánea todas las opciones de una lista

En la definición anterior, la notación || significa que el orden en el que se indican los valores de la propiedad es indiferente. El siguiente ejemplo indica que no se debe mostrar ni viñetas automáticas ni viñetas personalizadas:

```
| ul { list-style: none }
```

Cuando se utiliza una viñeta personalizada, es conveniente indicar la viñeta automática que se mostrará cuando no se pueda cargar la imagen:

```
| ul { list-style: url(imagenes/cuadrado_rojo.gif) square; }
```

9.1.2. Menú vertical sencillo

Las listas HTML se suelen emplear, además de para su función natural, para la creación de menús de navegación verticales y horizontales.

A continuación se muestra la transformación de una lista sencilla de enlaces en un menú vertical de navegación.

Lista de enlaces original:

```
<ul>
  <li><a href="#" title="Enlace genérico">Elemento 1</a></li>
  <li><a href="#" title="Enlace genérico">Elemento 2</a></li>
  <li><a href="#" title="Enlace genérico">Elemento 3</a></li>
  <li><a href="#" title="Enlace genérico">Elemento 4</a></li>
  <li><a href="#" title="Enlace genérico">Elemento 5</a></li>
  <li><a href="#" title="Enlace genérico">Elemento 6</a></li>
</ul>
```

Aspecto final del menú vertical:



Figura 9.4. Menú vertical sencillo creado con CSS

El proceso de transformación de la lista en un menú requiere de los siguientes pasos:

1) Definir la anchura del menú:

```
ul.menu { width:180px; }
```



Figura 9.5. Menú vertical: definiendo su anchura

2) Eliminar las viñetas automáticas y todos los márgenes y espaciados aplicados por defecto:

```
ul.menu {  
    width: 180px;  
    list-style: none;  
    margin: 0;  
    padding: 0;  
}
```



Figura 9.6. Menú vertical: eliminar viñetas por defecto

3) Añadir un borde al menú de navegación y establecer el color de fondo y los bordes de cada elemento del menú:

```
ul.menu {  
    width: 180px;  
    list-style: none;  
    margin: 0;  
    padding: 0;  
    border: 1px solid #7C7C7C;  
}
```

```
ul.menu li {
    border-bottom: 1px solid #7C7C7C;
    border-top: 1px solid #FFF;
    background: #F4F4F4;
}
```



Figura 9.7. Menú vertical: añadiendo bordes

4) Aplicar estilos a los enlaces: mostrarlos como un elemento de bloque para que ocupen todo el espacio de cada del menú, añadir un espacio de relleno y modificar los colores y la decoración por defecto:

```
ul.menu li a:link, ul.menu li a:visited {
    padding: .2em 0 .2em .5em;
    display: block;
    text-decoration: none;
    color: #333;
}
```

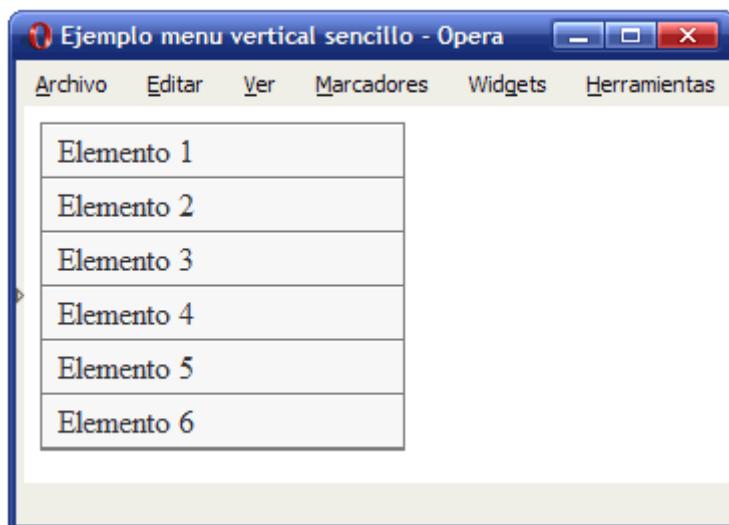


Figura 9.8. Aspecto final del menú vertical sencillo creado con CSS

Los tipos de menús verticales que se pueden definir mediante las propiedades CSS son innumerables, como se puede observar en la página <http://www.exploding-boy.com/images/EBmenus/menus.html>

9.1.3. Menú vertical avanzado

Ejercicio 10 Ver enunciado en la página 202

9.2. Estilos avanzados

9.2.1. Menú horizontal básico

A partir de un menú vertical sencillo, es posible crear un menú horizontal sencillo aplicando las propiedades CSS conocidas hasta el momento.

A continuación se muestra la transformación del anterior menú vertical sencillo en un menú horizontal sencillo. En este ejemplo, las propiedades para establecer el aspecto de los elementos del menú se definen en los elementos `<a>` en lugar de definirlas para los elementos `` como en el ejemplo anterior. En cualquier caso, es indiferente el elemento en el que se aplican los estilos que definen el aspecto de cada opción del menú.

Código HTML del menú horizontal:

```
<ul>
  <li><a href="#" title="Enlace genérico">Elemento 1</a></li>
  <li><a href="#" title="Enlace genérico">Elemento 2</a></li>
  <li><a href="#" title="Enlace genérico">Elemento 3</a></li>
  <li><a href="#" title="Enlace genérico">Elemento 4</a></li>
  <li><a href="#" title="Enlace genérico">Elemento 5</a></li>
  <li><a href="#" title="Enlace genérico">Elemento 6</a></li>
</ul>
```

Código CSS del menú vertical anterior:

```
ul.menu {
  width: 180px;
  list-style: none;
  margin: 0;
  padding: 0;
  border: 1px solid #7C7C7C;
}
ul.menu li {
  border-bottom: 1px solid #7C7C7C;
  border-top: 1px solid #FFF;
  background: #F4F4F4;
}
ul.menu li a:link, ul.menu li a:visited {
  padding: .2em 0 .2em .5em;
  display: block;
  text-decoration: none;
  color: #333;
}
```

Aspecto final del menú horizontal:

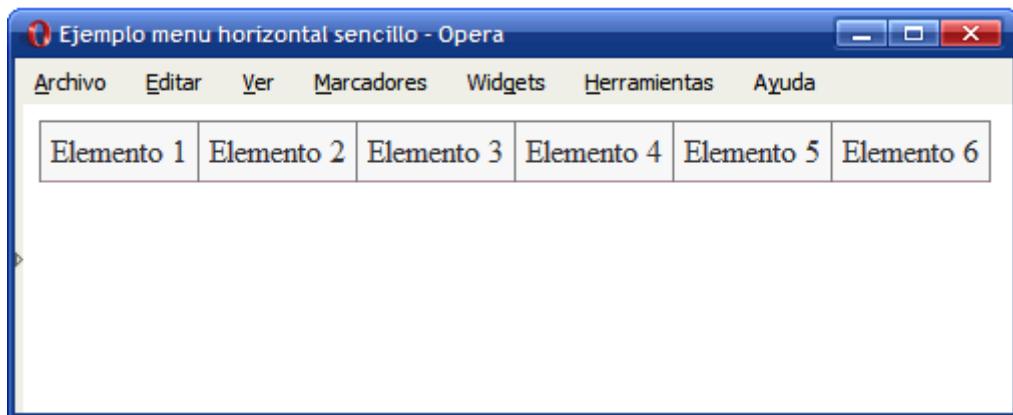


Figura 9.9. Menú horizontal sencillo creado con CSS

El proceso de transformación del menú vertical en un menú horizontal consta de los siguientes pasos:

- 1) Eliminar la anchura y el borde del elemento `` y aplicar las propiedades `float` y `clear`:

```
ul.menu {
    clear: both;
    float: left;
    width: 100%;
    list-style: none;
    margin: 0;
    padding: 0;
}
```



Figura 9.10. Menú horizontal: definiendo anchura y bordes

- 2) La clave de la transformación reside en modificar la propiedad `float` de los elementos `` del menú:

```
ul.menu li {
    float: left;
}
```

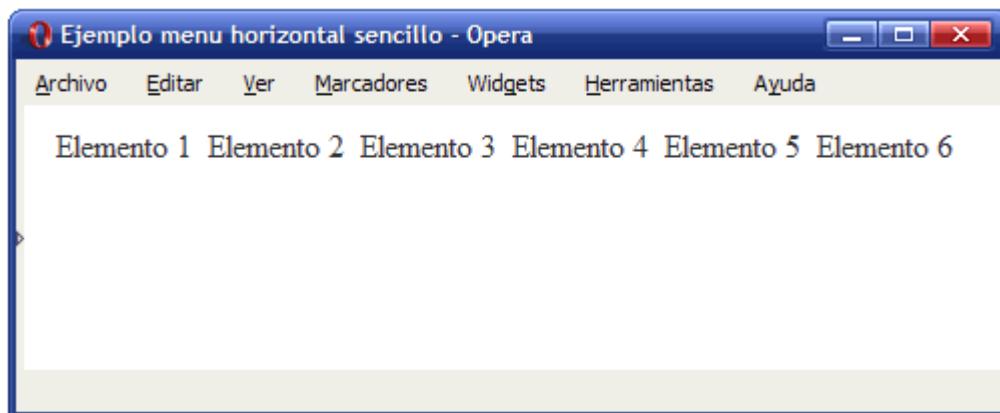


Figura 9.11. Menú horizontal: propiedades float y display

3) Modificar el padding y los bordes de los enlaces que forman el menú:

```
ul.menu li a:link, ul.menu li a:visited {
    padding: .3em;
    display: block;
    text-decoration: none;
    color: #333;
    background: #F4F4F4;
    border-top: 1px solid #7C7C7C;
    border-right: 1px solid #7C7C7C;
    border-bottom: 1px solid #9C9C9C;
}
```

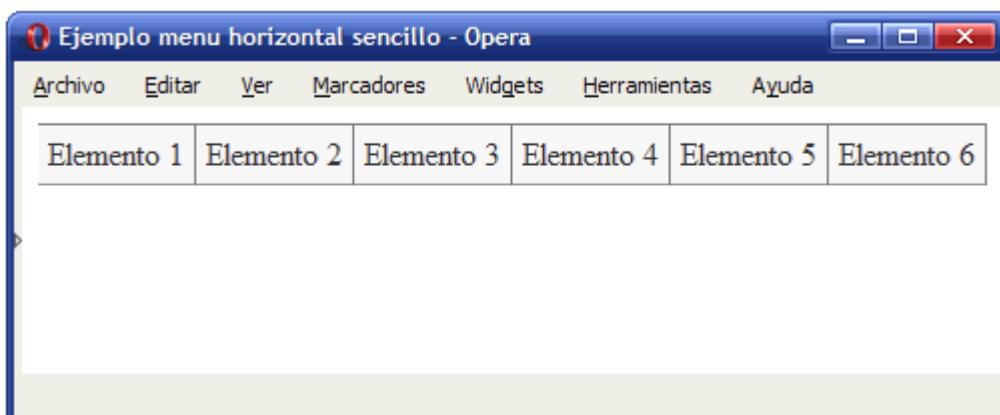


Figura 9.12. Menú horizontal: relleno y borde de los elementos

4) Por último, se añade un borde izquierdo en el elemento para homogeneizar el aspecto de los elementos del menú:

```
ul.menu {
    clear: both;
    float: left;
    width: 100%;
    list-style: none;
    margin: 0;
    padding: 0;
    border-left: 1px solid #7C7C7C;
}
```

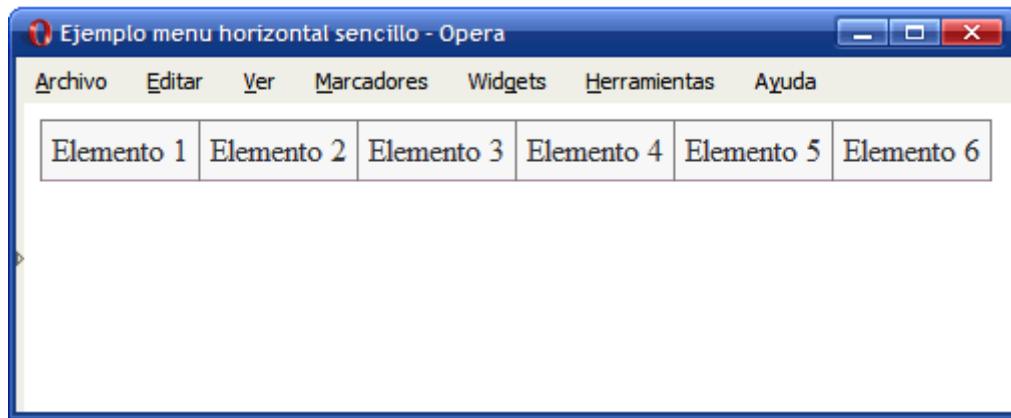


Figura 9.13. Aspecto final del menú horizontal sencillo creado con CSS

El código CSS final se muestra a continuación:

```
ul.menu {  
    clear: both;  
    float: left;  
    width: 100%;  
    list-style: none;  
    margin: 0;  
    padding: 0;  
    border-left:1px solid #7C7C7C;  
}  
ul.menu li {  
    display: inline;  
    float: left;  
}  
ul.menu li a:link, ul.menu li a:visited {  
    padding: .3em;  
    display: block;  
    text-decoration: none;  
    color: #333;  
    background: #F4F4F4;  
    border-top:1px solid #7C7C7C;  
    border-right:1px solid #7C7C7C;  
    border-bottom:1px solid #9C9C9C;  
}
```

9.2.2. Menú horizontal con solapas

Modificando los estilos de cada elemento del menú y utilizando imágenes de fondo y las pseudo-clases `:hover` y `:active`, se pueden crear menús horizontales complejos, incluso con el aspecto de un menú de solapas o pestañas:



Figura 9.14. Ejemplos de menús horizontales con pestañas creados con CSS

El código fuente de los menús de la imagen anterior y muchos otros se puede encontrar en <http://exploding-boy.com/images/cssmenus/menus.html>

9.2.3. Menú horizontal avanzado

Además de los menús horizontales de solapas, existen muchos otros tipos diferentes de menús horizontales (y verticales) que se pueden realizar empleando exclusivamente CSS:

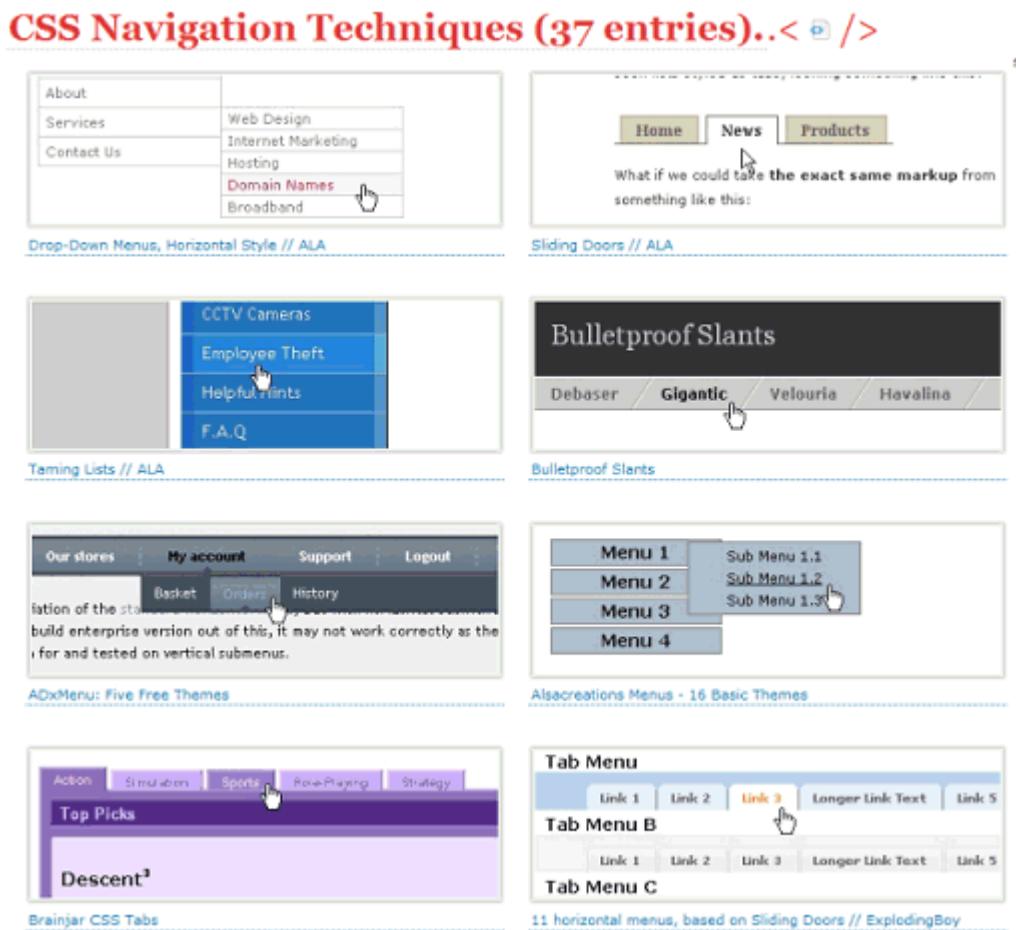


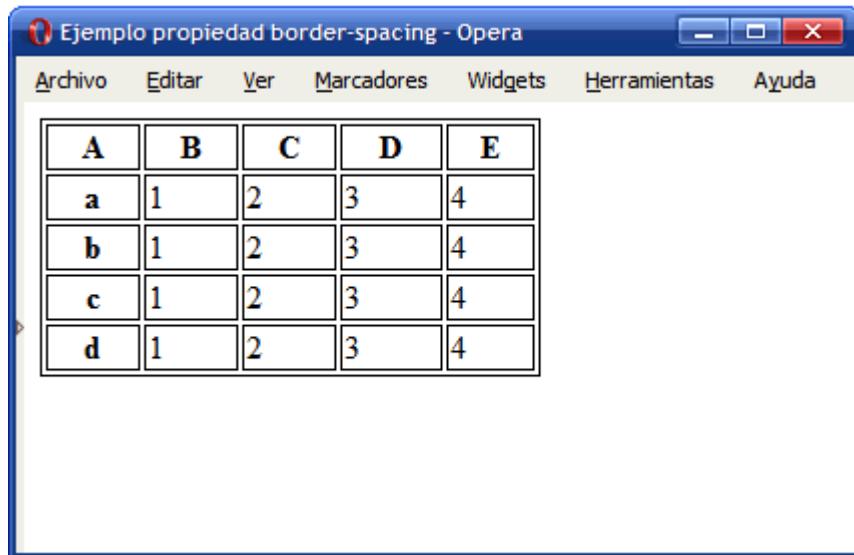
Figura 9.15. Ejemplos de menús horizontales y verticales complejos creados con CSS

El código CSS de todos los ejemplos de la imagen anterior y muchos otros se pueden encontrar en: <http://alvit.de/css-showcase/css-navigation-techniques-showcase.php>

Capítulo 10. Tablas

10.1. Estilos básicos

Cuando se aplican bordes a las celdas de una tabla, el aspecto por defecto con el que se muestra en un navegador es el siguiente:



The screenshot shows a 5x5 grid of cells. The columns are labeled A through E at the top, and the rows are labeled a through d on the left. Each cell contains a value: A1, B1, C1, D1, E1; A2, B2, C2, D2, E2; A3, B3, C3, D3, E3; A4, B4, C4, D4, E4. The table has a border around the entire grid, and each cell has its own individual border.

A	B	C	D	E
a	1	2	3	4
b	1	2	3	4
c	1	2	3	4
d	1	2	3	4

Figura 10.1. Aspecto por defecto de los bordes de una tabla

El código HTML y CSS del ejemplo anterior se muestra a continuación:

```
.normal {  
    width: 250px;  
    border: 1px solid #000;  
}  
.normal th, .normal td {  
    border: 1px solid #000;  
}  
  
<table class="normal" summary="Tabla genérica">  
    <tr>  
        <th scope="col">A</th>  
        <th scope="col">B</th>  
        <th scope="col">C</th>  
        <th scope="col">D</th>  
        <th scope="col">E</th>  
    </tr>  
    ...  
</table>
```

El estándar CSS 2.1 define dos modelos diferentes para el tratamiento de los bordes de las celdas. La propiedad que permite seleccionar el modelo de bordes es `border-collapse`:

border-collapse	Fusión de bordes
Valores	collapse separate inherit
Se aplica a	Todas las tablas
Valor inicial	separate
Descripción	Define el mecanismo de fusión de los bordes de las celdas adyacentes de una tabla

El modelo **collapse** fusiona de forma automática los bordes de las celdas adyacentes, mientras que el modelo **separate** fuerza a que cada celda muestre sus cuatro bordes. Por defecto, los navegadores utilizan el modelo **separate**, tal y como se puede comprobar en el ejemplo anterior.

En general, los diseñadores prefieren el modelo **collapse** porque estéticamente resulta más atractivo y más parecido a las tablas de datos tradicionales. El ejemplo anterior se puede rehacer para mostrar la tabla con bordes sencillos y sin separación entre celdas:

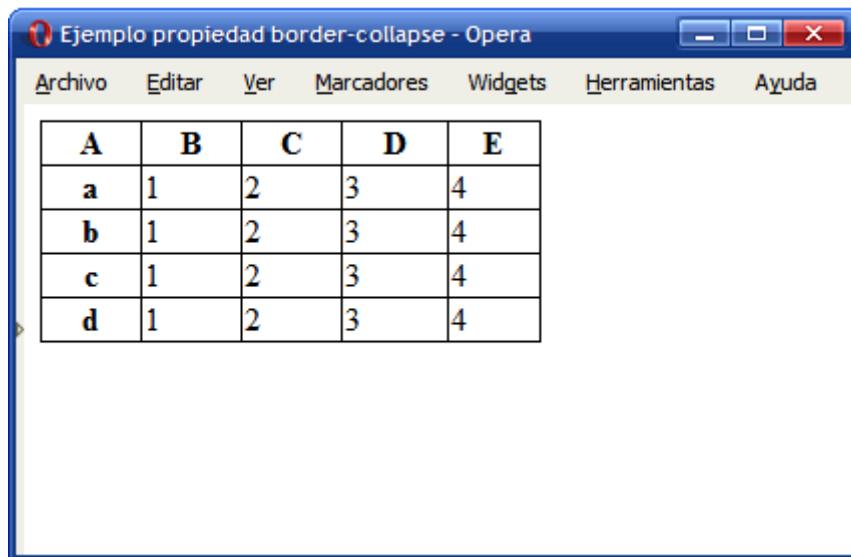


Figura 10.2. Ejemplo de propiedad border-collapse

El código CSS completo del ejemplo anterior se muestra a continuación:

```
.normal {
    width: 250px;
    border: 1px solid #000;
    border-collapse: collapse;
}
.normal th, .normal td {
    border: 1px solid #000;
}

<table class="normal" summary="Tabla genérica">
    <tr>
        <th scope="col">A</th>
        <th scope="col">B</th>
        <th scope="col">C</th>
        <th scope="col">D</th>
```

```

<th scope="col">E</th>
</tr>
...
</table>
```

Aunque parece sencillo, el mecanismo que utiliza el modelo `collapse` es muy complejo, ya que cuando los bordes que se fusionan no son exactamente iguales, debe tener en cuenta la anchura de cada borde, su estilo y el tipo de celda que contiene el borde (columna, fila, grupo de filas, grupo de columnas) para determinar la prioridad de cada borde.

Si se opta por el modelo `separate` (que es el modelo por defecto si no se indica lo contrario) se puede utilizar la propiedad `border-spacing` para controlar la separación entre los bordes de cada celda.

border-spacing	Espaciado entre bordes
Valores	<code><medida> <medida>? inherit</code>
Se aplica a	Todas las tablas
Valor inicial	0
Descripción	Establece la separación entre los bordes de las celdas adyacentes de una tabla

Si solamente se indica como valor una medida, se asigna ese valor como separación horizontal y vertical. Si se indican dos medidas, la primera es la separación horizontal y la segunda es la separación vertical entre celdas.

La propiedad `border-spacing` sólo controla la separación entre celdas y por tanto, no se puede utilizar para modificar el tipo de modelo de bordes que se utiliza. En concreto, si se establece un valor igual a 0 para la separación entre los bordes de las celdas, el resultado es muy diferente al modelo `collapse`:

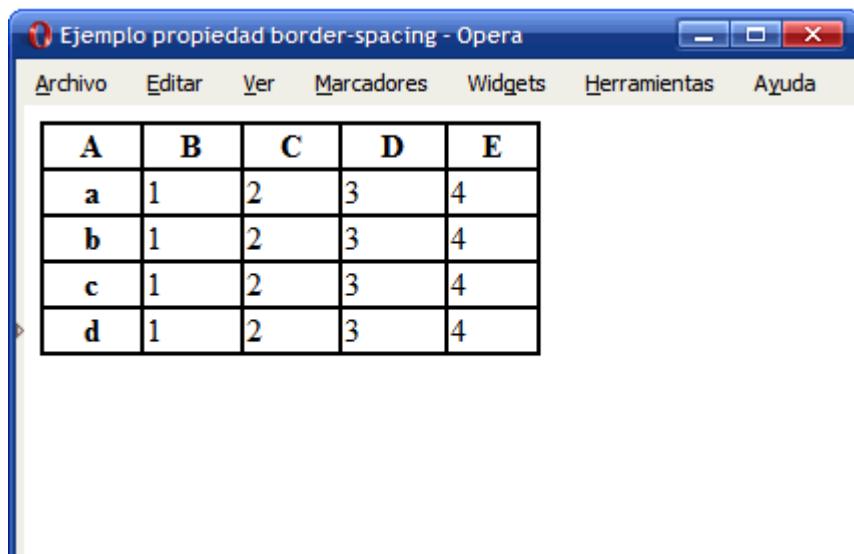


Figura 10.3. Ejemplo de propiedad border-spacing

En la tabla del ejemplo anterior, se ha establecido la propiedad `border-spacing: 0`, por lo que el navegador no introduce ninguna separación entre los bordes de las celdas. Además, como no se

ha establecido de forma explícita ningún modelo de bordes, el navegador utiliza el modelo **separate**.

El resultado es que **border-spacing: 0** muestra los bordes con una anchura doble, ya que en realidad se trata de dos bordes iguales sin separación entre ellos. Por tanto, las diferencias visuales con el modelo **border-collapse: collapse** son muy evidentes.

10.2. Estilos avanzados

CSS define otras propiedades específicas para el control del aspecto de las tablas. Una de ellas es el tratamiento que reciben las celdas vacías de una tabla, que se controla mediante la propiedad **empty-cells**.

empty-cells	Tratamiento de las celdas vacías
Valores	<code>show hide inherit</code>
Se aplica a	Celdas de una tabla
Valor inicial	<code>show</code>
Descripción	Define el mecanismo utilizado para el tratamiento de las celdas vacías de una tabla

El valor `hide` indica que las celdas vacías no se deben mostrar. Una celda vacía es aquella que no tiene ningún contenido, ni siquiera un espacio en blanco o un `&nbsp`.

La siguiente imagen muestra las diferencias entre una tabla normal y una tabla con la propiedad **empty-cells: hide**:

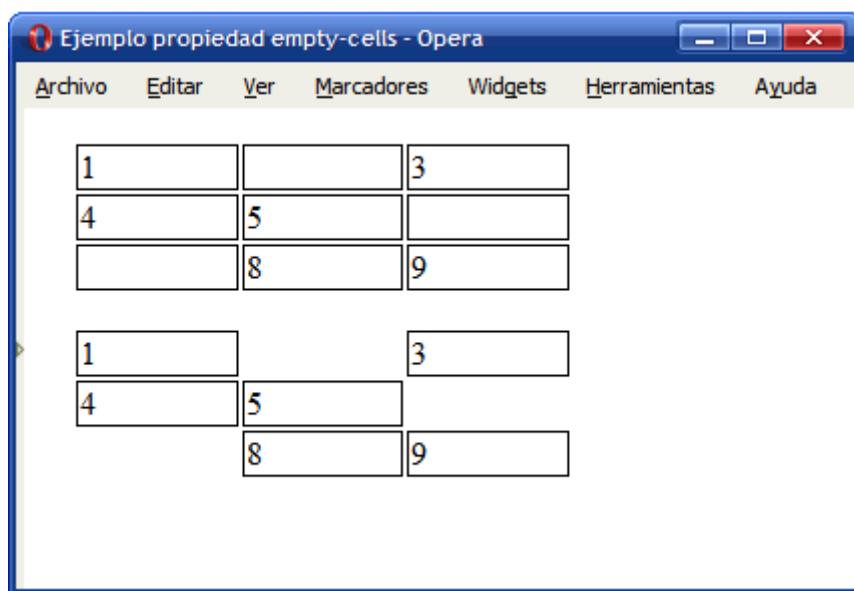


Figura 10.4. Ejemplo de propiedad empty-cells

Desafortunadamente, Internet Explorer 6 y las versiones anteriores no interpretan correctamente esta propiedad y muestran el ejemplo anterior de la siguiente manera:

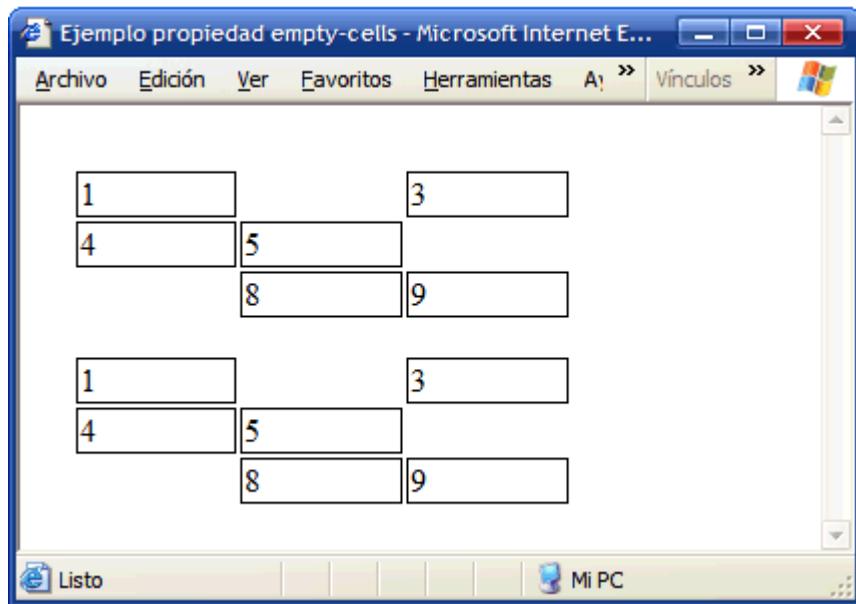


Figura 10.5. Internet Explorer no soporta la propiedad empty-cells

Por otra parte, el título de las tablas se establece mediante el elemento <caption>, que por defecto se muestra encima de los contenidos de la tabla. La propiedad **caption-side** permite controlar la posición del título de la tabla.

caption-side	Posición del título de la tabla
Valores	top bottom inherit
Se aplica a	Los elementos caption
Valor inicial	top
Descripción	Establece la posición del título de la tabla

El valor **bottom** indica que el título de la tabla se debe mostrar después de los contenidos de la tabla. La alineación horizontal se controla mediante la propiedad **text-align**.

A continuación se muestra el código HTML y CSS de un ejemplo sencillo de uso de la propiedad **caption-side**:

```
.especial {
    caption-side: bottom;
}

<table class="especial" summary="Tabla genérica">
    <caption>Tabla 2.- Título especial</caption>
    <tr>
        <td>1</td>
        <td>2</td>
        <td>3</td>
    </tr>
    ...
</table>
```

Desafortunadamente, el navegador Opera 9 y versiones anteriores y el navegador Internet Explorer 6 y versiones anteriores no soportan esta propiedad y muestran el título de la tabla siempre encima de sus contenidos:

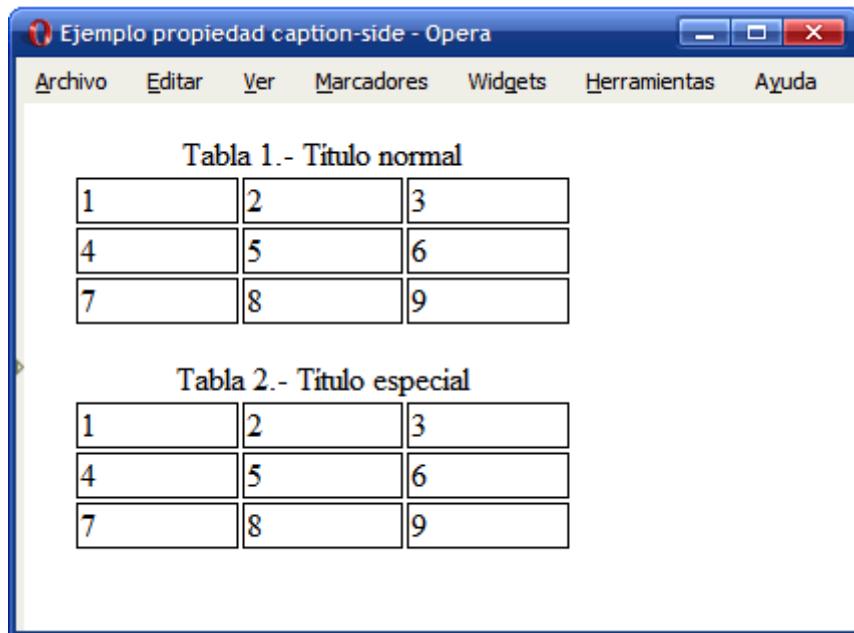


Figura 10.6. Opera e Internet Explorer no soportan la propiedad caption-side

El navegador Firefox sí que soporta esta propiedad y muestra el título de la segunda tabla debajo de sus contenidos, tal y como se ha indicado en el ejemplo:

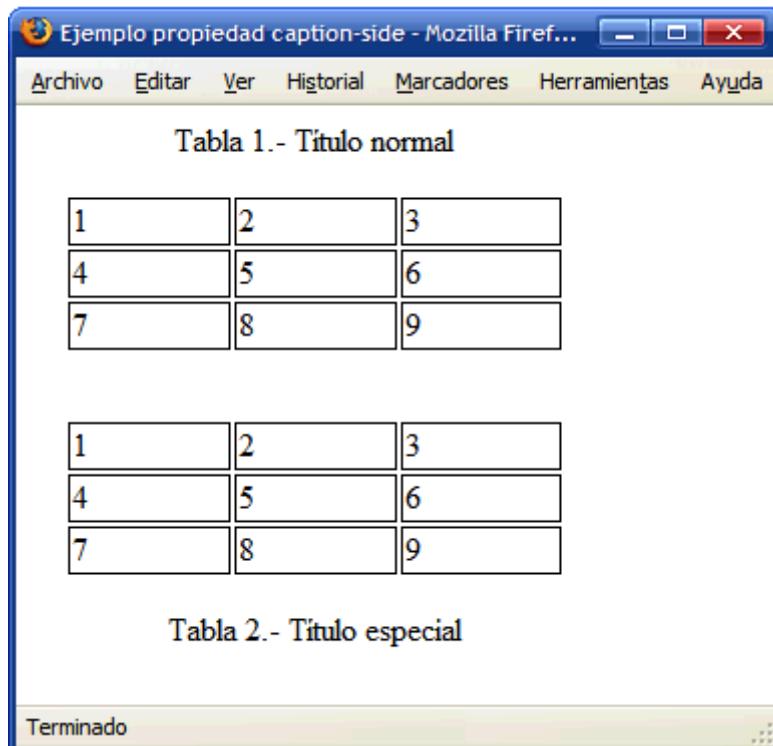


Figura 10.7. Ejemplo de propiedad caption-side

Ejercicio 11 Ver enunciado en la página 204

El resultado final del ejercicio anterior se podría completar añadiendo una pequeña mejora: que el color de las filas varíe cuando el usuario pasa el ratón por encima de cada fila. La *pseudo-clase* :hover permite añadir fácilmente esta característica:

Cambio	Compra	Venta	Máximo	Mínimo
€ Euro/Dolar	1.2524	1.2527	1.2539	1.2488
\$ Dolar/Yen	119.01	119.05	119.82	119.82
£ Libra/Dolar	1.8606	1.8611	1.8651	1.8522
¥ Yen/Euro	0.6711	0.6705	0.6676	0.6713

Figura 10.8. Pseudo-clase :hover en las filas de una tabla

La regla CSS necesaria se muestra a continuación:

```
table tr:hover {  
    background: #FFFF66;  
}
```

Desafortunadamente, Internet Explorer 6 y las versiones anteriores no soportan la *pseudo-clase* :hover en elementos diferentes a los enlaces, por lo que se debe recurrir a soluciones con JavaScript para mostrar de otro color la fila activa.

Capítulo 11. Formularios

11.1. Estilos básicos

11.1.1. Mostrar un botón como un enlace

Como ya se vio anteriormente, el estilo por defecto de los enlaces se puede modificar para que se muestren como botones de formulario. Ahora, los botones de formulario también se pueden modificar para que parezcan enlaces.

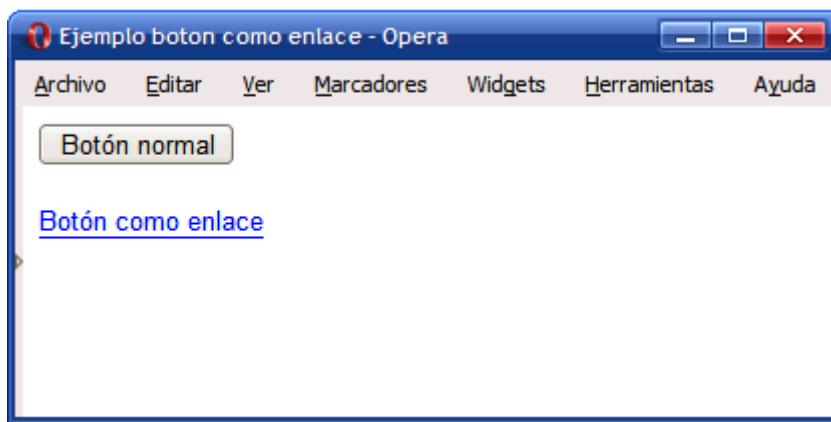


Figura 11.1. Mostrando un botón como si fuera un enlace

Las reglas CSS del ejemplo anterior son las siguientes:

```
.enlace {  
    border: 0;  
    padding: 0;  
    background-color: transparent;  
    color: blue;  
    border-bottom: 1px solid blue;  
}  
  
<input type="button" value="Botón normal" />  
  
<input class="enlace" type="button" value="Botón como enlace" />
```

11.1.2. Mejoras en los campos de texto

Por defecto, los campos de texto de los formularios no incluyen ningún espacio de relleno, por lo que el texto introducido por el usuario aparece *pegado* a los bordes del cuadro de texto.

Añadiendo un pequeño *padding* a cada elemento `<input>`, se mejora notablemente el aspecto del formulario:

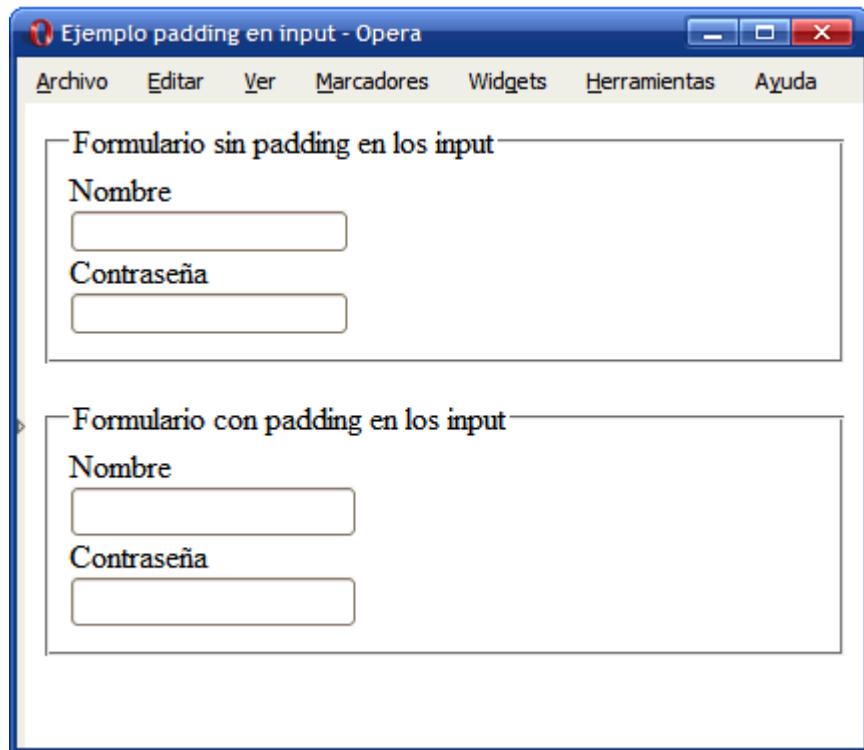


Figura 11.2. Mejorando el aspecto de los formularios gracias al padding

La regla CSS necesaria para mejorar el formulario es muy sencilla:

```
form.elegante input {  
    padding: .2em;  
}
```

11.1.3. Labels alineadas y formateadas

Los elementos `<input>` y `<label>` de los formularios son elementos en línea, por lo que el aspecto que muestran los formularios por defecto, es similar al de la siguiente imagen:

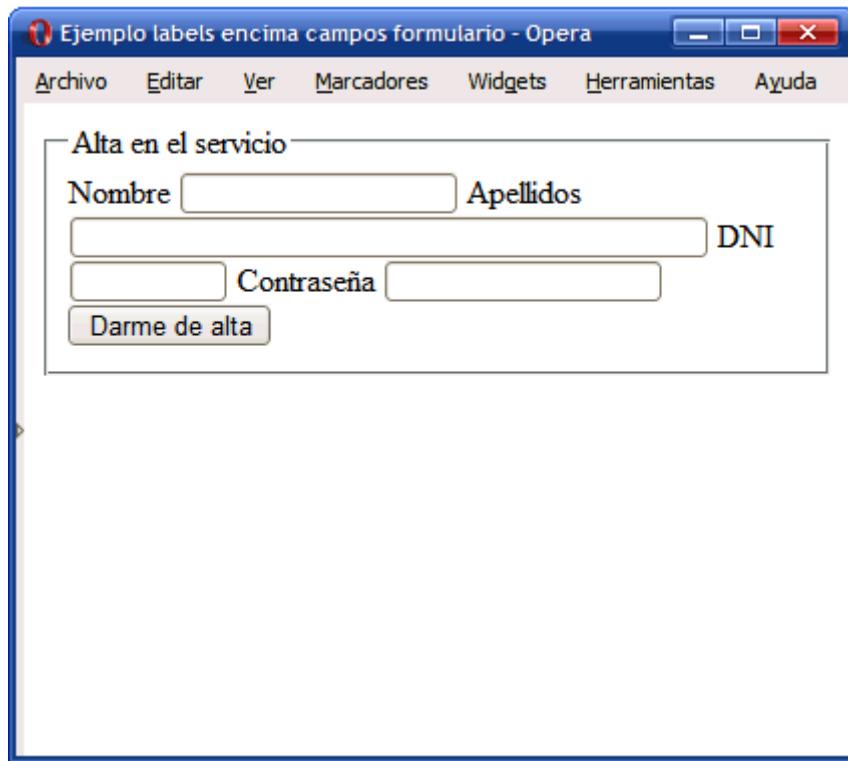


Figura 11.3. Aspecto por defecto que muestran los formularios

El código HTML del ejemplo anterior es el siguiente:

```
<form>
<fieldset>
  <legend>Alta en el servicio</legend>

  <label for="nombre">Nombre</label>
  <input type="text" id="nombre" />

  <label for="apellidos">Apellidos</label>
  <input type="text" id="apellidos" size="50" />

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

  <label for="contrasena">Contraseña</label>
  <input type="password" id="contrasena" />

  <input class="btn" type="submit" value="Darle de alta" />
</fieldset>
</form>
```

Aprovechando los elementos `<label>`, se pueden aplicar unos estilos CSS sencillos que permitan mostrar el formulario con el aspecto de la siguiente imagen:

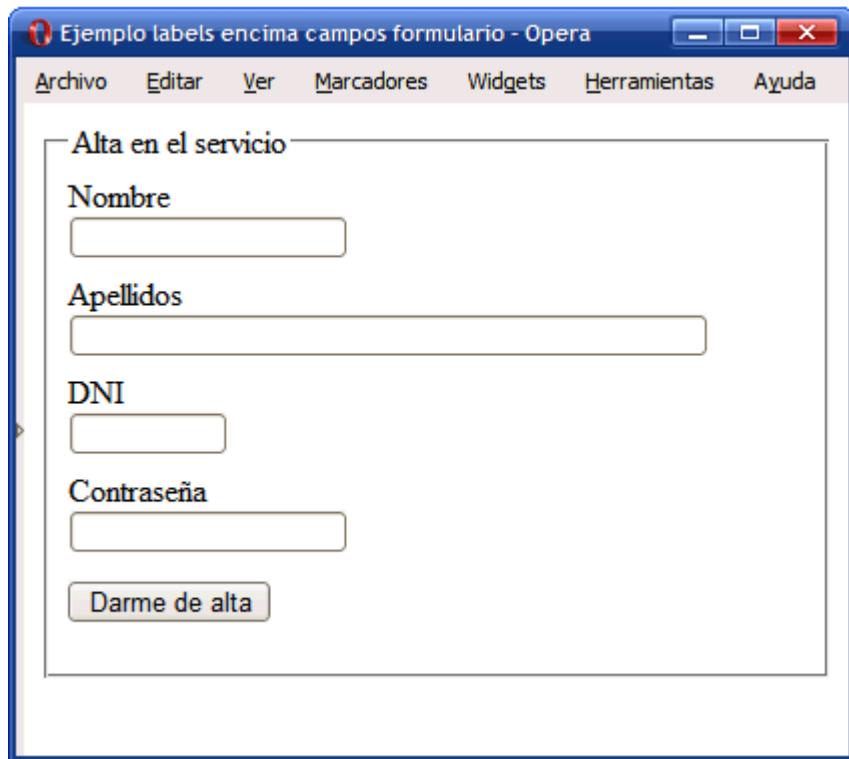


Figura 11.4. Mostrando las etiquetas label encima de los campos del formulario

En primer lugar, se muestran los elementos `<label>` como elementos de bloque, para que añadan una separación para cada campo del formulario. Además, se añade un margen superior para no mostrar juntas todas las filas del formulario:

```
label {  
    display: block;  
    margin: .5em 0 0 0;  
}
```

El botón del formulario también se muestra como un elemento de bloque y se le añade un margen para darle el aspecto final deseado:

```
.btn {  
    display: block;  
    margin: 1em 0;  
}
```

En ocasiones, es más útil mostrar todos los campos del formulario con su `<label>` alineada a la izquierda y el campo del formulario a la derecha de cada `<label>`, como muestra la siguiente imagen:

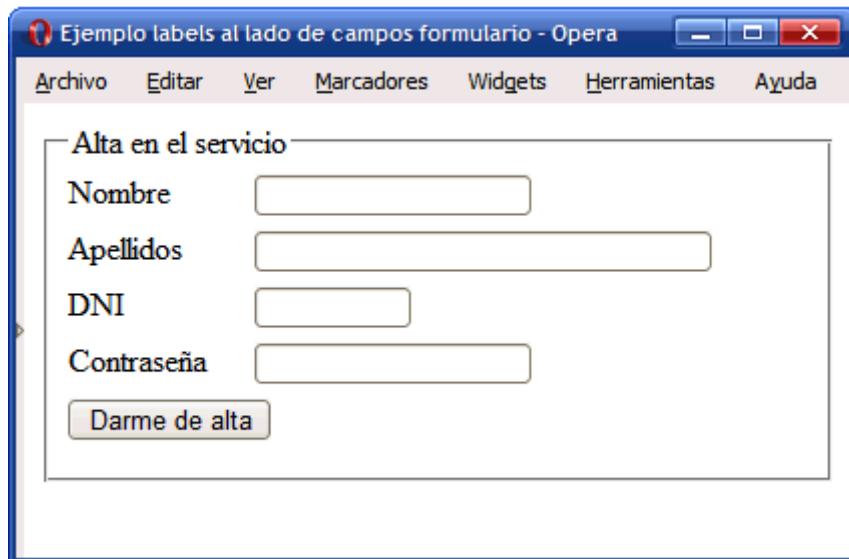


Figura 11.5. Mostrando las etiquetas label alineadas con los campos del formulario

Para mostrar un formulario tal y como aparece en la imagen anterior no es necesario crear una tabla y controlar la anchura de sus columnas para conseguir una alineación perfecta. Sin embargo, sí que es necesario añadir un nuevo elemento (`<div>`) que encierre a cada uno de los campos del formulario (`<label>` y `<input>`). El esquema de la solución propuesta es el siguiente:

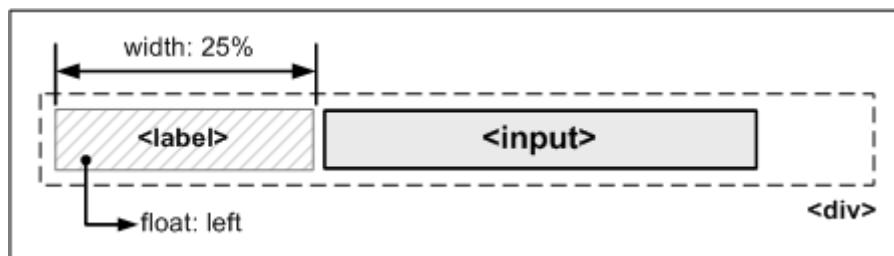


Figura 11.6. Esquema de la técnica de alineación de etiquetas label y campos de formulario

Por tanto, en el código HTML del formulario anterior se añaden los elementos `<div>`:

```

<form>
  <fieldset>
    <legend>Alta en el servicio</legend>

    <div>
      <label for="nombre">Nombre</label>
      <input type="text" id="nombre" />
    </div>

    <div>
      <label for="apellidos">Apellidos</label>
      <input type="text" id="apellidos" size="35" />
    </div>
    ...
  </fieldset>
</form>

```

Y en el código CSS se añaden las reglas necesarias para alinear los campos del formulario:

```
div {  
    margin: .4em 0;  
}  
div label {  
    width: 25%;  
    float: left;  
}
```

11.2. Estilos avanzados

11.2.1. Formulario en varias columnas

Los formularios complejos con decenas de campos pueden ocupar mucho espacio en la ventana del navegador. Además del uso de solapas para agrupar los campos relacionados en un formulario, también es posible mostrar el formulario a dos columnas, para aprovechar mejor el espacio.

The screenshot shows a window titled "Ejemplo formulario 2 columnas - Opera". The menu bar includes Archivo, Editar, Ver, Marcadores, Widgets, Herramientas, and Ayuda. The main content area contains two side-by-side `<fieldset>` elements. The left fieldset is labeled "Alta en el servicio" and contains four input fields: "Nombre", "Apellidos", "DNI", and "Contraseña". The right fieldset is labeled "Datos de contacto" and also contains four input fields: "Teléfono", "Email", "Dirección", and "Código Postal". Below these two columns is a single "Dar me de alta" button.

Figura 11.7. Ejemplo de formulario a dos columnas

La solución consiste en añadir un elemento `<div>` que agrupe a cada elemento `<fieldset>` y aplicar las siguientes reglas CSS:

```
form div {  
    display: inline;  
    float: left;  
    width: 49%;  
}  
  
<form>  
    <div>  
        <fieldset>
```

```
...
</fieldset>
</div>

...
</form>
```

11.2.2. Resaltar el campo seleccionado

Una de las mejoras más útiles para los formularios HTML consiste en resaltar de alguna forma especial el campo en el que el usuario está introduciendo datos. Para ello, CSS define la pseudo-clase `:focus`, que permite aplicar estilos especiales al elemento que en ese momento tiene el *foco* o atención del usuario.

La siguiente imagen muestra un formulario que resalta claramente el campo en el que el usuario está introduciendo la información:

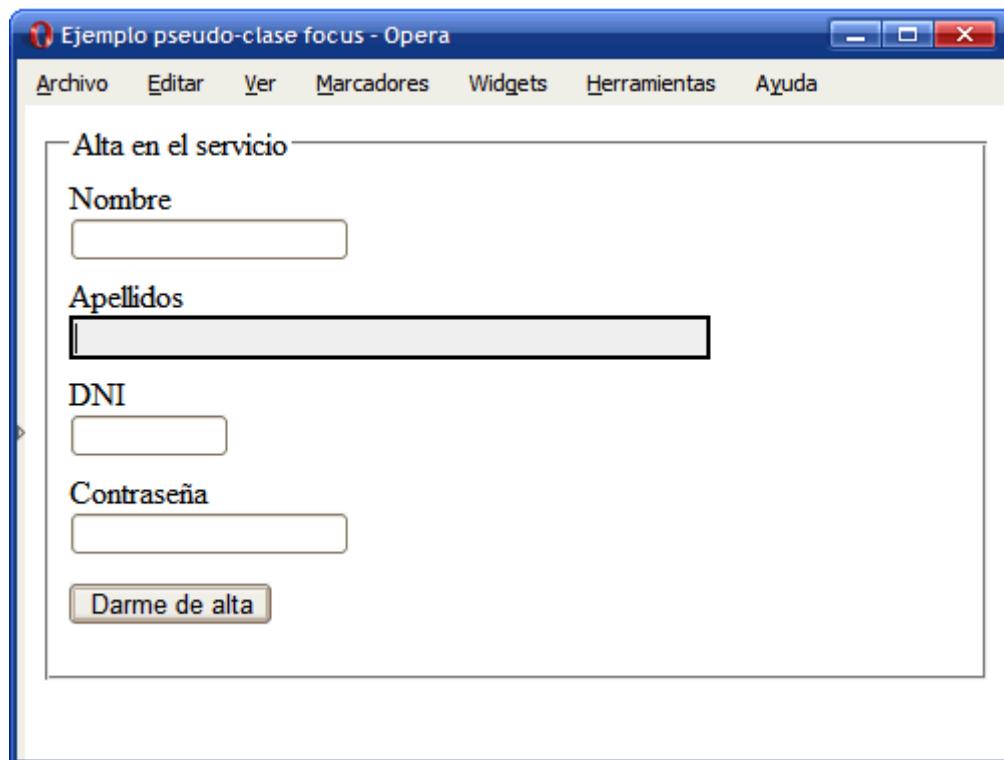


Figura 11.8. Ejemplo de pseudo-clase :focus

Añadiendo la pseudo-clase `:focus` después del selector normal, el navegador se encarga de aplicar esos estilos cuando el usuario activa el elemento:

```
input:focus {
    border: 2px solid #000;
    background: #F3F3F3;
}
```

Desafortunadamente, la pseudo-clase `:focus` no funciona en navegadores obsoletos como Internet Explorer 6, por lo que si la página debe visualizarse de la misma forma en todos los navegadores, es preciso recurrir a soluciones con JavaScript.

Ejercicio 12 Ver enunciado en la página 206

Capítulo 12. Layout

El diseño de las páginas web habituales se divide en bloques: cabecera, menú, contenidos y pie de página. Visualmente, los bloques se disponen en varias filas y columnas. Por este motivo, hace varios años la estructura de las páginas HTML se definía mediante tablas.

El desarrollo de CSS ha permitido que se puedan realizar los mismos diseños sin utilizar tablas HTML. Las principales ventajas de diseñar la estructura de las páginas web con CSS en vez de con tablas HTML son las siguientes:

- Mantenimiento: una página diseñada exclusivamente con CSS es mucho más fácil de mantener que una página diseñada con tablas. Cambiar el aspecto de una página creada con CSS es tan fácil como modificar unas pocas reglas en las hojas de estilos. Sin embargo, realizar la misma modificación en una página creada con tablas supone un esfuerzo muy superior y es más probable cometer errores.
- Accesibilidad: las páginas creadas con CSS son más accesibles que las páginas diseñadas con tablas. De hecho, los navegadores que utilizan las personas discapacitadas (en especial las personas invidentes) pueden tener dificultades con la estructura de las páginas complejas creadas con tablas HTML. No obstante, diseñar una página web exclusivamente con CSS no garantiza que la página sea accesible.
- Velocidad de carga: el código HTML de una página diseñada con tablas es mayor que el código de la misma página diseñada exclusivamente con CSS. En cualquier caso, si el usuario accede al sitio con una conexión de banda ancha y la página es de un tamaño medio o reducido, las diferencias son casi imperceptibles.
- Semántica: aunque resulta obvio, las tablas HTML sólo se deben utilizar para mostrar datos cuya información sólo se entiende en forma de filas y columnas. Utilizar tablas para crear la estructura completa de una página es tan absurdo como utilizar por ejemplo la etiqueta `` para crear párrafos de texto.

Por estos motivos, la estructura basada en tablas ha dado paso a la estructura basada exclusivamente en CSS. Aunque crear la estructura de las páginas sólo con CSS presenta en ocasiones retos importantes, en general es más sencilla y flexible.

En este capítulo se muestra cómo crear algunas de las estructuras o *layouts* más habituales de los diseños web utilizando exclusivamente CSS.

12.1. Centrar una página horizontalmente

A medida que aumenta el tamaño y la resolución de las pantallas de ordenador, se hace más difícil diseñar páginas que se adapten al tamaño de la ventana del navegador. El principal reto que se presenta con resoluciones superiores a 1024 x 768 píxel, es que las líneas de texto son demasiado largas como para leerlas con comodidad. Por ese motivo, normalmente se opta por diseños con una anchura fija limitada a un valor aceptable para mantener la legibilidad del texto.

Por otra parte, los navegadores alinean por defecto las páginas web a la izquierda de la ventana. Cuando la resolución de la pantalla es muy grande, la mayoría de páginas de anchura fija alineadas a la izquierda parecen muy estrechas y provocan una sensación de vacío.

La solución más sencilla para evitar los grandes espacios en blanco consiste en crear páginas con una anchura fija adecuada y centrar la página horizontalmente respecto de la ventana del navegador. Las siguientes imágenes muestran el aspecto de una página centrada a medida que aumenta la anchura de la ventana del navegador.



Figura 12.1. Página de anchura fija centrada mediante CSS



Figura 12.2. Página de anchura fija centrada mediante CSS



Figura 12.3. Página de anchura fija centrada mediante CSS

Utilizando la propiedad `margin` de CSS, es muy sencillo centrar una página web horizontalmente. La solución consiste en agrupar todos los contenidos de la página en un elemento `<div>` y asignarle a ese `<div>` unos márgenes laterales automáticos. El `<div>` que encierra los contenidos se suele llamar **contenedor** (en inglés se denomina *wrapper* o *container*):

```
#contenedor {  
    width: 300px;  
    margin: 0 auto;  
}  
  
<body>  
    <div id="contenedor">  
        <h1>Lorem ipsum dolor sit amet</h1>  
        ...  
    </div>  
</body>
```

Como se sabe, el valor `0 auto` significa que los márgenes superior e inferior son iguales a `0` y los márgenes laterales toman un valor de `auto`. Cuando se asignan márgenes laterales automáticos a un elemento, los navegadores centran ese elemento respecto de su elemento padre. En este ejemplo, el elemento padre del `<div>` es la propia página (el elemento `<body>`), por lo que se consigue centrar el elemento `<div>` respecto de la ventana del navegador.

Modificando ligeramente el código CSS anterior se puede conseguir un diseño dinámico o *líquido* que se adapta a la anchura de la ventana del navegador y permanece siempre centrado:

```
#contenedor {  
    width: 70%;  
    margin: 0 auto;  
}
```

Estableciendo la anchura del elemento contenedor mediante un porcentaje, su anchura se adapta de forma continua a la anchura de la ventana del navegador. De esta forma, si se reduce la anchura de la ventana del navegador, la página se verá más estrecha y si se maximiza la ventana del navegador, la página se verá más ancha.

Las siguientes imágenes muestran cómo se adapta el diseño dinámico a la anchura de la ventana del navegador, mostrando cada vez más contenidos a medida que se agranda la ventana.



Figura 12.4. Página de anchura variable centrada mediante CSS



Figura 12.5. Página de anchura variable centrada mediante CSS



Figura 12.6. Página de anchura variable centrada mediante CSS

12.2. Centrar una página verticalmente

Cuando se centra una página web de forma horizontal, sus márgenes laterales se adaptan dinámicamente de forma que la página siempre aparece en el centro de la ventana del navegador, independientemente de la anchura de la ventana. De la misma forma, cuando se centra una página web verticalmente, el objetivo es que sus contenidos aparezcan en el centro de la ventana del navegador y por tanto, que sus márgenes verticales se adapten de forma dinámica en función del tamaño de la ventana del navegador.

Aunque centrar una página web horizontalmente es muy sencillo, centrarla verticalmente es mucho más complicado. Afortunadamente, no es muy común que una página web aparezca centrada de forma vertical. El motivo es que la mayoría de páginas web son más altas que la ventana del navegador, por lo que no es posible centrarlas verticalmente.

A continuación se muestra la forma de centrar una página web respecto de la ventana del navegador, es decir, centrarla tanto horizontalmente como verticalmente.

Siguiendo el mismo razonamiento que el planteado para centrar la página horizontalmente, se podrían utilizar las siguientes reglas CSS para centrar la página respecto de la ventana del navegador:

```
#contenedor {  
    width: 300px;  
    height: 250px;  
    margin: auto;  
}  
  
<body>  
    <div id="contenedor">  
        <h1>Lorem ipsum dolor sit amet</h1>  
        ...  
    </div>  
</body>
```

Si el valor `auto` se puede utilizar para que los márgenes laterales se adapten dinámicamente, también debería ser posible utilizar el valor `auto` para los márgenes verticales. Desafortunadamente, la propiedad `margin: auto` no funciona tal y como se espera para los márgenes verticales y la página no se muestra centrada.

La solución correcta para centrar verticalmente una página web se basa en el posicionamiento absoluto e implica realizar un cálculo matemático sencillo. A continuación se muestra el esquema gráfico de los cuatro pasos necesarios para centrar una página web en la ventana del navegador:

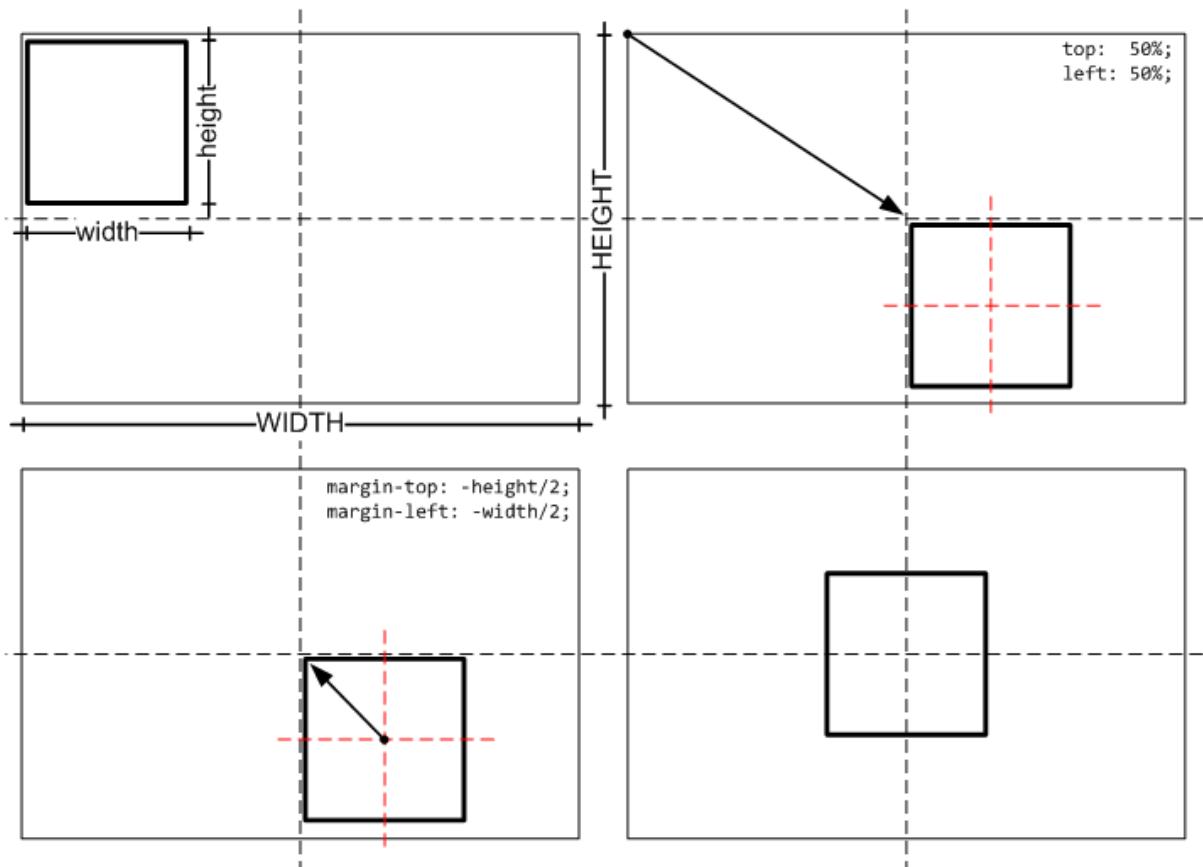


Figura 12.7. Pasos necesarios para centrar verticalmente una página web

En primer lugar, se asigna una altura y una anchura al elemento que encierra todos los contenidos de la página. En la primera figura, los contenidos de la página tienen una anchura llamada `width` y una altura llamada `height` que son menores que la anchura y altura de la ventana del navegador. En el siguiente ejemplo, se supone que tanto la anchura como la altura de la página es igual a 500px:

```
#contenedor {
    width: 500px;
    height: 500px;
}

<body>
    <div id="contenedor">
        <h1>Lorem ipsum dolor sit amet</h1>
        ...
    </div>
</body>
```

A continuación, se posiciona de forma absoluta el elemento `contenedor` y se asigna un valor de 50% tanto a la propiedad `top` como a la propiedad `left`. El resultado es que la esquina superior izquierda del elemento `contenedor` se posiciona en el centro de la ventana del navegador:

```
#contenedor {
    width: 500px;
    height: 500px;
```

```
position: absolute;  
top: 50%;  
left: 50%;  
}
```

Si la página se debe mostrar en el centro de la ventana del navegador, es necesario desplazar hacia arriba y hacia la izquierda los contenidos de la página web. Para determinar el desplazamiento necesario, se realiza un cálculo matemático sencillo. Como se ve en la tercera figura del esquema anterior, el punto central de la página debe desplazarse hasta el centro de la ventana del navegador.

Como se desprende de la imagen anterior, la página web debe moverse hacia arriba una cantidad igual a la mitad de su altura y debe desplazarse hacia la izquierda una cantidad equivalente a la mitad de su anchura. Utilizando las propiedades `margin-top` y `margin-left` con valores negativos, la página se desplaza hasta el centro de la ventana del navegador.

```
#contenedor {  
    width: 500px;  
    height: 500px;  
  
    position: absolute;  
    top: 50%;  
    left: 50%;  
  
    margin-top: -250px; /* height/2 = 500px / 2 */  
    margin-left: -250px; /* width/2 = 500px / 2 */  
}
```

Con las reglas CSS anteriores, la página web siempre aparece centrada verticalmente y horizontalmente respecto de la ventana del navegador. El motivo es que la anchura/altura de la página son fijas (propiedades `width` y `height`), el desplazamiento necesario para centrarla también es fijo (propiedades `margin-top` y `margin-left`) y el desplazamiento hasta el centro de la ventana del navegador se calcula dinámicamente gracias al uso de porcentajes en las propiedades `top` y `left`.

Para centrar una página sólo verticalmente, se debe prescindir tanto del posicionamiento horizontal como del desplazamiento horizontal:

```
#contenedor {  
    width: 500px;  
    height: 500px;  
  
    position: absolute;  
    top: 50%;  
  
    margin-top: -250px; /* height/2 = 500px / 2 */  
}
```

12.3. Estructura o layout

12.3.1. Diseño a 2 columnas con cabecera y pie de página

El objetivo de este diseño es definir una estructura de página con cabecera y pie, un menú lateral de navegación y una zona de contenidos. La anchura de la página se fija en 700px, la anchura del menú es de 150px y la anchura de los contenidos es de 550px:

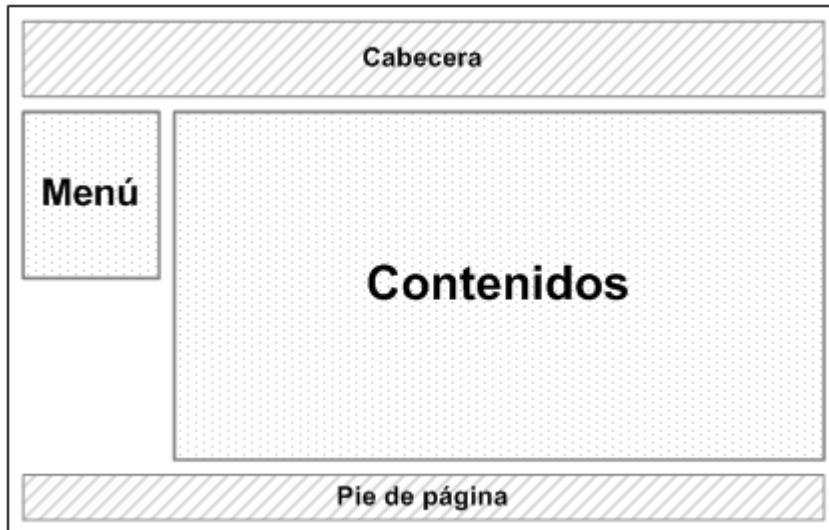


Figura 12.8. Esquema del diseño a 2 columnas con cabecera y pie de página

La solución CSS se basa en el uso de la propiedad `float` para los elementos posicionados como el menú y los contenidos y el uso de la propiedad `clear` en el pie de página para evitar los solapamientos ocasionados por los elementos posicionados con `float`.

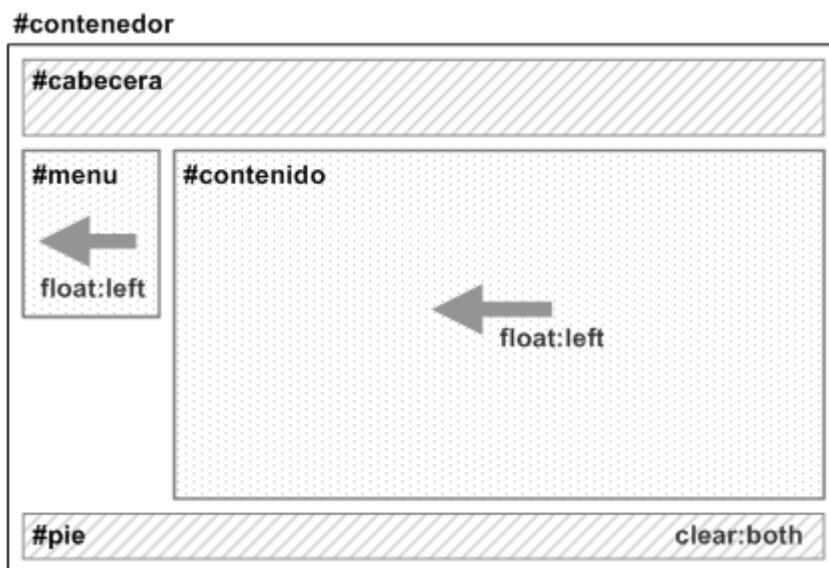


Figura 12.9. Propiedades CSS necesarias en el diseño a dos columnas con cabecera y pie de página

El código HTML y CSS mínimos para definir la estructura de la página sin aplicar ningún estilo adicional son los siguientes:

```
#contenedor {
    width: 700px;
}
#cabecera {
```

```
}

#menu {
    float: left;
    width: 150px;
}
#contenido {
    float: left;
    width: 550px;
}
#pie {
    clear: both;
}

<body>
<div id="contenedor">
    <div id="cabecera">
        </div>

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

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

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

En los estilos CSS anteriores se ha optado por desplazar tanto el menú como los contenidos hacia la izquierda de la página (`float: left`). Sin embargo, en este caso también se podría desplazar el menú hacia la izquierda (`float:left`) y los contenidos hacia la derecha (`float: right`).

El diseño anterior es de anchura fija, lo que significa que no se adapta a la anchura de la ventana del navegador. Para conseguir una página de anchura variable y que se adapte de forma dinámica a la ventana del navegador, se deben aplicar las siguientes reglas CSS:

```
#contenedor {
}
#cabecera {
}
#menu {
    float: left;
    width: 15%;
}
#contenido {
    float: left;
    width: 85%;
}
#pie {
    clear: both;
}
```

Si se indican la anchuras de los bloques que forman la página en porcentajes, el diseño final es dinámico. Para crear diseños de anchura fija, basta con establecer las anchuras de los bloques en píxel.

12.3.2. Diseño a 3 columnas con cabecera y pie de página

Además del diseño a dos columnas, el diseño más utilizado es el de tres columnas con cabecera y pie de página. En este caso, los contenidos se dividen en dos zonas diferenciadas: zona principal de contenidos y zona lateral de contenidos auxiliares:

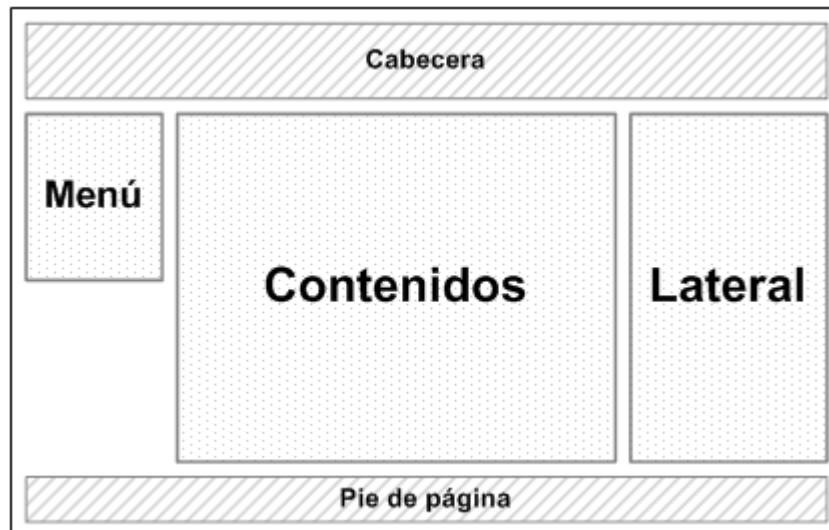


Figura 12.10. Esquema del diseño a tres columnas con cabecera y pie de página

La solución CSS emplea la misma estrategia del diseño a dos columnas y se basa en utilizar las propiedades `float` y `clear`:

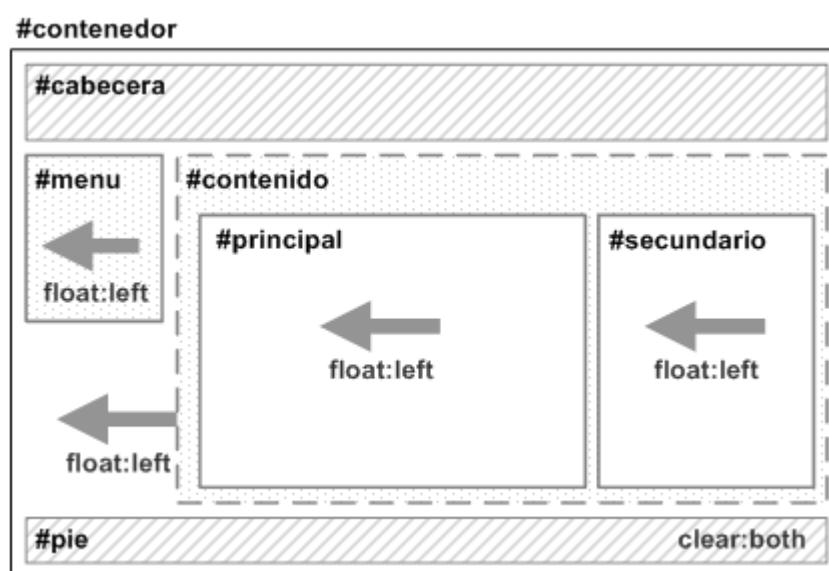


Figura 12.11. Propiedades CSS necesarias en el diseño a 3 columnas con cabecera y pie de página

El código HTML y CSS mínimos para definir la estructura de la página sin aplicar ningún estilo adicional son los siguientes:

```
#contenedor {  
}  
#cabecera {  
}  
#menu {  
    float: left;  
    width: 15%;  
}  
#contenido {  
    float: left;  
    width: 85%;  
}  
#contenido #principal {  
    float: left;  
    width: 80%;  
}  
#contenido #secundario {  
    float: left;  
    width: 20%;  
}  
  
#pie {  
    clear: both;  
}  
  
<body>  
<div id="contenedor">  
    <div id="cabecera">  
        </div>  
  
    <div id="menu">  
        </div>  
  
    <div id="contenido">  
        <div id="principal">  
            </div>  
  
        <div id="secundario">  
            </div>  
        </div>  
  
        <div id="pie">  
            </div>  
    </div>  
</body>
```

El código anterior crea una página con anchura variable que se adapta a la ventana del navegador. Para definir una página con anchura fija, solamente es necesario sustituir las anchuras en porcentajes por anchuras en píxel.

Al igual que sucedía en el diseño a dos columnas, se puede optar por posicionar todos los elementos mediante `float: left` o se puede utilizar `float: left` para el menú y la zona principal de contenidos y `float: right` para el contenedor de los contenidos y la zona secundaria de contenidos.

Ejercicio 13 Ver enunciado en la página 212

12.4. Alturas/anchuras máximas y mínimas

Cuando se diseña la estructura de una página web, se debe tomar la decisión de optar por un diseño de anchura fija o un diseño cuya anchura se adapta a la anchura de la ventana del navegador.

Sin embargo, la mayoría de las veces sería conveniente una solución intermedia: que la anchura de la página sea variable y se adapte a la anchura de la ventana del navegador, pero respetando ciertos límites. En otras palabras, que la anchura de la página no sea tan pequeña como para que no se puedan mostrar correctamente los contenidos y tampoco sea tan ancha como para que las líneas de texto no puedan leerse cómodamente.

CSS define cuatro propiedades que permiten limitar la anchura y altura mínima y máxima de cualquier elemento de la página. Las propiedades son `max-width`, `min-width`, `max-height` y `min-height`.

max-width	Anchura máxima
Valores	<code><medida></code> <code><porcentaje></code> <code>none</code> <code>inherit</code>
Se aplica a	Todos los elementos salvo filas y grupos de filas de tablas
Valor inicial	<code>none</code>
Descripción	Permite definir la anchura máxima de un elemento

min-width	Anchura mínima
Valores	<code><medida></code> <code><porcentaje></code> <code>inherit</code>
Se aplica a	Todos los elementos salvo filas y grupos de filas de tablas
Valor inicial	<code>0</code>
Descripción	Permite definir la anchura mínima de un elemento

max-height	Altura máxima
Valores	<code><medida></code> <code><porcentaje></code> <code>none</code> <code>inherit</code>
Se aplica a	Todos los elementos salvo columnas y grupos de columnas de tablas
Valor inicial	<code>none</code>
Descripción	Permite definir la altura máxima de un elemento

min-height	Altura mínima
Valores	<medida> <porcentaje> inherit
Se aplica a	Todos los elementos salvo columnas y grupos de columnas de tablas
Valor inicial	0
Descripción	Permite definir la altura mínima de un elemento

De esta forma, para conseguir un diseño de anchura variable pero controlada, se podrían utilizar reglas CSS como la siguiente:

```
#contenedor {
    min-width: 500px;
    max-width: 900px;
}
```

Las propiedades que definen la altura y anchura máxima y mínima se pueden aplicar a cualquier elemento, aunque solamente suelen utilizarse para estructurar la página. En general, las propiedades más utilizadas son `max-width` y `min-width`, ya que no es muy habitual definir alturas máximas y mínimas.

Desafortunadamente, Internet Explorer 6 y las versiones anteriores no soportan ninguna de las cuatro propiedades sobre ningún elemento. Hasta que se incorpore en las nuevas versiones del navegador, es preciso recurrir a trucos que simulen el comportamiento de las propiedades:

`max-width` equivalente para Internet Explorer:

```
div {
    max-width: 800px;
    width: expression(document.body.clientWidth > 801? "800px": "auto");
}
```

`min-width` equivalente para Internet Explorer:

```
div {
    min-width:800px;
    width: expression(document.body.clientWidth < 801? "800px": "auto" );
}
```

`max-height` equivalente para Internet Explorer:

```
div {
    max-height: 300px;
    overflow: hidden;
    height: expression(this.scrollHeight > 301? "300px" : "auto" );
}
```

`min-height` equivalente para Internet Explorer:

```
div {
    min-height:300px;
    overflow: hidden;
    height: expression(this.scrollHeight < 301? "300px" : "auto" );
}
```

Los equivalentes para Internet Explorer han sido extraídos de: http://www.svendtofte.com/code/max_width_in_ie/

12.5. Estilos avanzados

En general, la columna de los contenidos es la más larga y la columna de navegación es la más corta. El principal inconveniente de los diseños mostrados anteriormente es que no se puede garantizar que todas las columnas se muestren con la misma altura.

Si las columnas tienen algún color o imagen de fondo, este comportamiento no es admisible, ya que se vería que alguna columna no llega hasta el final de la columna más larga y el diseño final parecería inacabado.

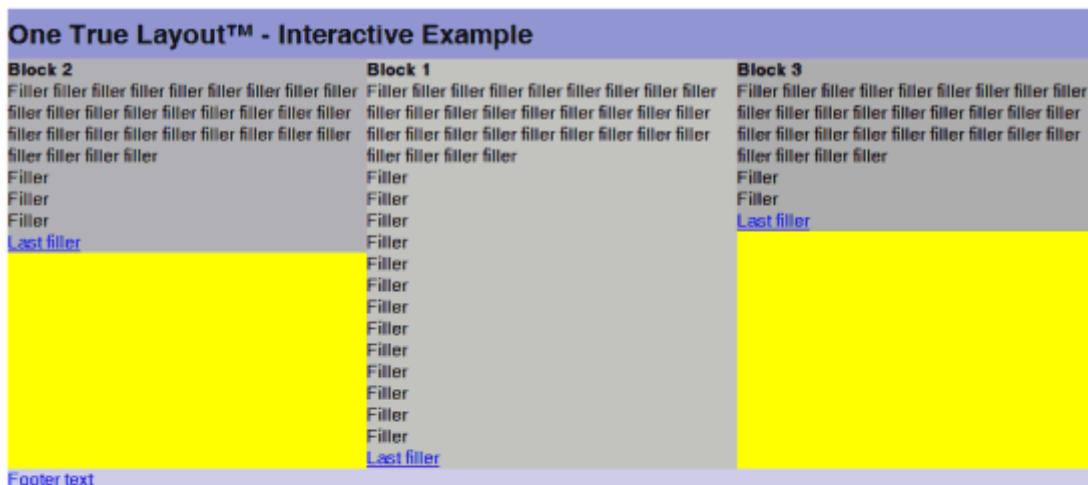
Desde la aparición de este problema se han presentado numerosas soluciones. La más conocida es la técnica *faux columns* ("columnas falsas") y que simula el color/imagen de fondo de las columnas laterales mediante la imagen de fondo de la columna central de contenidos.

La técnica fue presentada originalmente por Dan Cederholm en su célebre artículo "*Faux Columns*" (<http://alistapart.com/articles/fauxcolumns/>).

Más recientemente se ha presentado el proyecto "*In Search of the One True Layout*" que busca definir una serie de técnicas que permitan crear de forma sencilla cualquier estructura de página basada en columnas.

La página principal del proyecto se puede encontrar en: <http://www.positioniseverything.net/articles/onetruelayout/>

Además, está disponible una herramienta interactiva para crear diseños basados en columnas con la posibilidad de definir el número de columnas, su anchura y obligar a que todas las columnas muestren la misma altura:



This example forms part of the Position is Everything article, [In Search of the One True Layout](#).



Figura 12.12. Herramienta online para diseñar layouts de varias columnas con CSS

La herramienta interactiva se puede encontrar en: <http://www.fu2k.org/alex/css/onetruelayout/example/interactive>

Capítulo 13. Otros

13.1. Propiedades shorthand

Las propiedades de tipo "*shorthand*" son propiedades de CSS que permiten establecer de forma simultánea el valor de varias propiedades diferentes pero relacionadas. El uso de las propiedades "*shorthand*" es muy extendido, ya que permiten crear hojas de estilos más compactas.

A continuación se incluye a modo de referencia todas las propiedades de tipo "*shorthand*" que se han mostrado anteriormente.

font	Tipografía
Valores	((<font-style> <font-variant> <font-weight>)? <font-size> (/ <line-height>)? <font-family>) caption icon menu message-box small-caption status-bar inherit
Se aplica a	Todos los elementos
Valor inicial	-
Descripción	Permite indicar de forma directa todas las propiedades de la tipografía de un texto

margin	Margen
Valores	(<medida> <porcentaje> auto) {1, 4} inherit
Se aplica a	Todos los elementos salvo algunos casos especiales de elementos mostrados como tablas
Valor inicial	-
Descripción	Establece de forma directa todos los márgenes de un elemento

padding	Relleno
Valores	(<medida> <porcentaje>){1, 4} inherit
Se aplica a	Todos los elementos excepto algunos elementos de tablas como grupos de cabeceras y grupos de pies de tabla
Valor inicial	-
Descripción	Establece de forma directa todos los rellenos de los elementos

border	Estilo completo de todos los bordes
Valores	(<medida_borde> <color_borde> <estilo_borde>) inherit
Se aplica a	Todos los elementos
Valor inicial	-
Descripción	Establece el estilo completo de todos los bordes de los elementos
background	Fondo de un elemento
Valores	(<background-color> <background-image> <background-repeat> <background-attachment> <background-position>) inherit
Se aplica a	Todos los elementos
Valor inicial	-
Descripción	Establece todas las propiedades del fondo de un elemento
list-style	Estilo de una lista
Valores	(<list-style-type> <list-style-position> <list-style-image>) inherit
Se aplica a	Elementos de una lista
Valor inicial	-
Descripción	Propiedad que permite establecer de forma simultánea todas las opciones de una lista

13.2. Versión para imprimir

La mayoría de sitios web de calidad ofrecen al usuario la posibilidad de imprimir sus contenidos mediante una versión específica para impresora de cada página.

Estas versiones optimizadas para impresora eliminan los contenidos superfluos, modifican o eliminan las imágenes y colores de fondo y sobre todo, optimizan los contenidos de texto para facilitar su lectura.

CSS simplifica al máximo la creación de una versión para imprimir gracias al concepto de los medios CSS. Como se sabe, los estilos CSS que se aplican a los contenidos pueden variar en función del medio a través del que se acceden (pantalla, televisor, móvil, impresora, etc.)

De esta forma, realizar una versión para imprimir de una página HTML es tan sencillo como crear unas cuantas reglas CSS que optimicen sus contenidos para conseguir la mejor impresión.

El sitio web A List Apart (<http://www.alistapart.com>) es un excelente ejemplo de cómo los sitios web de calidad crean versiones específicas para impresora de las páginas web originales. Cuando se visualiza un artículo de ese sitio web en una pantalla normal, su aspecto es el siguiente:



Figura 13.1. Aspecto de un artículo de A List Apart como se ve en la pantalla

Además de sus contenidos, las páginas de los artículos muestran el logotipo del sitio, el menú principal de navegación y una barra lateral con varias utilidades como un buscador.

Sin embargo, cuando se imprime la página del mismo artículo, su aspecto es el que muestra la siguiente imagen:



Figura 13.2. Aspecto de un artículo de A List Apart como se ve cuando se imprime

La página impresa elimina todos los contenidos superfluos como los menús de navegación, el buscador y el formulario para añadir comentarios en el artículo. Además, modifica la estructura de la página para que la zona de contenidos ocupe toda la anchura de la página y el espacio se aproveche mejor.

Crear una versión para imprimir similar a la mostrada en el ejemplo anterior es una tarea que no lleva más de unos pocos minutos.

Las reglas CSS de la versión para imprimir se aplican solamente al medio `print`. Por lo tanto, en primer lugar se crea una nueva hoja de estilos y al enlazarla se especifica que sólo debe aplicarse en las impresoras:

```
| <link rel="stylesheet" type="text/css" href="/css/imprimir.css" media="print" />
```

Normalmente, las hojas de estilos para la pantalla se aplican a todos los medios (por utilizar el valor `media="all"` al enlazarla o por no indicar ningún valor en el atributo `media`). Por este motivo, cuando se imprime una página se aplican los mismos estilos que se aplican al visualizarla en la pantalla.

Aprovechando este comportamiento, las hojas de estilos para impresoras son muy sencillas, ya que sólo deben modificar algunos estilos aplicados en el resto de hojas de estilos.

Por este motivo, normalmente las hojas de estilos para impresora se construyen siguiendo los pasos que se muestran a continuación:

1) Ocultar los elementos que no se van a imprimir:

```
#cabecera, #menu, #lateral, #comentarios {  
    display: none !important;  
}
```

Los bloques (normalmente encerrados en elementos de tipo `<div>`) que no se van a imprimir se ocultan con la propiedad `display` y su valor `none`. La palabra clave `!important` aumenta la prioridad de esta regla CSS y más adelante se explica su significado.

2) Corregir la estructura de la página:

```
body, #contenido, #principal, #pie {  
    float: none !important;  
    width: auto !important;  
    margin: 0 !important;  
    padding: 0 !important;  
}
```

Normalmente, las páginas web complejas están formadas por varias columnas posicionadas mediante la propiedad `float`. Si al imprimir la página se eliminan las columnas laterales, es conveniente reajustar la anchura y el posicionamiento de la zona de contenidos y de otras zonas que sí se van a imprimir.

3) Modificar los colores y tipos de letra:

```
| body { color: #000; font: 100%/150% Georgia, "Times New Roman", Times, serif; }
```

Aunque el uso de impresoras en color es mayoritario, suele ser conveniente imprimir todo el texto de las páginas de color negro, para ahorrar costes y para aumentar el contraste cuando se imprime sobre hojas de color blanco. También suele ser conveniente modificar el tipo de letra y escoger uno que facilite al máximo la lectura del texto.

13.2.1. Imprimiendo los enlaces

Uno de los principales problemas de las páginas HTML impresas es la pérdida de toda la información relativa a los enlaces. En principio, imprimir los enlaces de una página es absurdo porque no se pueden utilizar en el medio impreso.

Sin embargo, lo que puede ser realmente útil es mostrar al lado de un enlace la dirección a la que apunta. De esta forma, al imprimir la página no se pierde la información relativa a los enlaces.

CSS incluye una propiedad llamada `content` que permite crear nuevos contenidos de texto para añadirlos a la página HTML. Si se combina la propiedad `content` y el *pseudo-elemento* `:after`, es posible insertar de forma dinámica la dirección a la que apunta un enlace justo después de su texto:

```
a:after {
  content: " (" attr(href) ")";
}
```

El código CSS anterior añade después de cada enlace de la página un texto formado por la dirección a la que apunta el enlace mostrada entre paréntesis. Si se quiere añadir las direcciones antes de cada enlace, se puede utilizar el *pseudo-elemento* `:before`:

```
a:before {
  content: " (" attr(href) ")";
}
```

13.3. Personalizar el cursor

CSS no permite modificar los elementos propios del navegador o de la interfaz de usuario del sistema operativo. Sin embargo, el puntero del ratón es una excepción muy importante, ya que se puede modificar mediante la propiedad `cursor`.

cursor	Puntero del ratón
Valores	((<url> ,)* (auto crosshair default pointer move e-resize ne-resize nw-resize n-resize se-resize sw-resize s-resize w-resize text wait help progress)) inherit
Se aplica a	Todos los elementos
Valor inicial	auto
Descripción	Permite personalizar el puntero del ratón

La propiedad `cursor` no sólo permite seleccionar un puntero entre los disponibles en el sistema operativo (flecha, mano, reloj de arena, redimensionar, etc.) sino que incluso permite indicar la URL de una imagen que se quiere mostrar como puntero personalizado.

Se pueden indicar varias URL para que CSS intente cargar varias imágenes: si la primera imagen del puntero no se carga o no la soporta el navegador, se pasa a la siguiente imagen y así sucesivamente hasta que se pueda cargar alguna imagen.

El siguiente ejemplo muestra el caso de un puntero definido con varias URL y un valor estándar:

```
| :link, :visited { cursor: url(puntero.svg), url(puntero.cur), pointer }
```

Si el navegador soporta las imágenes en formato SVG, el puntero del ratón cambia su aspecto por la imagen `puntero.svg`. Si el navegador no soporta el formato SVG, intenta cargar la siguiente URL que define un puntero en formato `.cur`. Si no se puede cargar correctamente, se mostraría el valor preestablecido `pointer`.

Los valores preestablecidos para el puntero se muestran a continuación:

Puntero	Navegadores que lo soportan
 Figura 13.3. cursor: default	Todos
 Figura 13.4. cursor: crosshair	Todos
 Figura 13.5. cursor: hand	Solo Internet Explorer
 Figura 13.6. cursor: pointer	Todos salvo Internet Explorer
 Figura 13.7. cursor:pointer; cursor: hand	Todos
 Figura 13.8. cursor: move	Todos
 Figura 13.9. cursor: text	Todos
 Figura 13.10. cursor: wait	Todos
 Figura 13.11. cursor: help	Todos
 Figura 13.12. cursor: n-resize	Todos

 Figura 13.13. cursor: ne-resize	Todos
 Figura 13.14. cursor: e-resize	Todos
 Figura 13.15. cursor: se-resize	Todos
 Figura 13.16. cursor: s-resize	Todos
 Figura 13.17. cursor: sw-resize	Todos
 Figura 13.18. cursor: w-resize	Todos
 Figura 13.19. cursor: nw-resize	Todos
 Figura 13.20. cursor: progress	Solo Internet Explorer
 Figura 13.21. cursor: not-allowed	Solo Internet Explorer
 Figura 13.22. cursor: no-drop	Solo Internet Explorer
 Figura 13.23. cursor: vertical-text	Solo Internet Explorer
 Figura 13.24. cursor: all-scroll	Solo Internet Explorer
 Figura 13.25. cursor: col-resize	Solo Internet Explorer

 Figura 13.26. cursor: row-resize	Solo Internet Explorer
 Figura 13.27. cursor: url(...)	Solo Internet Explorer

El puntero personalizado más utilizado es la opción `cursor: pointer` y `cursor: hand` que muestra en el puntero una mano que puede pinchar sobre el elemento. Otro puntero muy utilizado es `cursor: move` que permite indicar en las aplicaciones web dinámicas los elementos de la página que se pueden mover.

Se puede ver un ejemplo de cada uno de los punteros y la compatibilidad con los diferentes navegadores en la siguiente página: <http://www.echoecho.com/csscursors.htm>

13.4. Hacks y filtros

Los diferentes navegadores y las diferentes versiones de cada navegador incluyen defectos y carencias en su implementación del estándar CSS 2.1. Algunos navegadores no soportan ciertas propiedades, otros las soportan a medias y otros ignoran el estándar e incorporan su propio comportamiento.

De esta forma, diseñar una página compleja que presente un aspecto homogéneo en varios navegadores y varias versiones diferentes de cada navegador es una tarea que requiere mucho esfuerzo. Para facilitar la creación de hojas de estilos homogéneas, se han introducido los filtros y los *hacks*.

A pesar de que utilizar filtros y *hacks* es una solución poco ortodoxa, en ocasiones es la única forma de conseguir que una página web muestre un aspecto idéntico en cualquier navegador.

En primer lugar, los filtros permiten definir u ocultar ciertas reglas CSS para algunos navegadores específicos. Los filtros se definen aprovechando los errores de algunos navegadores (sobre todo los antiguos) a la hora de procesar las hojas de estilos.

Un caso especial de filtro son los comentarios condicionales, que es un mecanismo propietario del navegador Internet Explorer. Los comentarios condicionales permiten incluir hojas de estilos o definir reglas CSS específicamente para una versión de Internet Explorer.

El siguiente ejemplo carga la hoja de estilos `basico_ie.css` solamente para los navegadores de tipo Internet Explorer:

```
<!--[if IE]>
<style type="text/css">
  @import ("basico_ie.css");
</style>
<![endif]-->
```

Los navegadores que no son Internet Explorer ignoran las reglas CSS anteriores ya que interpretan el código anterior como un comentario de HTML (gracias a los caracteres `<!--` y

-->) mientras que los navegadores de la familia Internet Explorer lo interpretan como una instrucción propia y válida.

El filtro [if IE] indica que esos estilos CSS sólo deben tenerse en cuenta si el navegador es cualquier versión de Internet Explorer. Utilizando comentarios condicionales, también es posible incluir reglas CSS para versiones específicas de Internet Explorer:

```
<!--[if gte IE 6]>
<style type="text/css">
    @import ("basico_ie6.css");
</style>
<![endif]-->
```

El anterior ejemplo solamente carga la hoja de estilos `basico_ie6.css` si el navegador es la versión 6 o superior de Internet Explorer.

Una de las mejores listas actualizadas con todos los filtros disponibles para los navegadores de los diferentes sistemas operativos se puede encontrar en <http://centricle.com/ref/css/filters/>

Por otra parte, los *hacks* permiten forzar el comportamiento de un navegador para que se comporte tal y como se espera. Se trata de una forma poco elegante de crear las hojas de estilos y los *hacks* se pueden considerar pequeños *parches* y *chapuzas* que permiten que la hoja de estilos completa se muestre tal y como se espera.

Uno de los *hacks* más conocidos y utilizados es el llamado `* html`. Todas las propiedades CSS que se establezcan mediante el selector `* html` son interpretadas exclusivamente por el navegador Internet Explorer 6 y sus versiones anteriores:

```
div {
    border-bottom: 1px dotted #000;
}
* html div {
    border-bottom: 1px solid #000;
}
```

El ejemplo anterior utiliza el *hack* `* html` para mostrar un borde inferior punteado en los `<div>` en todos los navegadores salvo Internet Explorer 6. Como en este navegador no se muestran correctamente los bordes punteados de 1 píxel de anchura, se decide mostrar un borde formado por una línea continua.

El otro *hack* más conocido y utilizado por su sencillez es el "*underscore hack*". Las propiedades cuyos nombres se indiquen con un guión bajo por delante, sólo son interpretadas por el navegador Internet Explorer 6 y sus versiones anteriores:

```
#menu {
    position: fixed;
    _position: static;
}
```

Los navegadores más modernos soportan el valor `fixed` para la propiedad `position`, pero Internet Explorer 6 no la soporta. Por este motivo, la regla CSS anterior establece el valor de la propiedad `position` y seguidamente define la propiedad `_position`.

Los navegadores que siguen los estándares rechazan la propiedad `_position`, ya que su nombre no se corresponde con ninguna propiedad válida de CSS. Internet Explorer 6 y las versiones anteriores, consideran correcta tanto `position` como `_position`, por lo que el valor utilizado será el que se haya definido en último lugar (`static` en este caso).

Una de las mejores listas actualizadas con los hacks más útiles para varios navegadores de diferentes sistemas operativos se puede encontrar en: <http://css-discuss.incutio.com/?page=CssHack>

13.5. Prioridad de las declaraciones CSS

Además de las hojas de estilos definidas por los diseñadores, los navegadores aplican a cada página otras dos hojas de estilos: la del navegador y la del usuario.

La hoja de estilos del navegador es la primera que se aplica y se utiliza para establecer el estilo inicial por defecto a todos los elementos HTML (tamaños de letra iniciales, decoración del texto, márgenes entre elementos, etc.)

La otra hoja de estilos externa que se aplica es la que puede definir el usuario mediante su navegador. Aunque la inmensa mayoría de usuarios no utiliza esta característica, en teoría es posible que los usuarios puedan establecer el tipo de letra, color y tamaño de los textos de las páginas que ven en sus navegadores.

El orden normal en el que se aplican las hojas de estilos es el siguiente:



Figura 13.28. Orden en el que se aplican las diferentes hojas de estilos

Por tanto, las reglas que menos prioridad tienen son las del CSS por defecto de los navegadores, ya que son las primeras que se aplican. A continuación se aplican las reglas definidas por los usuarios y por último se aplican las reglas CSS definidas por el diseñador, que por tanto son las que más prioridad tienen.

Además de estas hojas de estilos, CSS define la palabra reservada `!important` para controlar la prioridad de las declaraciones de las diferentes hojas de estilos.

Si a una declaración CSS se le añade la palabra reservada `!important`, se aumenta su prioridad. El siguiente ejemplo muestra el uso de `!important`:

```
p {\n    color: red !important;\n    color: blue;\n}
```

Si la primera declaración no tuviera añadido el valor `!important`, el color de los párrafos sería azul, ya que en el caso de declaraciones de la misma importancia, prevalece la indicada en último lugar.

Sin embargo, como la primera declaración se ha marcado como de alta prioridad (gracias al valor `!important`), el color de los párrafos será el rojo.

El valor `!important` no sólo afecta a las declaraciones simples, sino que varía la prioridad de las hojas de estilo. Cuando se indican declaraciones de alta prioridad, el orden en el que se aplican las hojas de estilo es el siguiente:



Figura 13.29. Orden en el que se aplican las diferentes hojas de estilos cuando se utiliza la palabra reservada `important`

Los estilos del usuario marcados como `!important` tienen más prioridad que los estilos marcados como `!important` en la hoja de estilos del diseñador. De esta forma, ninguna página web puede sobreescibir o redefinir ninguna propiedad de alta prioridad establecida por el usuario.

13.6. Validador

La validación del código CSS y de las reglas que lo forman es un concepto similar a la validación de documentos XHTML.

La validación es un mecanismo que permite comprobar que el código CSS creado cumple las reglas de la sintaxis del lenguaje CSS y que por tanto es una hoja de estilos válida para aplicarla a cualquier documento XHTML.

La validación suele ser útil cuando se producen errores en los estilos definidos o comportamientos no deseados al aplicar las reglas CSS. En muchas ocasiones, los errores se producen porque el navegador está ignorando algunas reglas que contienen propiedades mal definidas o errores de sintaxis.

El W3C (*World Wide Web Consortium*) dispone de un validador online que permite validar reglas CSS sueltas, páginas XHTML con CSS incluido y archivos CSS independientes. El validador se puede acceder en <http://jigsaw.w3.org/css-validator/>

13.7. Recomendaciones generales sobre CSS

13.7.1. Atributos ID y class

El atributo `id` se emplea para identificar a cada elemento HTML, por lo que los identificadores deben ser únicos en una misma página. En otras palabras, dos elementos HTML diferentes de una misma página no pueden tener un mismo valor en el atributo `id`.

Por otra parte, el atributo `class` se emplea para indicar la clase o clases a las que pertenece el elemento. Una misma clase se puede aplicar a varios elementos diferentes y un único elemento puede tener asignadas varias clases (se indican separadas por espacios en blanco).

Aunque los dos atributos tienen muchos otros propósitos (sobre todo el atributo `id`), CSS los emplea principalmente con los selectores para indicar los elementos sobre los que se aplican los diferentes estilos.

En el siguiente ejemplo, las dos listas están formadas por un mismo elemento HTML ``, pero sus atributos `id` las distinguen de forma adecuada:

```
<ul id="menu">
  ...
</ul>

<ul id="enlaces">
  ...
</ul>
```

Una de las recomendaciones más importantes relacionadas con el diseño de páginas XHTML y hojas de estilos CSS está relacionada con los valores asignados a los atributos `id` y `class`. Siempre que sea posible, estos atributos se deben utilizar para mejorar la semántica del documento, es decir, para añadir significado a cada elemento de la página.

Por este motivo, los valores asignados a `id` y `class` deberían indicar la función del elemento y no deberían estar relacionados con su aspecto o su posición:

Valores incorrectos	Valores correctos
negrita	importante
arial15	titular
verdanaPequena	normal
menulzquierdo	menuSecundario
letraRoja	error

Técnicamente, los valores de los atributos `id` y `class` deben cumplir las siguientes restricciones:

- Sólo pueden empezar por un guión medio (-), un guión bajo (_) o una letra.
- El resto de caracteres, pueden ser números, guiones medios (-), guiones bajos (_) y letras.
- Los navegadores distinguen entre mayúsculas y minúsculas.
- Aunque es posible utilizar letras como ñ y acentos, no se recomienda hacerlo porque no es seguro que funcione correctamente en todas las versiones de todos los navegadores.

13.7.2. CLASSitis y DIVitis

Un error común al comenzar a desarrollar páginas con estilos CSS es la utilización excesiva de etiquetas `<div>` y atributos `class`.

Ejemplo de *divitis* y *classitis*:

```
<div id="menu">
<ul class="menu">
    <li class="elemento_menu"><span class="texto_elemento_menu">...</span></li>
    <li class="elemento_menu"><span class="texto_elemento_menu">...</span></li>
    <li class="elemento_menu"><span class="texto_elemento_menu">...</span></li>
    <li class="elemento_menu"><span class="texto_elemento_menu">...</span></li>
</ul>
</div>
```

Los selectores de CSS permiten prescindir de la mayoría de etiquetas `<div>` y atributos `id` y `class`. Diseñar páginas con exceso de etiquetas `<div>` no mejora la semántica del documento y sólo consigue complicar el código HTML.

13.7.3. Estructuración del código CSS

La posibilidad de incluir todo el código CSS en archivos externos exclusivamente dedicados a contener las reglas CSS, permite ordenar de forma lógica las reglas, mejorando su legibilidad y facilitando su actualización.

Las reglas CSS de las hojas de estilos complejas se suelen agrupar según su funcionalidad y se suelen incluir en el siguiente orden:

- Estilos básicos (`<body>`, tipo de letra por defecto, márgenes de ``, `` y ``, etc.)
- Estilos de la estructura o layout (anchura, altura y posición de la cabecera, pie de página, zonas de contenidos, menús de navegación, etc.)
- Enlaces (estilos normales, estilos `:hover`, etc.)
- Estilos de cada una de las zonas (elementos de la cabecera, titulares y texto de la zona de contenidos, enlaces, listas e imágenes de las zonas laterales, etc.)

Otra característica común de los mejores sitios web es el uso de comentarios CSS para mejorar la estructura de las hojas de estilos muy largas.

Ejemplo de código CSS estructurado de <http://veerle.duoh.com/>

```
/* VeerLe's blog Main stylesheet
-----
/* Wide browser windows
-----
#wrap {
    width:995px;
}
```

```

/* Global
-----
html, body, form, h1, h2, h3, h4, h5, h6, p, pre, blockquote, ul, ol, dl {
    margin:0;
    padding:0;
}
ul,li {
    list-style-type:none;
}
...

/* Wide Layout
-----
.wide #wrap-main {
    ...
}
...

/* Links
-----
a:link,
a:visited {
    text-decoration:none;
    color:#e45a49;
}

/* Header
-----
#header {
    ...
}
...

/* Main navigation
-----
ul#nav {
    ...
}
...

```

Ejemplo de código CSS estructurado de <http://www.uxmag.com/>

```

/*
-----  

    UX Magazine  

    Design | Technology | Strategy | Common Sense  

-----  

    Description:      Base setup styles  

    Filename:        uxm.base.css  

    Version:         1.9  

    Date:            Feb 9, 2006  

----- */  

/*
-----  

    Base Body Styles  

----- */

```

```
/*
-----  
Print Styles  
----- */  
  
/* -----  
Top Bar Styles  
----- */  
  
/* Slogan  
----- */  
/* Search Form  
----- */  
/* Channels  
----- */
```

13.7.4. División de los estilos en varios archivos CSS

Normalmente, los estilos de una página compleja se dividen en varios archivos CSS diferentes para hacerlos más manejables. En primer lugar, se suele utilizar un archivo común que contiene todos los estilos básicos de las páginas HTML del sitio web.

Además, si existe alguna sección especial del sitio web que requiera nuevos estilos, se crea un archivo CSS con todos esos estilos. También es habitual preparar una hoja de estilos específica para impresora y otra preparada para los dispositivos móviles.

Una vez creados los archivos CSS, existen dos estrategias para enlazar varios archivos CSS en las páginas HTML:

Si se puede modificar fácilmente la cabecera del documento (por ejemplo porque las páginas se generan dinámicamente) lo habitual es incluir tantos elementos `<link>` como archivos CSS se enlazan:

```
<head>  
...  
<link rel="stylesheet" type="text/css" href="/css/basico.css" media="screen" />  
<link rel="stylesheet" type="text/css" href="/css/seccion.css" media="screen" />  
<link rel="stylesheet" type="text/css" href="/css/impresora.css" media="print" />  
<link rel="stylesheet" type="text/css" href="/css/movil.css" media="handheld" />  
...  
</head>
```

Si no se puede modificar de forma sencilla la cabecera de los documentos para añadir, eliminar y modificar los archivos CSS que se enlazan, lo habitual es enlazar un único archivo CSS que se encarga de importar todos los demás:

```
<head>  
...  
<link rel="stylesheet" type="text/css" href="/css/estilos.css" media="all" />  
...  
</head>
```

El contenido del archivo `estilos.css` debería ser el siguiente para ser equivalente al ejemplo anterior:

```
| @import url("basico.css") screen;
| @import url("seccion.css") screen;
| @import url("impresora.css") print;
| @import url("movil.css") handheld;
```

Capítulo 14. Recursos útiles

Disponer de una buena colección de recursos realmente útiles es una de las características que diferencian a los diseñadores web profesionales del resto.

En primer lugar, resulta imprescindible instalar en el navegador Firefox varias extensiones relacionadas con el diseño web. Todas estas extensiones facilitan el trabajo, ayudan a descubrir rápidamente la causa de los errores del diseño y en general mejoran notablemente la productividad de los diseñadores.

Además, siempre es necesario disponer de varias galerías de páginas como fuente de inspiración y colecciones de enlaces a buenos recursos relacionados con el diseño web. Por último, también es necesario estar al día de las últimas técnicas y novedades mediante sitios web y blogs dedicados en exclusiva al diseño web.

14.1. Extensiones de Firefox

Si te dedicas profesionalmente al diseño de páginas y aplicaciones web, seguramente tu navegador preferido para trabajar es Firefox. Si no lo conoces, puedes descargar gratuitamente Firefox desde su sitio web oficial: <http://www.mozilla.com/firefox>

Las principales ventajas de Firefox desde el punto de vista del diseñador y creador de páginas web es que respeta los estándares del W3C mucho más que los navegadores de la familia Internet Explorer. Además, permite instalar pequeños añadidos, llamados extensiones, que añaden funcionalidades al navegador.

Una extensión se puede considerar como un pequeño programa que se instala dentro del navegador y que añade alguna característica interesante que el navegador no incorpora de serie. Lo mejor de las extensiones de Firefox es que existen cientos de extensiones, prácticamente todas son gratuitas y casi todas son realmente útiles.

El sitio web de Firefox incluye una sección especial llamada "Complementos" en la que se puede encontrar el listado completo de extensiones disponibles para Firefox (<http://addons.mozilla.org/es-ES/firefox/>). A continuación se muestra una pequeña selección de algunas de las extensiones más interesantes para los diseñadores de páginas web.

14.1.1. Firebug

Firebug (<https://addons.mozilla.org/firefox/1843/>) es la extensión más útil y completa de todas las que están relacionadas con el diseño web. No importa si tu especialidad es XHTML, CSS, JavaScript, DOM o AJAX, ya que Firebug proporciona toda la información posible sobre cada uno de estos temas.

Como Firebug tiene tantas opciones, sería necesario un libro entero para explicar sus posibilidades. Por ello, lo mejor es que instales la extensión y la pruebes en tus proyectos web.

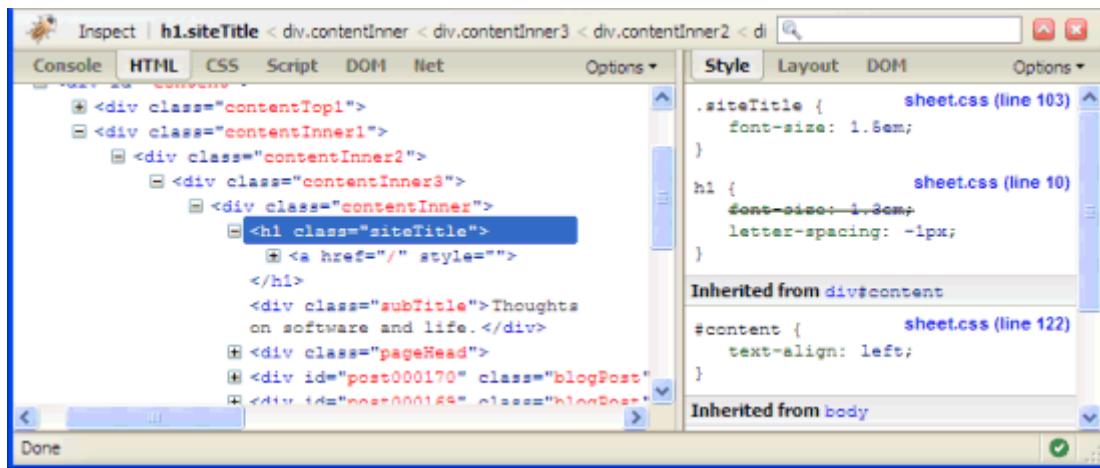


Figura 14.1. Extensión Firebug para Firefox

Probablemente, después de varios meses de uso de Firebug, seguirás descubriendo nuevas opciones muy útiles y seguirás agradeciendo a su creador las horas de trabajo que te ha ahorrado.

14.1.2. Web Developer

Antes de que existiera Firebug, la extensión Web Developer (<https://addons.mozilla.org/firefox/60/>) era la más útil para los diseñadores web. Se trata de una barra que se instala junto con el resto de herramientas del navegador y que básicamente se puede utilizar para obtener información sobre la página:

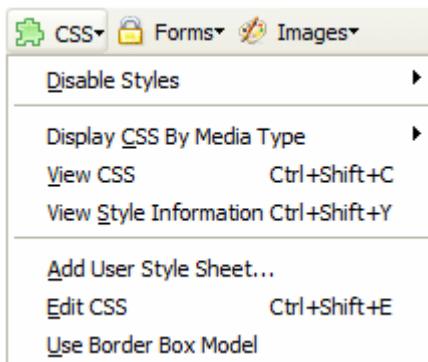


Figura 14.2. Extensión Web Developer para Firefox

Además de proporcionar información, la extensión Web Developer incluye utilidades que Firebug todavía no incorpora, como la redimensión de la ventana del navegador a las dimensiones más utilizadas, recuadrar todos los elementos de un determinado tipo (celdas de tabla, divs, etc.), mostrar una lupa, una regla redimensionable, mostrar los elementos de tipo `hidden`, mostrar la ruta de cada imagen, etc.

14.1.3. HTML Validator

Una buena práctica, y un requisito impuesto por muchos clientes, es que las páginas XHTML creadas sean válidas y por tanto, pasen el validador de HTML y de CSS disponible en el W3C. Para facilitar la validación de las páginas, la extensión HTML Validator

(<https://addons.mozilla.org/firefox/249/>) indica en todo momento los errores y las recomendaciones sobre el código HTML de la página que muestra el navegador.



Figura 14.3. Extensión HTML Validator para Firefox

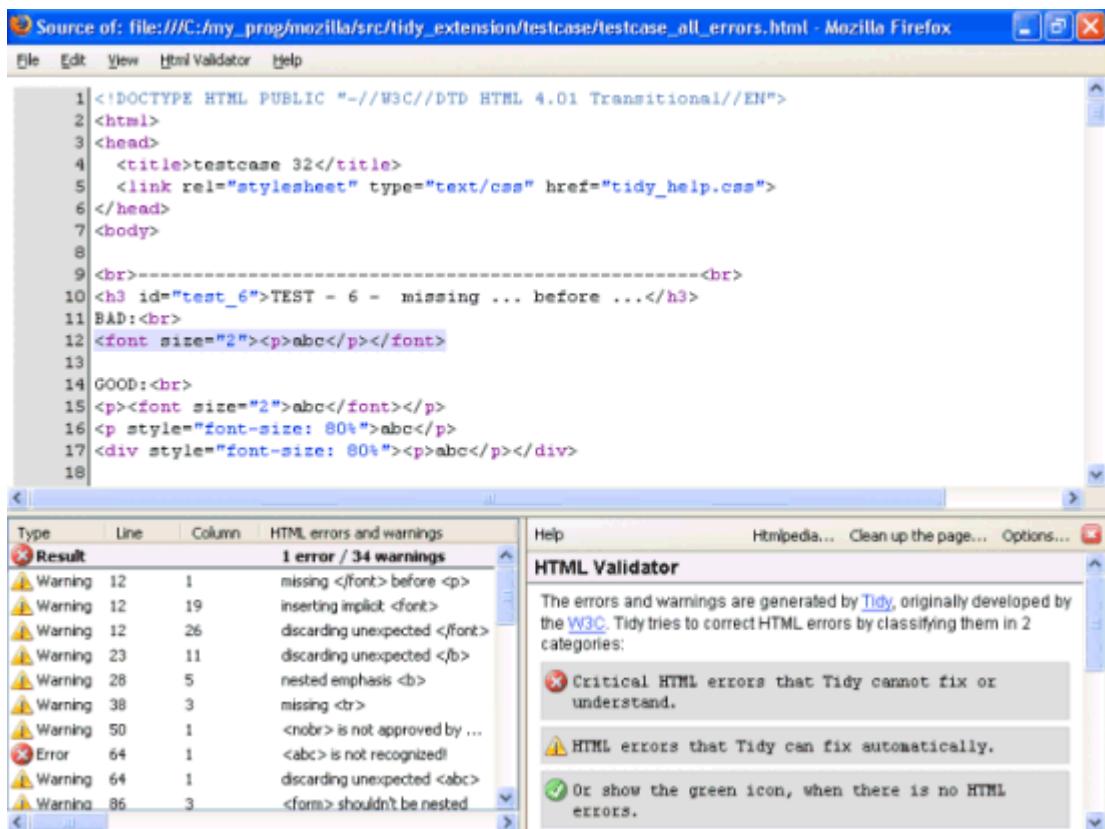


Figura 14.4. Extensión HTML Validator para Firefox

14.1.4. Otras extensiones

ColorZilla (<https://addons.mozilla.org/firefox/271/>) permite obtener los colores de cualquier elemento de la página mediante una herramienta similar a la de los programas de diseño gráfico:



Figura 14.5. Extensión ColorZilla para Firefox

MeasureIt (<https://addons.mozilla.org/firefox/539/>) permite medir la altura y anchura de cualquier elemento de la página:

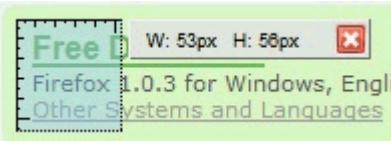


Figura 14.6. Extensión MeasureIt para Firefox

View Source Chart (<https://addons.mozilla.org/firefox/655/>) , muestra el código fuente de una página estructurado según el anidamiento de los diferentes elementos. Facilita el aprendizaje de HTML y mejora la búsqueda de posibles errores en el código:



Figura 14.7. Extensión View Source Chart para Firefox

14.2. Aplicaciones web

A continuación se indican algunas aplicaciones web que pueden ser de utilidad para el diseñador CSS:

- Clean CSS (<http://www.cleancss.com>) : optimiza, ordena, limpia, corrige y reduce el tamaño de las hojas de estilos.
- Typetester (<http://typetester.maratz.com>) : permite comparar de forma sencilla diferentes tipos de letra y propiedades relacionadas con la tipografía y el texto.

- [Browsershots](http://browsershots.org) (<http://browsershots.org>) : muestra cómo se visualiza una misma página web en diferentes navegadores de diferentes sistemas operativos (55 navegadores en total). El uso de la aplicación es gratuito y se pueden ver y/o descargar las imágenes que muestran el aspecto de la página en cada navegador.
- [Stripe Generator](http://www.stripegenerator.com/) (<http://www.stripegenerator.com/>) : permite generar fácilmente imágenes preparadas para poder repetirse en todas direcciones de forma correcta y por tanto, para que puedan ser utilizadas como imágenes de fondo.

14.3. Sitios web de inspiración

Muchas veces resulta útil disponer de buenos ejemplos de páginas diseñadas completamente con CSS para tomarlas como referencia y posible inspiración de los diseños propios:

- [Web Creme](http://www.webcreme.com) (<http://www.webcreme.com>) : incluye diariamente varios ejemplos de las mejores páginas diseñadas con CSS y permite realizar búsquedas a partir del color utilizado en la página.
- [CSS Remix](http://www.cssremix.com) (<http://www.cssremix.com>) : muestra centenares de páginas diseñadas exclusivamente con CSS y con la posibilidad de puntuar su diseño.
- [CSS Zen Garden](http://www.csszengarden.com) (<http://www.csszengarden.com>) : es una galería diferente a las tradicionales, pero se ha convertido en una referencia en cuanto a diseños complejos realizados mediante CSS.
- [Open Source Web Design](http://www.oswd.org) (<http://www.oswd.org>) : sitio web que ofrece cientos de plantillas gratuitas con posibilidad de utilizarlas libremente en aplicaciones personales y comerciales.

14.4. Referencias y colecciones de recursos

- Especificación oficial de CSS 2.1 (<http://www.w3.org/TR/CSS21/>) : la que incluyen todos los navegadores actuales.
- Especificación oficial de CSS 3 (<http://www.w3.org/Style/CSS/current-work#CSS3>) : la que sustituirá dentro de unos años a la actual versión 2.1.
- del.icio.us/webDesign (<http://del.icio.us/webDesign>) : colección de recursos relacionados con todos los aspectos del diseño web. Lista creada y mantenida por el diseñador Miguel Sánchez (<http://www.miguelsanchez.com/>).
- [netvibes.com/formacionweb](http://www.netvibes.com/formacionweb) (<http://www.netvibes.com/formacionweb>) : colección de noticias, artículos y recursos relacionados con el diseño web. Lectura diaria recomendada. Los recursos han sido seleccionados por el diseñador Miguel Sánchez (<http://www.miguelsanchez.com/>)
- [Web Developers Handbook](http://www.alvit.de/handbook/) (<http://www.alvit.de/handbook/>) : cientos de enlaces con todos los recursos útiles para diseñadores web.
- [Blue Vertigo](http://www.bluevertigo.com.ar/bluevertigo.htm) (<http://www.bluevertigo.com.ar/bluevertigo.htm>) : otra colección de cientos de enlaces con recursos útiles para diseñadores web.

- [Foros del Web](http://www.forosdelweb.com/) (<http://www.forosdelweb.com/>) : incluyen foros específicos dedicados a resolver dudas relacionadas con el diseño web con CSS.
- [Ovillo](http://www.ovillo.org/) (<http://www.ovillo.org/>) : una de las mejores listas de distribución en castellano.
- [Los foros de SitePoint](http://www.sitepoint.com/forums/) (<http://www.sitepoint.com/forums/>) son una de las mejores referencias en inglés para resolver dudas relacionadas con el diseño web.

Capítulo 15. Ejercicios

15.1. Ejercicio 1

A partir del código HTML y CSS que se muestra, añadir los selectores CSS que faltan para aplicar los estilos deseados. Cada regla CSS incluye un comentario en el que se explica los elementos a los que debe aplicarse:

```
<!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">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
<title>Ejercicio de selectores</title>
<style type="text/css">
/* Todos los elementos de la pagina */
{ font: 1em/1.3 Arial, Helvetica, sans-serif; }

/* Todos los párrafos de la pagina */
{ color: #555; }

/* Todos los párrafos contenidos en #primero */
{ color: #336699; }

/* Todos los enlaces la pagina */
{ color: #CC3300; }

/* Los elementos "em" contenidos en #primero */
{ background: #FFFFCC; padding: .1em; }

/* Todos los elementos "em" de clase "especial" en toda la pagina */
{ background: #FFCC99; border: 1px solid #FF9900; padding: .1em; }

/* Elementos "span" contenidos en .normal */
{ font-weight: bold; }

</style>
</head>

<body>

<div id="primero">
<p>Lorem ipsum dolor sit amet, <a href="#">consectetuer adipiscing elit</a>. Praesent blandit nibh at felis. Sed nec diam in dolor vestibulum aliquet. Duis ullamcorper, nisi non facilisis molestie, <em>lorem sem aliquam nulla</em>, id lacinia velit mi vestibulum enim.</p>
</div>

<div class="normal">
<p>Phasellus eu velit sed lorem sodales egestas. Ut feugiat. <span><a href="#">Donec
```

```

porttitor</a>, magna eu varius luctus,</span> metus massa tristique massa, in imperdiet
est velit vel magna. Phasellus erat. Duis risus. <a href="#">Maecenas dictum</a>, nibh
vitae pellentesque auctor, tellus velit consectetur tellus, tempor pretium felis
tellus at metus.</p>

<p>Cum sociis natoque <em class="especial">penatibus et magnis</em> dis parturient
montes, nascetur ridiculus mus. Proin aliquam convallis ante. Pellentesque habitant
morbi tristique senectus et netus et malesuada fames ac turpis egestas. Nunc aliquet.
Sed eu metus. Duis justo.</p>

<p>Donec facilisis blandit velit. Vestibulum nisi. Proin volutpat, <em
class="especial">enim id iaculis congue</em>, orci justo ultrices tortor, <a
href="#">quis lacinia eros libero in eros</a>. Sed malesuada dui vel quam. Integer at
eros.</p>
</div>

</body>
</html>

```

15.2. Ejercicio 2

A partir del código HTML proporcionado, añadir las reglas CSS necesarias para que la página resultante tenga el mismo aspecto que el de la siguiente imagen:

Lorem ipsum dolor sit amet

Nulla pretium. Sed tempus nunc vitae neque. **Suspendisse gravida**, metus a scelerisque sollicitudin, lacus velit ultricies nisl, nonummy tempus neque diam quis felis. **Etiam sagittis tortor** sed arcu sagittis tristique.

Aliquam tincidunt, sem eget volutpat porta

Vivamus velit dui, placerat vel, feugiat in, ornare et, urna. **Aenean turpis metus, aliquam non, tristique in**, pretium varius, sapien. Proin vitae nisi. Suspendisse **porttitor purus ac elit**. Suspendisse eleifend odio at dui. In in elit sed metus pretium elementum.

Título de la tabla

	 Título columna 1	 Título columna 2
 Título fila 1	Donec purus ipsum	Curabitur <i>blandit</i>
 Título fila 2	Donec purus ipsum	Curabitur blandit
	 Título columna 1	 Título columna 2

Donec purus ipsum, posuere id, venenatis at, placerat ac, lorem. Curabitur blandit, eros sed gravida aliquet, risus justo porta lorem, ut mollis lectus tortor in orci. Pellentesque nec augue.

Fusce nec felis eu diam pretium adipiscing. **Nunc elit elit, vehicula vulputate**, venenatis in, posuere id, lorem. Etiam sagittis, tellus in ultrices accumsan, diam nisi feugiat ante, eu congue magna mi non nisl.

Vivamus ultrices aliquet augue. **Donec arcu pede, pretium vitae, rutrum aliquet, tincidunt blandit, pede**. Aliquam in nisi. Suspendisse volutpat. Nulla facilisi. Ut ullamcorper nisi quis mi.

Figura 15.1. Aspecto final de la página

A continuación se muestra el código HTML de la página sin estilos:

```
<!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">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>Ejercicio de selectores</title>
</head>

<body>
<h1 id="titulo">Lorem ipsum dolor sit amet</h1>

<p>Nulla pretium. Sed tempus nunc vitae neque. <strong>Suspendisse gravida</strong>, metus a scelerisque sollicitudin, lacinia velit ultricies nisl, nonummy tempus neque diam quis felis. <span class="destacado">Etiam sagittis tortor</span> sed arcu sagittis tristique.</p>

<h2 id="subtitulo">Aliquam tincidunt, sem eget volutpat porta</h2>

<p>Vivamus velit dui, placerat vel, feugiat in, ornare et, urna. <a href="#">Aenean turpis metus, <em>aliquam non</em>, tristique in</a>, pretium varius, sapien. Proin vitae nisi. Suspendisse <span class="especial">porttitor purus ac elit</span>. Suspendisse eleifend odio at dui. In in elit sed metus pretium elementum.</p>

<table summary="Descripción de la tabla y su contenido">
<caption>Título de la tabla</caption>
<thead>
<tr>
<th scope="col"></th>
<th scope="col" class="especial">Título columna 1</th>
<th scope="col" class="especial">Título columna 2</th>
</tr>
</thead>

<tfoot>
<tr>
<th scope="col"></th>
<th scope="col">Título columna 1</th>
<th scope="col">Título columna 2</th>
</tr>
</tfoot>

<tbody>
<tr>
<th scope="row" class="especial">Título fila 1</th>
<td>Donec purus ipsum</td>
<td>Curabitur <em>blandit</em></td>
</tr>
<tr>
<th scope="row">Título fila 2</th>
<td>Donec <strong>purus ipsum</strong></td>
<td>Curabitur blandit</td>
</tr>
</tbody>
```

```
</table>

<div id="adicional">
<p>Donec purus ipsum, posuere id, venenatis at, <span>placerat ac, lorem</span>.
Curabitur blandit, eros sed gravida aliquet, risus justo
porta lorem, ut mollis lectus tortor in orci. Pellentesque nec augue.</p>

<p>Fusce nec felis eu diam pretium adipiscing. <span id="especial">Nunc elit elit,
vehicula vulputate</span>, venenatis in,
posuere id, lorem. Etiam sagittis, tellus in ultrices accumsan, diam nisi feugiat ante,
eu congue magna mi non nisl.</p>

<p>Vivamus ultrices aliquet augue. <a href="#">Donec arcu pede, pretium vitae</a>,
rutrum aliquet, tincidunt blandit, pede.
Aliquam in nisi. Suspendisse volutpat. Nulla facilisi. Ut ullamcorper nisi quis mi.</p>
</div>

</body>
</html>
```

Aunque la propiedad que modifica el color del texto se explica detalladamente en los próximos capítulos, en este ejercicio solamente es preciso conocer que la propiedad se llama `color` y que como valor se puede indicar directamente el nombre del color.

Los nombres de los colores también están estandarizados y se corresponden con el nombre en inglés de cada color. En este ejercicio, se deben utilizar los colores: `teal`, `red`, `blue`, `orange`, `purple`, `olive`, `fuchsia` y `green`.

15.3. Ejercicio 3

A partir del código HTML y CSS proporcionados, determinar las reglas CSS necesarias para añadir los siguientes márgenes y rellenos:

The screenshot shows a website layout with the following structure:

- Header:** A navigation bar with a search input field labeled "Buscar".
- Left Sidebar:**
 - Noticias:** A list of links: dd/mm/aaaa [Lorem ipsum dolor sit amet](#), dd/mm/aaaa [Consectetuer adipiscing elit](#), dd/mm/aaaa [Donec molestie nunc eu sapien](#), dd/mm/aaaa [Maecenas aliquam dolor eget metus](#), dd/mm/aaaa [Fusce tristique lorem id metus](#).
 - Enlaces relacionados:** A list of links: [Proin placerat](#), [Nulla in felis](#), [Nam luctus](#).
 - Publicidad:** A block of text: Etiam fermentum, nisl tincidunt blandit interdum, massa velit posuere dolor, sed euismod sem odio at mi. Duis porta placerat arcu. Nullam felis pede, commodo vel, suscipit a, molestie vel, felis. Maecenas mattis est vel est.
 - Links:** [Seguir leyendo...](#)
- Main Content Area:**
 - Section 1:** **Lorem ipsum dolor sit amet, consectetuer adipiscing elit**. It contains a gray square with a large white X and the following text: Nullam est lacus, suscipit ut, dapibus quis, condimentum ac, risus. Vivamus vestibulum, ipsum sollicitudin faucibus pharetra, dolor metus fringilla dui, vel aliquet pede diam tempor tortor. Vestibulum pulvinar urna et quam. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Nullam vel turpis vitae dui imperdiet laoreet. Quisque eget ipsum. Donec commodo, turpis vel venenatis sollicitudin, quam ante convallis justo, sed eleifend justo lectus quis sapien. Ut consequat libero eget est. [Seguir leyendo...](#)
 - Section 2:** **Vivamus lobortis turpis ac ante fringilla faucibus**. It contains a gray square with a large white X and the following text: Quisque eget ipsum. Donec commodo, turpis vel venenatis sollicitudin, quam ante convallis justo, sed eleifend justo lectus quis sapien. Ut consequat libero eget est. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Nullam dictum hendrerit neque. Mauris id ligula non elit mattis semper. Fusce arcu ipsum, tempus eget, tincidunt at, imperdiet in, mi. Sed fermentum cursus dolor. Aenean a diam. Phasellus feugiat. Donec tempor dignissim sem. [Seguir leyendo...](#)
- Right Sidebar:**
 - Phasellus blandit**: Praesent sodales imperdiet augue. Mauris lorem felis, semper eu, tincidunt eu, sollicitudin eget, sem. Nulla facilisi. Morbi ut enim ut enim ultricies dapibus.
 - Links:** [Seguir leyendo...](#)
 - Nullam vel turpis**: Donec commodo, turpis vel venenatis sollicitudin, quam ante convallis justo, sed eleifend justo lectus quis sapien. Ut consequat libero eget est.
 - Links:** [Seguir leyendo...](#)

Figura 15.2. Página original

1. El elemento `#cabecera` debe tener un relleno de `1em` en todos los lados.
2. El elemento `#menu` debe tener un relleno de `0.5em` en todos los lados y un margen inferior de `0.5em`.
3. El resto de elementos (`#noticias`, `#publicidad`, `#principal`, `#secundario`) deben tener `0.5em` de relleno en todos sus lados, salvo el elemento `#pie`, que sólo debe tener relleno en la zona superior e inferior.
4. Los elementos `.articulo` deben mostrar una separación entre ellos de `1em`.
5. Las imágenes de los artículos muestran un margen de `0.5em` en todos sus lados.
6. El elemento `#publicidad` está separado `1em` de su elemento superior.
7. El elemento `#pie` debe tener un margen superior de `1em`.

LOGOTIPO

[LoremIpsumDolorSitAmet](#)

Noticias

dd/mm/aaaa [Lorem ipsum dolor sit amet](#)
dd/mm/aaaa [Consectetuer adipiscing elit](#)
dd/mm/aaaa [Donec molestie nunc eu sapien](#)
dd/mm/aaaa [Maecenas aliquam dolor eget metus](#)
dd/mm/aaaa [Fusce tristique lorem id metus](#)

Enlaces relacionados

[Proin placerat](#)
[Nulla in felis](#)
[Nam luctus](#)

Publicidad

Etiam fermentum, nisl tincidunt blandit interdum, massa vel posuere dolor, sed euismod sem odio at mi.
Duis porta placerat arcu. Nullam felis pede, commodo vel, suscipit a, molestie vel, felis. Maecenas mattis est vel est.

[Seguir leyendo...](#)

Lore ipsum dolor sit amet, consectetuer adipiscing elit



Nullam est lacus, suscipit ut, dapibus quis, condimentum ac, risus. Vivamus vestibulum, ipsum sollicitudin faucibus pharetra, dolor metus fringilla dui, vel aliquet pede diam tempor tortor. Vestibulum pulvinar urna et quam. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Nullam vel turpis vitae dui imperdiet laoreet. Quisque eget ipsum.

Donec commodo, turpis vel venenatis sollicitudin, quam ante convallis justo, sed eleifend justo lectus quis sapien. Ut consequat libero eget est.

[Seguir leyendo...](#)

Vivamus lobortis turpis ac ante fringilla faucibus



Quisque eget ipsum. Donec commodo, turpis vel venenatis sollicitudin, quam ante convallis justo, sed eleifend justo lectus quis sapien. Ut consequat libero eget est.

Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Nullam dictum hendrerit neque. Mauris id ligula non elit mattis semper. Fusce arcu ipsum, tempus eget, tincidunt at, imperdiet in, mi.

Sed fermentum cursus dolor. Aenean a diam. Phasellus feugiat. Donec tempor dignissim sem.

[Seguir leyendo...](#)

Phasellus blandit

Praesent sodales imperdiet augue. Mauris lorem felis, semper eu, tincidunt eu, sollicitudin eget, sem. Nulla facilisi. Morbi ut enim ut enim ultricies dapibus.

[Seguir leyendo...](#)

Nullam vel turpis

Donec commodo, turpis vel venenatis sollicitudin, quam ante convallis justo, sed eleifend justo lectus quis sapien. Ut consequat libero eget est.

[Seguir leyendo...](#)

[Nulla](#) | [Pharetra](#) | [Luctus](#) | [Ipsum](#) | [Proin](#) | [Placerat](#)

© Copyright Lorem ipsum

Figura 15.3. Página con márgenes y rellenos

15.4. Ejercicio 4

A partir del código HTML y CSS proporcionados, determinar las reglas CSS necesarias para añadir los siguientes bordes:

LOGOTIPO

Buscar

[LoremIpsumDolorSitAmet](#)

Noticias

dd/mm/aaaa [Lorem ipsum dolor sit amet](#)
 dd/mm/aaaa [Consectetuer adipiscing elit](#)
 dd/mm/aaaa [Donec molestie nunc eu sapien](#)
 dd/mm/aaaa [Maecenas aliquam dolor eget metus](#)
 dd/mm/aaaa [Fusce tristique lorem id metus](#)

Enlaces relacionados

[Proin placerat](#)
[Nulla in felis](#)
[Nam luctus](#)

Publicidad

Etiam fermentum, nisl tincidunt blandit interdum, massa velit posuere dolor, sed euismod sem odio at mi. Duis porta placerat arcu. Nullam felis pede, commodo vel, suscipit a, molestie vel, felis. Maecenas mattis est vel est.

[Seguir leyendo...](#)

LOGOTIPO

Phasellus blandit

Praesent sodales imperdiet augue. Mauris lorem felis, semper eu, tincidunt eu, sollicitudin eget, sem. Nulla facilisi. Morbi ut enim ut enim ultricies dapibus.

[Seguir leyendo...](#)

Nullam vel turpis

Donec commodo, turpis vel venenatis sollicitudin, quam ante convallis justo, sed eleifend justo lectus quis sapien. Ut consequat libero eget est.

[Seguir leyendo...](#)

[Nulla](#) | [Pharetra](#) | [Luctus](#) | [Ipsum](#) | [Proin](#) | [Placerat](#)

© Copyright Lorem ipsum

Figura 15.4. Página original

1. Eliminar el borde gris que muestran por defecto todos los elementos.
2. El elemento `#menu` debe tener un borde inferior de 1 píxel y azul (#004C99).
3. El elemento `#noticias` muestra un borde de 1 píxel y gris claro (#C5C5C5).
4. El elemento `#publicidad` debe mostrar un borde discontinuo de 1 píxel y de color #CC6600.
5. El lateral formado por el elemento `#secundario` muestra un borde de 1 píxel y de color #CC6600.
6. El elemento `#pie` debe mostrar un borde superior y otro inferior de 1 píxel y color gris claro #C5C5C5.

LOGOTIPO

Buscar [LoremIpsumDolorSitAmet](#)

Noticias
dd/mm/aaaa [Lorem ipsum dolor sit amet](#)

dd/mm/aaaa [Consectetuer adipiscing elit](#)

dd/mm/aaaa [Donec molestie nunc eu sapien](#)

dd/mm/aaaa [Maecenas aliquam dolor eget metus](#)

dd/mm/aaaa [Fusce tristique lorem id metus](#)

Enlaces

[relacionados](#)

[Pron placherat](#)

[Nulla in felis](#)

[Nam luctus](#)

Publicidad

Etiam fermentum, nisl tincidunt blandit interdum, massa velit posuere dolor, sed euismod sem odio at mi.

Duis porta placerat arcu. Nullam felis pede, commodo vel, suscipit a, molestie vel, felis.

Maecenas mattis est vel est.

[Seguir leyendo...](#)

Lore ipsum dolor sit amet, consectetur adipiscing elit



Nullam est lacus, suscipit ut, dapibus quis, condimentum ac, risus. Vivamus vestibulum, ipsum sollicitudin faucibus pharetra, dolor metus fringilla dui, vel aliquet pede diam tempor tortor. Vestibulum pulvinar urna et quam. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Nullam vel turpis vitae dui imperdiet laoreet. Quisque eget ipsum.

Donec commodo, turpis vel venenatis sollicitudin, quam ante convallis justo, sed eleifend justo lectus quis sapien. Ut consequat libero eget est.

[Seguir leyendo...](#)

Vivamus lobortis turpis ac ante fringilla faucibus



Quisque eget ipsum. Donec commodo, turpis vel venenatis sollicitudin, quam ante convallis justo, sed eleifend justo lectus quis sapien. Ut consequat libero eget est.

Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Nullam dictum hendrerit neque. Mauris id ligula non elit mattis semper. Fusce arcu ipsum, tempus eget, tincidunt at, imperdiet in, mi.

Sed fermentum cursus dolor. Aenean a diam. Phasellus feugiat. Donec tempor dignissim sem.

[Seguir leyendo...](#)

Phasellus blandit

Praesent sodales imperdiet augue. Mauris lorem felis, semper eu, tincidunt eu, sollicitudin eget, sem. Nulla facilisi. Morbi ut enim ut enim ultricies dapibus.

[Seguir leyendo...](#)

Nullam vel turpis

Donec commodo, turpis vel venenatis sollicitudin, quam ante convallis justo, sed eleifend justo lectus quis sapien. Ut consequat libero eget est.

[Seguir leyendo...](#)

[Nulla](#) | [Pharetra](#) | [Luctus](#) | [Ipsum](#) | [Pron](#) | [Placerat](#)

© Copyright Lorem ipsum

Figura 15.5. Página con bordes

15.5. Ejercicio 5

A partir del código HTML y CSS proporcionados, determinar las reglas CSS necesarias para añadir los siguientes colores e imágenes de fondo:

LOGOTIPO

Buscar [Lorem ipsum dolor sit amet](#)

Noticias

dd/mm/aaaa [Lorem ipsum dolor sit amet](#)

dd/mm/aaaa [Consectetuer adipiscing elit](#)

dd/mm/aaaa [Donec molestie nunc eu sapien](#)

dd/mm/aaaa [Maecenas aliquam dolor eget metus](#)

dd/mm/aaaa [Fusce tristique lorem id metus](#)

Enlaces relacionados

[Proin placerat](#)

[Nulla in felis](#)

[Nam luctus](#)

Publicidad

Etiam fermentum, nisl tincidunt blandit interdum, massa velit posuere dolor, sed euismod sem odio at mi.

Duis porta placerat arcu. Nullam felis pede, commodo vel, suscipit a, molestie vel, felis.

Maecenas mattis est vel est.

[Seguir leyendo...](#)

Lorem ipsum dolor sit amet, consectetuer adipiscing elit



Nullam est lacus, suscipit ut, dapibus quis, condimentum ac, risus. Vivamus vestibulum, ipsum sollicitudin faucibus pharetra, dolor metus fringilla dui, vel aliquet pede diam tempore tortor. Vestibulum pulvinar urna et quam. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Nullam vel turpis vitae dui imperdiet laoreet. Quisque eget ipsum.

Donec commodo, turpis vel venenatis sollicitudin, quam ante convallis justo, sed eleifend justo lectus quis sapien. Ut consequat libero eget est.

[Seguir leyendo...](#)

Vivamus lobortis turpis ac ante fringilla faucibus



Quisque eget ipsum. Donec commodo, turpis vel venenatis sollicitudin, quam ante convallis justo, sed eleifend justo lectus quis sapien. Ut consequat libero eget est.

Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Nullam dictum hendrerit neque. Mauris id ligula non elit mattis semper. Fusce arcu ipsum, tempus eget, tincidunt at, imperdiet in, mi.

Sed fermentum cursus dolor. Aenean a diam. Phasellus feugiat. Donec tempor dignissin sem.

[Seguir leyendo...](#)

Phasellus blandit

Praesent sodales imperdiet augue. Mauris lorem felis, semper eu, tincidunt eu, sollicitudin eget, sem. Nulla facilisi. Morbi ut enim ut enim ultricies dapibus.

[Seguir leyendo...](#)

Nullam vel turpis

Donec commodo, turpis vel venenatis sollicitudin, quam ante convallis justo, sed eleifend justo lectus quis sapien. Ut consequat libero eget est.

[Seguir leyendo...](#)

[Nulla](#) | [Pharetra](#) | [Luctus](#) | [Ipsum](#) | [Proin](#) | [Placerat](#)

© Copyright Lorem ipsum

Figura 15.6. Página original

1. Los elementos #noticias y #pie tiene un color de fondo gris claro (#F8F8F8).
2. El elemento #publicidad muestra un color de fondo amarillo claro (#FFF6CD).
3. Los elementos <h2> del lateral #secundario muestran un color de fondo #DB905C y un pequeño padding de .2em.
4. El fondo del elemento #menu se construye mediante una pequeña imagen llamada fondo_menu.gif.
5. El logotipo del sitio se muestra mediante una imagen de fondo del elemento <h1> contenido en el elemento #cabecera (la imagen se llama logo.gif).

Logotipo

Buscar

Lorem ipsum dolor sit amet

Noticias

dd/mm/aaaa [Lorem ipsum dolor sit amet](#)
 dd/mm/aaaa [Consectetuer adipiscing elit](#)
 dd/mm/aaaa [Donec molestie nunc eu sapien](#)
 dd/mm/aaaa [Maecenas aliquam dolor eget metus](#)
 dd/mm/aaaa [Fusce tristique lorem id metus](#)

Enlaces relacionados

[Pron placerat](#)
[Nulla in felis](#)
[Nam luctus](#)

Publicidad

Eiam fermentum, nisl tincidunt blandit interdum, massa velit posuere dolor, sed euismod sem odio at mi.
 Duis porta placerat arcu. Nullam felis pede, commodo vel, suscipit a, molestie vel, felis. Maecenas mattis est vel est.

[Seguir leyendo...](#)

Lore ipsum dolor sit amet, consectetur adipiscing elit



Nullam est lacus, suscipit ut, dapibus quis, condimentum ac, risus. Vivamus vestibulum, ipsum sollicitudin faucibus pharetra, dolor metus fringilla dui, vel aliquet pede diam tempor tortor. Vestibulum pulvinar urna et quam. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas.

Nullam vel turpis vitae dui imperdiet laoreet. Quisque eget ipsum. Donec commodo, turpis vel venenatis sollicitudin, quam ante convallis justo, sed eleifend justo lectus quis sapien. Ut consequat libero eget est.

[Seguir leyendo...](#)

Vivamus lobortis turpis ac ante fringilla faucibus



Quisque eget ipsum. Donec commodo, turpis vel venenatis sollicitudin, quam ante convallis justo, sed eleifend justo lectus quis sapien. Ut consequat libero eget est.

Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Nullam dictum hendrerit neque. Mauris id ligula non elit mattis semper. Fusce arcu ipsum, tempus eget, tincidunt at, imperdiet in, mi.

Sed fermentum cursus dolor. Aenean a diam. Phasellus feugiat. Donec tempor dignissim sem.

[Seguir leyendo...](#)

Phasellus blandit

Praesent sodales imperdiet augue. Mauris lorem felis, semper eu, tincidunt eu, sollicitudin eget, sem. Nulla facilisi. Morbi ut enim ut enim ultricies dapibus.

[Seguir leyendo...](#)

Nullam vel turpis

Donec commodo, turpis vel venenatis sollicitudin, quam ante convallis justo, sed eleifend justo lectus quis sapien. Ut consequat libero eget est.

[Seguir leyendo...](#)

[Nulla](#) | [Pharetra](#) | [Luctus](#) | [Ipsum](#) | [Pron](#) | [Placerat](#)

© Copyright Lorem ipsum

Figura 15.7. Página con colores e imágenes de fondo

15.6. Ejercicio 6

A partir del código HTML proporcionado:

```
<!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">
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
  <title>Ejercicio posicionamiento float</title>
  <style type="text/css">
    </style>
</head>

<body>
  <div>
    &laquo; Anterior &nbsp; Siguiente &raquo;
  </div>
</body>
</html>
```

Determinar las reglas CSS necesarias para que el resultado sea similar al mostrado en la siguiente imagen:

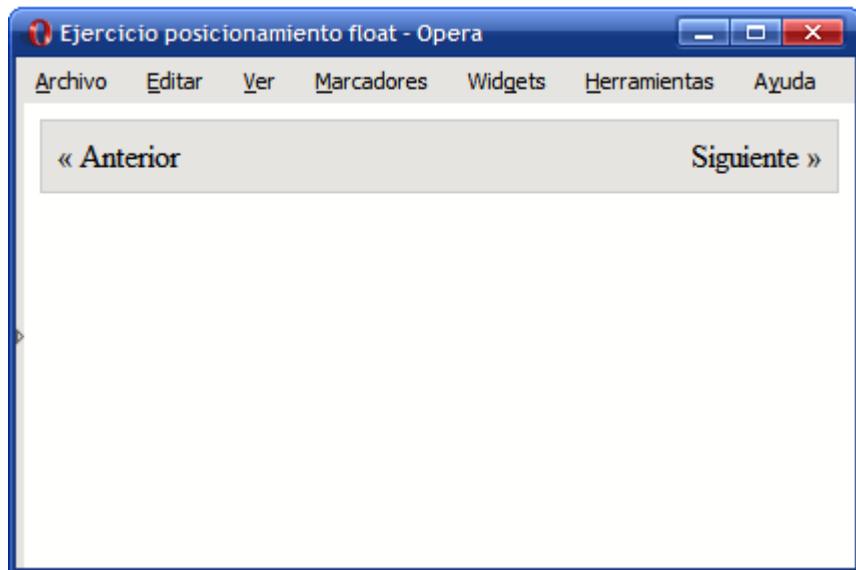


Figura 15.8. Elementos posicionados mediante float

15.7. Ejercicio 7

A partir del código HTML y CSS proporcionados, determinar las reglas CSS necesarias para añadir las siguientes propiedades a la tipografía de la página:

Logotipo

Buscar

[Lorem ipsum dolor sit amet](#)

Noticias

dd/mm/aaaa [Lorem ipsum dolor sit amet](#)
dd/mm/aaaa [Consectetuer adipiscing elit](#)
dd/mm/aaaa [Donec molestie nunc eu sapien](#)
dd/mm/aaaa [Maecenas aliquam dolor eget metus](#)
dd/mm/aaaa [Fusce tristique lorem id metus](#)
Enlaces relacionados
[Proin placerat](#)
[Nulla in felis](#)
[Nam luctus](#)

Publicidad

Etim fermentum, nisl tincidunt blandit interdum, massa velit posuere dolor, sed euismod sem odio at mi.
Duis porta placerat arcu. Nullam felis pede, commodo vel, suscipit a, molestie vel, felis.
Maecenas mattis est vel est.
[Seguir leyendo...](#)

Lorem ipsum dolor sit amet, consectetuer adipiscing elit

 Nullam est lacus, suscipit ut, dapibus quis, condimentum ac, risus. Vivamus vestibulum, ipsum sollicitudin faucibus pharetra, dolor metus fringilla dui, vel aliquet pede diam tempor tortor. Vestibulum pulvinar urna et quam. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Nullam vel turpis vitae dui imperdiet laoreet. Quisque eget ipsum.
Donec commodo, turpis vel venenatis sollicitudin, quam ante convallis justo, sed eleifend justo lectus quis sapien. Ut consequat libero eget est.
[Seguir leyendo...](#)

Vivamus lobortis turpis ac ante fringilla faucibus

 Quisque eget ipsum. Donec commodo, turpis vel venenatis sollicitudin, quam ante convallis justo, sed eleifend justo lectus quis sapien. Ut consequat libero eget est.
Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Nullam dictum hendrerit neque. Mauris id ligula non elit mattis semper. Fusce arcu ipsum, tempus eget, tincidunt at, imperdiet in, mi.
Sed fermentum cursus dolor. Aenean a diam. Phasellus feugiat. Donec tempor dignissim sem.
[Seguir leyendo...](#)

Phasellus blandit

Praesent sodales imperdiet augue. Mauris lorem felis, semper eu, tincidunt eu, sollicitudin eget, sem. Nulla facilisi. Morbi ut enim ut enim ultricies dapibus.

[Seguir leyendo...](#)

Nullam vel turpis

Donec commodo, turpis vel venenatis sollicitudin, quam ante convallis justo, sed eleifend justo lectus quis sapien. Ut consequat libero eget est.

[Seguir leyendo...](#)

[Nulla](#) | [Pharetra](#) | [Luctus](#) | [Ipsum](#) | [Proin](#) | [Placerat](#)

© Copyright Lorem ipsum

Figura 15.9. Página original

1. La fuente base de la página debe ser: color negro, tipo Arial, tamaño `0.9em`, interlineado `1.4`.
2. Los elementos `<h2>` de `.articulo` se muestran en color `#CC6600`, con un tamaño de letra de `1.6em`, un interlineado de `1.2` y un margen inferior de `0.3em`.
3. Los elementos del `#menu` deben mostrar un margen a su derecha de `1em` y los enlaces deben ser de color blanco y tamaño de letra `1.3em`.
4. El tamaño del texto de todos los contenidos de `#lateral` debe ser de `0.9em`. La fecha de cada noticia debe ocupar el espacio de toda su línea y mostrarse en color gris claro `#999`. El elemento `<h3>` de `#noticias` debe mostrarse de color `#003366`.
5. El texto del elemento `#publicidad` es de color gris oscuro `#555` y todos los enlaces de color `#CC6600`.
6. Los enlaces contenidos dentro de `.articulo` son de color `#CC6600` y todos los párrafos muestran un margen superior e inferior de `0.3em`.
7. Añadir las reglas necesarias para que el contenido de `#secundario` se vea como en la imagen que se muestra.
8. Añadir las reglas necesarias para que el contenido de `#pie` se vea como en la imagen que se muestra.

Logotipo

Buscar

Lorem Ipsum Dolor Sit Amet

Noticias

dd/mm/aaaa
[Lorem ipsum dolor sit amet](#)
dd/mm/aaaa
[Consectetuer adipiscing elit](#)
dd/mm/aaaa
[Donec molestie nunc eu sapien](#)
dd/mm/aaaa
[Maecenas aliquam dolor eget metus](#)
dd/mm/aaaa
[Fusce tristique lorem id metus](#)
[Enlaces relacionados](#)
[Proin placerat](#)
[Nulla in felis](#)
[Nam luctus](#)

Publicidad

Etiam fermentum, nisl tincidunt blandit interdum, massa velit posuere dolor, sed euismod sem odio at mi.
Duis porta placerat arcu. Nullam felis pede, commodo vel, suscipit a, molestie vel, felis.
Maecenas mattis est vel est.
[Seguir leyendo...](#)

Lorem ipsum dolor sit amet, consectetuer adipiscing elit



Nullam est lacus, suscipit ut, dapibus quis, condimentum ac, risus. Vivamus vestibulum, ipsum sollicitudin faucibus pharetra, dolor metus fringilla dui, vel aliquet pede diam tempor tortor.
Vestibulum pulvinar urna et quam. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Nullam vel turpis vitae dui imperdiet laoreet. Quisque eget ipsum.
Donec commodo, turpis vel venenatis sollicitudin, quam ante convallis justo, sed eleifend justo lectus quis sapien. Ut consequat libero eget est.
[Seguir leyendo...](#)

Vivamus lobortis turpis ac ante fringilla faucibus



Quisque eget ipsum. Donec commodo, turpis vel venenatis sollicitudin, quam ante convallis justo, sed eleifend justo lectus quis sapien. Ut consequat libero eget est.
Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Nullam dictum hendrerit neque. Mauris id ligula non elit mattis semper. Fusce arcu ipsum, tempus eget, tincidunt at, imperdiet in, mi.
Sed fermentum cursus dolor. Aenean a diam. Phasellus feugiat. Donec tempor dignissim sem.
[Seguir leyendo...](#)

Phasellus blandit

Praesent sodales imperdiet augue. Mauna lorem felis, semper eu, tincidunt eu, sollicitudin eget, sem. Nulla facilisi. Morbi ut enim ut enim ultricies dapibus.

[Seguir leyendo...](#)

Nullam vel turpis

Donec commodo, turpis vel venenatis sollicitudin, quam ante convallis justo, sed eleifend justo lectus quis sapien. Ut consequat libero eget est.

[Seguir leyendo...](#)

Nula | Pharetra | Luctus | Ibeam | Proin | Placerat

© Copyright Lorem ipsum

Figura 15.10. Página con propiedades tipográficas

15.8. Ejercicio 8

Definir las reglas CSS que permiten mostrar los enlaces con los siguientes estilos:

1. En su estado normal, los enlaces se muestran de color rojo #CC0000.
2. Cuando el usuario pasa su ratón sobre el enlace, se muestra con un color de fondo rojo #CC0000 y la letra de color blanco #FFF.
3. Los enlaces visitados se muestran en color gris claro #CCC.



Figura 15.11. Enlaces con estilos aplicados mediante CSS

15.9. Ejercicio 9

Definir las reglas CSS que permiten mostrar una galería de imágenes similar a la que se muestra en la siguiente imagen:

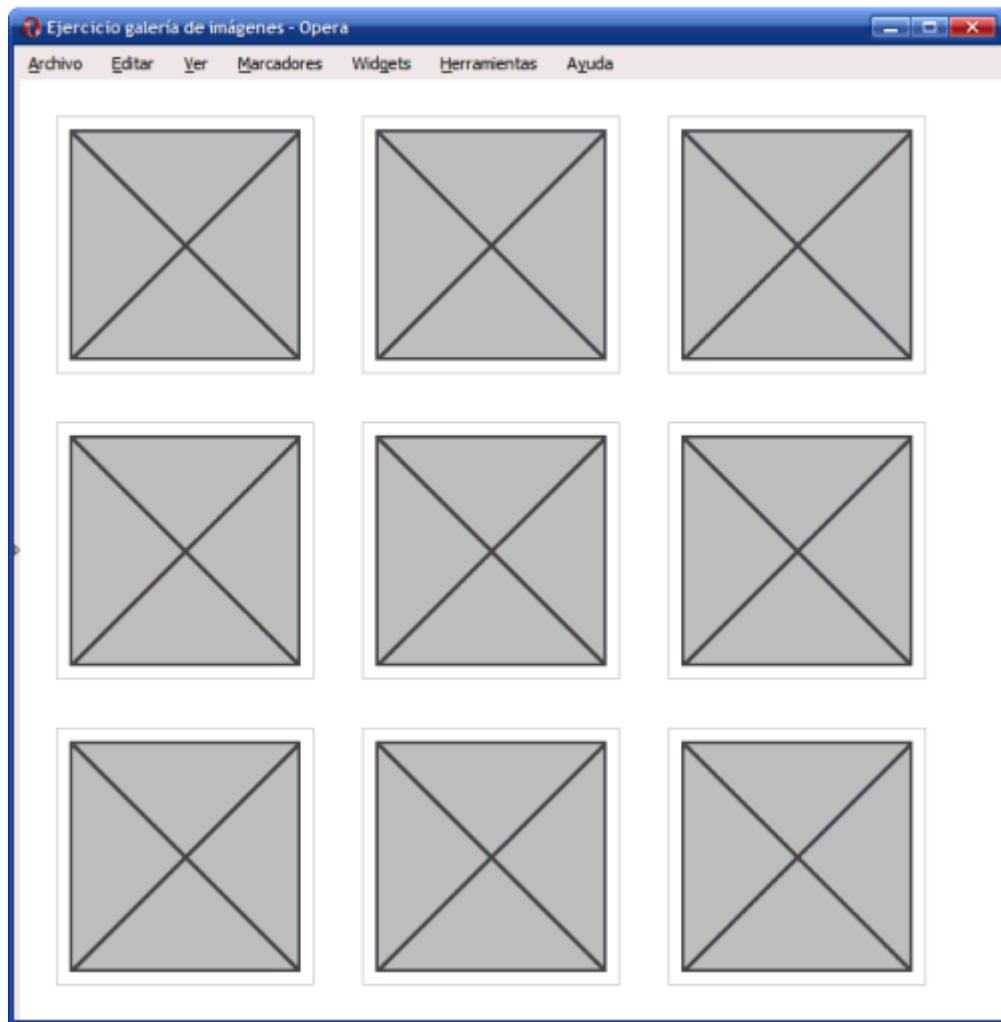


Figura 15.12. Galería de imágenes construida con CSS

15.10. Ejercicio 10

Modificar el menú vertical sencillo para que muestre el siguiente comportamiento:

1. Los elementos deben mostrar una imagen de fondo (`flecha_inactiva.png`):

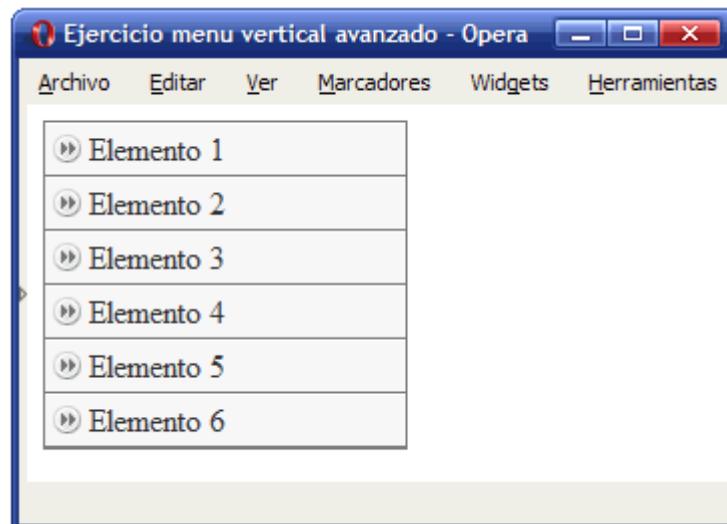


Figura 15.13. Menú vertical con imagen de fondo

2. Cuando se pasa el ratón por encima de un elemento, se debe mostrar una imagen alternativa (`flecha_activa.png`):

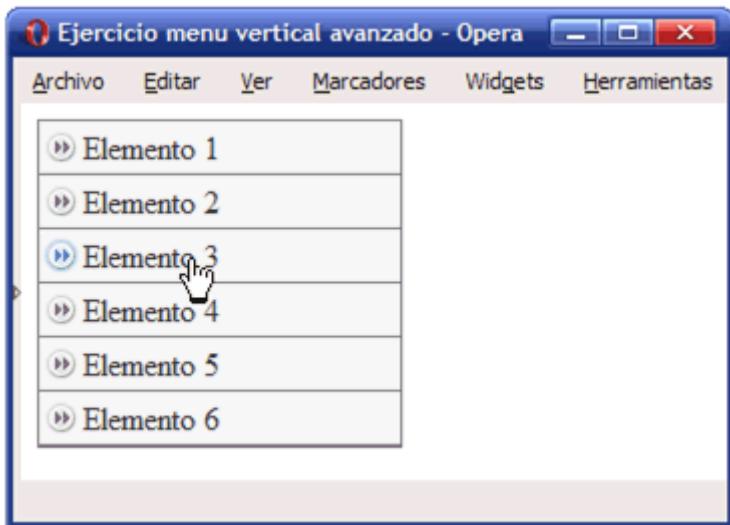


Figura 15.14. Menú vertical con imagen de fondo alternativa

3. El color de fondo del elemento también debe variar ligeramente y mostrar un color gris más oscuro (#E4E4E4) cuando se pasa el ratón por encima:

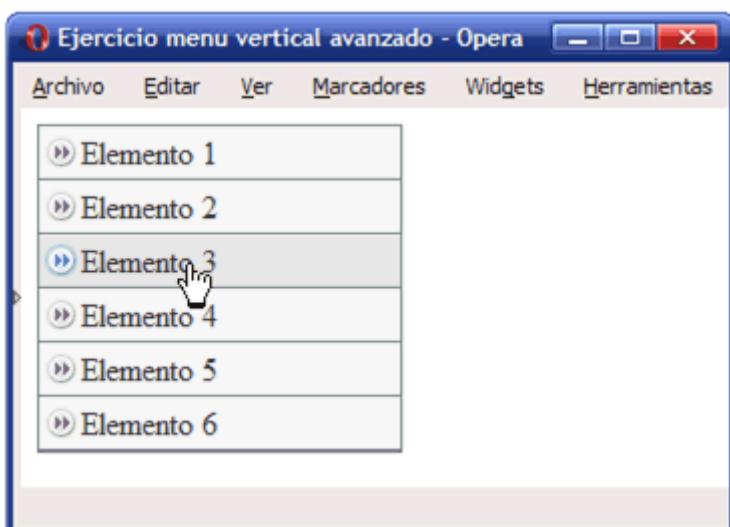


Figura 15.15. Menú vertical con imagen de fondo y color alternativos

4. El comportamiento anterior se debe producir cuando el usuario pasa el ratón por encima de cualquier zona del elemento del menú, no solo cuando se pasa el ratón por encima del texto del elemento (este problema solo sucede con Internet Explorer):



Figura 15.16. Aspecto final del menú vertical avanzado creado con CSS

15.11. Ejercicio 11

Determinar las reglas CSS necesarias para mostrar la siguiente tabla con el aspecto final mostrado en la imagen (modificar el código HTML que se considere necesario añadiendo los atributos class oportunos).

Tabla original:



Cambio	Compra	Venta	Máximo	Mínimo
Euro/Dolar	1.2524	1.2527	1.2539	1.2488
Dolar/Yen	119.01	119.05	119.82	119.82
Libra/Dolar	1.8606	1.8611	1.8651	1.8522
Euro/Yen	149.09	149.13	149.79	148.96

Figura 15.17. Aspecto original de la tabla

Tabla final:

Cambio	Compra	Venta	Máximo	Mínimo
€ Euro/Dolar	1.2524	1.2527	1.2539	1.2488
\$ Dolar/Yen	119.01	119.05	119.82	119.82
£ Libra/Dolar	1.8606	1.8611	1.8651	1.8522
¥ Yen/Euro	0.6711	0.6705	0.6676	0.6713

Figura 15.18. Aspecto definitivo de la tabla después de aplicar estilos CSS

1. Alinear el texto de las celdas, cabeceras y título. Definir los bordes de la tabla, celdas y cabeceras (color gris oscuro #333).

Cambio	Compra	Venta	Máximo	Mínimo
Euro/Dolar	1.2524	1.2527	1.2539	1.2488
Dolar/Yen	119.01	119.05	119.82	119.82
Libra/Dolar	1.8606	1.8611	1.8651	1.8522
Yen/Euro	0.6711	0.6705	0.6676	0.6713

Figura 15.19. Tabla con texto alineado y bordes

2. Formatear las cabeceras de fila y columna con la imagen de fondo correspondiente en cada caso (fondo_gris.gif, euro.png, dolar.png, yen.png, libra.png). Modificar el tipo de letra de la tabla y utilizar Arial. El color azul claro es #E6F3FF.

The screenshot shows a table with five rows and five columns. The columns are labeled: Cambio, Compra, Venta, Máximo, and Mínimo. The rows contain currency exchange rates:

Cambio	Compra	Venta	Máximo	Mínimo
€ Euro/Dolar	1.2524	1.2527	1.2539	1.2488
\$ Dolar/Yen	119.01	119.05	119.82	119.82
£ Libra/Dolar	1.8606	1.8611	1.8651	1.8522
¥ Yen/Euro	0.6711	0.6705	0.6676	0.6713

Figura 15.20. Tabla con colores e imágenes de fondo

3. Mostrar un color alterno en las filas de datos (color amarillo claro #FFFFCC).

The screenshot shows the same table as Figure 15.20, but with alternating row colors. The even-numbered rows (second, fourth, and fifth) have a light yellow background color (#FFFFCC). The odd-numbered rows (first and third) have a white background.

Cambio	Compra	Venta	Máximo	Mínimo
€ Euro/Dolar	1.2524	1.2527	1.2539	1.2488
\$ Dolar/Yen	119.01	119.05	119.82	119.82
£ Libra/Dolar	1.8606	1.8611	1.8651	1.8522
¥ Yen/Euro	0.6711	0.6705	0.6676	0.6713

Figura 15.21. Tabla con colores de fila alternos

15.12. Ejercicio 12

A partir del código HTML proporcionado:

- 1) Aplicar las reglas CSS necesarias para que el formulario muestre el siguiente aspecto:

Formulario de alta

Nombre y apellidos *	Dirección *
<input type="text"/>	<input type="text"/>
Nombre	Calle, número, piso, puerta
<input type="text"/>	<input type="text"/>
Primer apellido	Código postal
<input type="text"/>	<input type="text"/>
Segundo apellido	Municipio
	<input type="text"/>
	Provincia
	<input type="text"/>
	País
Email	Teléfono *
<input type="text"/>	<input type="text"/>
	<input type="text"/>
	Fijo
	<input type="text"/>
	Móvil
Darme de alta →	

Figura 15.22. Formulario estructurado a dos columnas

2) Cuando el usuario pasa el ratón por encima de cada grupo de elementos de formulario (es decir, por encima de cada ``) se debe modificar su color de fondo (sugerencia: color amarillo claro #FF9). Además, cuando el usuario se posiciona en un cuadro de texto, se debe modificar su borde para resaltar el campo que está activo cada momento (sugerencia: color amarillo #E6B700):

Formulario de alta

Nombre y apellidos *
<input type="text"/>
Nombre
<input type="text"/>
Primer apellido
<input type="text"/>
Segundo apellido

Figura 15.23. Mejoras en los campos de formulario

3) Utilizando el menor número de reglas CSS, cambiar el aspecto del formulario para que se muestre como la siguiente imagen:

Formulario de alta

Nombre y apellidos *

Nombre
 Primer apellido
 Segundo apellido

Dirección *

Calle, número, piso, puerta
 Código postal Municipio
 Provincia País

Email

Teléfono *

Fijo Móvil

Darme de alta →

Figura 15.24. Formulario estructurado a una columna

- 4) Cuando el usuario pasa el ratón por encima de un grupo de elementos de formulario (es decir, por encima de cada ``) se debe mostrar el mensaje de ayuda asociado. Añadir las reglas CSS necesarias para que el formulario tenga el aspecto definitivo mostrado en la siguiente imagen:

Formulario de alta

Nombre y apellidos *

Nombre

 Primer apellido

 Segundo apellido

Dirección *

Calle, número, piso, puerta
 Calle, número, piso, puerta
 Código postal
 Municipio
 Provincia
 País

El código postal es imprescindible para poder recibir los pedidos

Email

Teléfono *

Fijo Móvil

Darme de alta →

Figura 15.25. Aspecto final del formulario

Código HTML:

```
<!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>Ejercicio 12 - Formulario de alta</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />

<style type="text/css">
</style>
</head>

<body>
<div id="contenedor">

<h2>Formulario de alta</h2>

<form method="post" action="#">
<ul>
<li>
```

```
<label class="titulo" for="nombre">Nombre y apellidos <span>
<span></span></label>
<div>
  <span>
    <input id="nombre" name="nombre" value="" />
    <label for="nombre">Nombre</label>
  </span>

  <span>
    <input id="apellido1" name="apellido1" value="" />
    <label for="apellido1">Primer apellido</label>
  </span>

  <span>
    <input id="apellido2" name="apellido2" value="" />
    <label for="apellido2">Segundo apellido</label>
  </span>
</div>

<p class="ayuda">No te olvides de escribir también tu segundo apellido</p>
</li>

<li>
  <label class="titulo" for="direccion">Dirección <span>
  <span></span></label>

  <div>
    <span>
      <input id="direccion" name="direccion" value="" />
      <label for="direccion">Calle, número, piso, puerta</label>
    </span>

    <span>
      <input id="codigopostal" name="codigopostal" value="" />
      <label for="codigopostal">Código postal</label>
    </span>

    <span>
      <input id="municipio" name="municipio" value="" />
      <label for="municipio">Municipio</label>
    </span>

    <span>
      <select id="provincia" name="provincia">
        <option value=""></option>
        <option value="provincial1">Provincia 1</option>
        <option value="provincia2">Provincia 2</option>
        <option value="provincia3">Provincia 3</option>
      </select>
      <label for="provincia">Provincia</label>
    </span>

    <span>
      <select id="pais" name="pais">
        <option value=""></option>
```

```
<option value="pais1">País 1</option>
<option value="pais2">País 2</option>
<option value="pais3">País 3</option>
</select>
<label for="pais">País</label>
</span>
</div>

<p class="ayuda">El código postal es imprescindible para poder recibir los pedidos</p>
</li>

<li>
    <label class="titulo" for="email">Email</label>

    <div>
        <span>
            <input id="email" name="email" value="" />
        </span>
    </div>

    <p class="ayuda">Asegúrate de que sea válido</p>
</li>

<li>
    <label class="titulo" for="telefonofijo">Teléfono <span
class="requerido">*</span></label>

    <div>
        <span>
            <input id="telefonofijo" name="telefonofijo" value="" />
            <label for="telefonofijo">Fijo</label>
        </span>

        <span>
            <input id="telefonomovil" name="telefonomovil" value="" />
            <label for="telefonomovil">Móvil</label>
        </span>
    </div>

    <p class="ayuda">Sin prefijo de país y sin espacios en blanco</p>
</li>

<li>
    <input id="alta" type="submit" value="Darme de alta &rarr;" />
</li>

</ul>
</form>

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

15.13. Ejercicio 13

Determinar las reglas CSS necesarias para mostrar la página HTML que se proporciona con el estilo que se muestra en la siguiente imagen:



Figura 15.26. Aspecto final que debe mostrar la página HTML proporcionada

A continuación se indica una propuesta de los pasos que se pueden seguir para obtener el aspecto final deseado:

- Añadir los estilos básicos de la página (tipo de letra Verdana, color de letra #192666, imagen de fondo llamada `fondo.gif`, color de fondo #F2F5FE).
- Definir la estructura básica de la página: anchura fija de 770 píxel, centrada en la ventana del navegador, cabecera y pie, columna central de contenidos de anchura 530 píxel y columna secundaria de contenidos de 200 píxel de anchura.
- La cabecera tiene una altura de 100 píxel y una imagen de fondo llamada `cabecera.jpg`.
- Los elementos del menú de navegación tienen un color de fondo #253575, un color de letra #B5C4E3. Cuando el ratón pasa por encima de cada elemento, su color de fondo cambia a #31479B. Los elementos seleccionados se muestran con un color de fondo blanco y un color de letra #FF9000:

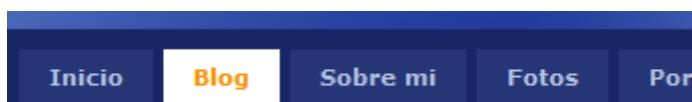


Figura 15.27. Imagen detallada del aspecto que muestran los elementos del menú de navegación

- Con la ayuda de las imágenes que se proporcionan, mostrar cada uno de los artículos de contenido con el estilo que se muestra en la siguiente imagen:

The screenshot shows a web page layout. At the top, there's a header bar with a light blue background. Below it, the main content area has a white background. The title of the article is "Lorem ipsum dolor sit amet" in bold black font. Below the title, there's a timestamp "dd-mm-aaaa 00:00", a category link "diseño", an author link "Nombre Apellido", and a comment link "Añadir comentario". The main text of the article is placeholder text: "Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aliquam pellentesque enim blandit enim bibendum blandit. Integer eu leo ac est aliquet imperdiet. Quisque nec justo id augue posuere malesuada. Nullam ac metus. Cras non leo ut est placerat condimentum.". At the bottom of the article, there's a link "Seguir leyendo...". To the right of the main content area, there's a sidebar with a light blue background. It contains sections for "Sobre mí" (with fields for "Nombre apellido", "Edad", and "Ciudad de residencia"), a link "Mi perfil público", and a section for "Categorías" with five categories: "Categoría 1", "Categoría 2", "Categoría 3", "Categoría 4", and "Categoría 5".

Figura 15.28. Aspecto de un artículo de la sección principal de contenidos

- Añadir los estilos adecuados para mostrar los elementos de la columna secundaria de contenidos con el siguiente aspecto.

The sidebar content sections are as follows:

- Sobre mí**: Includes fields for Nombre apellido (with a placeholder icon), Edad, Ciudad de residencia, and a link Mi perfil público.
- Categorías**: Includes five categories: Categoría 1, Categoría 2, Categoría 3, Categoría 4, and Categoría 5.

Figura 15.29. Aspecto de las secciones de la columna secundaria de contenidos

Capítulo 16. Ejercicios resueltos

16.1. Solución ejercicio 1

```
/* Todos los elementos de la pagina */
* { font: 1em/1.3 Arial, Helvetica, sans-serif; }

/* Todos los párrafos de la pagina */
p { color: #555; }

/* Solo los párrafos contenidos en #primero */
#primero p { color: #336699; }

/* Todos los enlaces la pagina */
a { color: #CC3300; }

/* Los elementos em contenidos en #primero */
#primero em { background: #FFFFCC; padding: .1em; }

/* Todos los elementos em de tipo especial en toda la pagina */
em.especial { background: #FFCC99; border: 1px solid #FF9900; padding: .1em; }

/* Elementos span contenidos en .normal */
.normal span { font-weight: bold; }
```

16.2. Solución ejercicio 2

```
<!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">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>Ejercicio de selectores</title>
<style type="text/css">
/* No borrar la siguiente regla CSS porque es necesaria para ver los bordes de la tabla */
table, tr, th, td {border:1px solid #000; border-collapse:collapse; padding:5px;}

h1#titulo { color: teal; }

strong { color: red; }
span.destacado { color: orange; }

h2#subtitulo { color: blue; }

span.especial { color: purple; }

a { color: red; }
a em { color: blue; }

div#adicional p { color: olive; }
div#adicional span#especial { color: fuchsia; }
```

```
div#adicional a { color: green; }

caption { color: blue; }
td { color: green; }
td strong { color: orange; }
th { color: red; }
th.especial { color: orange; }
</style>
</head>

<body>
<h1 id="titulo">Lorem ipsum dolor sit amet</h1>

<p>Nulla pretium. Sed tempus nunc vitae neque. <strong>Suspendisse gravida</strong>, metus a scelerisque sollicitudin, lacinia velit ultricies nisl, nonummy tempus neque diam quis felis. <span class="destacado">Etiam sagittis tortor</span> sed arcu sagittis tristique.</p>

<h2 id="subtitulo">Aliquam tincidunt, sem eget volutpat porta</h2>

<p>Vivamus velit dui, placerat vel, feugiat in, ornare et, urna. <a href="#">Aenean turpis metus, <em>aliquam non</em>, tristique in</a>, pretium varius, sapien. Proin vitae nisi. Suspendisse <span class="especial">porttitor purus ac elit</span>. Suspendisse eleifend odio at dui. In in elit sed metus pretium elementum.</p>

<table summary="Descripción de la tabla y su contenido">
<caption>Título de la tabla</caption>
<thead>
<tr>
<th scope="col"></th>
<th scope="col" class="especial">Título columna 1</th>
<th scope="col" class="especial">Título columna 2</th>
</tr>
</thead>

<tfoot>
<tr>
<th scope="col"></th>
<th scope="col">Título columna 1</th>
<th scope="col">Título columna 2</th>
</tr>
</tfoot>

<tbody>
<tr>
<th scope="row" class="especial">Título fila 1</th>
<td>Donec purus ipsum</td>
<td>Curabitur <em>blandit</em></td>
</tr>
<tr>
<th scope="row">Título fila 2</th>
<td>Donec <strong>purus ipsum</strong></td>
<td>Curabitur blandit</td>
</tr>
</tbody>
```

```

</table>

<div id="adicional">
<p>Donec purus ipsum, posuere id, venenatis at, <span>placerat ac, lorem</span>.
Curabitur blandit, eros sed gravida aliquet, risus justo
porta lorem, ut mollis lectus tortor in orci. Pellentesque nec augue.</p>

<p>Fusce nec felis eu diam pretium adipiscing. <span id="especial">Nunc elit elit,
vehicula vulputate</span>, venenatis in,
posuere id, lorem. Etiam sagittis, tellus in ultrices accumsan, diam nisi feugiat ante,
eu congue magna mi non nisl.</p>

<p>Vivamus ultrices aliquet augue. <a href="#">Donec arcu pede, pretium vitae</a>,
rutrum aliquet, tincidunt blandit, pede.
Aliquam in nisi. Suspendisse volutpat. Nulla facilisi. Ut ullamcorper nisi quis mi.</p>
</div>

</body>
</html>

```

16.3. Solución ejercicio 3

```

/* === IMPORTANTE =====
No modificar estos estilos, ya que son imprescindibles para
que la página se vea correctamente.
===== */

/*-- Básico -----*/
ul, ul li { margin: 0; padding: 0; list-style: none; }
h1, h2, h3, p, form { margin: 0; padding: 0; }
.clear { clear: both; }
img { border: none; }

/*-- Layout -----*/
#contenedor {
    width: 90%;
    max-width: 900px;
    width: expression(document.body.clientWidth > 901? "900px": "auto");
    margin: 0 auto;
}

#cabeza, #menu, #lateral, #contenido, #contenido #principal, #contenido #secundario,
#pie {
    border: 2px solid #777;
}

#cabeza { clear: both; }
#menu { clear: both; }
#lateral { float: left; width: 20%; }
#contenido { float: right; width: 78%; }
#contenido #principal { float: left; width: 78%; }
#contenido #secundario { float: right; width: 20%; }
#pie { clear: both; }

/*-- Cabecera -----*/

```

```
#cabecera #logo { float: left; }
#cabecera #buscador { float: right; }

/*-- Menu -----
#menu ul#menu_principal li { display: inline; float: left; }

/*-- Sección Principal -----
#contenido #principal .articulo img { width: 100px; float: left; }

/*-- Pie de página -----
#pie .enlaces { float: left; }
#pie .copyright { float: right; }

/* === IMPORTANTE =====
A partir de aquí, se pueden añadir todos los estilos propios que
sean necesarios.
===== */

#cabecera,
#menu,
#lateral,
#lateral #noticias,
#lateral #publicidad,
#contenido,
#contenido #principal,
#contenido #secundario,
#pie {
    padding: .5em;
}

#lateral {
    padding: 0;
}

#cabeza {
    padding: 1em;
}

#menu {
    margin-bottom: .5em;
}

#contenido {
    width: 77%;
    padding: 0;
}

#contenido #principal {
    width: 73%;
}

#pie {
    padding: .5em 0;
    margin-top: 1em;
```

```

}

#contenido #principal .articulo {
    margin-bottom: 1em;
}

#contenido #principal .articulo img {
    margin: .5em;
}

#lateral #publicidad {
    margin-top: 1em;
}

```

16.4. Solución ejercicio 4

```

/* === IMPORTANTE =====
No modificar estos estilos, ya que son imprescindibles para
que la página se vea correctamente.
===== */

/*-- Básico -----
ul, ul li { margin: 0; padding: 0; list-style: none; }
h1, h2, h3, p, form { margin: 0; padding: 0; }
.clear { clear: both; }
img { border: none; }

/*-- Layout -----
#contenedor {
    width: 90%;
    max-width: 900px;
    width: expression(document.body.clientWidth > 901? "900px": "auto");
    margin: 0 auto;
}

#cabeza, #menu, #lateral, #contenido, #contenido #principal, #contenido #secundario,
#pie {
    border: 2px solid #777;
}

#cabeza { clear: both; }
#menu { clear: both; }
#lateral { float: left; width: 20%; }
#contenido { float: right; width: 78%; }
#contenido #principal { float: left; width: 78%; }
#contenido #secundario { float: right; width: 20%; }
#pie { clear: both; }

/*-- Cabecera -----
#cabeza #logo { float: left; }
#cabeza #buscador { float: right; }

/*-- Menú -----
#menu ul#menu_principal li { display: inline; float: left; }

```

```
/*-- Sección Principal -----*/
#contenido #principal .articulo img { width: 100px; float: left; }

/*-- Pie de página -----*/
#pie .enlaces { float: left; }
#pie .copyright { float: right; }

/* === IMPORTANTE =====
   A partir de aquí, se pueden añadir todos los estilos propios que
   sean necesarios.
===== */

#cabecera,
#menu,
#lateral,
#lateral #noticias,
#lateral #publicidad,
#contenido,
#contenido #principal,
#contenido #secundario,
#pie {
    padding: .5em;
    border: none;
}

#lateral {
    padding: 0;
}

#cabecera {
    padding: 1em;
}

#menu {
    margin-bottom: .5em;
    border-bottom: 1px solid #004C99;
}

#contenido {
    width: 77%;
    padding: 0;
}

#contenido #principal {
    width: 73%;
}

#contenido #secundario {
    border: 1px solid #C60;
}

#pie {
    padding: .5em 0;
    margin-top: 1em;
```

```

border-top: 1px solid #C5C5C5;
border-bottom: 1px solid #C5C5C5;
}

#contenido #principal .articulo {
  margin-bottom: 1em;
}

#contenido #principal .articulo img {
  margin: .5em;
}

#lateral #noticias {
  border: 1px solid #C5C5C5;
}

#lateral #publicidad {
  margin-top: 1em;
  border: 1px dashed #C60;
}

```

16.5. Solución ejercicio 5

```

/* === IMPORTANTE =====
No modificar estos estilos, ya que son imprescindibles para
que la página se vea correctamente.
===== */

/*-- Básico -----
ul, ul li { margin: 0; padding: 0; list-style: none; }
h1, h2, h3, p, form { margin: 0; padding: 0; }
.clear { clear: both; }
img { border: none; }

/*-- Layout -----
#contenedor {
  width: 90%;
  max-width: 900px;
  width: expression(document.body.clientWidth > 901? "900px": "auto");
  margin: 0 auto;
}

#cabeza, #menu, #lateral, #contenido, #contenido #principal, #contenido #secundario,
#pie {
  border: 2px solid #777;
}

#cabeza { clear: both; }
#menu { clear: both; }
#lateral { float: left; width: 20%; }
#contenido { float: right; width: 78%; }
#contenido #principal { float: left; width: 78%; }
#contenido #secundario { float: right; width: 20%; }
#pie { clear: both; }

```

```
/*-- Cabecera -----*/
#cabecera #logo { float: left; }
#cabecera #buscador { float: right; }

/*-- Menu -----*/
#menu ul#menu_principal li { display: inline; float: left; }

/*-- Sección Principal -----*/
#contenido #principal .articulo img { width: 100px; float: left; }

/*-- Pie de página -----*/
#pie .enlaces { float: left; }
#pie .copyright { float: right; }

/* === IMPORTANTE ===== */
A partir de aquí, se pueden añadir todos los estilos propios que
sean necesarios.
===== */

#cabecera,
#menu,
#lateral,
#lateral #noticias,
#lateral #publicidad,
#contenido,
#contenido #principal,
#contenido #secundario,
#pie {
    padding: .5em;
    border: none;
}

#lateral {
    padding: 0;
}

#cabecera {
    padding: 1em;
}

#cabecera h1 {
    background: url(..//comun/imagenes/logo.gif) no-repeat -5px -10px;
}

#cabecera h1 span {
    visibility: hidden;
}

#menu {
    margin-bottom: .5em;
    border-bottom: 1px solid #004C99;
    background: url(..//comun/imagenes/fondo_menu.gif) repeat-x;
}
```

```
#contenido {  
    width: 77%;  
    padding: 0;  
}  
  
#contenido #principal {  
    width: 73%;  
}  
  
#contenido #secundario {  
    border: 1px solid #C60;  
}  
  
#contenido #secundario h2 {  
    background: #DB905C;  
    padding: .2em;  
}  
  
#pie {  
    padding: .5em 0;  
    margin-top: 1em;  
    border-top: 1px solid #C5C5C5;  
    border-bottom: 1px solid #C5C5C5;  
    background: #F8F8F8;  
}  
  
#contenido #principal .articulo {  
    margin-bottom: 1em;  
}  
  
#contenido #principal .articulo img {  
    margin: .5em;  
}  
  
#lateral #noticias {  
    border: 1px solid #C5C5C5;  
    background: #F8F8F8;  
}  
  
#lateral #publicidad {  
    margin-top: 1em;  
    border: 1px dashed #C60;  
    background: #FFF6CD;  
}
```

16.6. Solución ejercicio 6

```
<!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">  
<head>  
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />  
<title>Ejercicio posicionamiento float</title>  
<style type="text/css">  
div#paginacion {
```

```

        border: 1px solid #CCC;
        background-color: #E0E0E0;
        padding: .5em;
    }
    .derecha {
        float: right;
    }
    .izquierda {
        float: left;
    }
    div.clear {
        clear: both;
    }
</style>
</head>

<body>
<div id="paginacion">
<span class="izquierda">&laquo; Anterior</span> <span class="derecha">Siguiente
&raquo;</span>
<div class="clear"></div>
</div>
</body>
</html>

```

16.7. Solución ejercicio 7

```

/* === IMPORTANTE =====
   No modificar estos estilos, ya que son imprescindibles para
   que la página se vea correctamente.
===== */

/*-- Básico -----*/
ul, ul li { margin: 0; padding: 0; list-style: none; }
h1, h2, h3, p, form { margin: 0; padding: 0; }
.clear { clear: both; }
img { border: none; }

/*-- Layout -----*/
#contenedor {
    width: 90%;
    max-width: 900px;
    width: expression(document.body.clientWidth > 901? "900px": "auto");
    margin: 0 auto;
}

#cabecera, #menu, #lateral, #contenido, #contenido #principal, #contenido #secundario,
#pie {
    border: 2px solid #777;
}

#cabecera { clear: both; }
#menu { clear: both; }
#lateral { float: left; width: 20%; }
#contenido { float: right; width: 78%; }

```

```
#contenido #principal { float: left; width: 78%; }
#contenido #secundario { float: right; width: 20%; }
#pie { clear: both; }

/*-- Cabecera -----*/
#cabeza #logo { float: left; }
#cabeza #buscador { float: right; }

/*-- Menú -----*/
#menu ul#menu_principal li { display: inline; float: left; }

/*-- Sección Principal -----*/
#contenido #principal .articulo img { width: 100px; float: left; }

/*-- Pie de página -----*/
#pie .enlaces { float: left; }
#pie .copyright { float: right; }

/* === IMPORTANTE =====
   A partir de aquí, se pueden añadir todos los estilos propios que
   sean necesarios.
===== */

body {
    font: .9em/1.4 Arial, Helvetica, sans-serif;
    color: #000;
}

#cabeza,
#menu,
#lateral,
#lateral #noticias,
#lateral #publicidad,
#contenido,
#contenido #principal,
#contenido #secundario,
#pie {
    padding: .5em;
    border: none;
}

#lateral {
    padding: 0;
    font-size: .9em;
}

#cabeza {
    padding: 1em;
}

#cabeza h1 {
    background: url(..//comun/imagenes/logo.gif) no-repeat -5px -5px;
}

#cabeza h1 span {
```

```
    visibility: hidden;
}

#menu {
    margin-bottom: .5em;
    border-bottom: 1px solid #004C99;
    background: url(..../comun/imagenes/fondo_menu.gif) repeat-x;
}

#menu li {
    margin-right: 1em;
    font-size: 1.3em;
}

#menu li a {
    color: #FFF;
}

#contenido {
    width: 77%;
    padding: 0;
}

#contenido #principal {
    width: 73%;
}

#contenido #secundario {
    border: 1px solid #C60;
}

#contenido #secundario h2 {
    background: #DB905C;
    padding: .2em;
    font-size: 1em;
    color: #FFF;
}

#contenido #secundario p {
    margin: .5em 0;
}

#pie {
    padding: .5em 0;
    margin-top: 1em;
    border-top: 1px solid #C5C5C5;
    border-bottom: 1px solid #C5C5C5;
    background: #F8F8F8;
    color: #555;
    font-size: .75em;
}

#contenido #principal .articulo {
    margin-bottom: 1em;
}
```

```
#contenido #principal .articulo p {  
    margin: .3em 0;  
}  
  
#contenido #principal .articulo a {  
    color: #C60;  
}  
  
#contenido #principal .articulo h2 {  
    color: #C60;  
    font-size: 1.6em;  
    line-height: 1.2;  
    margin-bottom: .3em;  
}  
  
#contenido #principal .articulo img {  
    margin: .5em;  
}  
  
#lateral #noticias {  
    border: 1px solid #C5C5C5;  
    background: #F8F8F8;  
}  
  
#lateral #noticias h3 {  
    color: #036;  
}  
  
#lateral #noticias span.fecha {  
    display: block;  
    color: #999;  
}  
  
#lateral #publicidad {  
    margin-top: 1em;  
    border: 1px dashed #C60;  
    background: #FFF6CD;  
    color: #555;  
}  
  
#lateral #publicidad a {  
    color: #C60;  
}
```

16.8. Solución ejercicio 8

```
<!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">  
<head>  
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />  
<title>Ejercicio de pseudo clases de enlaces</title>  
<style type="text/css">  
a {
```

```
margin: 1em 0;
color: #CC0000;
float: left;
clear: left;
padding: 2px;
}
a:hover {
  text-decoration: none;
  background-color: #CC0000;
  color: #FFF;
}
a:visited {
  color: #CCC;
}
</style>
</head>

<body>
<a href="#">Enlace número 1</a>

<a href="#">Enlace número 2</a>

<a href="#">Enlace número 3</a>

<a href="#">Enlace número 4</a>

<a href="#">Enlace número 5</a>
</body>
</html>
```

16.9. Solución ejercicio 9

```
<!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">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
<title>Ejercicio galería de imágenes</title>
<style type="text/css">
#galeria {
  /* En el navegador Internet Explorer versión 6 y anteriores no funciona la propiedad
  "max-width" */
  max-width: 650px;
}
#galeria img {
  float: left;
  margin: 1em;
  padding: .5em;
  border: 1px solid #CCC;
}
</style>
</head>

<body>
<div id="galeria">
```

```










</div>
</body>
</html>

```

16.10. Solución ejercicio 10

```

<!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">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
<title>Ejercicio menu vertical avanzado</title>
<style type="text/css">
ul.menu {
    width: 180px;
    list-style: none;
    margin: 0;
    padding: 0;
    border: 1px solid #7C7C7C;
}
ul.menu li {
    border-bottom: 1px solid #7C7C7C;
    border-top: 1px solid #FFF;
    background: #F4F4F4;
}
ul.menu li a:link, ul.menu li a:visited {
    width: 158px;
    padding: .2em 0 .2em 1.3em;
    display: block;
    text-decoration: none;
    color: #333;
    background: #F4F4F4 url(imagenes/flecha_inactiva.png) no-repeat .2em center;
}
ul.menu li a:hover, ul.menu li a:active {
    background: #E4E4E4 url(imagenes/flecha_activa.png) no-repeat .2em center;
}
</style>
</head>

<body>
<ul class="menu">
    <li><a href="#" title="Enlace genérico">Elemento 1</a></li>
    <li><a href="#" title="Enlace genérico">Elemento 2</a></li>
    <li><a href="#" title="Enlace genérico">Elemento 3</a></li>
    <li><a href="#" title="Enlace genérico">Elemento 4</a></li>
    <li><a href="#" title="Enlace genérico">Elemento 5</a></li>

```

```
<li><a href="#" title="Enlace genérico">Elemento 6</a></li>
</ul>
</body>
</html>
```

16.11. Solución ejercicio 11

```
<!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">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
<title>Ejercicio formatear tabla</title>
<style type="text/css">
table {
    font: .9em Arial, Helvetica, sans-serif;
    border: 1px solid #333;
    border-collapse: collapse;
    text-align: center;
}
table th {
    background: #F5F5F5 url(imagenes/fondo_gris.gif) repeat-x;
    padding: 0 .3em;
}
table th.euro {
    background: #E6F3FF url(imagenes/euro.png) no-repeat left center;
    padding: 0 .3em 0 1.2em;
}
table th.dolar {
    background: #E6F3FF url(imagenes/dolar.png) no-repeat left center;
    padding: 0 .3em 0 1.2em;
}
table th.libra {
    background: #E6F3FF url(imagenes/libra.png) no-repeat left center;
    padding: 0 .3em 0 1.2em;
}
table th.yen {
    background: #E6F3FF url(imagenes/yen.png) no-repeat left center;
    padding: 0 .3em 0 1.2em;
}
table th, table, td {
    border: 1px solid #333;
    line-height: 2em;
}
.fila {
    text-align: left;
}
.par {
    background-color:#FFFFCC;
}
table tr:hover {
    background: #FFFF66 !important;
}
</style>
</head>
```

```

<body>
<table summary="Tipos de cambio">
    <tr>
        <th scope="col">Cambio</th>
        <th scope="col">Compra</th>
        <th scope="col">Venta</th>
        <th scope="col">M&aacute;ximo</th>
        <th scope="col">M&iacute;nimo</th>
    </tr>
    <tr class="par">
        <th scope="row" class="fila euro">Euro/Dolar</th>
        <td>1.2524</td>
        <td>1.2527</td>
        <td>1.2539</td>
        <td>1.2488</td>
    </tr>
    <tr>
        <th scope="row" class="fila dolar">Dolar/Yen</th>
        <td>119.01</td>
        <td>119.05</td>
        <td>119.82</td>
        <td>119.82</td>
    </tr>
    <tr class="par">
        <th scope="row" class="fila libra">Libra/Dolar</th>
        <td>1.8606</td>
        <td>1.8611</td>
        <td>1.8651</td>
        <td>1.8522</td>
    </tr>
    <tr>
        <th scope="row" class="fila yen">Yen/Euro</th>
        <td>0.6711</td>
        <td>0.6705</td>
        <td>0.6676</td>
        <td>0.6713</td>
    </tr>
</table>

</body>
</html>

```

16.12. Solución ejercicio 12

```

<!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>Ejercicio 12 - Formulario de alta</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />

<style type="text/css">
/* Formatear el formulario a dos columnas */

```

```
body {  
    font: 13px/1.6 Tahoma, sans-serif;  
    background: #F5F5F5;  
}  
  
.izquierda {  
    float: left;  
    clear: left;  
}  
  
.derecha {  
    float: right;  
    clear: right;  
}  
  
ul {  
    list-style: none;  
    margin: 0;  
    padding: 0;  
}  
  
#contenedor {  
    background: #FFF;  
    border: 1px solid silver;  
    margin: 1em auto;  
    padding: 1em;  
    width: 768px;  
}  
  
span.requerido {  
    font-size: 1.3em;  
    font-weight: bold;  
    color: #C00;  
}  
  
h2 {  
    font: normal 2em arial, sans-serif;  
    margin: 0;  
}  
  
ul li.botones {  
    border-top: 2px solid #CCC;  
    clear: both;  
    float: none;  
    padding: 1em 0;  
    margin-top: 1em;  
    text-align: right;  
    width: 100%;  
}  
  
ul li.botones input {  
    font-size: 1.3em;  
}  
  
ul li {
```

```
margin: 0.5em 0;
padding: 0.5em;
width: 46%;
}

ul li label.titulo {
  font-weight: bold;
}

ul li div span {
  float: left;
  padding: 0.3em 0;
}

ul li div span.completo {
  width: 100%;
}

ul li div span.mitad {
  width: 50%;
}

ul li div span.tercio {
  width: 33%;
}

ul li div span.dostercios {
  width: 66%;
}

ul li div span label {
  display: block;
  font: normal 0.8em arial, sans-serif;
  color: #333;
}

ul li p.ayuda {
  display: none;
}

ul li input {
  padding: 0.2em;
}

input#apellido1, input#apellido2, input#direccion, input#email {
  width: 260px;
}

input#codigopostal {
  width: 80px;
}

select#provincia {
  width: 90px;
}
```

```
select#pais {
    width: 150px;
}

input#telefonofijo, input#telefonomovil {
    width: 135px;
}

/* Cambiar el color en el :hover y resaltar los campos en el :focus */
ul li:hover {
    background-color: #FF9;
}

ul li.botones:hover {
    background-color: transparent;
}

ul li input:focus {
    border: 2px solid #E6B700;
}

/* Formatear el formulario a una columna */
ul li.izquierda, ul li.derecha {
    float: none;
    width: auto;
}

ul li {
    overflow: hidden;
}

ul li label.titulo {
    float: left;
    width: 150px;
}

ul li div {
    margin-left: 160px;
}

/* Aspecto final del formulario con Los mensajes de ayuda */
h2 {
    margin-bottom: 0.3em;
}

ul li {
    border-top: 1px solid #CCC;
    margin: 0;
    padding: 1em;
}

ul li.botones {
    margin: 0;
}
```

```
ul li label.titulo {  
    text-align: right;  
    width: 100px;  
}  
  
ul li div {  
    margin-left: 110px;  
    overflow: hidden;  
}  
  
ul li {  
    position: relative;  
}  
  
ul li:hover p.ayuda {  
    display: block;  
    margin: 0.3em;  
    position: absolute;  
    top: 0;  
    right: 0;  
    width: 150px;  
}  
</style>  
</head>  
  
<body>  
<div id="contenedor">  
  
<h2>Formulario de alta</h2>  
  
<form method="post" action="#">  
<ul>  
<li class="izquierda">  
    <label class="titulo" for="nombre">Nombre y apellidos <span  
    class="requerido">*</span></label>  
    <div>  
        <span class="completo">  
            <input id="nombre" name="nombre" value="" />  
            <label for="nombre">Nombre</label>  
        </span>  
  
        <span class="completo">  
            <input id="apellido1" name="apellido1" value="" />  
            <label for="apellido1">Primer apellido</label>  
        </span>  
  
        <span class="completo">  
            <input id="apellido2" name="apellido2" value="" />  
            <label for="apellido2">Segundo apellido</label>  
        </span>  
    </div>  
  
    <p class="ayuda">No te olvides de escribir también tu segundo apellido</p>  
</li>
```

```
<li class="derecha">
    <label class="titulo" for="direccion">Dirección <span
    class="requerido">*</span></label>

    <div>
        <span class="completo">
            <input id="direccion" name="direccion" value="" />
            <label for="direccion">Calle, número, piso, puerta</label>
        </span>

        <span class="tercio">
            <input id="codigopostal" name="codigopostal" value="" />
            <label for="codigopostal">Código postal</label>
        </span>

        <span class="dostercios">
            <input id="municipio" name="municipio" value="" />
            <label for="municipio">Municipio</label>
        </span>

        <span class="tercio">
            <select id="provincia" name="provincia">
                <option value=""></option>
                <option value="provincial">Provincia 1</option>
                <option value="provincia2">Provincia 2</option>
                <option value="provincia3">Provincia 3</option>
            </select>
            <label for="provincia">Provincia</label>
        </span>

        <span class="dostercios">
            <select id="pais" name="pais">
                <option value=""></option>
                <option value="pais1">País 1</option>
                <option value="pais2">País 2</option>
                <option value="pais3">País 3</option>
            </select>
            <label for="pais">País</label>
        </span>
    </div>

    <p class="ayuda">El código postal es imprescindible para poder recibir los pedidos</p>
</li>

<li class="izquierda">
    <label class="titulo" for="email">Email</label>

    <div>
        <span class="completo">
            <input id="email" name="email" value="" />
        </span>
    </div>

    <p class="ayuda">Asegúrate de que sea válido</p>
```

```

</li>

<li class="derecha">
  <label class="titulo" for="telefonofijo">Teléfono <span
  class="requerido">*</span></label>

  <div>
    <span class="mitad">
      <input id="telefonofijo" name="telefonofijo" value="" />
      <label for="telefonofijo">Fijo</label>
    </span>

    <span class="mitad">
      <input id="telefonomovil" name="telefonomovil" value="" />
      <label for="telefonomovil">Móvil</label>
    </span>
  </div>

  <p class="ayuda">Sin prefijo de país y sin espacios en blanco</p>
</li>

<li class="botones">
  <input id="alta" type="submit" value="Darle de alta &rarr;" />
</li>

</ul>
</form>

</div>
</body>
</html>

```

16.13. Solución ejercicio 13

```

/* ----- Estilos básicos ----- */

body {
  margin: 0;
  padding: 0;
  background: #F2F5FE url("../imagenes/fondo.gif") 0 0 repeat-x;
  font: 70%/160% "verdana", sans-serif;
  color: #192666;
}

a {
  color:#192666;
}
a:hover {
  color:#4F6AD7;
}

p {
  margin: 1em 0;
  padding: 0;
}

```

```
.clear {  
    clear: both;  
}  
  
h1, h2, h3, h4, h5 {  
    margin: 1em 0;  
    padding: 0;  
}  
  
h1 {  
    font-size: 260%;  
    font-family: "georgia", serif;  
    font-weight: normal;  
}  
  
h2 {  
    font-size: 180%;  
    font-family: "georgia", serif;  
    font-weight: normal;  
}  
  
h3 {  
    font-size: 120%;  
    font-weight: bold;  
}  
  
ul, ol {  
    margin: 1em 0 1em 2em;  
    padding: 0;  
}  
  
/* ----- Layout ----- */  
  
#contenedor {  
    width: 770px;  
    margin: 50px auto 0 auto;  
}  
  
#cabecera {  
    width: 770px;  
    position: relative;  
    height: 100px;  
    margin: 0;  
    padding: 0;  
    background: #233C9B url("../imagenes/cabecera.jpg") no-repeat;  
    color: #FFF;  
}  
  
#contenido {  
    width: 760px;  
    margin: 0 5px;  
    background: #FFF;  
}
```

```
#contenido #principal {  
    float: left;  
    width: 530px;  
    margin-top: 15px;  
    padding: 0 0 0 20px;  
    background: #FFF;  
}  
  
#contenido #secundario {  
    float: left;  
    width: 200px;  
    margin: 15px 0 0 0;  
    padding: 0;  
    background: #CEDBF9 url("../imagenes/fondo_columna.gif") no-repeat;  
}  
  
#pie {  
    clear: both;  
    height: 60px;  
    margin-bottom: 50px;  
    background: url("../imagenes/fondo_pie.jpg") no-repeat;  
    color: #6685CC;  
    position: relative;  
}  
  
/* ----- Cabecera ----- */  
#cabecera #logo {  
    position: absolute;  
    top: 35px;  
    left: 35px;  
    margin: 0;  
}  
  
#cabecera #logo a {  
    color: #FFF;  
    display: block;  
    line-height: 35px;  
}  
  
#cabecera #logo a:hover {  
    color: #B5C4E3;  
    text-decoration: underline;  
}  
  
#cabecera #buscador {  
    position: absolute;  
    top: 40px;  
    right: 20px;  
}  
  
#cabecera #buscador legend {  
    display: none;  
}  
  
#cabecera #buscador fieldset {
```

```
border: none;
}

/* ----- Menú ----- */
#menu {
    background: #192666;
    margin: 0 5px;
    padding: 10px 0 0 0;
}

#menu ul {
    margin: 0 10px;
    padding: 0;
    list-style:none;
}

#menu ul li {
    margin: 0 5px 0 0;
    padding: 0;
    float:left;
}

#menu ul li a {
    display: block;
    position: relative;
    padding: 5px 15px;
    border: 0;
    background: #253575;
    color: #B5C4E3;
    font-weight: bold;
    text-decoration: none;
    cursor: pointer;
}

#menu ul li a:hover {
    background: #31479B;
}

#menu ul li.seleccionado a {
    background: #FFF;
    color: #FF9000;
}

/* ----- Contenidos ----- */
#contenido .articulo {
    clear: both;
    margin: 0;
    padding: 20px;
    background: url("../imagenes/fondo_articulo.jpg") no-repeat;
}

#contenido .articulo h2 {
    margin: 0 -20px;
    padding: 10px;
    background: #DEE5FD;
```

```
color: #192666;
}

#contenido .info {
    margin: 10px 0;
    padding-bottom: 8px;
    border-bottom: 1px solid #DEE5FD;
    color:#6685CC;
}

#contenido .info a { color:#6685CC; }
#contenido .info a:hover { color:#FF9000; }

#contenido .info span.fecha, #contenido .info span.categoría, #contenido .info
span.autor, #contenido .info span.comentarios {
    padding-left:15px;
}

#contenido .info span.fecha { background: url("../imagenes/icono_fecha.gif") 0 50%
no-repeat; }
#contenido .info span.categoría { background: url("../imagenes/icono_categoria.gif") 0
50% no-repeat; margin-left: 8px; }
#contenido .info span.autor { background: url("../imagenes/icono_autor.gif") 0 50%
no-repeat; margin-left: 8px; }
#contenido .info span.comentarios { background: url("../imagenes/
icono_comentarios.gif") 0 50% no-repeat; margin-left: 8px; }

#contenido #secundario h3 {
    padding: 10px 0 10px 10px;
    margin-top: 20px;
    background: #A0B9F3;
    color: #192666;
}

#contenido #secundario #sobre mí img {
    float: left;
    margin: 4px 5px 0 0;
}

#contenido #secundario #sobre mí p {
}

#contenido #secundario div {
    padding: 0 10px;
}

#contenido #secundario ul {
    margin: 15px 0;
    padding: 0;
    list-style:none;
}

#contenido #secundario li {
    margin: 0;
```

```
padding: 0;
border-bottom: 1px solid #E0E8FA;
}

#contenido #secundario li a {
    display: block;
    padding: 3px 0 3px 22px;
    background: url("../imagenes/icono_elemento.gif") 8px no-repeat;
    text-decoration: none;
}

#contenido #secundario li a:hover {
    background-color: #E0E8FA;
    color: #192666;
}

#contenido #secundario li.seleccionado a {
    background: #E0E8FA url("../imagenes/icono_elemento_seleccionado.gif") 8px no-repeat;
    font-weight: bold;
}

/* ----- Pie ----- */
#pie p {
    margin: 0;
    padding: 0;
    position: absolute;
    top: 10px;
    left: 40px;
}
```