



Arkaitz Garro

css2.1

Esta página se ha dejado vacía a propósito

CSS 2.1

Publication date: 14/06/2013

This book was published with *easybook v5.0-DEV*, a free and open-source book publishing application developed by Javier Eguiluz (<http://javiereguiluz.com>) using several Symfony components (<http://components.symfony.com>) .

Esta página se ha dejado vacía a propósito

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

Esta obra está basada en el trabajo previo de Javier Eguiluz, Introducción a CSS y CSS Avanzado, publicadas en la siguientes direcciones, respectivamente: <http://www.librosweb.es/css/> y http://www.librosweb.es/css_avanzado/. Puedes copiar, distribuir y comunicar públicamente la obra, incluso transformándola, siempre que cumplas todas las condiciones siguientes:

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

Esta página se ha dejado vacía a propósito

Capítulo 1 Introducción.....	11
1.1 ¿Qué es CSS?.....	11
1.2 Especificación oficial	11
1.3 Funcionamiento básico de CSS	12
1.4 Cómo incluir CSS en un documento XHTML.....	14
1.5 Glosario básico.....	18
1.6 Medios CSS	19
1.7 Comentarios	22
1.8 Sintaxis de la definición de cada propiedad CSS.....	22
Capítulo 2 Selectores	25
2.1 Selectores básicos	25
2.2 Selectores avanzados.....	36
2.3 Agrupación de reglas	39
2.4 Herencia	40
2.5 Colisiones de estilos	42
Capítulo 3 Unidades de medida y colores.....	45
3.1 Unidades de medida.....	45
3.2 Unidades absolutas	46
3.3 Unidades relativas.....	47
3.4 Colores	51
Capítulo 4 Modelo de cajas	55
4.1 Anchura y altura	57
4.2 Margen y relleno	59
4.3 Bordes.....	66
4.4 Margen, relleno, bordes y modelo de cajas	73
4.5 Fondos	74
Capítulo 5 Posicionamiento y visualización	77
5.1 Tipos de elementos.....	78
5.2 Posicionamiento	79
5.3 Posicionamiento normal	82

5.4 Posicionamiento relativo	83
5.5 Posicionamiento absoluto	85
5.6 Posicionamiento fijo	90
5.7 Posicionamiento flotante	90
5.8 Visualización	98
5.9 Propiedades display y visibility	98
Capítulo 6 Texto	105
6.1 Tipografía	105
6.2 Texto	115
Capítulo 7 Enlaces	131
7.1 Estilos básicos	131
7.2 Estilos avanzados	135
Capítulo 8 Listas	139
8.1 Estilos básicos	139
8.2 Estilos avanzados	148
Capítulo 9 Formularios	153
9.1 Estilos básicos	153
9.2 Estilos avanzados	158
Capítulo 10 Tablas	161
10.1 Estilos básicos	161
10.2 Estilos avanzados	165
Capítulo 11 Layout básico	169
11.1 Centrar una página horizontalmente	170
11.2 Centrar una página verticalmente	173
11.3 Estructura o layout	177
11.4 Alturas/anchuras máximas y mínimas	182
11.5 Estilos avanzados	184
Capítulo 12 Otros	187
12.1 Versión para imprimir	187
12.2 Hacks y filtros	191
12.3 Prioridad de las declaraciones CSS	193
Capítulo 13 CSS Avanzado	197

13.1 Técnicas imprescindibles	197
13.2 Selectores	211
13.3 Pseudo-elementos	220
13.4 Propiedades avanzadas	222
Capítulo 14 Ejercicios de CSS	231
14.1 Capítulo 2	231
14.2 Capítulo 2	233
14.3 Capítulo 4	235
14.4 Capítulo 5	241
14.5 Capítulo 6	242
14.6 Capítulo 7	244
14.7 Capítulo 8	244
14.8 Capítulo 9	246
14.9 Capítulo 10	249
14.10 Ejercicio final	252

Esta página se ha dejado vacía a propósito

Capítulo 1

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.

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 7 de junio de 2011 y que se puede consultar libremente en [w3.org/TR/CSS21/](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 multitud de cambios importantes en todos los niveles y es mucho más avanzada y compleja que CSS 2. Puedes consultar el estado actual de cada componente de CSS 3 en [w3.org/Style/CSS/current-work](http://www.w3.org/Style/CSS/current-work) (<http://www.w3.org/Style/CSS/current-work>) . También existe un blog oficial (<http://www.w3.org/blog/CSS>) en el que se publican todas las novedades relacionadas con el estándar CSS.

Mientras CSS 3 se convierte en un estandar, los distintos navegadores pueden incluir o no las mejoras propuestas, o incluso añadir las suyas propias. La web [Can I use...](http://caniuse.com/) (<http://caniuse.com/>) incluye un listado completo de funcionalidades soportadas por los ditintos navegadores.

Antes de que se generalizara el uso de CSS, los diseñadores de páginas web utilizaban etiquetas HTML especiales para modificar el aspecto de los elementos 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=UTF-8" />
  <title>Ejemplo de estilos sin CSS</title>
</head>
<body>
  <h1><font color="red" face="Arial">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, el tipo y el tamaño de letra de cada elemento de la página.

El problema de utilizar este método para 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 tres atributos, habría que definir 1.5 millones de atributos.

Como el diseño de los sitios web está en constante evolución, es habitual modificar cada cierto tiempo el aspecto de las páginas del sitio. Siguiendo con el ejemplo anterior, cambiar el aspecto del sitio requeriría modificar 500.000 etiquetas y 1.5 millones de atributos.

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="charset=UTF-8" />
  <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 la información sobre su aspecto. En el ejemplo anterior, dentro de la propia página HTML se crea una zona especial en la que se incluye toda la información relacionada con los estilos de la página.

Utilizando CSS, se pueden establecer los mismos estilos con menos esfuerzo y sin ensuciar el código HTML de los contenidos con etiquetas . Como se verá más adelante, la etiqueta <style> crea una zona especial donde se incluyen todas las reglas CSS que se aplican en la página.

En el ejemplo anterior, dentro de la zona de CSS 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.

Definir los estilos de esta forma ahorra miles de etiquetas y millones de atributos respecto a la solución anterior, pero sigue sin ser una solución ideal. Como los estilos CSS sólo se aplican en la página que los incluye, si queremos que las 10.000 páginas diferentes del sitio tengan el mismo aspecto, se deberían copiar 10.000 veces esas mismas reglas CSS. Más adelante se explica la solución que propone CSS para evitar este problema.

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.

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=UTF-8" />
  <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.

Los ejemplos mostrados en este libro utilizan este método para aplicar CSS al contenido HTML de las páginas. De esta forma el código de los ejemplos es más conciso y se aprovecha mejor el espacio.

En este caso, todos los estilos CSS se incluyen en un archivo de tipo CSS que las páginas HTML enlazan mediante la etiqueta . 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.

Si se quieren incluir los estilos del ejemplo anterior en un archivo CSS externo, se deben seguir los siguientes pasos:

1) Se crea un archivo de texto y se le añade solamente el siguiente contenido:

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

2) Se guarda el archivo de texto con el nombre estilos.css Se debe poner especial atención a que el archivo tenga extensión .css y no .txt

3) En la página HTML se enlaza el archivo CSS externo mediante la etiqueta <link>:

```
<!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>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>
```

Cuando el navegador carga la página HTML anterior, antes de mostrar sus contenidos también descarga los archivos CSS externos enlazados mediante la etiqueta `<link>` y aplica los estilos a los contenidos de la página.

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

- `rel`: indica el tipo de relación que existe entre 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=UTF-8" />
```

```
<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>
</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");
```

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=UTF-8" />
    <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.

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:

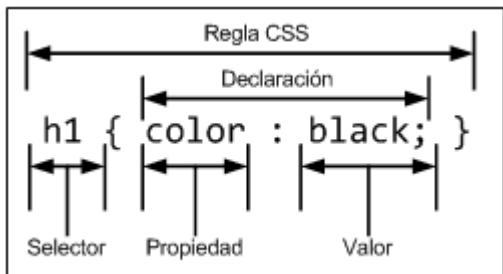


Figura 1.1 Componentes de un estilo CSS básico

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 "*declaración*" 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: característica que se modifica en el elemento seleccionado, como por ejemplo su tamaño de letra, su color de fondo, etc.
- Valor: establece el nuevo valor de la característica modificada en el elemento. Un archivo CSS puede contener un número ilimitado de reglas CSS, cada regla se puede aplicar a varios selectores diferentes y cada declaración puede incluir tantos pares propiedad/valor como se desee.

El estándar CSS 2.1 define 115 propiedades, cada una con su propia lista de valores permitidos. Por su parte, los últimos borradores del estándar CSS 3 ya incluyen 239 propiedades.

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:

all	Todos los medios definidos
braille	Dispositivos táctiles que emplean el sistema braille
embosed	Impresoras braille
handheld	Dispositivos de mano: móviles, PDA, etc.
print	Impresoras y navegadores en el modo "Vista Previa para Imprimir"
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).

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.

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.

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 sobre-

tiende que el medio es `all`, es decir, que los estilos se aplican en todos los medios.

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.

Si la etiqueta `<link>` no indica el medio CSS, se sobreentiende que los estilos se deben aplicar a todos los medios, por lo que es equivalente a indicar `media="all"`.

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`.

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 -->
```

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 (carácter |) 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 (símbolo ||) 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.

Esta página se ha dejado vacía a propósito

Capítulo 2

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 aplicarlo*". 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 aplicar varias reglas CSS y cada regla CSS puede aplicarse a un número ilimitado 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.

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.

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; }
```

Selecciona los elementos que se encuentran 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 descendiente directo del otro. La única condición es que un elemento debe estar dentro de otro elemento, sin importar el nivel de profundidad en el 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 aumentar 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:

```
| selector1 selector2 selector3 ... selectorN
```

Los selectores descendentes siempre están formados por dos o más selectores separados entre sí por espacios en blanco. El último selector indica el elemento sobre el que se aplican los estilos y todos los selectores anteriores indican el lugar en el que se debe encontrar ese elemento.

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

```
| 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; }
```

Se puede restringir el alcance del selector descendente combinándolo con el selector universal. 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 interpreta 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`.

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 utilizar 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 adipisci</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 pixel:

```
.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*".

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 { ... }
```

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 (#ej01)

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 quieres 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/>)

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.

El selector adyacente se emplea para seleccionar elementos que en el código HTML de la página se encuentran justo a continuación de otros elementos. Su sintaxis emplea el signo `+` para separar los dos elementos:

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

Si se considera el siguiente código HTML:

```
<body>  
  <h1>Tituto1</h1>  
  
  <h2>Subtítulo</h2>  
  ...  
  
  <h2>Otro subtítulo</h2>  
  ...  
</body>
```

La página anterior dispone de dos elementos `<h2>`, pero sólo uno de ellos se encuentra inmediatamente después del elemento `<h1>`. Si se quiere aplicar diferentes colores en función de esta circunstancia, el selector adyacente es el más adecuado:

```
h2 { color: green; }  
h1 + h2 { color: red }
```

Las reglas CSS anteriores hacen que todos los `<h2>` de la página se vean de color verde, salvo aquellos `<h2>` que se encuentran inmediatamente después de cualquier elemento `<h1>` y que se muestran de color rojo.

Técnicamente, los elementos que forman el selector adyacente deben cumplir las dos siguientes condiciones:

- elemento1 y elemento2 deben ser *elementos 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.

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 de la página que estén precedidos por otro párrafo, por lo que no se aplica al primer párrafo de la página.

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 }
```

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;}
```

Una de las características principales de CSS es la herencia de los estilos definidos para los elementos. Cuando se establece el valor de una propiedad CSS en un elemento, sus elementos descendientes heredan de forma automática el valor de esa propiedad. Si se considera 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=UTF-8" />  
    <title>Ejemplo de herencia de estilos</title>  
    <style type="text/css">  
      body { color: blue; }  
    </style>  
  </head>  
  
<body>  
  <h1>Titular de la página</h1>  
  <p>Un párrafo de texto no muy largo.</p>
```

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

En el ejemplo anterior, el selector `body` solamente establece el color de la letra para el elemento `<body>`. No obstante, la propiedad `color` es una de las que se heredan de forma automática, por lo que todos los elementos descendientes de `<body>` muestran ese mismo color de letra. Por tanto, establecer el color de la letra en el elemento `<body>` de la página implica cambiar el color de letra de todos los elementos de la página.

Aunque la herencia de estilos se aplica automáticamente, se puede anular su efecto estableciendo de forma explícita otro valor para la propiedad que se hereda, como se muestra 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=UTF-8" />
  <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 establece en primer lugar el color y tipo de letra del elemento `<body>`, por lo que todos los elementos de la página se mostrarían con ese mismo color y tipo de letra. No obstante, las otras reglas CSS modifican alguno de los estilos heredados.

De esta forma, los elementos `<h1>` de la página se muestran con el tipo de letra Verdana establecido por el selector `h1` y se muestran de color negro que es el valor heredado del elemento `<body>`. Igualmente, los elementos `<p>` de la página se muestran del color rojo establecido por el selector `p` y con un tipo de letra Arial heredado del elemento `<body>`.

La mayoría de propiedades CSS aplican la herencia de estilos de forma automática. Además, para aquellas propiedades que no se heredan automáticamente, CSS incluye un mecanismo para forzar a que se hereden sus valores, tal y como se verá más adelante.

Por último, aunque la herencia automática de estilos puede parecer complicada, simplifica en gran medida la creación de hojas de estilos complejas. Como se ha visto en los ejemplos anteriores, si se quiere establecer por ejemplo la tipografía base de la página, simplemente se debe establecer en el elemento `<body>` de la página y el resto de elementos la heredarán de forma automática.

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.

El método seguido por CSS para resolver las colisiones de estilos se muestra a continuación:

- Determinar todas las declaraciones que se aplican al elemento para el medio CSS seleccionado.
- Ordenar las declaraciones según su origen (CSS de navegador, de usuario o de diseñador) y su prioridad (palabra clave `!important`).
- Ordenar las declaraciones según lo específico que sea el selector. Cuanto más genérico es un selector, menos importancia tienen sus declaraciones.
- 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.

Hasta que no se expliquen más adelante los conceptos de tipo de hoja de estilo y la prioridad, el mecanismo simplificado que se puede aplicar es el siguiente:

- Cuanto más específico sea un selector, más importancia tiene su regla asociada.
- 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 \(#ej02\)](#)

Esta página se ha dejado vacía a propósito

Capítulo 3

Muchas de las propiedades de CSS que se ven en los próximos capítulos permiten indicar medidas 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, cuando una propiedad pueda tomar como valor una medida o un color, no se volverán a explicar todas estas alternativas.

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 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 uno de los errores más habituales de los diseñadores que empiezan con CSS. Algunas propiedades permiten indicar medidas negativas, aunque habitualmente sus valores son positivos. Si el valor decimal de una medida es inferior a `1`, se puede omitir el `0` de la izquierda (`0.5em` es equivalente a `.5em`).

Una medida indicada mediante unidades absolutas está completamente definida, ya que su valor no depende de otro valor de referencia. A continuación se muestra la lista completa de unidades absolutas definidas por CSS y su significado:

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

A continuación se muestran ejemplos de utilización de unidades absolutas:

```
/* El cuerpo de la página debe mostrar un margen de media pulgada */
body { margin: 0.5in; }

/* Los elementos <h1> deben mostrar un interlineado de 2 centímetros */
h1 { line-height: 2cm; }

/* Las palabras de todos los párrafos deben estar separadas 4 milímetros
entre si */
p { word-spacing: 4mm; }

/* Los enlaces se deben mostrar con un tamaño de letra de 12 puntos */
a { font-size: 12pt }

/* Los elementos <span> deben tener un tamaño de letra de 1 pica */
span { font-size: 1pc }
```

La principal ventaja de las unidades absolutas es que su valor es directamente el valor que se debe utilizar, sin necesidad de realizar cálculos intermedios. Su principal desventaja es que son muy poco flexibles y no se adaptan fácilmente a los diferentes medios.

De todas las unidades absolutas, la única que suele utilizarse es el punto (pt). Se trata de la unidad de medida preferida para establecer el tamaño del texto en los documentos que se van a imprimir, es decir, para el medio print de CSS, tal y como se verá más adelante.

Las unidades relativas, a diferencia de las absolutas, no están completamente definidas, ya que su valor siempre está referenciado respecto a otro valor. A pesar de su aparente dificultad, son las más utilizadas en el diseño web por la flexibilidad con la que se adaptan a los diferentes medios.

A continuación se muestran las tres unidades de medida relativas definidas por CSS y la referencia que toma cada una para determinar su valor real:

- em, (no confundir con la etiqueta de HTML) relativa respecto del tamaño de letra del elemento.
- ex, relativa respecto de la altura de la letra x ("*equis minúscula*") del tipo y tamaño de letra del elemento.
- px, (píxel) relativa respecto de la resolución de la pantalla del dispositivo en el que se visualiza la página HTML.

Las unidades em y ex no han sido creadas por CSS, sino que llevan décadas utilizándose en el campo de la tipografía. Aunque no es una definición exacta, la unidad 1em equivale a la anchura de la letra M ("eme mayúscula") del tipo y tamaño de letra del elemento.

La unidad em hace referencia al tamaño en puntos de la letra que se está utilizando. Si se utiliza una tipografía de 12 puntos, 1em equivale a 12 puntos. El valor de 1ex se puede aproximar por 0.5 em.

Si se considera el siguiente ejemplo:

```
| p { margin: 1em; }
```

La regla CSS anterior indica que los párrafos deben mostrar un margen de anchura igual a 1em. Como se trata de una unidad de medida relativa, es necesario realizar un cálculo matemático para determinar la anchura real de ese margen.

La unidad de medida em siempre hace referencia al tamaño de letra del elemento. Por otra parte, todos los navegadores muestran por defecto el texto de los párrafos con un tamaño de letra de 16 píxel. Por tanto, en este caso el margen de 1em equivale a un margen de anchura 16px.

A continuación se modifica el ejemplo anterior para cambiar el tamaño de letra de los párrafos:

```
| p { font-size: 32px; margin: 1em; }
```

El valor del margen sigue siendo el mismo en unidades relativas (1em) pero su valor real ha variado porque el tamaño de letra de los párrafos ha variado. En este caso, el margen tendrá una anchura de 32px, ya que 1em siempre equivale al tamaño de letra del elemento.

Si se quiere reducir la anchura del margen a 16px pero manteniendo el tamaño de letra de los párrafos en 32px, se debe utilizar la siguiente regla CSS:

```
| p { font-size: 32px; margin: 0.5em; }
```

El valor 0.5em se interpreta como "*la mitad del tamaño de letra del elemento*", ya que se debe multiplicar por 0.5 su tamaño de letra (32px × 0.5 = 16px). De la misma forma, si se quiere mostrar un margen de 8px de anchura, se debería utilizar el valor 0.25em, ya que 32px × 0.25 = 8px.

La gran ventaja de las unidades relativas es que siempre mantienen las proporciones del diseño de la página. Establecer el margen de un elemento con el valor 1em equivale a indicar que "*el margen del elemento debe ser del mismo tamaño que su letra y debe cambiar proporcionalmente*".

En efecto, si el tamaño de letra de un elemento aumenta hasta un valor enorme, su margen de 1em también será enorme. Si su tamaño de letra se reduce hasta un valor diminuto, el margen de 1em también será diminuto. El uso de unidades relativas permite mantener las proporciones del diseño cuando se modifica el tamaño de letra de la página.

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, por lo que su valor es aproximadamente la mitad que el de la unidad `em`.

Por último, 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 la página HTML. Si un elemento tiene una anchura de `400px`, ocupará la mitad de una pantalla con una resolución de `800x600`, pero ocupará menos de la tercera parte en una pantalla con resolución de `1440x900`.

Las unidades de medida se pueden mezclar en 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 `10px` 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, el valor de la mayoría de propiedades CSS se hereda de padres a 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, el valor de las medidas relativas no se hereda directamente, sino que se hereda su valor real una vez calculado. 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 $3em \times 15px = 45px$. 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>`: $3em \times 12px = 36px$. Una vez calculado el valor real, este es el valor que se hereda para el resto de elementos.

El porcentaje también es una unidad de medida relativa, aunque por su importancia CSS la trata de forma separada a `em`, `ex` y `px`. 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 $80\% \times 600px = 480px$.

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 de los elementos de las páginas) y em y porcentajes para el tamaño de letra de los textos.

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.

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 3.1 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 en.wikipedia.org/wiki/Websafe (<http://en.wikipedia.org/wiki/Websafe>) .

En el campo del diseño gráfico, se han definido varios modelos para hacer referencia a los colores. Los dos modelos más conocidos son RGB y CMYK. Simplificando su explicación, el modelo RGB consiste en definir un color indicando la cantidad de color rojo, verde y azul que se debe mezclar para obtener ese color. Técnicamente, el modelo RGB es un modelo de tipo "aditivo", ya que los colores se obtienen sumando sus componentes.

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 y 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.

Las componentes RGB de un color también se pueden indicar mediante un porcentaje. El funcionamiento y la sintaxis de este método es el mismo que el del RGB decimal. La única diferencia es que en este caso el valor de las componentes RGB puede tomar valores entre 0% y 100%. Por tanto, para transformar un valor RGB decimal en un valor RGB porcentual, es preciso realizar una regla de tres considerando que 0 es igual a 0% y 255 es igual a 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%.

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

Para entender el modelo RGB hexadecimal, en primer lugar es preciso introducir un concepto matemático llamado *sistema numérico hexadecimal*. Cuando realizamos operaciones matemáticas, siempre utilizamos 10 símbolos para representar los números (del 0 al 9). Por este motivo, se dice que utilizamos un sistema numérico decimal.

No obstante, el sistema decimal es solamente uno de los muchos sistemas numéricos que se han definido. Entre los sistemas numéricos alternativos más utilizados se encuentra el sistema hexadecimal, que utiliza 16 símbolos para representar sus números.

Como sólo conocemos 10 símbolos numéricos, el sistema hexadecimal utiliza también seis letras (de la A a la F) para representar los números. De esta forma, en el sistema hexadecimal, después del 9 no va el 10, sino la

A. La letra B equivale al número 11, la C al 12, la D al 13, la E al 14 y la F al número 15.

Definir un color en CSS con el método RGB hexadecimal requiere realizar los siguientes pasos:

- Determinar las componentes RGB decimales del color original, por ejemplo: R = 71, G = 98, B = 176
- Transformar el valor decimal de cada componente al sistema numérico hexadecimal. Se trata de una operación exclusivamente matemática, por lo que puedes utilizar una calculadora. En el ejemplo anterior, el valor hexadecimal de cada componente es: R = 47, G = 62, B = B0
- Para obtener el color completo en formato RGB hexadecimal, se concatenan los valores hexadecimales de las componentes RGB en ese orden y se les añade el prefijo #. De esta forma, el color del ejemplo anterior es #4762B0 en formato RGB hexadecimal.

Siguiendo el mismo ejemplo de las secciones anteriores, el color del párrafo se indica de la siguiente forma utilizando el formato RGB hexadecimal:

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

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; }
```

Las letras que forman parte del color en formato RGB hexadecimal se pueden escribir en mayúsculas o minúsculas indistintamente. No obstante, se recomienda escribirlas siempre en mayúsculas o siempre en minúsculas para que la hoja de estilos resultante sea más limpia y homogénea.

Capítulo 4

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 modelo de cajas es el comportamiento de CSS que hace que todos los elementos de las páginas se representen mediante cajas rectangulares.

Las cajas de una página se crean automáticamente. Cada vez que se inserta una etiqueta HTML, se crea una nueva caja rectangular que encierra los contenidos de ese elemento. La siguiente imagen muestra las tres cajas rectangulares que crean las tres etiquetas HTML que incluye la página:

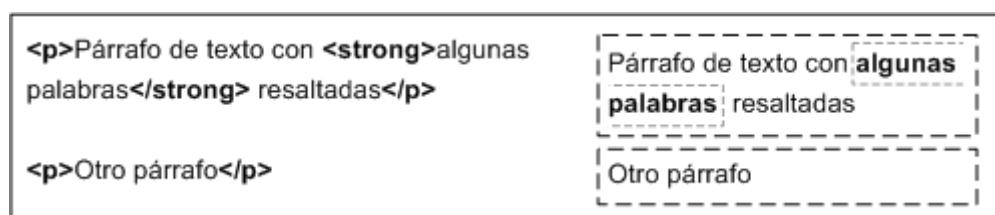


Figura 4.1 Las cajas se crean automáticamente al definir cada elemento HTML

Las cajas de las páginas no son visibles a simple vista porque inicialmente no muestran ningún color de fondo ni ningún borde. 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 su borde:



Figura 4.2 Cajas que forman la página `alistapart.com`

Los navegadores crean y colocan las cajas de forma automática, pero CSS permite modificar todas sus características. Cada una de las cajas está formada por seis partes, tal y como muestra la siguiente imagen:

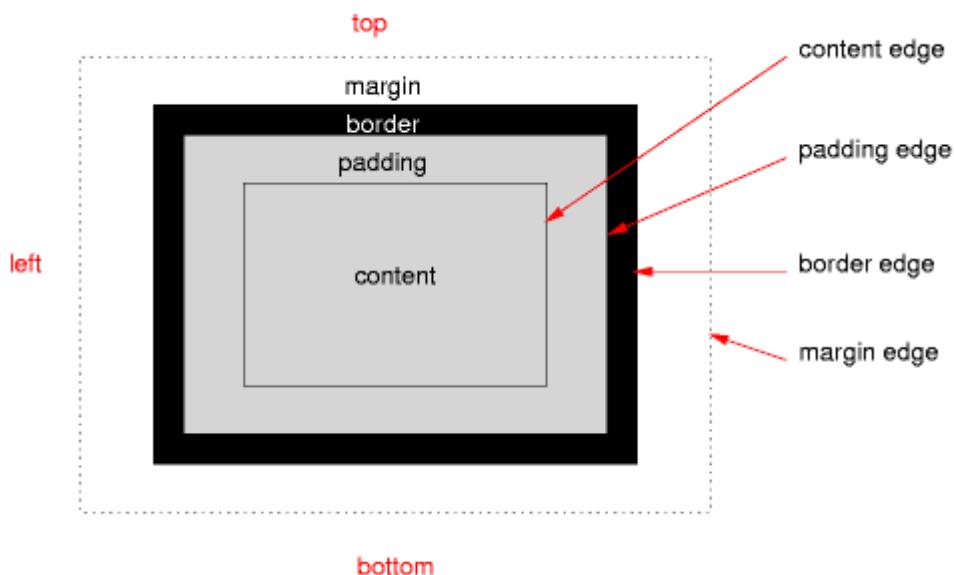


Figura 4.3 Representación bidimensional del box model de CSS

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 existente entre el contenido y el borde.
- 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*): separación opcional existente entre la caja y el resto de 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. No obstante, si la imagen de fondo no cubre totalmente la caja del elemento o si la imagen tiene zonas transparentes, también se visualiza el color de fondo. Combinando imágenes transparentes y colores de fondo se pueden lograr efectos gráficos muy interesantes.

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

width	
Valores	unidad de medida porcentaje auto inherit

	width
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	auto
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.

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

	height
Valores	unidad de medida porcentaje auto inherit
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	auto

height

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.

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

	<code>margin-top, margin-right, margin-bottom, margin-left</code>
Valores	unidad de medida porcentaje <code>auto</code> <code>inherit</code>
Se aplica a	Todos los elementos, salvo <code>margin-top</code> y <code>margin-bottom</code> que sólo se aplican a los elementos de bloque y a las imágenes
Valor inicial	0

margin-top, margin-right, margin-bottom, margin-left

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:

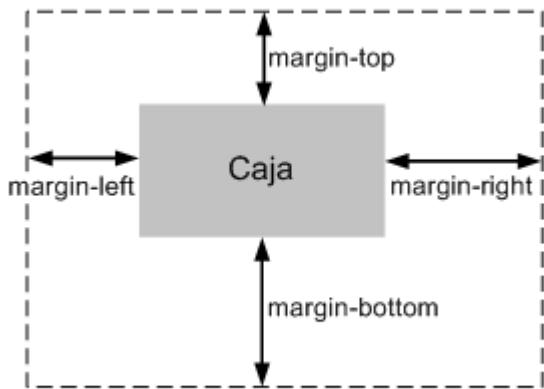


Figura 4.4 Las cuatro propiedades relacionadas con los márgenes

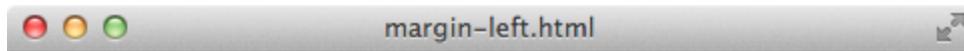
Las unidades más utilizadas para indicar los márgenes de un elemento son los píxeles (cuando se requiere una precisión total), los em (para hacer diseños que mantengan las proporciones) y los porcentajes (para hacer diseños líquidos o fluidos).

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:



Etiam euismod, **temporibus** etiam **adipiscing** elit. Nam **et** **elit.** Vivamus **placerat** lorem. Maecenas **sapien.** Integer **ut** massa. **Cras** diam **ipsum,** laoreet **non,** tincidunt **a,** viverra **sed,** tortor.

Vestibulum lectus diam, luctus vel, venenatis ultrices, cursus vel, tellus. Etiam placerat erat non sem. Nulla molestie odio non nisl tincidunt faucibus.

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

Figura 4.5 Ejemplo de propiedad margin-left

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:

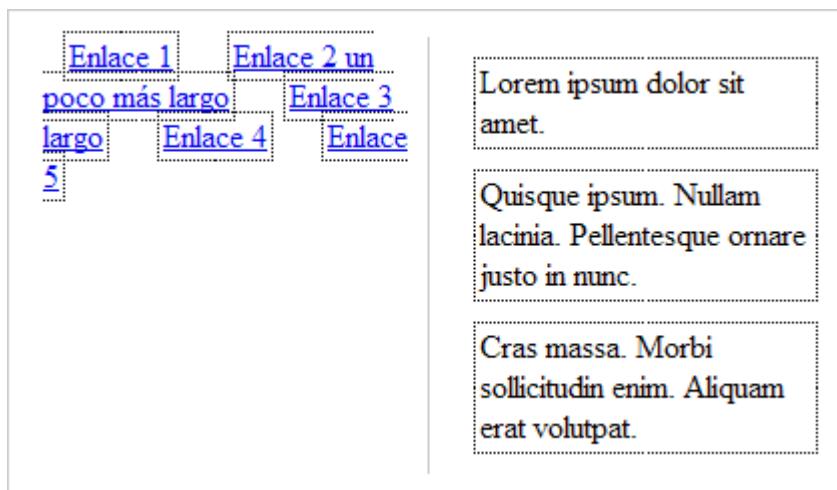


Figura 4.6 Los márgenes verticales sólo se aplican a los elementos de bloque e imágenes

La imagen anterior muestra el resultado de aplicar los mismos márgenes a varios enlaces (elementos en línea) y varios párrafos (elementos de blo-

que). 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.

Además de las cuatro propiedades que controlan cada uno de los márgenes del elemento, CSS define una propiedad especial que permite establecer los cuatro márgenes de forma simultánea. Estas propiedades especiales se denominan "*propiedades shorthand*" y CSS define varias propiedades de este tipo, como se verá más adelante.

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

margin	
Valores	(unidad de 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.

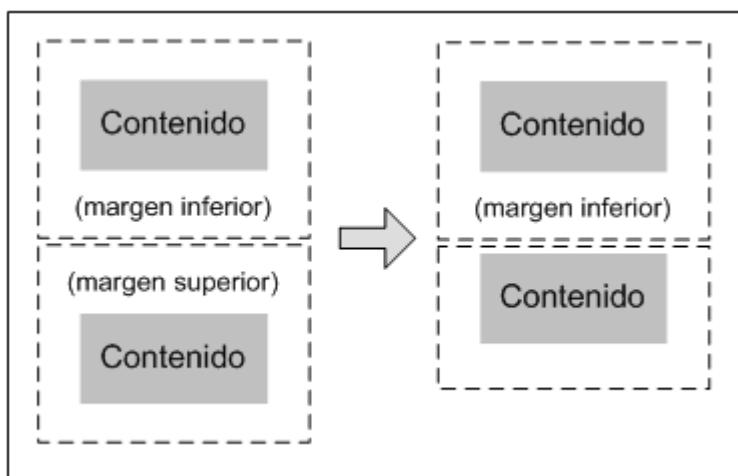


Figura 4.7 Fusión automática de los márgenes verticales

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:

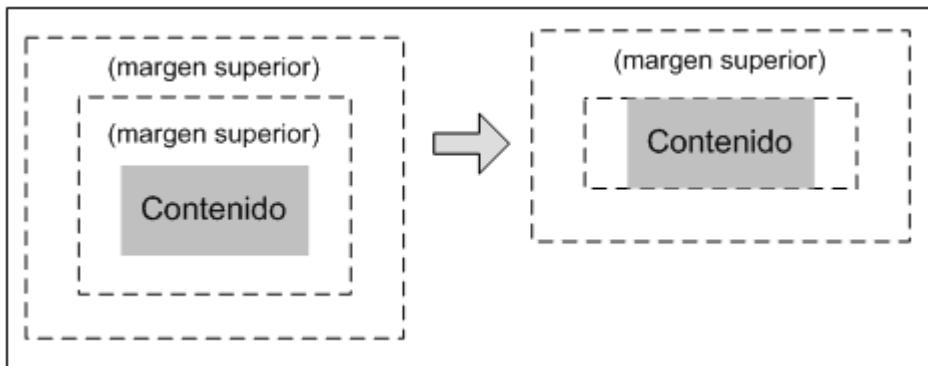


Figura 4.8 Fusión automática de los márgenes interiores

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.

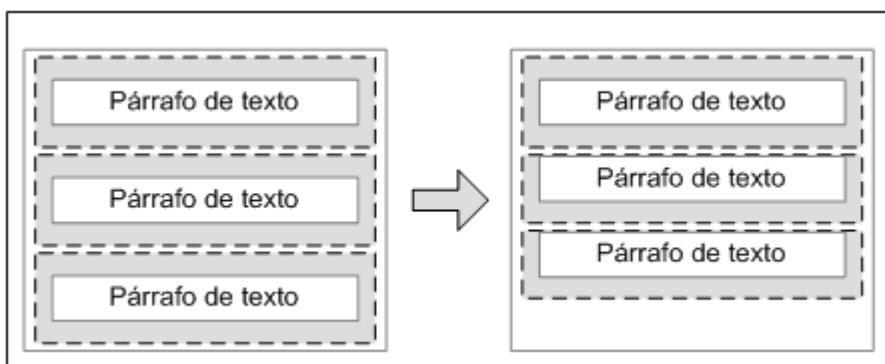


Figura 4.9 Motivo por el que se fusionan automáticamente los márgenes verticales

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.

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

Valores	unidad de medida porcentaje inherit
----------------	---

	padding-top, padding-right, padding-bottom, padding-left
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 estas propiedades establece la separación entre el contenido y los bordes laterales de la caja del elemento:

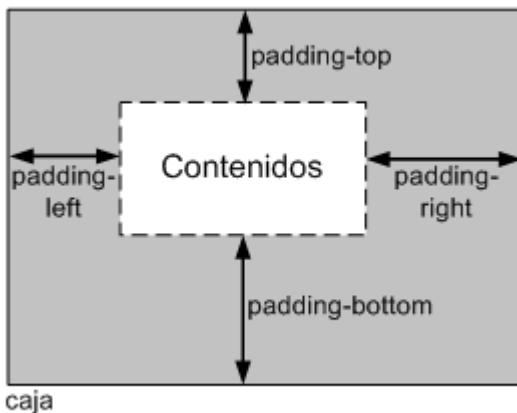


Figura 4.10 Las cuatro propiedades relacionadas con los rellenos

Como sucede con los márgenes, CSS también define una propiedad de tipo "shorthand" llamada padding para establecer los cuatro rellenos de un elemento de forma simultánea.

	padding
Valores	(unidad de 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:

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

Ejercicio 3

[Ver enunciado \(#ej03\)](#)

CSS permite modificar el aspecto de cada uno de los cuatro bordes de la caja de un elemento. Para cada borde se puede establecer su anchura o grosor, su color y su estilo, por lo que en total CSS define 20 propiedades relacionadas con los bordes.

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

	border-top-width, border-right-width, border-bottom-width, border-left-width
Valores	(unidad de 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 indica mediante una medida (en cualquier unidad de medida absoluta o relativa) o mediante las palabras clave thin (borde delgado), medium (borde normal) y thick (borde ancho).

La unidad de medida más habitual para establecer el grosor de los bordes es el píxel, ya que es la que permite un control más preciso sobre el grosor. Las palabras clave apenas se utilizan, ya que el estándar CSS no indica explícitamente el grosor al que equivale cada palabra clave, por lo que pueden producirse diferencias visuales entre navegadores. Así por ejemplo, el grosor `medium` equivale a 4px en algunas versiones de Internet Explorer y a 3px en el resto de navegadores.

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

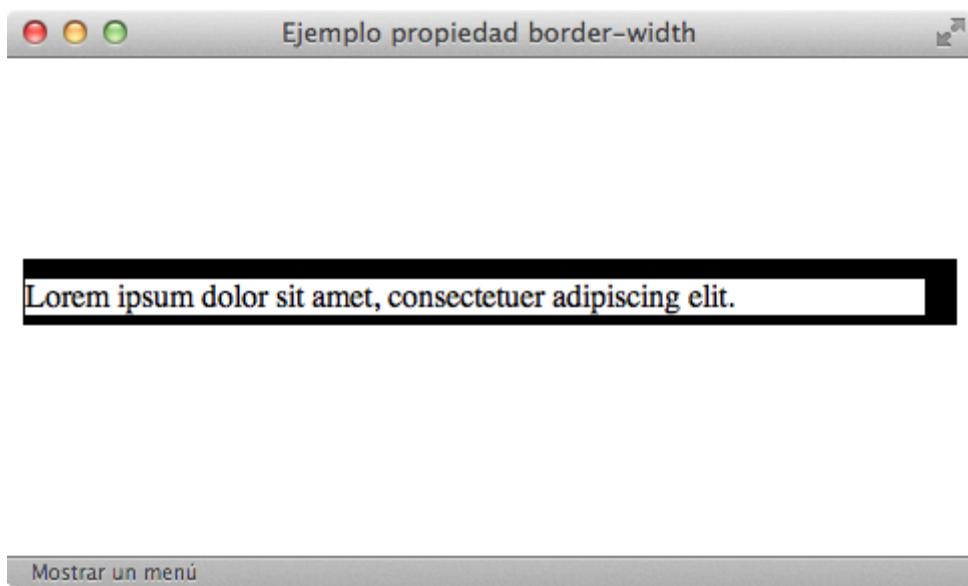


Figura 4.11 Ejemplo de propiedad border-width

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 de forma simultánea la anchura de todos los bordes de una caja, es necesario utilizar una propiedad "shorthand" llamada `border-width`:

	border-width
Valores	(unidad de medida thin medium thick) {1, 4} inherit

	border-width
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.

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

	<code>border-top-color</code> , <code>border-right-color</code> , <code>border-bottom-color</code> , <code>border-left-color</code>
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:

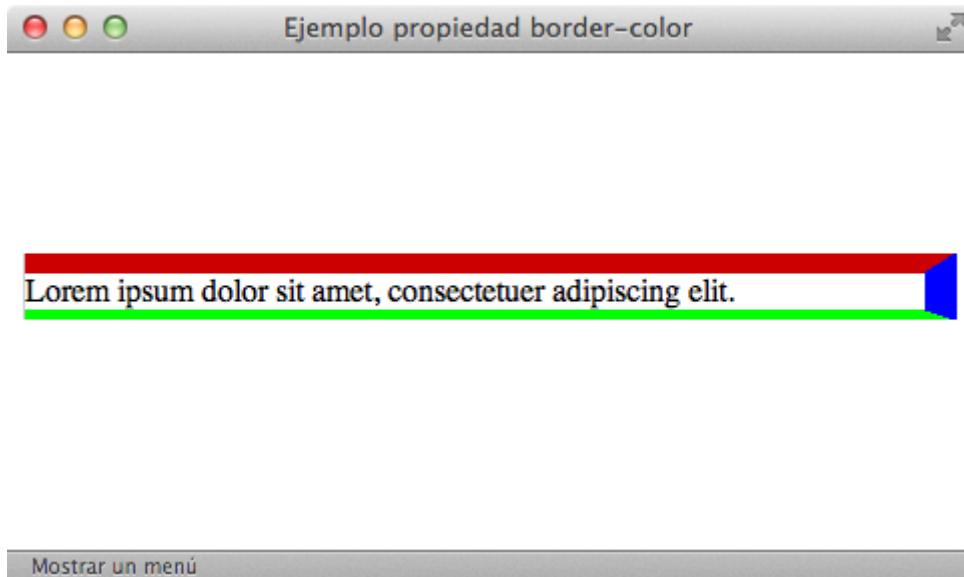


Figura 4.12 Ejemplo de propiedad border-color

CSS incluye una propiedad "shorthand" llamada border-color para establecer de forma simultánea el color de todos los bordes de una caja:

	border-color
Valores	(color transparent) {1, 4} inherit
Se aplica a	Todos los elementos
Valor inicial	medium
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 las de la propiedad border-width.

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
Valores	none hidden dotted dashed solid double groove ridge inset outset inherit
Se aplica a	Todos los elementos
Valor inicial	none

border-top-style, border-right-style, border-bottom-style, border-left-style

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:

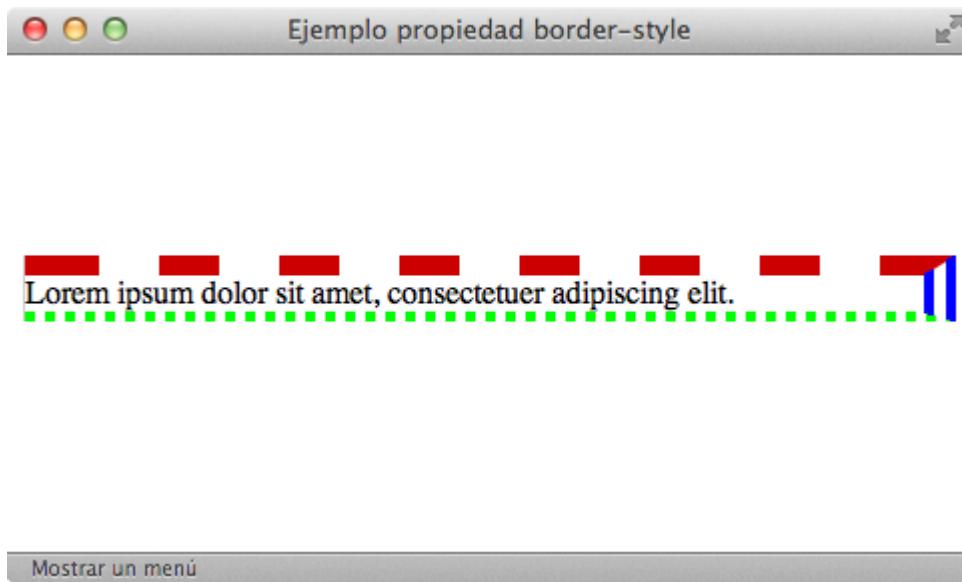


Figura 4.13 Ejemplo de propiedad border-style

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;  
}
```

Los bordes más utilizados 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.

Para establecer de forma simultánea los estilos de todos los bordes de una caja, es necesario utilizar la propiedad "*shorthand*" llamada border-style:

border-style	
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*".

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 simultánea. CSS incluye una propiedad "*shorthand*" para cada uno de los cuatro bordes y una propiedad "*shorthand*" global.

border-top, border-right, border-bottom, border-left	
Valores	(unidad de 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

El significado de cada uno de los valores especiales es el siguiente:

- <medida_borde>: una medida CSS o alguna de las siguientes palabras clave: thin, medium, thick.
- <color_borde>: un color de CSS o la palabra clave transparent

- <estilo_borde>: una de las siguientes palabras clave: none, hidden, dotted, dashed, solid, double, groove, ridge, inset, outset.

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	
Valores	(unidad_de_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

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 \(#ej04\)](#)

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 también se añaden todos sus márgenes, rellenos y bordes:

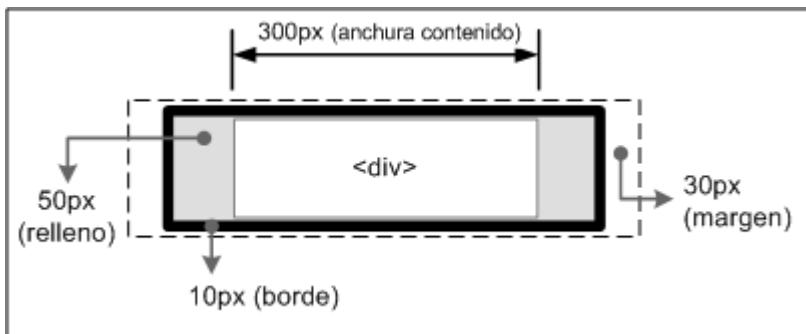


Figura 4.14 La anchura total de un elemento tiene en cuenta los márgenes, rellenos y bordes

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:

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

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.

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
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 \(#ej05\)](#)

Capítulo 5

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.

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:

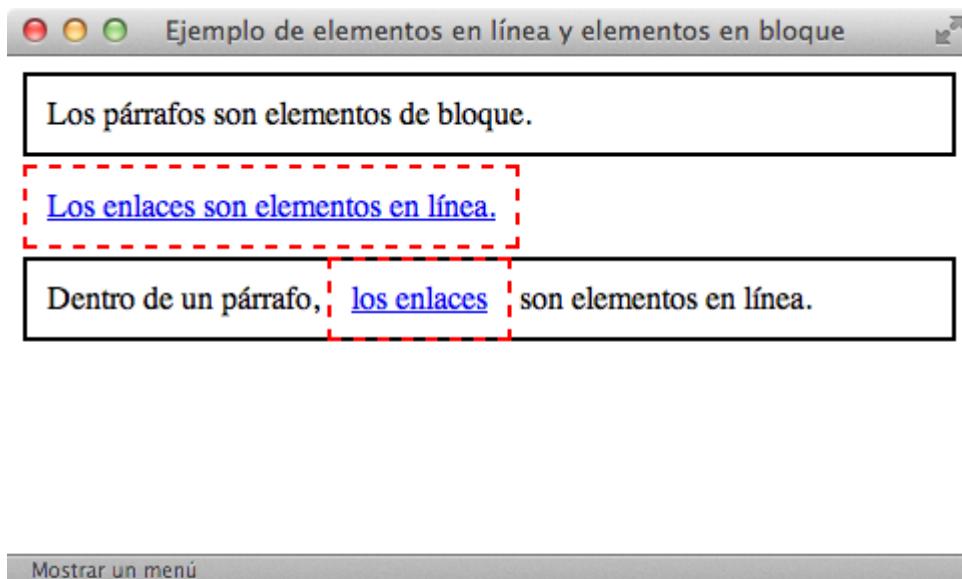


Figura 5.1 Cajas creadas por los elementos de línea y los elementos de bloque

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,noframes, 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.

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	
Valores	static relative absolute fixed inherit
Se aplica a	Todos los elementos
Valor inicial	static
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:

<code>top, right, bottom, left</code>	
Valores	unidad de medida porcentaje auto inherit
Se aplica a	Todos los elementos posicionados
Valor inicial	auto
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 parente 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.

El posicionamiento normal o estático es el modelo que utilizan por defecto los navegadores para mostrar los elementos de las páginas. En este modelo, sólo se tiene en cuenta si el elemento es de bloque o en línea, sus propiedades width y height y su contenido.

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.

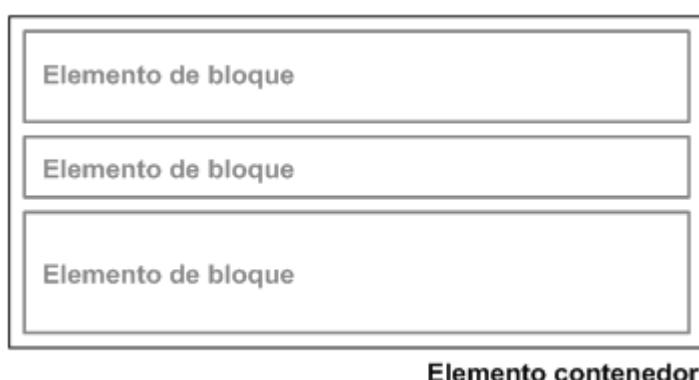


Figura 5.2 Posicionamiento normal de los elementos de bloque

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.

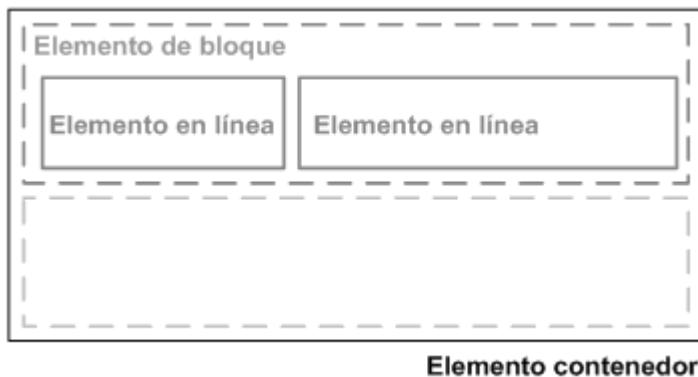


Figura 5.3 Posicionamiento normal de los elementos en línea

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.

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

El posicionamiento relativo desplaza 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 valor de la propiedad `top` se interpreta como el desplazamiento entre el borde superior de la caja en su posición final y el borde superior de la misma caja en su posición original.

De la misma forma, el valor de las propiedades `left`, `right` y `bottom` indica respectivamente el desplazamiento entre el borde izquierdo/derecho/inferior de la caja en su posición final y el borde izquierdo/derecho/inferior de la caja original.

Por tanto, la propiedad `top` se emplea para mover las cajas de forma descendente, la propiedad `bottom` mueve las cajas de forma ascendente, la propiedad `left` se utiliza para desplazar las cajas hacia la derecha y la propiedad `right` mueve las cajas hacia la izquierda. Este comportamiento parece poco intuitivo y es causa de errores cuando se empiezan a di-

señar páginas con CSS. Si se utilizan valores negativos en las propiedades top, right, bottom y left, su efecto es justamente el inverso.

El desplazamiento relativo 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.

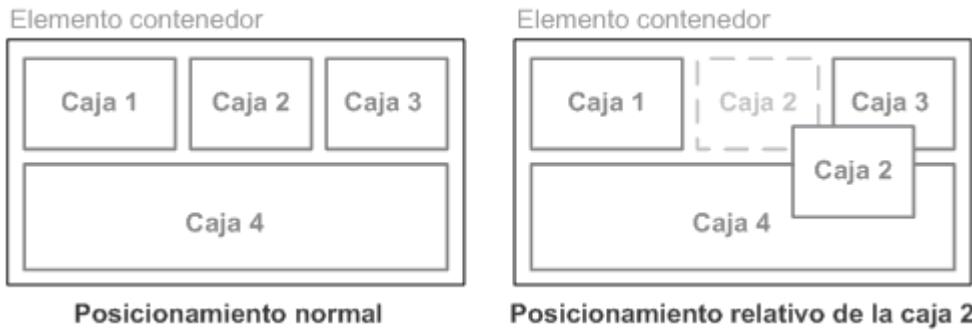


Figura 5.4 Ejemplo de posicionamiento relativo de un elemento

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.

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

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 posicionamiento absoluto se emplea para establecer de forma exacta 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 se ven afectados y modifican su posición. 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:

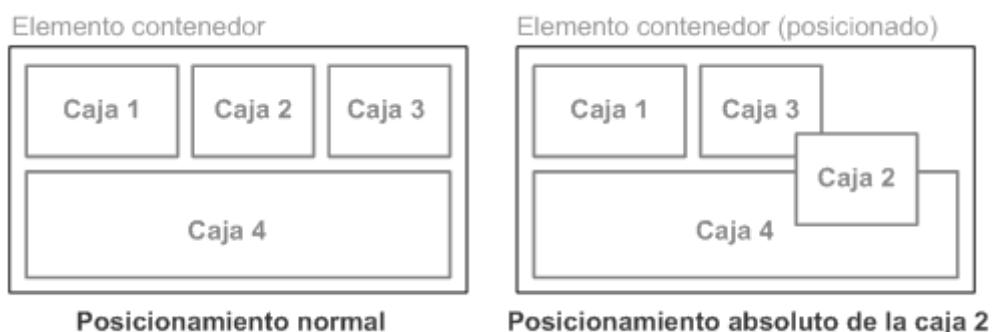


Figura 5.5 Ejemplo de posicionamiento absoluto de un elemento

La caja 2 está posicionada de forma absoluta, lo que provoca que el resto de elementos de la página modifiquen su posición. En concreto, la caja 3 deja su lugar original y pasa a ocupar el hueco dejado por la caja 2.

El estándar de CSS 2.1 indica que las cajas posicionadas de forma absoluta "*salen del flujo normal de la página*", lo que provoca que el resto de elementos de la página se muevan y en ocasiones, ocupen la posición original en la que se encontraba la caja.

Por otra parte, el desplazamiento de una caja posicionada de forma absoluta se controla mediante las propiedades top, right, bottom y left.

A diferencia del posicionamiento relativo, la interpretación de los valores de estas propiedades depende del elemento contenedor de la caja posicionada.

Determinar la referencia utilizada para interpretar los valores de `top`, `right`, `bottom` y `left` de 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>`
- El primer elemento contenedor que esté posicionado de cualquier forma diferente a `position: static` se convierte en la referencia que determina la posición de la caja posicionada de forma absoluta.
- Si ningún elemento contenedor está posicionado, la referencia es la ventana del navegador, que no debe confundirse con el elemento `<body>` de la página.

Una vez determinada la referencia del posicionamiento absoluto, la interpretación de los valores de las propiedades `top`, `right`, `bottom` y `left` se realiza como sigue:

- El valor de la propiedad `top` indica el desplazamiento desde el borde superior de la caja hasta el borde superior del elemento contenedor que se utiliza como referencia.
- El valor de la propiedad `right` indica el desplazamiento desde el borde derecho de la caja hasta el borde derecho del elemento contenedor que se utiliza como referencia.
- El valor de la propiedad `bottom` indica el desplazamiento desde el borde inferior de la caja hasta el borde inferior del elemento contenedor que se utiliza como referencia.
- El valor de la propiedad `left` indica el desplazamiento desde el borde izquierdo de la caja hasta el borde izquierdo del elemento contenedor que se utiliza como referencia.

En los siguientes ejemplos, se utiliza la página HTML que muestra la siguiente imagen:

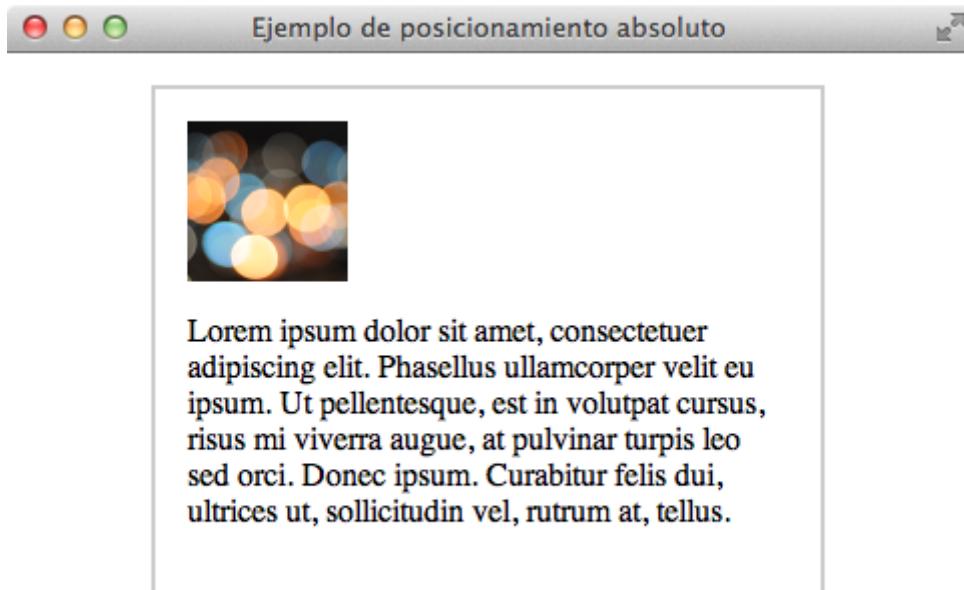


Figura 5.6 Situación original antes de modificar el posicionamiento

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, consectetur 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: 50px;
```

```
    left: 50px;  
}
```

El resultado visual se muestra en la siguiente imagen:

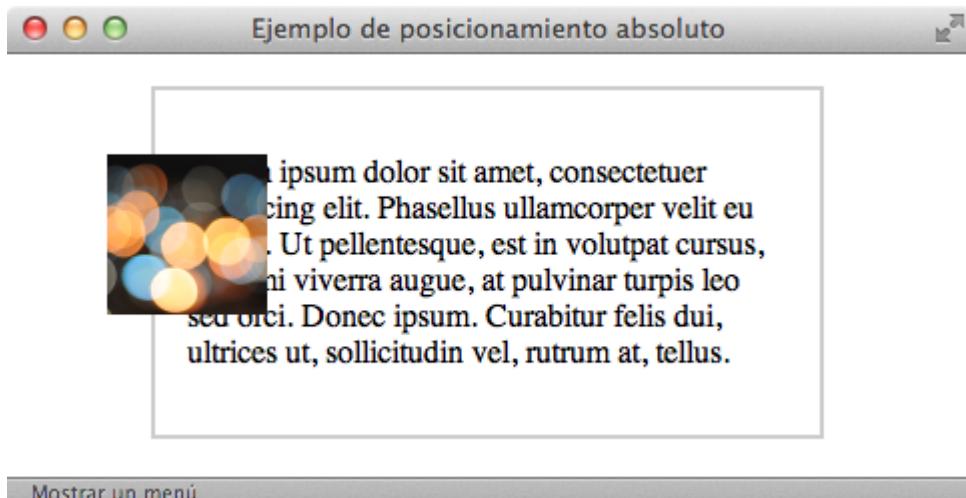


Figura 5.7 Imagen posicionada de forma absoluta

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 contenedores desde el más cercano a la imagen (el `<div>`) hasta terminar en el `<body>` buscando el primer elemento contenedor que esté posicionado.
3. El posicionamiento de todos los elementos contenedores es el normal o estático, ya que ni siquiera tienen establecida la propiedad `position`
4. Como ningún elemento contenedor está posicionado, la referencia es la ventana del navegador.
5. A partir de esa referencia, la caja de la imagen se desplaza 50px hacia la derecha (`left: 50px`) y otros 50px de forma descendente (`top: 50px`).

Como la imagen se posiciona de forma absoluta, el resto de elementos de la página se mueven para ocupar el lugar libre dejado por la imagen. Por este motivo, el párrafo sube hasta el principio del `<div>` y se produce un solapamiento con la imagen posicionada que impide ver parte de los contenidos del párrafo.

A continuación, se modifica el ejemplo anterior posicionando de forma relativa el elemento `<div>` que contiene la imagen y el párrafo. 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: 50px;  
    left: 50px;  
}
```

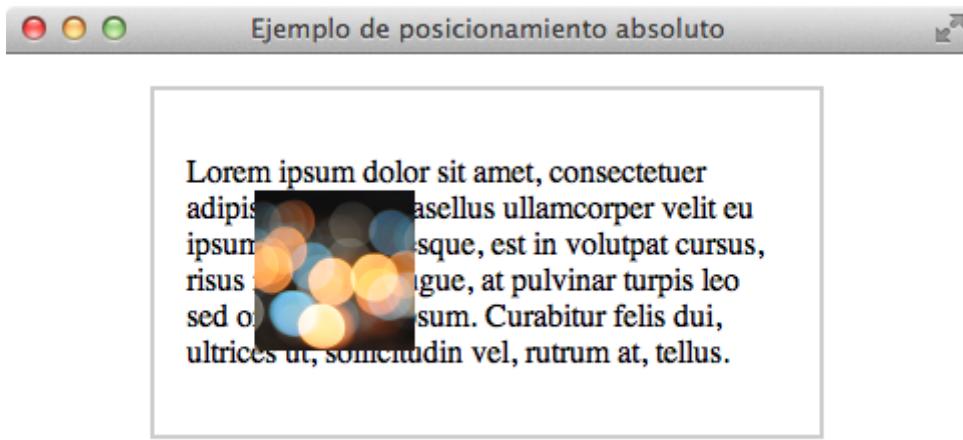


Figura 5.8 Imagen posicionada de forma absoluta

En este caso, como el elemento contenedor de la imagen está posicionado, se convierte en la referencia para el posicionamiento absoluto. El resultado es que la posición de la imagen es muy diferente a la del ejemplo anterior. Por tanto, si se quiere posicionar un elemento de forma absoluta respecto de su elemento contenedor, es imprescindible posicionar este último. 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.

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.

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:

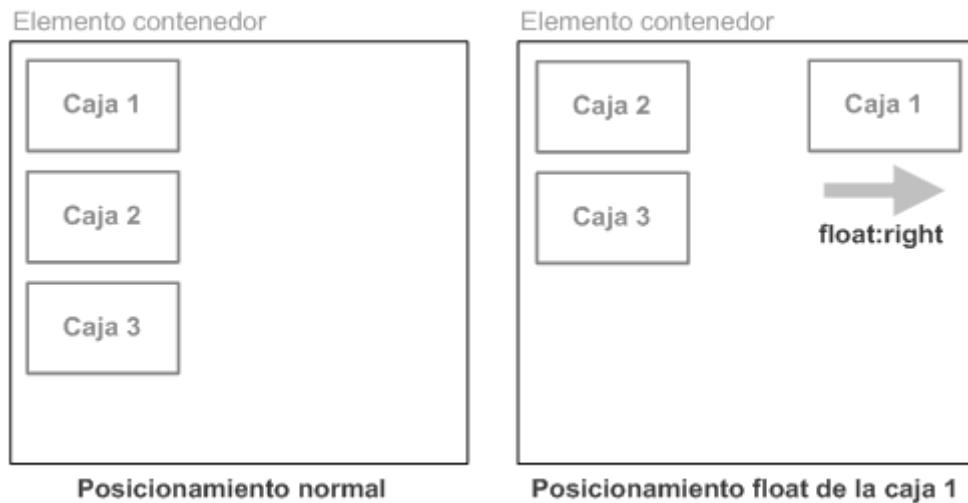


Figura 5.9 Ejemplo de posicionamiento float de una caja

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:

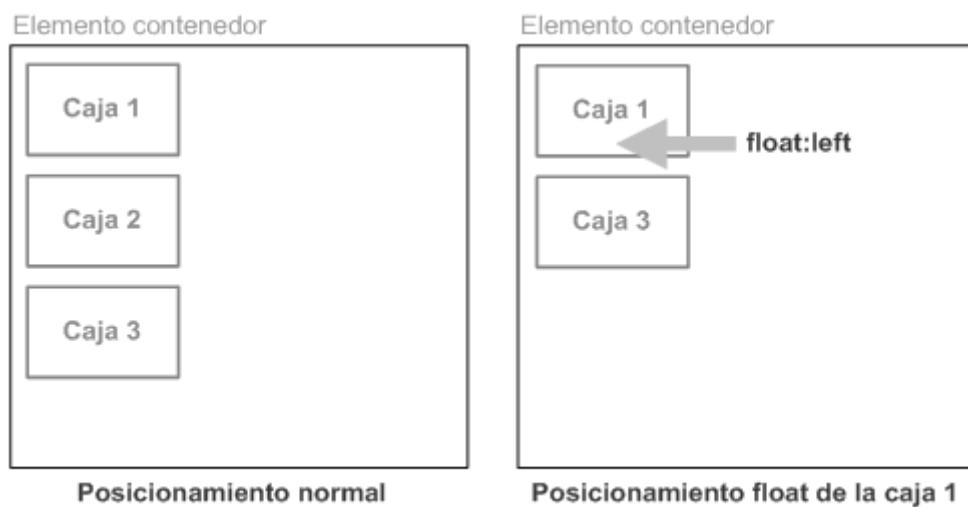


Figura 5.10 Ejemplo de posicionamiento float de una caja

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 posicianan de forma flotante hacia la izquierda las tres cajas:

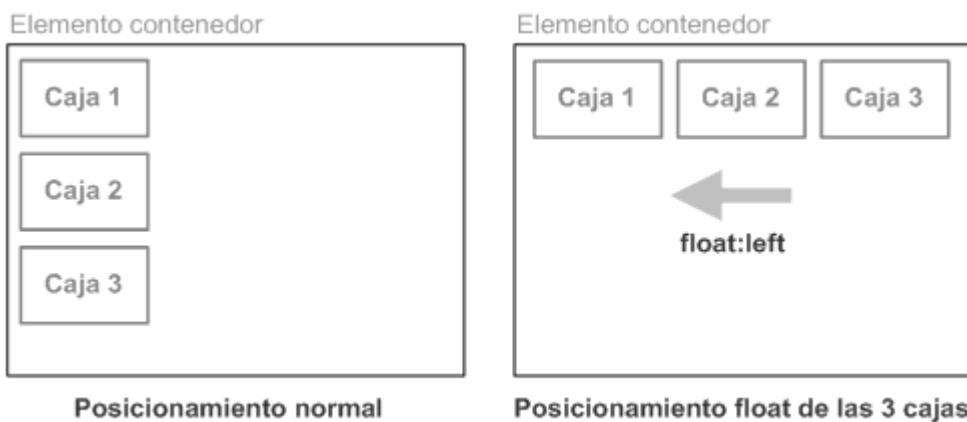


Figura 5.11 Ejemplo de posicionamiento float de una caja

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:

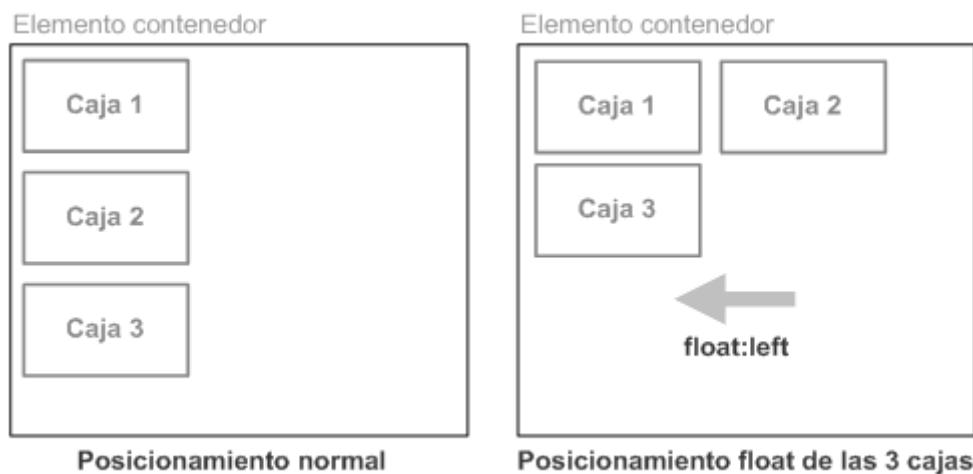


Figura 5.12 Ejemplo de posicionamiento float de una caja

Las cajas flotantes influyen en la disposición de todas las demás cajas. Los elementos en línea *hacen 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`:

<code>float</code>	
Valores	left right none inherit
Se aplica a	Todos los elementos
Valor inicial	none
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 \(#ej06\)](#)

Los elementos que se encuentran alrededor de una caja flotante adaptan sus contenidos para que fluyan alrededor del elemento posicionado:

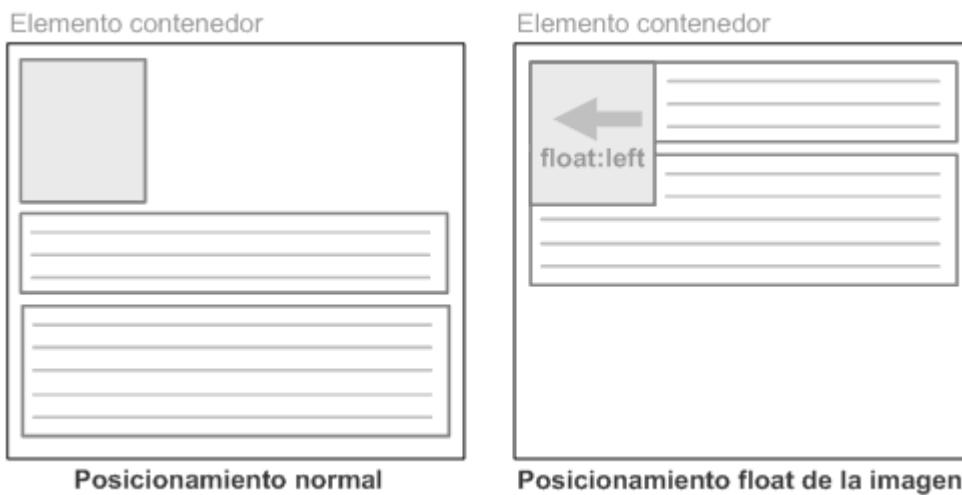


Figura 5.13 Elementos que fluyen alrededor de un elemento posicionado mediante float

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:

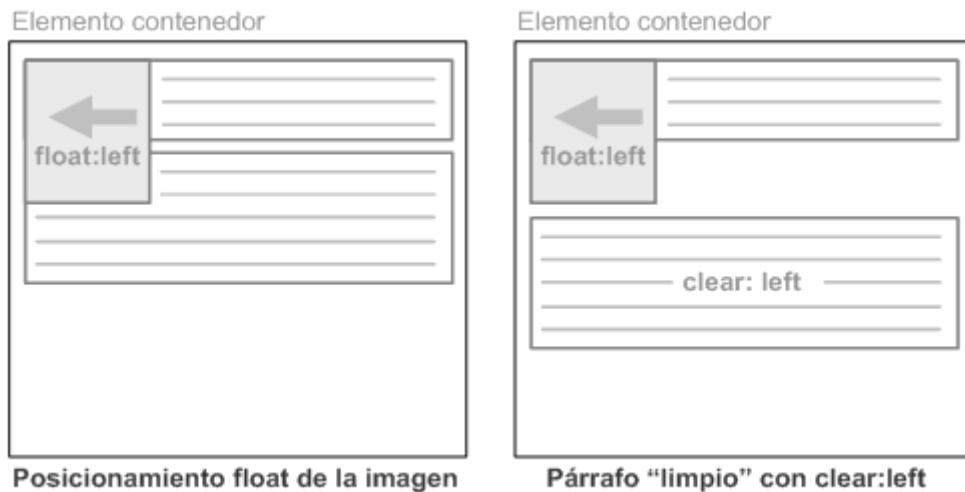


Figura 5.14 Forzando a que un elemento no fluya alrededor de otro elemento posicionado mediante float

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	
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.

Si se considera el siguiente código CSS y HTML:

```
#paginacion {  
    border: 1px solid #CCC;  
    background-color: #E0E0E0;  
    padding: .5em;  
}  
  
.derecha { float: right; }  
.izquierda { float: left; }  
<div id="paginacion">  
    <span class="izquierda">&lquo; Anterior</span>  
    <span class="derecha">Siguiente &raquo;</span>  
</div>
```

Si se visualiza la página anterior en cualquier navegador, el resultado es el que muestra la siguiente imagen:

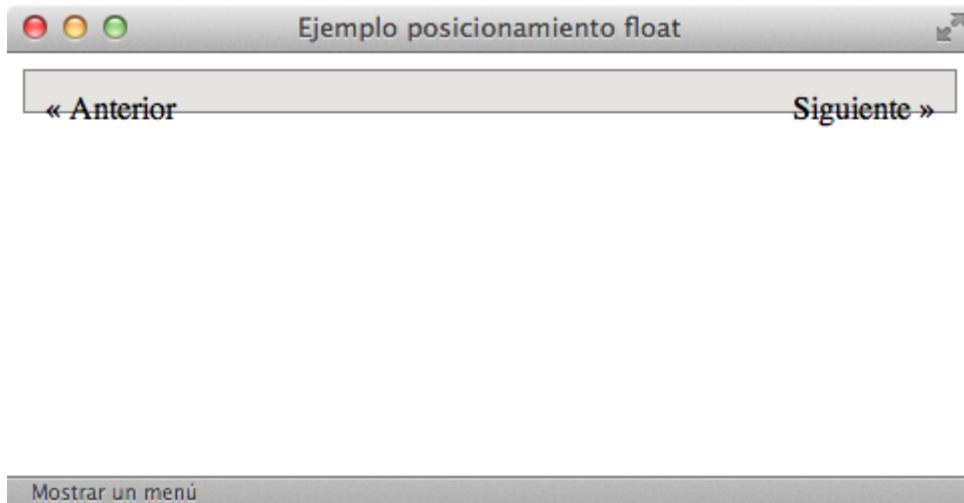


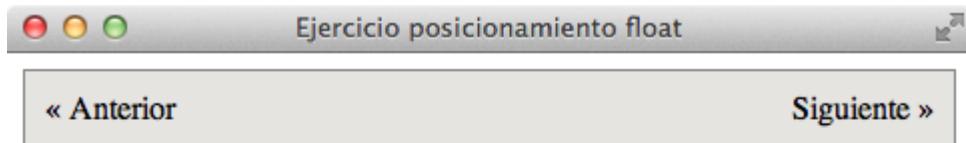
Figura 5.15 Visualización incorrecta de dos elementos posicionados mediante float

Los elementos Anterior y Siguiente se salen de su elemento contenedor y el resultado es visualmente incorrecto. El motivo de este comportamiento es que un elemento posicionado de forma flotante ya no pertenece al flujo normal de la página HTML. Por tanto, el elemento `<div id="paginacion">` en realidad no encierra ningún contenido y por eso se visualiza incorrectamente.

La solución consiste en utilizar la propiedad overflow (que se explica más adelante) sobre el elemento contenedor:

```
#paginacion {  
    border: 1px solid #CCC;  
    background-color: #E0E0E0;  
    padding: .5em;  
    overflow: hidden;  
}  
  
.derecha { float: right; }  
.izquierda { float: left; }
```

Si se visualiza de nuevo la página anterior en cualquier navegador, el resultado ahora sí que es el esperado:



[Mostrar un menú](#)

Figura 5.16 Visualización correcta de dos elementos posicionados mediante float

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.

Las propiedades `display` y `visibility` controlan la visualización de los elementos. Las dos propiedades permiten ocultar cualquier elemento de la página. Habitualmente se utilizan junto con JavaScript para crear efectos dinámicos como mostrar y ocultar determinados textos o imágenes cuando el usuario pincha sobre ellos.

La propiedad `display` permite ocultar completamente un elemento haciendo que desaparezca de la página. Como el elemento oculto no se muestra, el resto de elementos de la página se mueven para ocupar su lugar.

Por otra parte, la propiedad `visibility` permite hacer invisible un elemento, lo que significa que el navegador crea la caja del elemento pero no la muestra. En este caso, el resto de elementos de la página no modifican su posición, ya que aunque la caja no se ve, sigue ocupando sitio.

La siguiente imagen muestra la diferencia entre ocultar la caja número 5 mediante la propiedad `display` o hacerla invisible mediante la propiedad `visibility`:

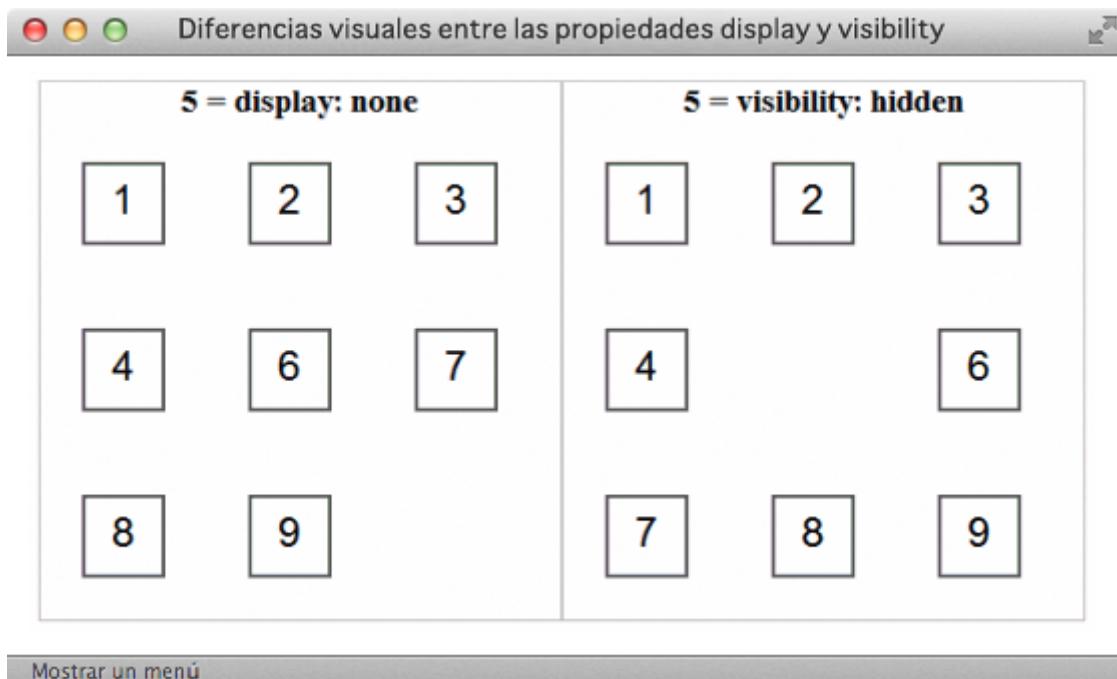


Figura 5.17 Diferencias visuales entre las propiedades `display` y `visibility`

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`:

	<code>display</code>
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 más avanzadas que simplemente ocultar elementos. 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` muestra un elemento como si fuera un elemento de bloque, independientemente del tipo de elemento que se trate. El valor `inline` visualiza un elemento en forma de elemento en línea, independientemente del tipo de elemento que se trate.

El valor `none` oculta un elemento y hace que desaparezca de la página. El resto de elementos de la página se visualizan como si no existiera el elemento oculto, es decir, pueden ocupar el espacio en el que se debería visualizar el elemento.

Como se verá más adelante, la propiedad `display: inline` se puede utilizar en 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 su parte, la definición completa de la propiedad `visibility` es mucho más sencilla:

visibility	
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.

Inicialmente todas las cajas que componen la página son visibles. Empleando el valor `hidden` es posible convertir una caja en invisible para que no muestre sus contenidos. El resto de elementos de la página se muestran como si la caja todavía fuera visible, por lo que en el lugar donde originalmente se mostraba la caja invisible, 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.

Cuando se establecen las propiedades display, float y position sobre una misma caja, su interpretación es la siguiente:

- Si display vale none, se ignoran las propiedades float y position y la caja no se muestra en la página.
- 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.

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.

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	
Valores	visible hidden scroll auto inherit

	overflow
Se aplica a	Elementos de bloque y celdas de tablas
Valor inicial	visible
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:

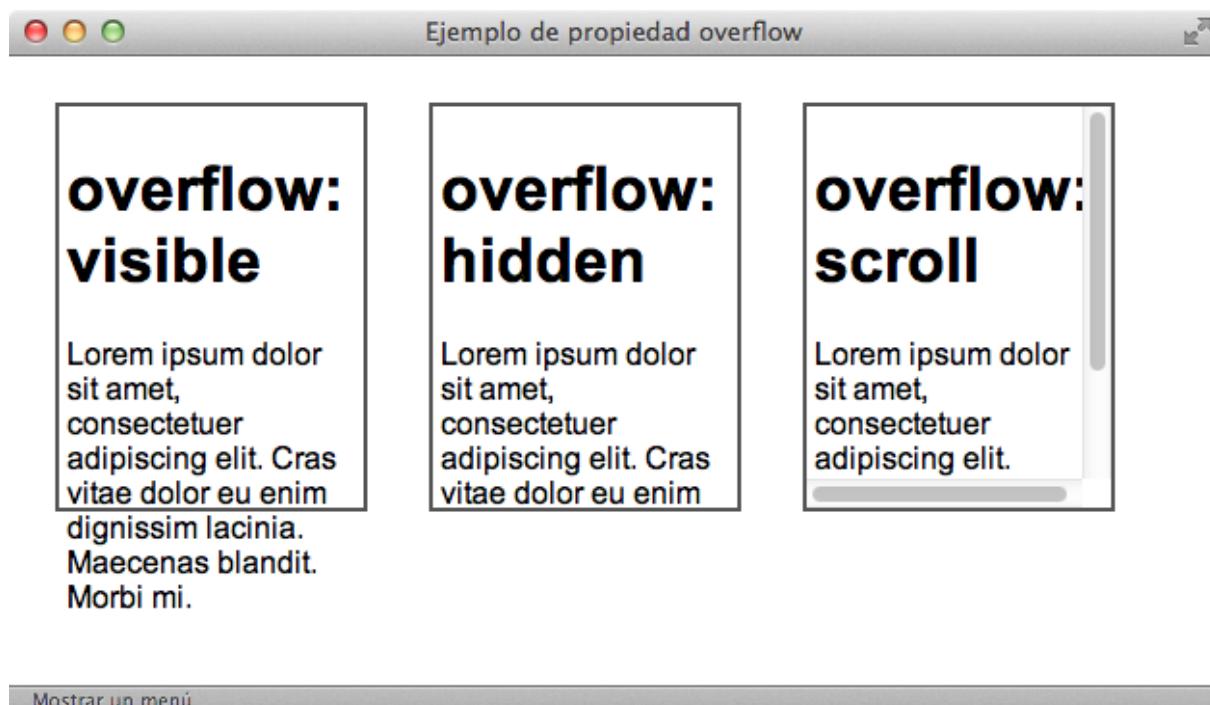


Figura 5.18 Ejemplo de propiedad overflow

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 indicar las cajas que se muestran delante o detrás de otras cajas cuando se producen solapamientos.

La posición tridimensional de un elemento 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	
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`.

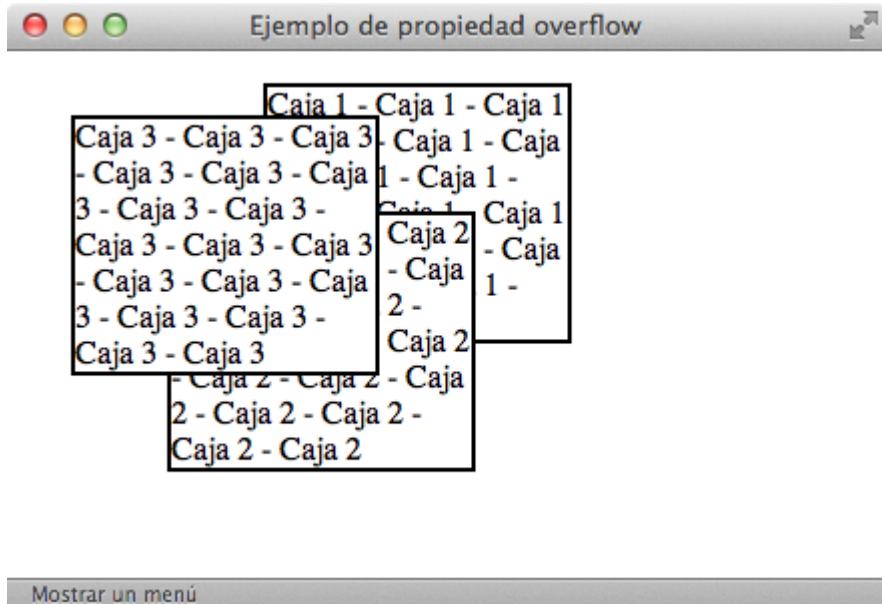


Figura 5.19 Ejemplo de propiedad z-index

La propiedad z-index sólo tiene efecto en los elementos posicionados, por lo que es obligatorio que la propiedad z-index vaya acompañada de la propiedad position. Si debes posicionar un elemento pero no quieres moverlo de su posición original ni afectar al resto de elementos de la página, puedes utilizar el posicionamiento relativo (position: relative).

Capítulo 6

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	
Valores	color inherit
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:

```
h1 { 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	
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 o dispositivo 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`.

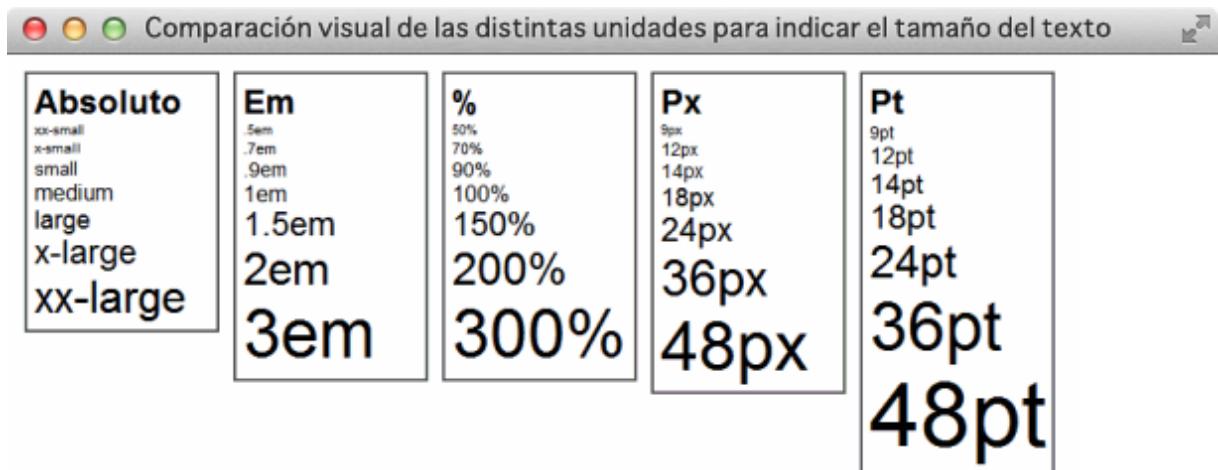
font-size	
Valores	tamaño_absoluto tamaño_relativo unidad de medida porcentaje inherit
Se aplica a	Todos los elementos
Valor inicial	medium
Descripción	Establece el tamaño de letra utilizado para el texto

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:



Mostrar un menú

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 imprimirlo.

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`.

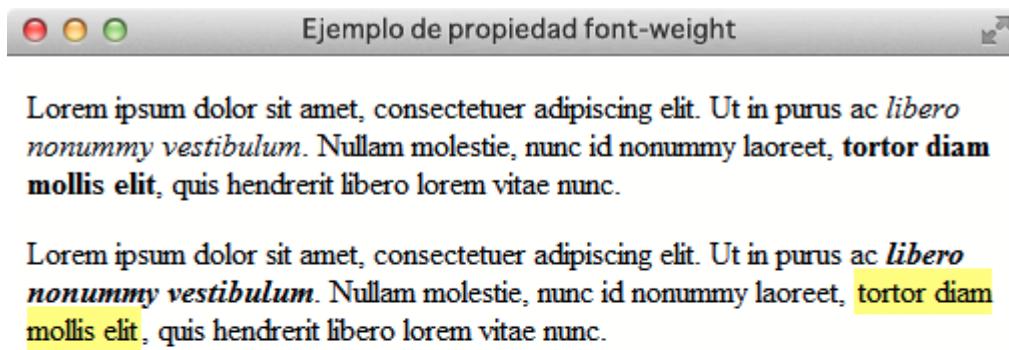
Una vez indicado el tipo y el tamaño de letra, es habitual modificar otras características como su grosor (texto en negrita) y su estilo (texto en cursiva). La propiedad que controla la anchura de la letra es `font-weight`.

font-weight	
Valores	<code>normal</code> <code>bold</code> <code>bolder</code> <code>lighter</code> <code>100</code> <code>200</code> <code>300</code> <code>400</code> <code>500</code> <code>600</code> <code>700</code> <code>800</code> <code>900</code> <code>inherit</code>

	font-weight
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`:



Mostrar un menú

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, consectetuer 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, consectetuer 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	
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 ``:

The screenshot shows a web browser window with a title bar "Ejemplo de propiedad font-style". Below the title bar are three colored circular icons (red, yellow, green). The main content area displays two paragraphs of text. The first paragraph contains the text: "Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut in purus ac **libero nonummy vestibulum**. Nullam molestie, nunc id nonummy laoreet, **tortor diam mollis elit**, quis hendrerit libero lorem vitae nunc." The second paragraph contains the same text, but the words "libero", "nonummy", "vestibulum", "tortor", "diam", and "mollis" are highlighted with a yellow background. A toolbar at the bottom has a "Mostrar un menú" button.

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;  
}
```

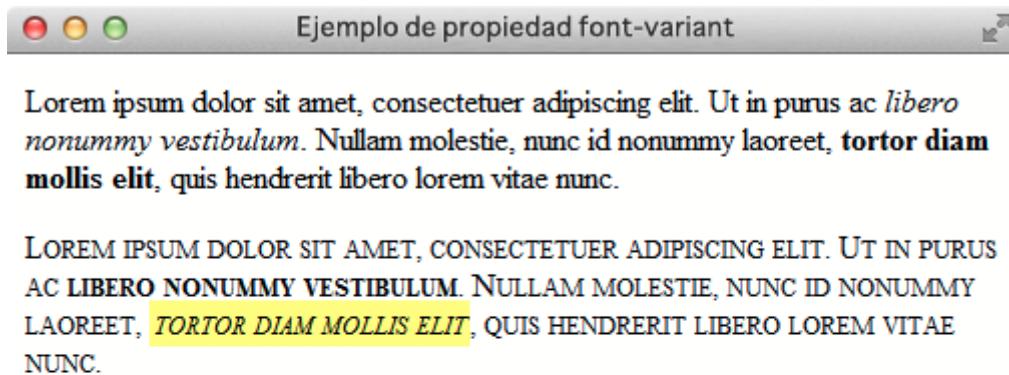
Por último, CSS permite otra variación en el estilo del tipo de letra, controlado mediante la propiedad `font-variant`.

	font-variant
Valores	normal small-caps inherit

	font-variant
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:



Mostrar un menú

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
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 \(#ej07\)](#)

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`.

<code>text-align</code>	
Valores	<code>left right center justify 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.

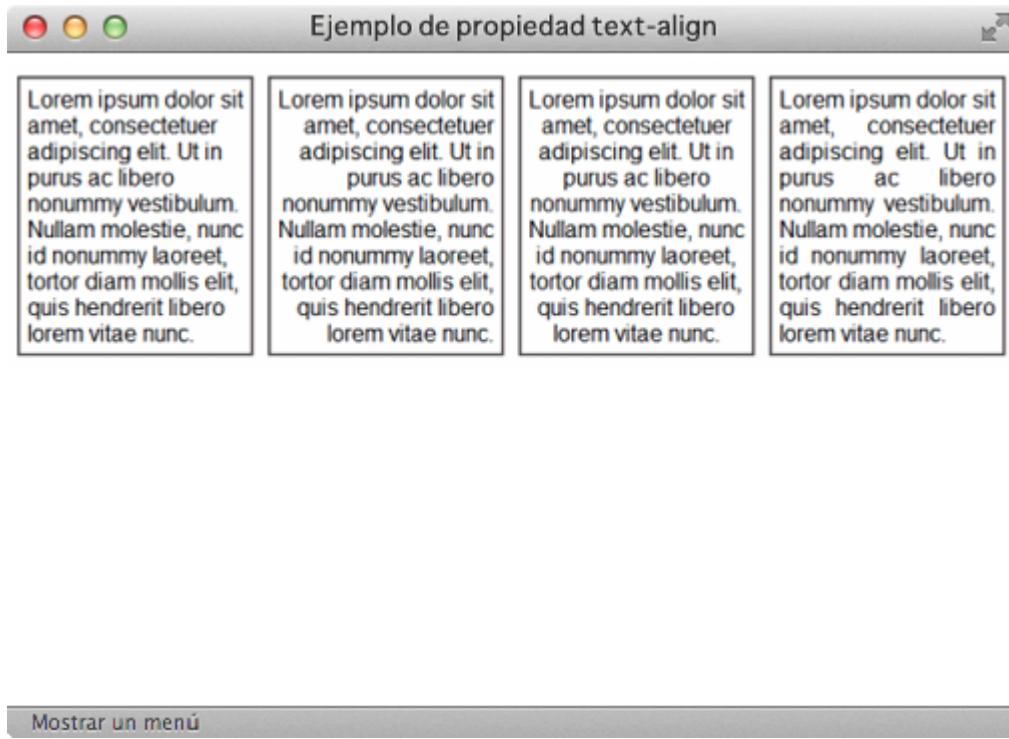


Figura 6.5 Ejemplo de propiedad text-align

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:

	<code>line-height</code>
Valores	<code>normal</code> <code>numero</code> <code>unidad de 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 de letra del elemento. 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`.

<code>text-decoration</code>	
Valores	none (<code>underline</code> <code>overline</code> <code>line-through</code> <code>blink</code>) <code>inherit</code>
Se aplica a	Todos los elementos
Valor inicial	<code>none</code>
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.

<code>text-transform</code>	
Valores	capitalize uppercase lowercase none inherit
Se aplica a	Todos los elementos
Valor inicial	none
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:

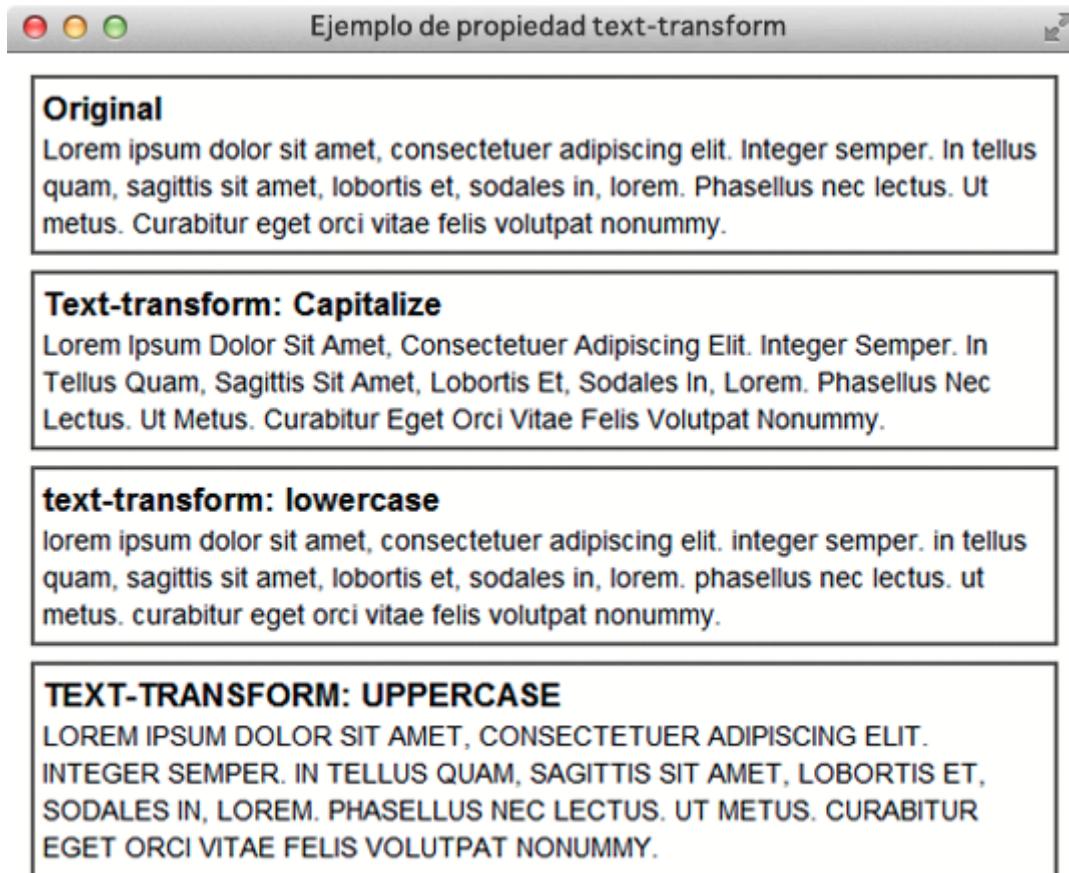


Figura 6.7 Ejemplo de propiedad text-transform

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`.

<code>vertical-align</code>	
Valores	baseline sub super top text-top middle bottom text-bottom porcentaje unidad de 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`:

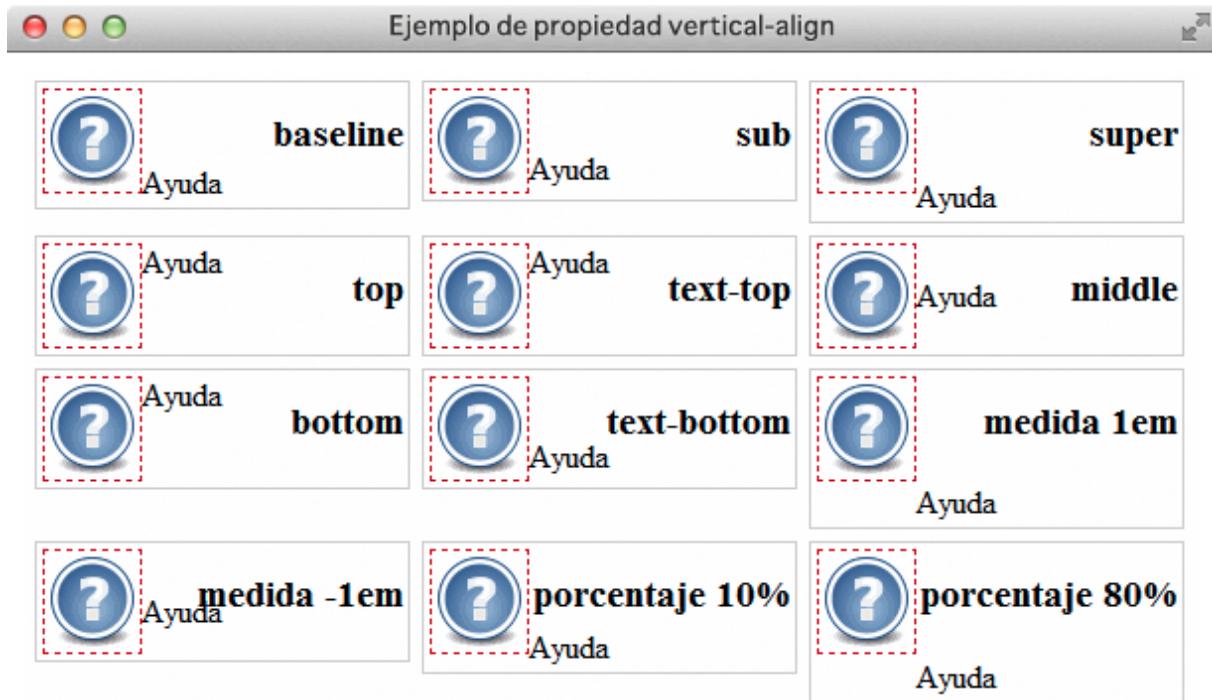


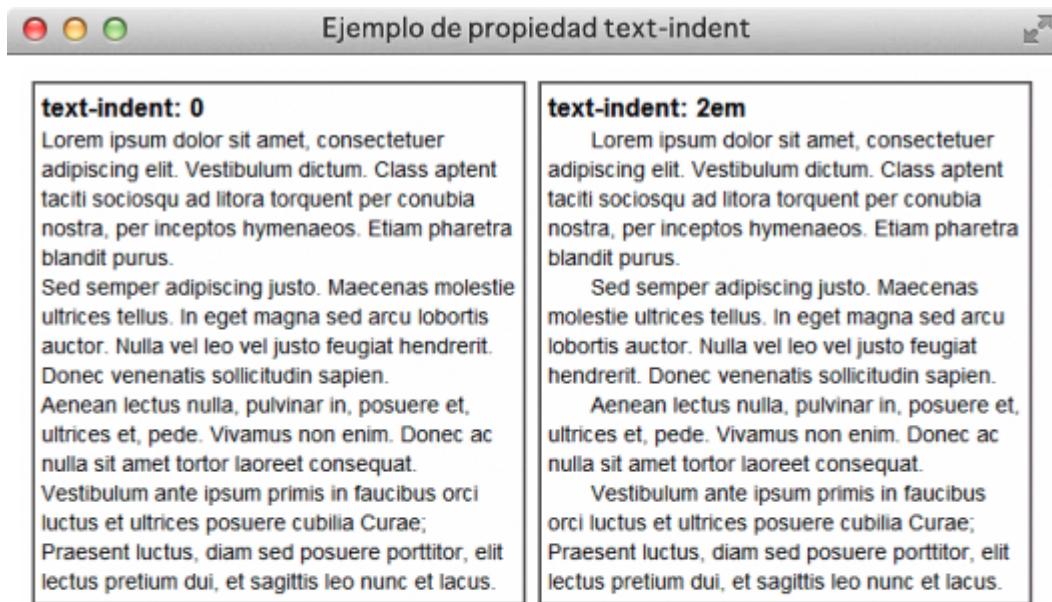
Figura 6.8 Ejemplo de propiedad vertical-align

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.

Valores	unidad de medida porcentaje inherit
Se aplica a	Los elementos de bloque y las celdas de tabla
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:



Mostrar un menú

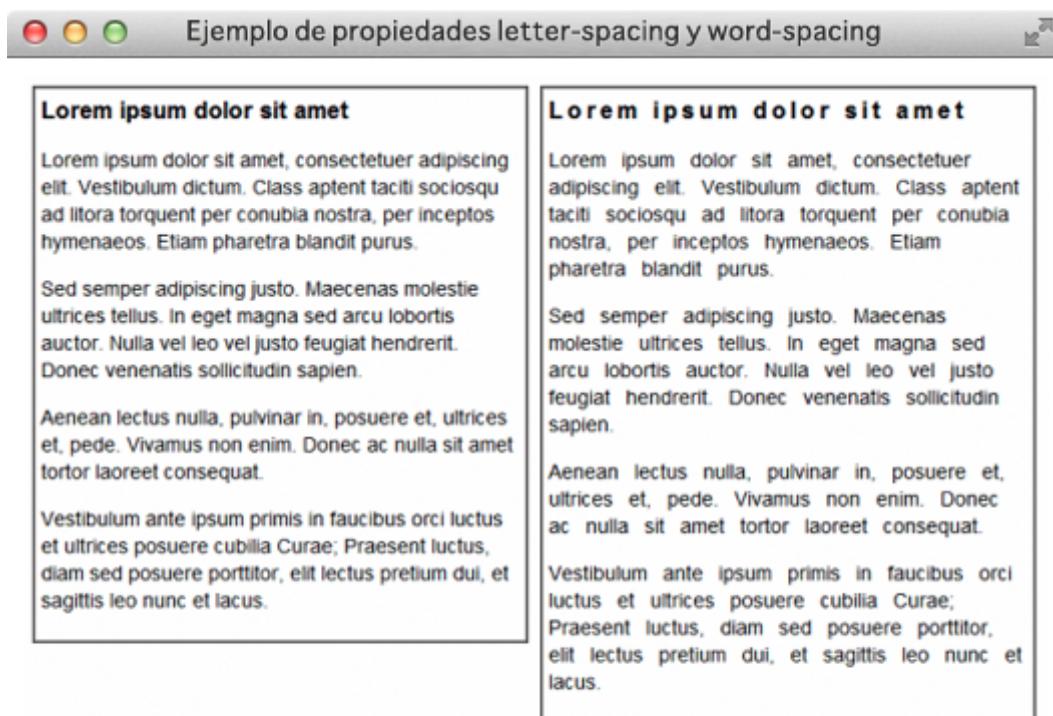
Figura 6.9 Ejemplo de propiedad text-indent

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`.

<code>letter-spacing</code>	
Valores	normal unidad de 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
Valores	normal unidad de 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:



Mostrar un menú

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 &nbs; 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	
Valores	normal pre nowrap pre-wrap pre-line inherit
Se aplica a	Todos los elementos

white-space	
Valor inicial	normal
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:

normal	no	no	si
pre	si	si	no
nowrap	no	no	no
pre-wrap	si	si	si
pre-line	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:

The screenshot shows a web browser window with a title bar "Ejemplo de propiedad white-space". Below the title bar are four separate boxes, each demonstrating a different value for the white-space property:

- white-space: pre;** The text is displayed with two line breaks and three spaces between the first and second lines.

```
  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.
```
- white-space: pre-wrap;** The text is displayed with one line break and three spaces between the first and second lines.

```
  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.
```
- white-space: nowrap;** The text is displayed as a single horizontal line.

```
  Lorem ipsum dolor sit amet, consectetur adipiscing elit. Vestibulum dictum. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos hymenaeos. Etiam pharetra blandit purus.
```
- white-space: pre-line;** The text is displayed with one line break and no spaces between the first and second lines.

```
  Lorem ipsum dolor sit amet, consectetur adipiscing elit.  
Vestibulum dictum. Class aptent taciti sociosqu ad litora  
torquent per conubia nostra, per inceptos hymenaeos.  
Etiam pharetra blandit purus.
```

Mostrar un menú

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:

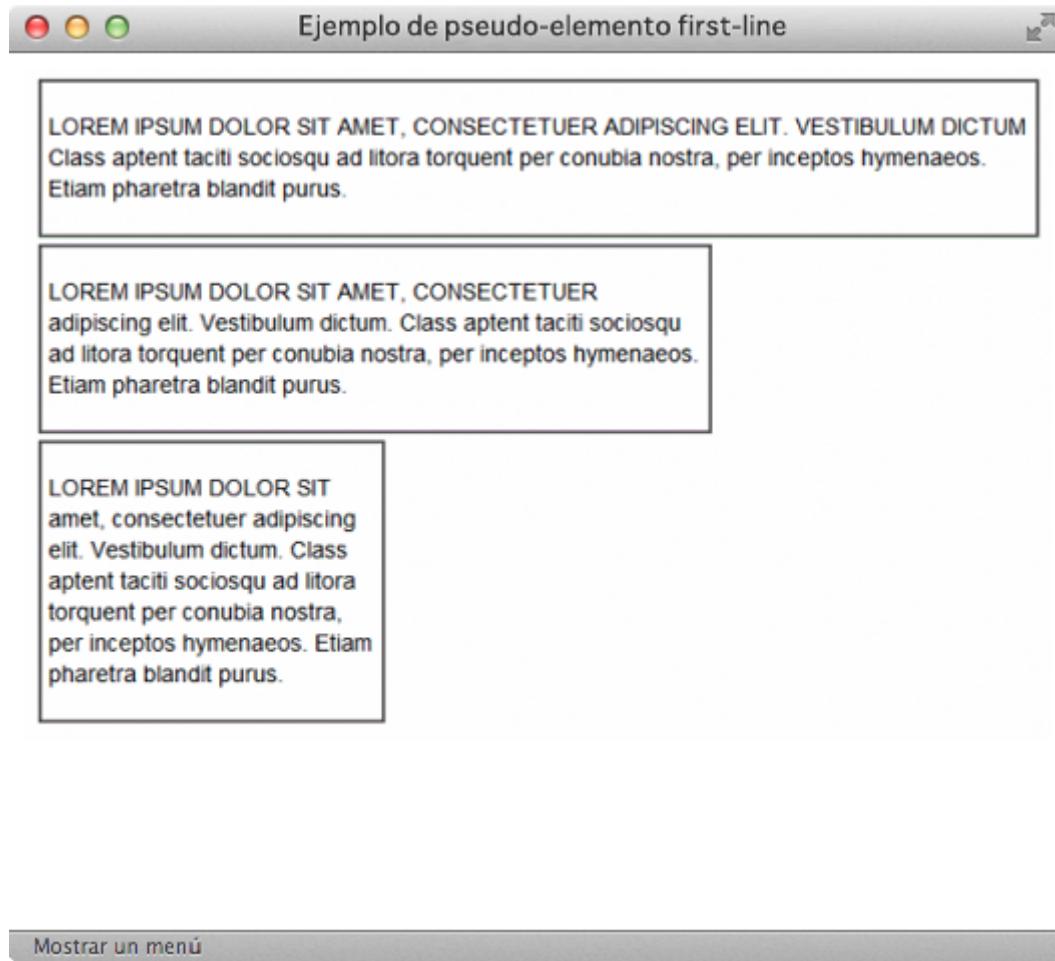


Figura 6.12 Ejemplo de pseudo-elemento first-line

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:

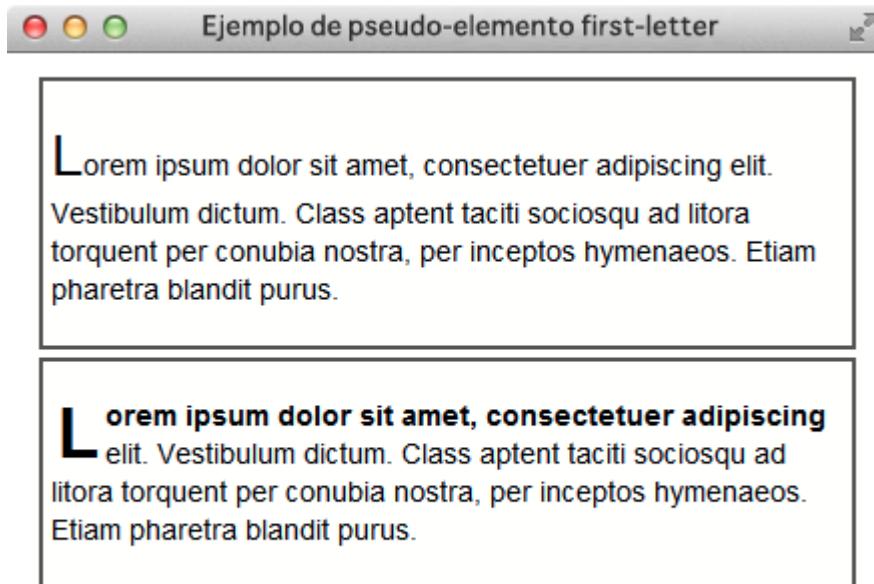


Figura 6.13 Ejemplo de pseudo-elemento first-letter

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, consectetur 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, consectetur adipiscing elit.  
    Vestibulum  
        dictum. Class aptent taciti sociosqu ad litora torquent per conubia
```

```
    nostra, per inceptos hymenaeos. Etiam pharetra blandit purus.</p>
</div>
```

Esta página se ha dejado vacía a propósito

Capítulo 7

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:

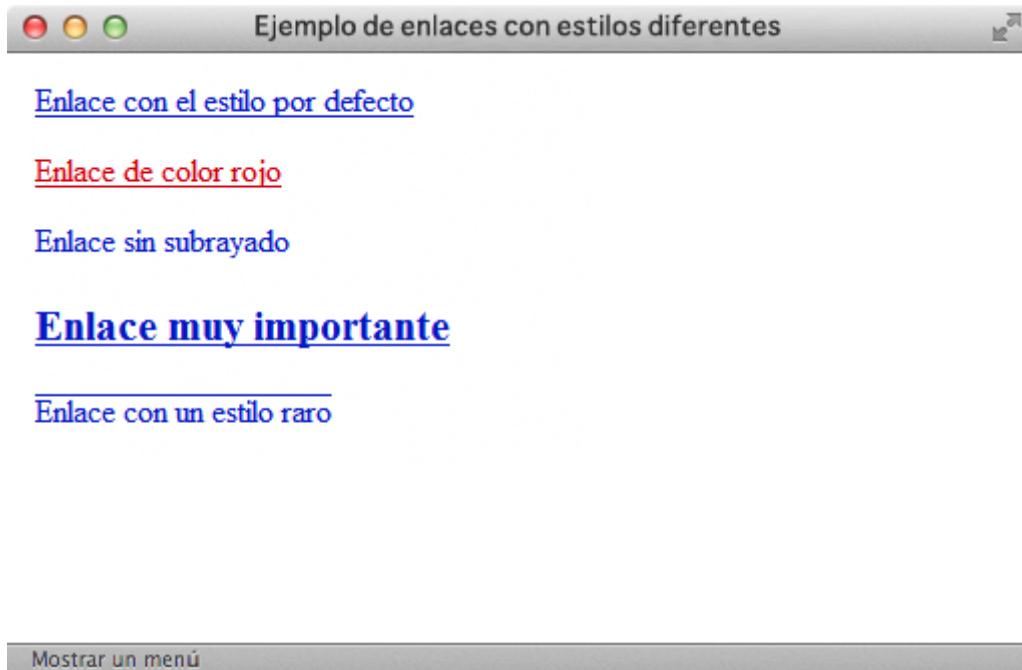


Figura 7.1 Ejemplo de enlaces con estilos diferentes

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>
```

CSS también permite aplicar diferentes estilos a un mismo enlace en función de su estado. De esta forma, es posible cambiar el aspecto de un enlace cuando por ejemplo el usuario pasa el ratón por encima o cuando el usuario pincha sobre ese enlace.

Como con los atributos `id` o `class` no es posible aplicar diferentes estilos a un mismo elemento en función de su estado, CSS introduce un nuevo concepto llamado **pseudo-clases**. En concreto, CSS define las siguientes cuatro pseudo-clases:

- `:link`, aplica estilos a los enlaces que apuntan a páginas o recursos que aún no han sido visitados por el usuario.
- `:visited`, aplica estilos a los enlaces que apuntan a recursos que han sido visitados anteriormente por el usuario. El historial de enlaces visitados se borra automáticamente cada cierto tiempo y el usuario también puede borrarlo manualmente.
- `:hover`, aplica estilos al enlace sobre el que el usuario ha posicionando el puntero del ratón.
- `:active`, aplica estilos al enlace que está pinchando el usuario. Los estilos sólo se aplican desde que el usuario pincha el botón del ratón hasta que lo suelta, por lo que suelen ser unas pocas décimas de segundo. Como se sabe, por defecto los navegadores muestran los enlaces no visitados de color azul y subrayados y los enlaces visitados de color morado. Las pseudo-clases anteriores permiten modificar completamente ese aspecto por defecto y por eso casi todas las páginas las utilizan.

El siguiente ejemplo muestra cómo ocultar el subrayado cuando el usuario pasa el ratón por encima de cualquier enlace de la página:

```
| a:hover { text-decoration: none; }
```

Aplicando las reglas anteriores, los navegadores ocultan el subrayado del enlace sobre el que se posiciona el ratón:

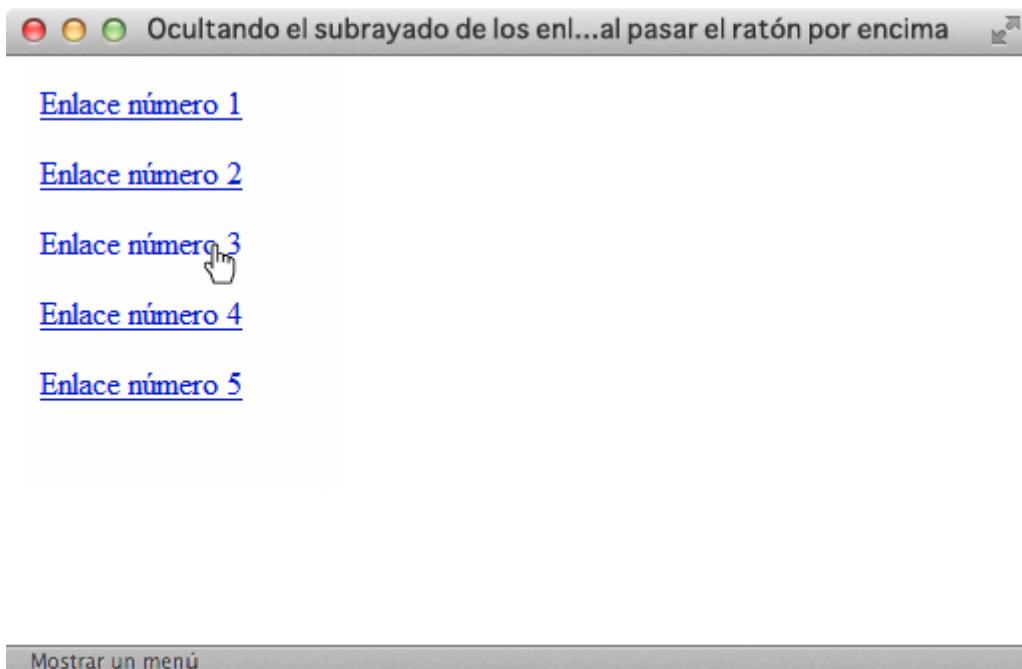


Figura 7.2 Ocultando el subrayado de los enlaces al pasar el ratón por encima

Las pseudo-clases siempre incluyen dos puntos (:) por delante de su nombre y siempre se añaden a continuación del elemento al que se aplican, sin dejar ningún espacio en blanco por delante:

```
/* Incorrecto: el nombre de la pseudo-clase no se puede separar de los  
dos puntos iniciales */  
a: visited { ... }  
  
/* Incorrecto: la pseudo-clase siempre se añade a continuación del  
elemento al que modifica */  
a :visited { ... }  
  
/* Correcto: la pseudo-clase modifica el comportamiento del elemento <a>  
*/  
a:visited { ... }
```

Las pseudo-clases también se pueden combinar con cualquier otro selector complejo:

```
a.importante:visited { ... }  
a#principal:hover { ... }  
div#menu ul li a.primero:active { ... }
```

Cuando se aplican varias pseudo-clases diferentes sobre un mismo enlace, se producen colisiones entre los estilos de algunas pseudo-clases. Si se pasa por ejemplo el ratón por encima de un enlace visitado, se aplican los estilos de las pseudo-clases :hover y :visited. Si el usuario pincha sobre un enlace no visitado, se aplican las pseudo-clases :hover, :link y :active y así sucesivamente.

Si se definen varias pseudo-clases sobre un mismo enlace, el único orden que asegura que todos los estilos de las pseudo-clases se aplican de forma coherente es el siguiente: :link, :visited, :hover y :active. De hecho, en muchas hojas de estilos es habitual establecer los estilos de los enlaces de la siguiente forma:

```
a:link, a:visited {  
    ...  
}  
  
a:hover, a: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.

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 \(#ej08\)](#)

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:

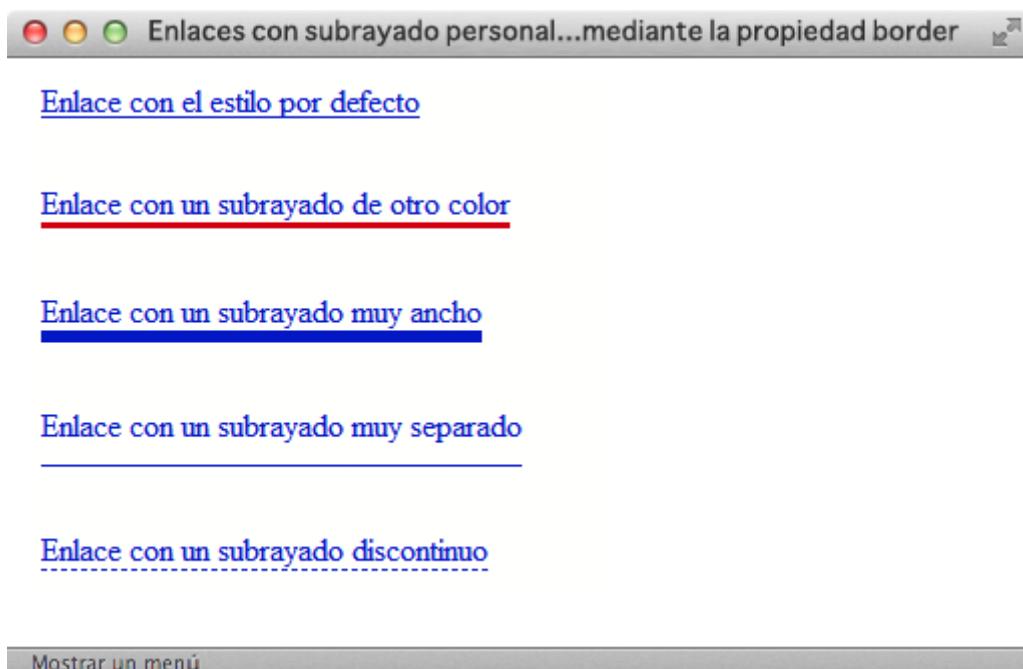


Figura 7.3 Enlaces con subrayado personalizado mediante la propiedad `border`

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>
```

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:

Personalizando el aspecto de los enlaces en función de su tipo

[Enlace con el estilo por defecto](#)

[Enlace a un archivo RSS](#)

[Enlace a un documento PDF](#)

Mostrar un menú

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>
```

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:

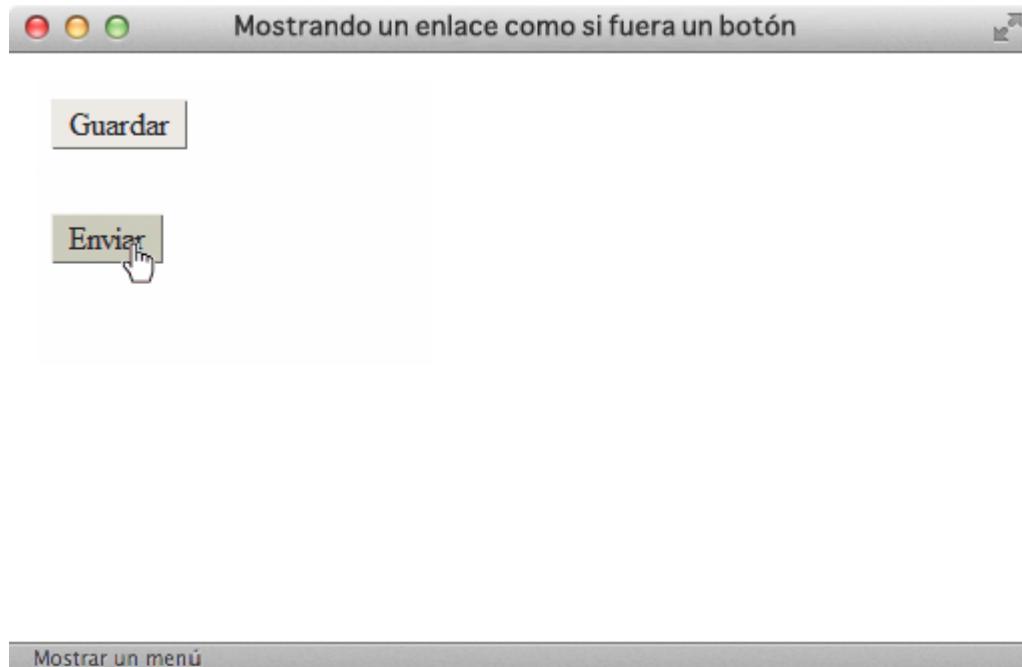


Figura 7.5 Mostrando un enlace como si fuera un botón

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

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.

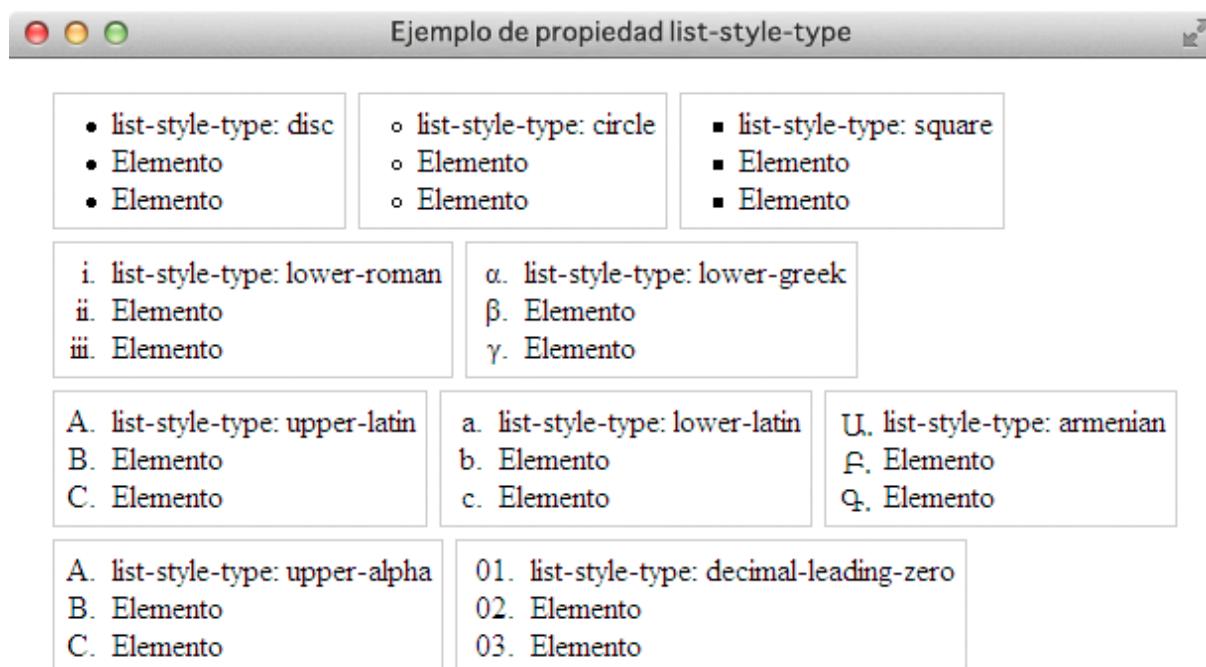
No obstante, CSS define varias propiedades para controlar el tipo de viñeta que muestran las listas, además de poder controlar la posición de la propia viñeta. La propiedad básica es la que controla el tipo de viñeta que se muestra y que se denomina `list-style-type`.

<code>list-style-type</code>	
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`:



Mostrar un menú

Figura 8.1 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	
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:

The screenshot shows a browser window with a title bar "Ejemplo de propiedad list-style-position". Below the title bar are three standard operating system window control buttons (minimize, maximize, close). The main content area is divided into two columns by a vertical border.

Left Column (list-style-position: outside):

- **list-style-position: outside**
- Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nunc in diam. Praesent a justo. Nam odio. Quisque a libero vel massa malesuada scelerisque. Curabitur metus.
- Nunc lobortis tortor. Etiam nec nibh. In tincidunt urna ut erat. Integer velit ante, tempus ut, egestas convallis, euismod in, erat.

Right Column (list-style-position: inside):

- **list-style-position: inside**
- Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nunc in diam. Praesent a justo. Nam odio. Quisque a libero vel massa malesuada scelerisque. Curabitur metus.
- Nunc lobortis tortor. Etiam nec nibh. In tincidunt urna ut erat. Integer velit ante, tempus ut, egestas convallis, euismod in, erat.

Mostrar un menú

Figura 8.2 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.

<code>list-style-image</code>	
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 8.3 Ejemplo de propiedad `list-style-image`

Las reglas CSS correspondientes al ejemplo anterior se muestran a continuación:

```
ul.ok { list-style-image: url("imagenes/ok.png"); }
ul.flecha { list-style-image: url("imagenes/flecha.png"); }
ul.circulo { list-style-image: url("imagenes/circulo_rojo.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`.

<code>list-style</code>	
Valores	(<code>list-style-type</code> <code>list-style-position</code> <code>list-style-image</code>) <code>inherit</code>
Se aplica a	Elementos de una lista

	list-style
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; }
```

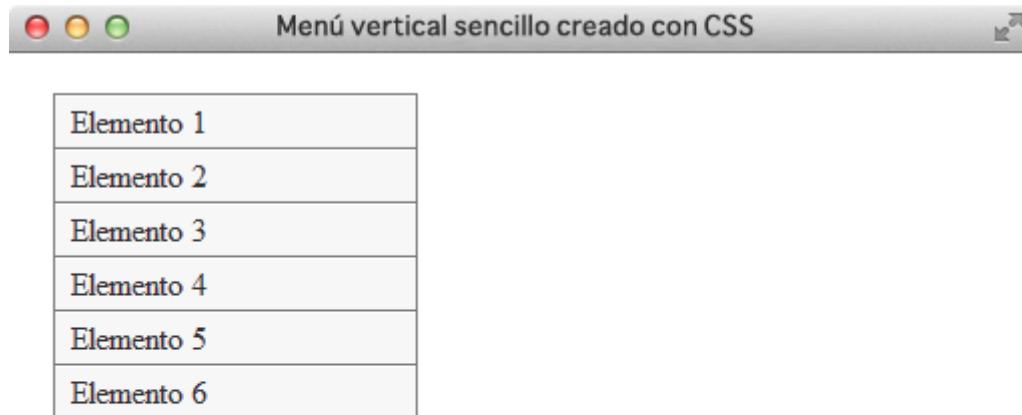
Los sitios web correctamente diseñados emplean las listas de elementos para crear todos sus menús de navegación. Utilizando la etiqueta de HTML se agrupan todas las opciones del menú y haciendo uso de CSS se modifica su aspecto para mostrar un menú horizontal o vertical.

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="#">Elemento 1</a></li>
  <li><a href="#">Elemento 2</a></li>
  <li><a href="#">Elemento 3</a></li>
  <li><a href="#">Elemento 4</a></li>
  <li><a href="#">Elemento 5</a></li>
  <li><a href="#">Elemento 6</a></li>
</ul>
```

Aspecto final del menú vertical:



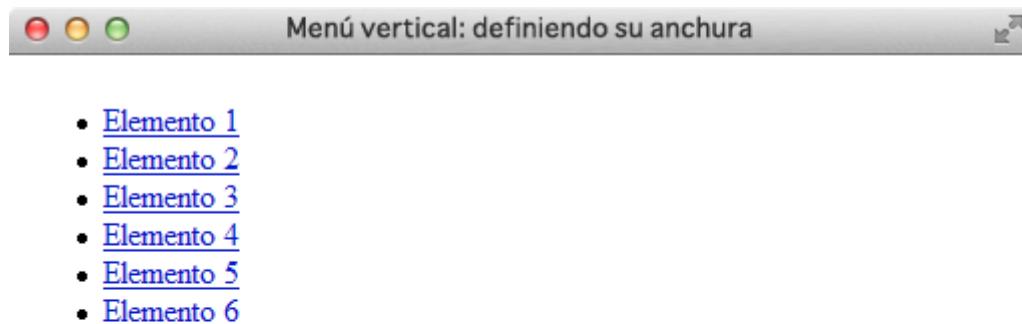
Mostrar un menú

Figura 8.4 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; }
```



Mostrar un menú

Figura 8.5 Menú vertical: definiendo su anchura

2) Eliminar las viñetas automáticas y todos los márgenes y espaciados aplicados por defecto:

```
ul.menu {  
    list-style: none;  
    margin: 0;  
    padding: 0;  
    width: 180px;  
}
```

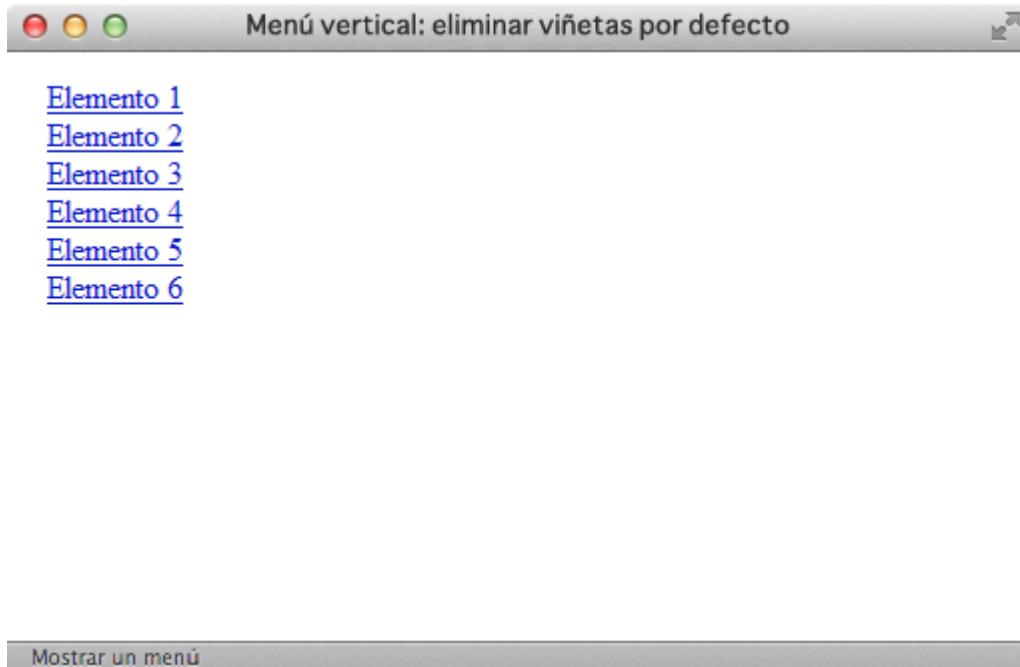
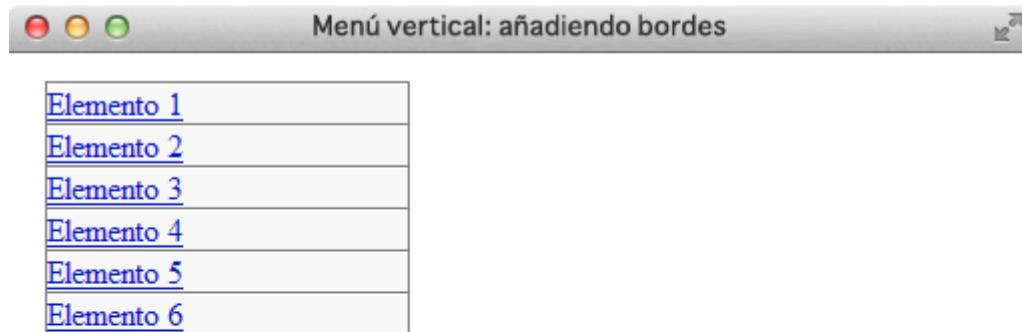


Figura 8.6 Menú vertical: eliminar viñetas por defecto

3) Añadir un borde al menú de navegación y establecer el color de fondo y los bordes de cada elemento del menú:

```
ul.menu {  
    border: 1px solid #7C7C7C;  
    border-bottom: none;  
    list-style: none;  
    margin: 0;  
    padding: 0;  
    width: 180px;  
}  
ul.menu li {  
    background: #F4F4F4;  
    border-bottom: 1px solid #7C7C7C;  
    border-top: 1px solid #FFF;  
}
```



Mostrar un menú

Figura 8.7 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 {  
    color: #333;  
    display: block;  
    padding: .2em 0 .2em .5em;  
    text-decoration: none;  
}
```

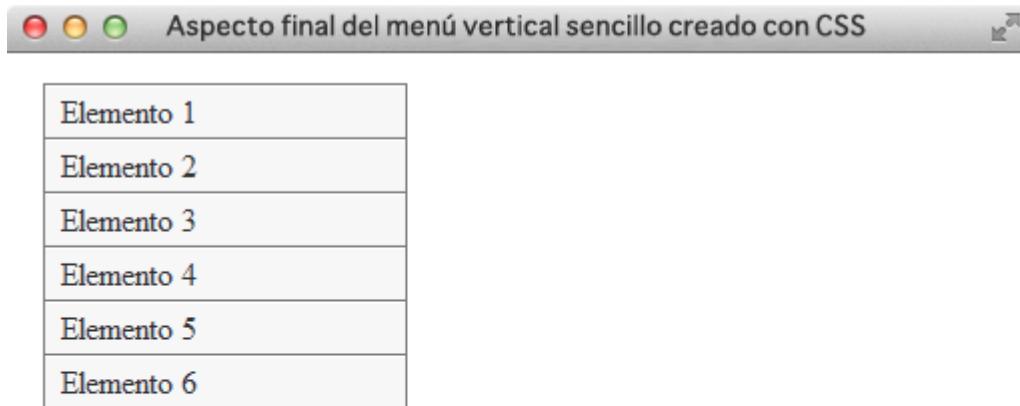


Figura 8.8 Aspecto final del menú vertical sencillo creado con CSS

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>.

Partiendo de la misma lista de elementos del menú vertical, resulta muy sencillo crear un menú de navegación horizontal. La clave reside en modificar el posicionamiento original de los elementos de la lista.

Código HTML del menú horizontal:

```
<ul>
  <li><a href="#">Elemento 1</a></li>
  <li><a href="#">Elemento 2</a></li>
  <li><a href="#">Elemento 3</a></li>
  <li><a href="#">Elemento 4</a></li>
  <li><a href="#">Elemento 5</a></li>
</ul>
```

Aspecto final del menú horizontal:

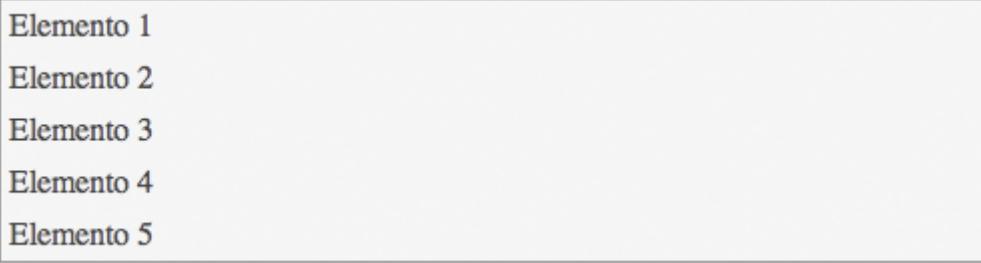


Figura 8.9 Menú horizontal creado con CSS

El proceso de creación del menú horizontal consta de los siguientes pasos:

1) Aplicar los estilos CSS básicos para establecer el estilo del menú (similares a los del menú vertical anterior):

```
ul.menu {  
    background: #F4F4F4;  
    border: 1px solid #7C7C7C;  
    list-style: none;  
    margin: 0;  
    padding: 0;  
}  
  
ul.menu li a {  
    color: #333;  
    display: block;  
    padding: .3em;  
    text-decoration: none;  
}
```



Elemento 1
Elemento 2
Elemento 3
Elemento 4
Elemento 5

Figura 8.10 Menú horizontal: estilo básico inicial

2) Establecer la anchura de los elementos del menú. Como el menú es de anchura variable y contiene cinco elementos, se asigna una anchura del 20% a cada elemento. Si se quiere tener un control más preciso sobre el aspecto de cada elemento, es necesario asignar una anchura fija al menú.

Además, se posiciona de forma flotante los elementos de la lista mediante la propiedad `float`. Esta es la clave de la transformación de una lista en un menú horizontal:

```
ul.menu li {  
    float: left;  
    width: 20%;  
}
```

Después de posicionar de forma flotante a todos los elementos de la lista, el elemento es un elemento vacío ya que en su interior no existe ningún elemento posicionado de forma normal.

Como ya se explicó en las secciones anteriores, la solución de este problema consiste en aplicar la propiedad overflow: hidden; al elemento , de forma que encierre a todos los elementos posicionados de forma flotante:

```
ul.menu {  
    overflow: hidden;  
}
```

Elemento 1	Elemento 2	Elemento 3	Elemento 4	Elemento 5
------------	------------	------------	------------	------------

Figura 8.11 Menú horizontal: anchura y posicionamiento flotante

3) Establecer los bordes de los elementos que forman el menú:

```
ul.menu li a {  
    border-left: 1px solid #FFF;  
    border-right: 1px solid #7C7C7C;  
}
```

Elemento 1	Elemento 2	Elemento 3	Elemento 4	Elemento 5
------------	------------	------------	------------	------------

Figura 8.12 Menú horizontal: borde de los elementos

4) Por último, se elimina el borde derecho del último elemento de la lista, para evitar el borde duplicado:

```
<ul>  
    <li><a href="#">Elemento 1</a></li>  
    <li><a href="#">Elemento 2</a></li>  
    <li><a href="#">Elemento 3</a></li>  
    <li><a href="#">Elemento 4</a></li>  
    <li><a href="#" style="border-right: none">Elemento 5</a></li>  
</ul>
```

Elemento 1	Elemento 2	Elemento 3	Elemento 4	Elemento 5
------------	------------	------------	------------	------------

Figura 8.13 Aspecto final del menú horizontal creado con CSS

El código CSS final se muestra a continuación:

```

ul.menu {
    background: #F4F4F4;
    border: 1px solid #7C7C7C;
    list-style: none;
    margin: 0;
    padding: 0;
    overflow: hidden;
}

ul.menu li {
    float: left;
    width: 20%;
}

ul.menu li a {
    border-left: 1px solid #FFF;
    border-right: 1px solid #7C7C7C;
    color: #333;
    display: block;
    padding: .3em;
    text-decoration: none;
}

```

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:

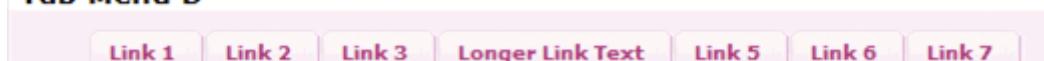
Tab Menu**Tab Menu B****Tab Menu C****Tab Menu D****Tab Menu E**

Figura 8.14 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>

Además de los menús horizontales de pestañas, haciendo uso de las propiedades de CSS se pueden crear menús verticales y horizontales muy avanzados:

CSS Navigation Techniques (37 entries)..< />

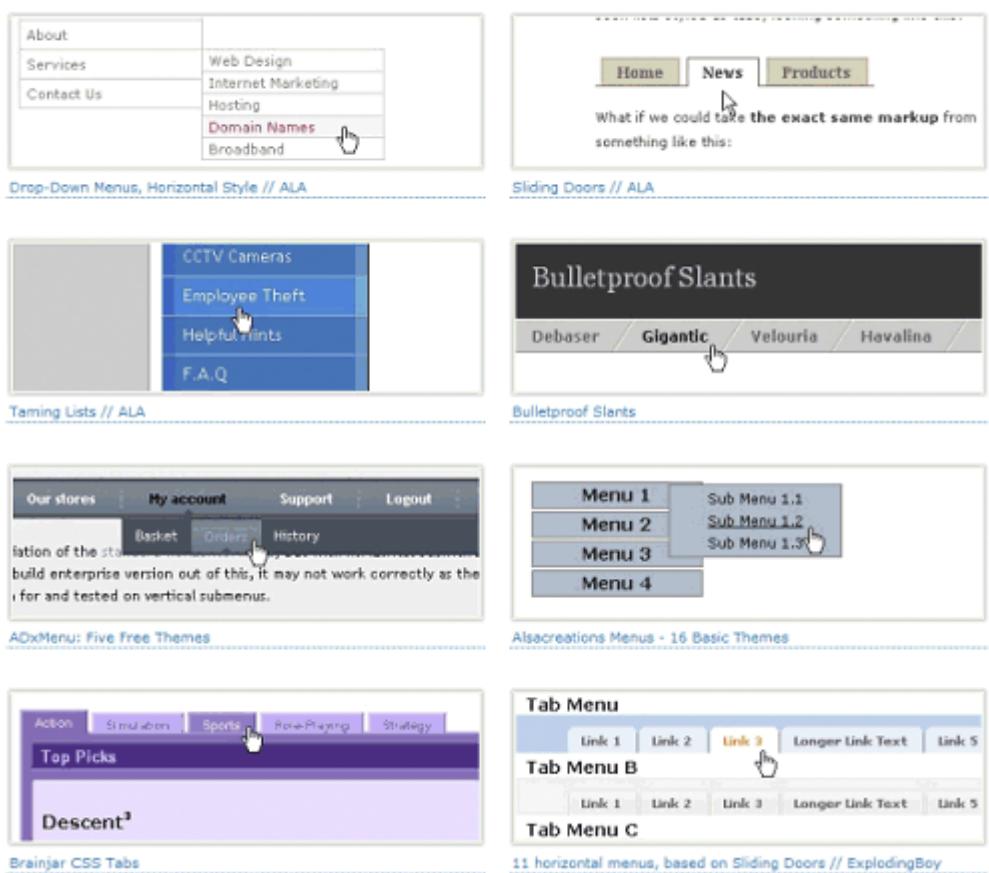


Figura 8.15 Ejemplos de menús horizontales y verticales complejos creados con CSS

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>

Ejercicio 9

[Ver enunciado \(#ej09\)](#)

Capítulo 9

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.

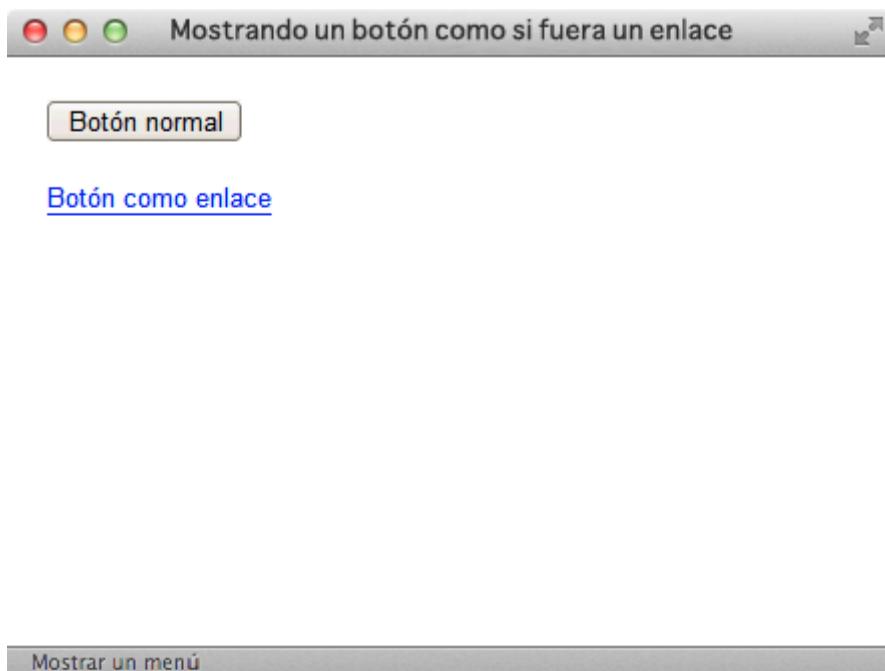


Figura 9.1 Mostrando un botón como si fuera un enlace

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" />
```

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:

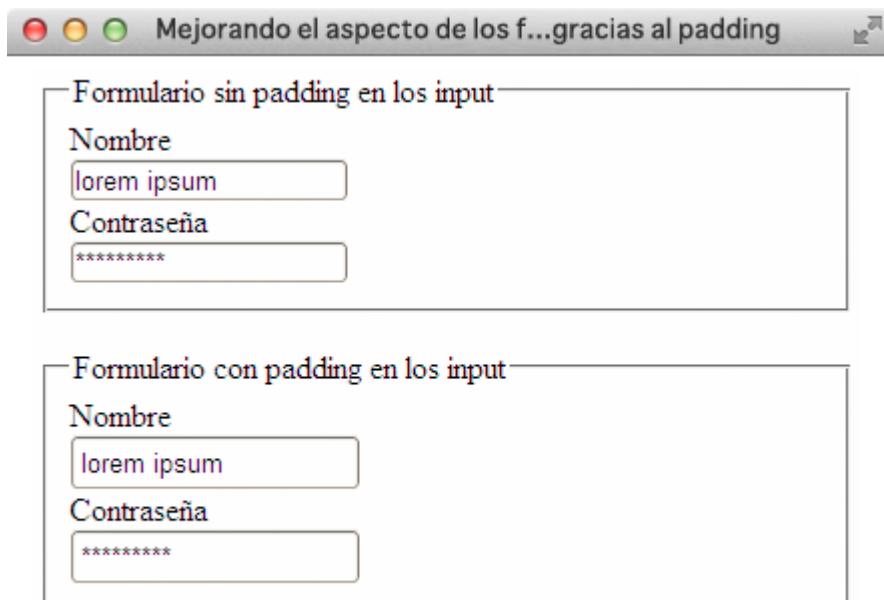


Figura 9.2 Mejorando el aspecto de los formularios gracias al padding

La regla CSS necesaria para mejorar el formulario es muy sencilla:

```
form.elegante input {  
    padding: .2em;  
}
```

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:

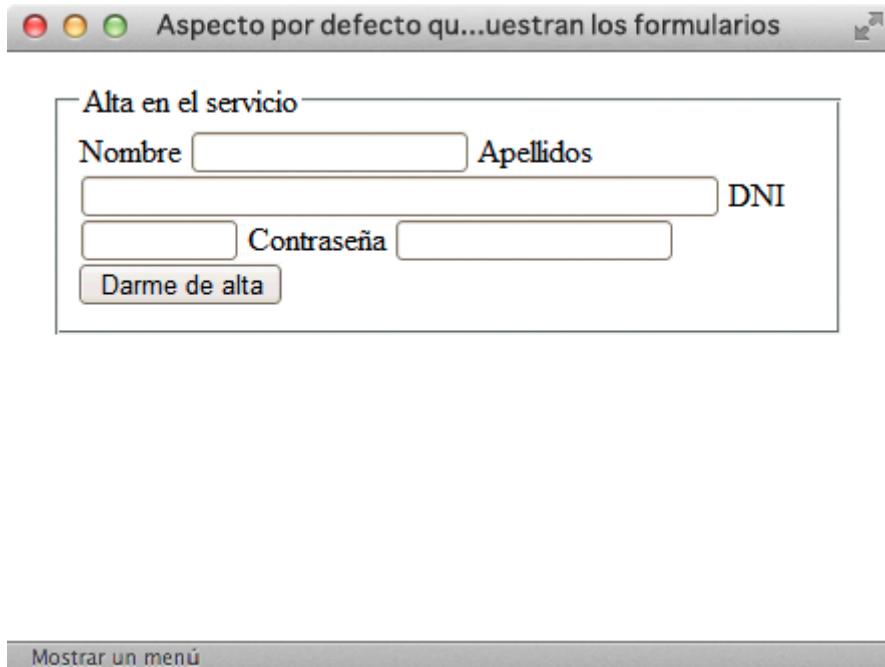


Figura 9.3 Aspecto por defecto que muestran los formularios

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="Darme 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 9.4 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:

The screenshot shows a registration form titled "Alta en el servicio". It contains four input fields: "Nombre", "Apellidos", "DNI", and "Contraseña", each with a corresponding text input box. Below these fields is a button labeled "Darme de alta". The browser window has a title bar with three colored buttons (red, yellow, green) and the text "Mostrando las etiquetas...campos del formulario".

Mostrar un menú

Figura 9.5 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 (por ejemplo un `<div>`) que encierre a cada uno de los campos del formulario (`<label>` y `<input>`). El esquema de la solución propuesta es el siguiente:

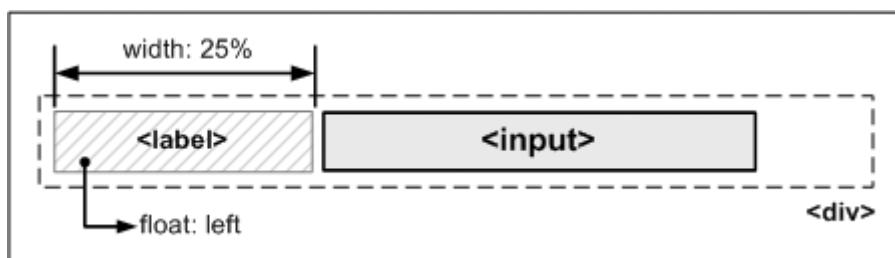


Figura 9.6 Esquema de la técnica de alineación de etiquetas label y campos de formulario

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;
}
```

Los formularios complejos con decenas de campos pueden ocupar mucho espacio en la ventana del navegador. Además del uso de pestañas 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 web browser window with a title bar 'Ejemplo de formulario a dos columnas'. Below the title bar are two columns of form fields. The left column is titled 'Alta en el servicio' and includes fields for 'Nombre' (with an input box), 'Apellidos' (with an input box), 'DNI' (with an input box), and 'Contraseña' (with an input box). The right column is titled 'Datos de contacto' and includes fields for 'Telefono' (with an input box), 'Email' (with an input box), 'Dirección' (with an input box), and 'Código Postal' (with an input box). At the bottom of the form is a blue 'Dar de alta' button.

Figura 9.7 Ejemplo de formulario a dos columnas

La solución consiste en aplicar la siguiente regla CSS a los <fieldset> del formulario:

```
form fieldset {
    float: left;
    width: 48%;
}

<form>
    <fieldset>
        ...
    </fieldset>

    ...
</form>
```

Si se quiere mostrar el formulario con más de dos columnas, se aplica la misma regla pero modificando el valor de la propiedad width de cada <fieldset>. Si el formulario es muy complejo, puede ser útil agrupar los <fieldset> de cada fila mediante elementos <div>.

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 per-

mite 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:

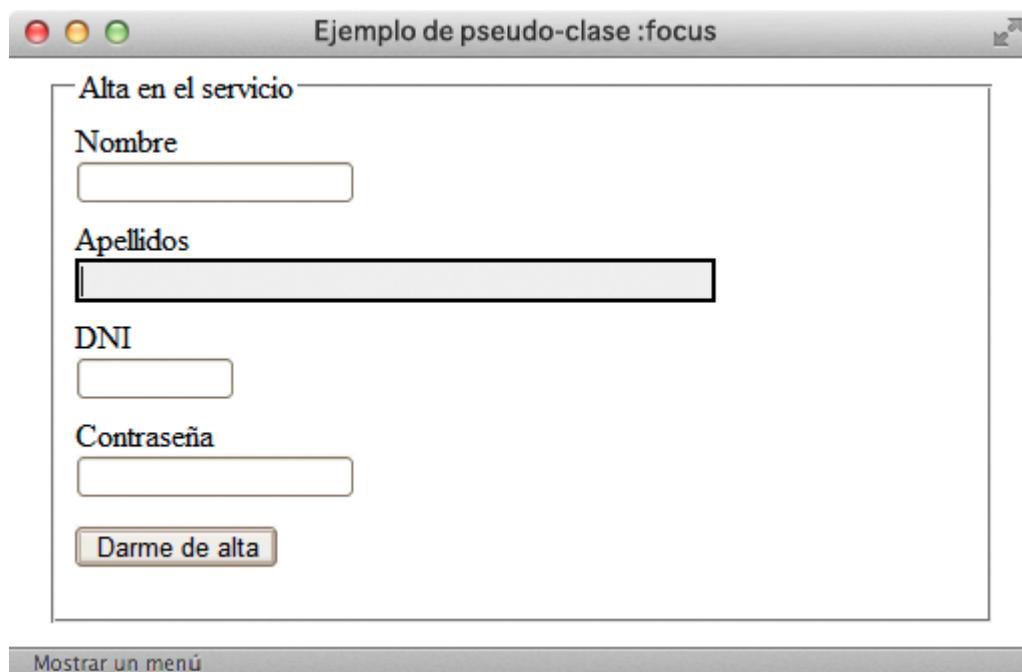


Figura 9.8 Ejemplo de pseudo-clase :focus

Añadiendo la pseudo-clase :focus después del selector normal, el navegador se encarga de aplicar esos estilos cuando el usuario activa el elemento:

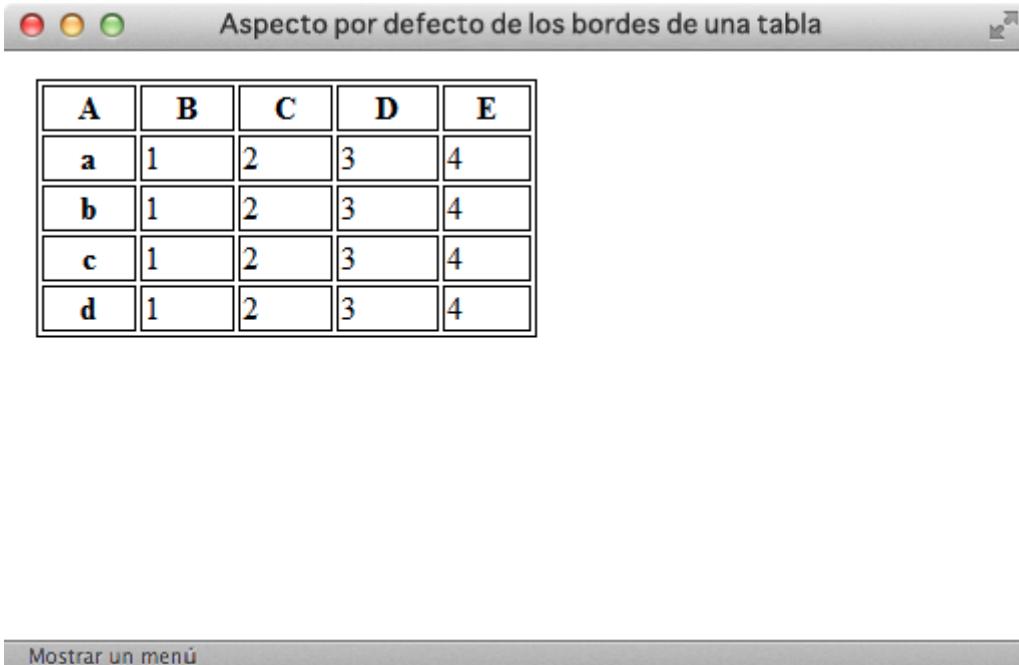
```
input:focus {  
    border: 2px solid #000;  
    background: #F3F3F3;  
}
```

Ejercicio 10

[Ver enunciado \(#ej10\)](#)

Capítulo 10

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 web browser window with a title bar that reads "Aspecto por defecto de los bordes de una tabla". Below the title bar is a toolbar with three colored buttons (red, yellow, green). The main content area displays a 5x5 grid table. 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: row 1 has values 1, 2, 3, 4, 5; row 2 has values 1, 2, 3, 4, 5; row 3 has values 1, 2, 3, 4, 5; and row 4 has values 1, 2, 3, 4, 5. All cells have a thin black border. The browser interface includes a menu bar at the bottom with a "Mostrar un menú" option.

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;
}



| A | B | C | D | E |
|---|---|---|---|---|
|   |   |   |   |   |
|   |   |   |   |   |
|   |   |   |   |   |
|   |   |   |   |   |


```

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	
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:

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

Mostrar un menú

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 es-

tilo 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 que se aplica 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	
Valores	unidad de medida inherit
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:

The screenshot shows a web browser window with a title bar that reads "Ejemplo de propiedad border-spacing". Below the title bar is a toolbar with three icons: red, yellow, and green. The main content area displays a table with 5 columns and 4 rows. The columns are labeled A, B, C, D, and E. The rows are labeled a, b, c, and d. Each cell contains a blue number (1, 2, 3, or 4). The table has a border and the cells are separated by a horizontal border-spacing of 10 pixels and a vertical border-spacing of 10 pixels, creating a clear gap between adjacent cells. At the bottom of the browser window, there is a menu bar with a single item: "Mostrar un menú".

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 10.3 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.

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. Esta propiedad sólo se aplica cuando el modelo de bordes de la tabla es de tipo separate.

empty-cells	
Valores	show hide inherit
Se aplica a	Celdas de una tabla
Valor inicial	show
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 .

La siguiente imagen muestra las diferencias entre una tabla normal y una tabla con la propiedad empty-cells: hide:

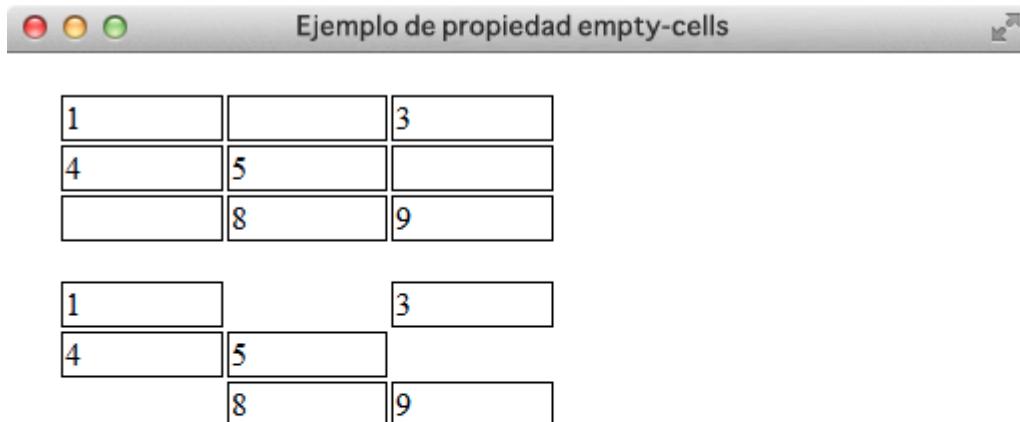


Figura 10.4 Ejemplo de propiedad empty-cells

Desafortunadamente, Internet Explorer 6 y las versiones anteriores no interpretan correctamente esta propiedad y muestran el ejemplo anterior de la siguiente manera:

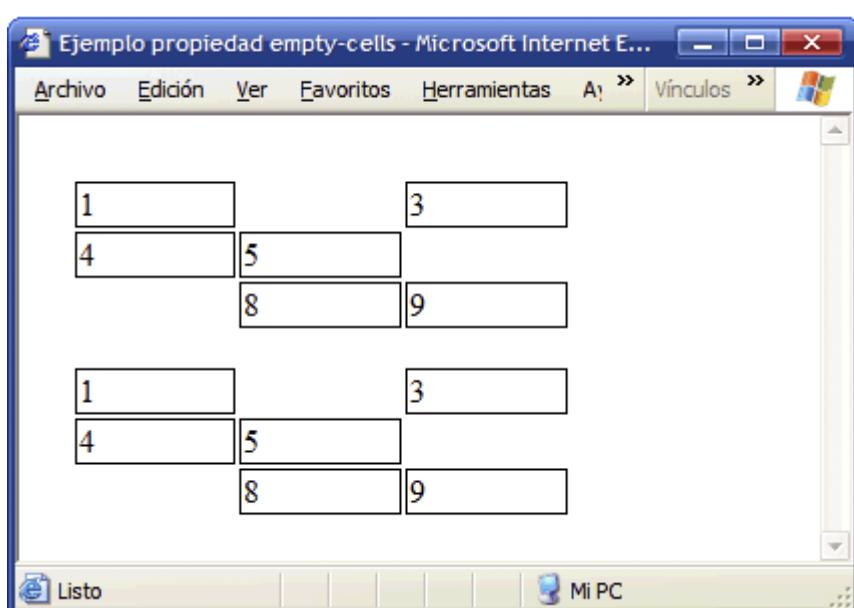


Figura 10.5 Internet Explorer no soporta la propiedad empty-cells

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
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 debajo de los contenidos de la tabla. Si también se quiere modificar la alineación horizontal del título, debe utilizarse 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>
```

La siguiente imagen muestra el resultado de visualizar el ejemplo anterior en cualquier navegador:

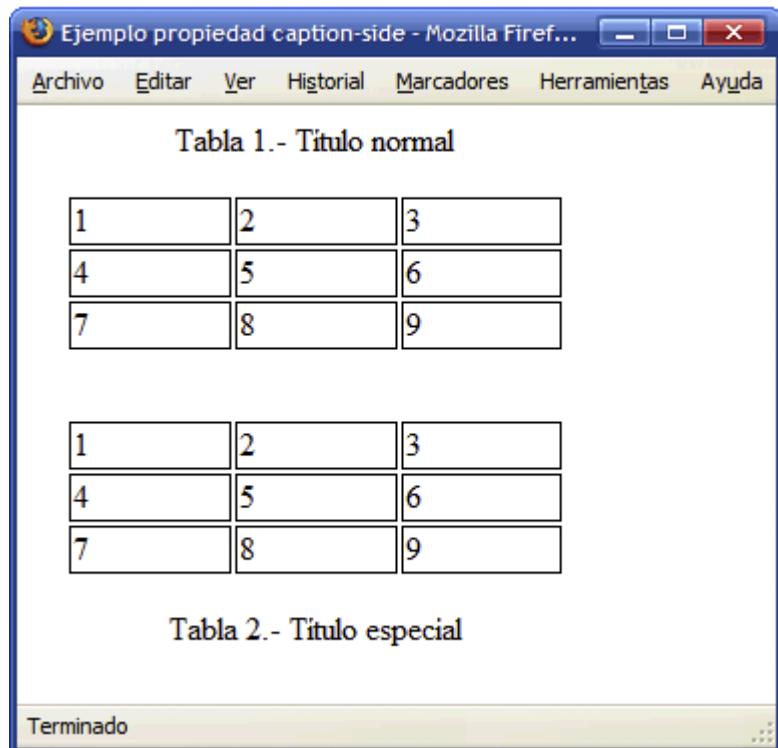


Figura 10.6 Ejemplo de propiedad caption-side

Ejercicio 11

[Ver enunciado \(#ej11\)](#)

Capítulo 11

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 mucho mayor que el código de la misma página diseñada

exclusivamente con CSS, por lo que tarda más tiempo en descargarse. 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.

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 11.1 Página de anchura fija centrada mediante CSS



Figura 11.2 Página de anchura fija centrada mediante CSS



Figura 11.3 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 (también llamado fluido) 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 11.4 Página de anchura variable centrada mediante CSS



Figura 11.5 Página de anchura variable centrada mediante CSS



Figura 11.6 Página de anchura variable centrada mediante CSS

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:

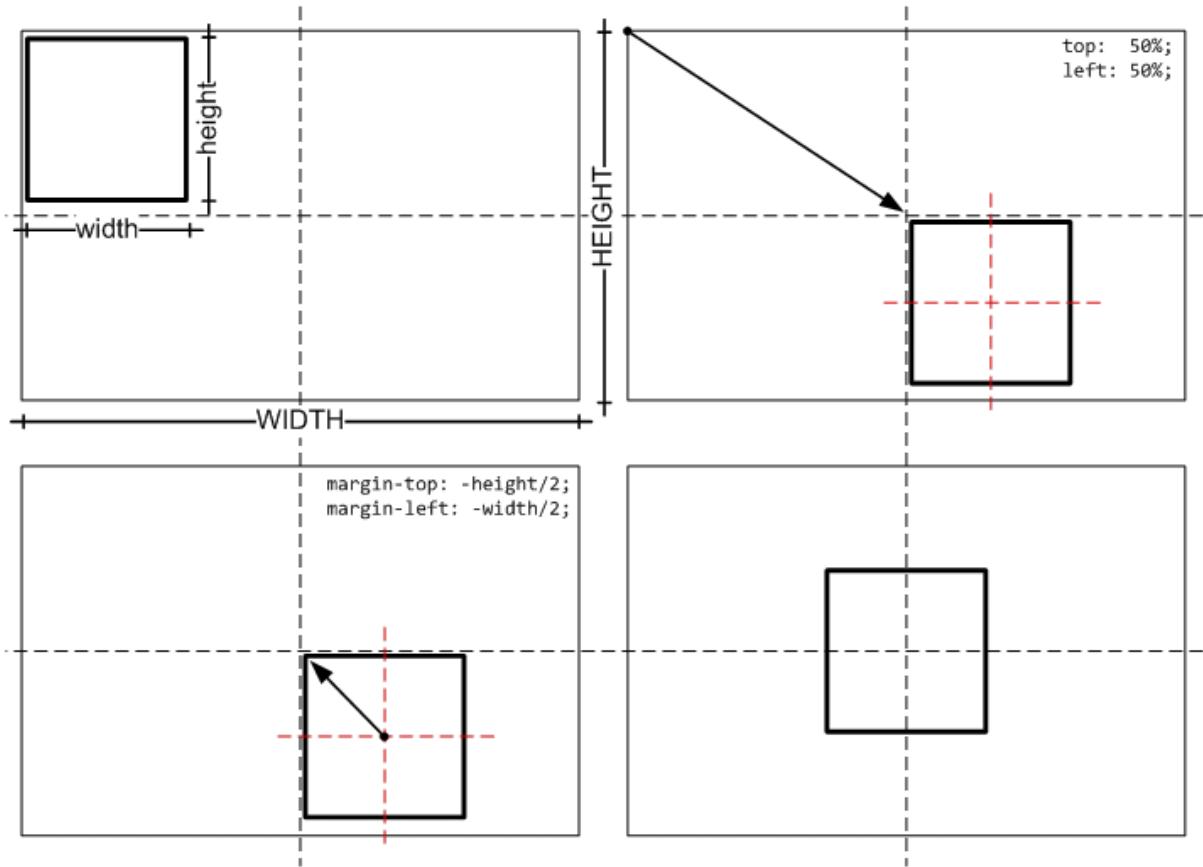


Figura 11.7 Pasos necesarios para centrar verticalmente una página web

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 */  
}
```

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:

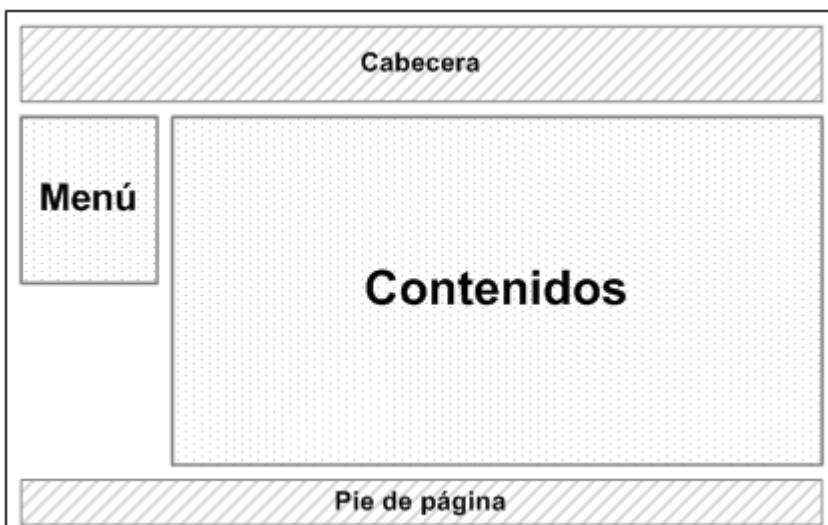


Figura 11.8 Esquema del diseño a 2 columnas con cabecera y pie de página

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`.

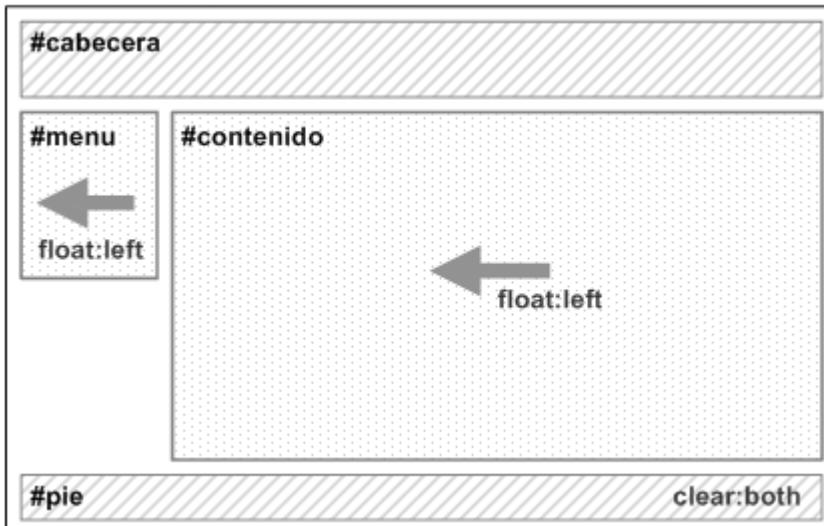
#contenedor

Figura 11.9 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.

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:

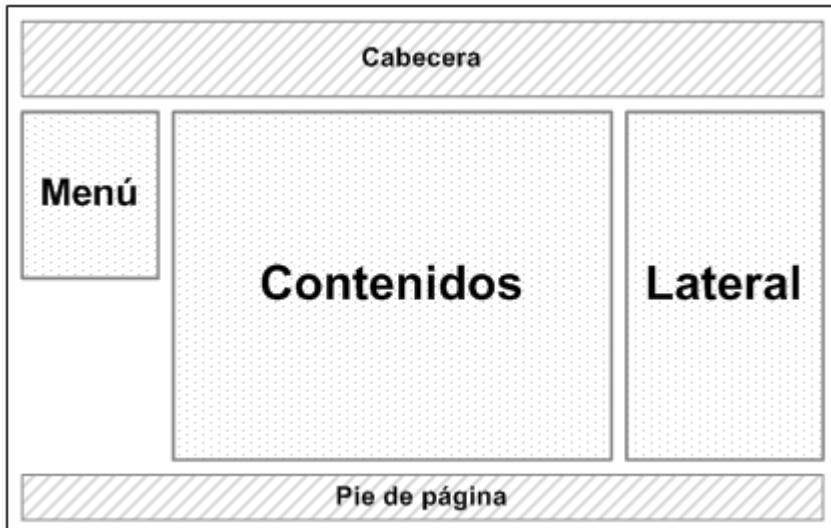


Figura 11.10 Esquema del diseño a tres columnas con cabecera y pie de página

La solución CSS emplea la misma estrategia del diseño a dos columnas y se basa en utilizar las propiedades `float` y `clear`:

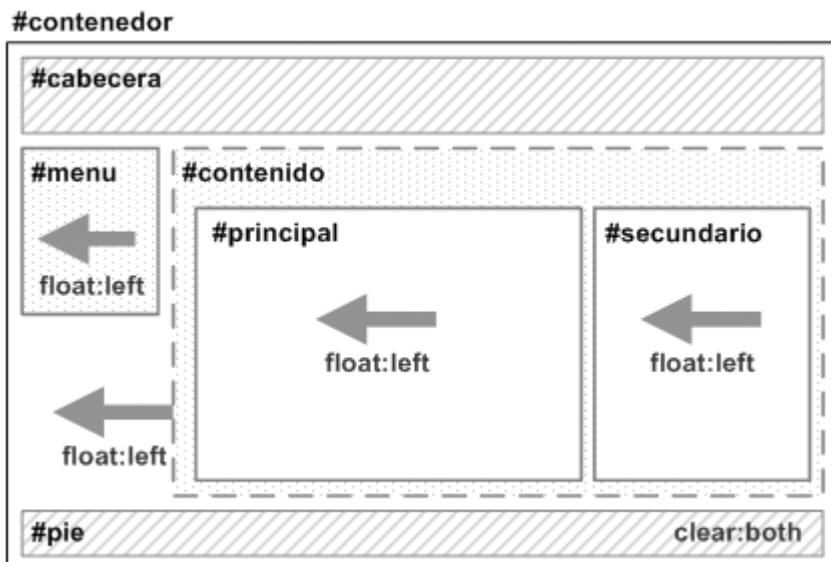


Figura 11.11 Propiedades CSS necesarias en el diseño a 3 columnas con cabecera y pie de página

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>
    </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, so-

lamente 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.

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`.

<code>max-width</code>	
Valores	unidad de medida porcentaje none inherit
Se aplica a	Todos los elementos salvo filas y grupos de filas de tablas
Valor inicial	none
Descripción	Permite definir la anchura máxima de un elemento

<code>min-width</code>	
Valores	unidad de medida porcentaje inherit
Se aplica a	Todos los elementos salvo filas y grupos de filas de tablas

	<code>min-width</code>
Valor inicial	0
Descripción	Permite definir la anchura mínima de un elemento
	<code>max-height</code>
Valores	unidad de medida porcentaje none inherit
Se aplica a	Todos los elementos salvo columnas y grupos de columnas de tablas
Valor inicial	none
Descripción	Permite definir la altura máxima de un elemento
	<code>min-height</code>
Valores	unidad de 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/

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:

One True Layout™ - Interactive Example

The screenshot shows a web-based layout editor. At the top, it says "One True Layout™ - Interactive Example". Below that, there are three columns labeled "Block 2", "Block 1", and "Block 3". Each column contains several lines of placeholder text starting with "Filler". In "Block 2", the last line is "Last filler". In "Block 1", there are 15 lines of "Filler" text. In "Block 3", there are 10 lines of "Filler" text, with the last one being "Last filler". At the bottom left, there's a "Footer text" section. At the very bottom, it says "This example forms part of the Position is Everything article, [In Search of the One True Layout](#)".

Display Controls

element	col 1	col 2	col 3	unit	Percent	total width	100%
width	33	34	33				
longest	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>				

If column wrapping occurs, try reducing total width to 99%.

make columns equal height
 include bottom-aligned elements
 direction:rtl

include Opera 8 fix
 include Firefox 1.0 fix

show display controls
 show HTML and CSS rules
 include image

add column remove column update display

Figura 11.12 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 12

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 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:

The screenshot shows a web page from A List Apart. At the top left is the site's logo 'A LIST apart' with the tagline 'FOR PEOPLE WHO MAKE WEBSITES'. To its right is a dark circular badge with the number '231'. The top navigation bar includes links for 'ARTICLES', 'TOPICS', 'ABOUT', 'CONTACT', 'CONTRIBUTE', and 'FEED'. Below the navigation is the date 'JANUARY 23, 2007'. The main title of the article is 'Paper Prototyping' by 'SHAWN MEDERO'. A small note below the title says 'Published in: Information Architecture, Layout, User Interface Design | Discuss this article >'. The main content discusses the benefits of paper prototyping over digital prototypes. To the right of the text is a cartoon illustration of a character sitting at a desk, working on a paper prototype with a pen and scissors. On the far right, there's a sidebar titled 'Search ALA' with a search input field and a 'GO' button, a checked checkbox for 'include discussions', and a 'Topics' section listing categories like 'Code', 'Content', 'Culture', 'Design', 'Process', and 'User Science'. Below that is a 'Snapshot' section with a short summary of the article's content.

Figura 12.1 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:

This screenshot shows the same article from A List Apart as it would appear when printed. The layout is much simpler and cleaner. It includes the date 'JANUARY 23, 2007' at the top left, the article title 'Paper Prototyping' by 'SHAWN MEDERO' in the center, and the date 'No. 23' at the top right. Below the title is the publication information 'Published in: Information Architecture, Layout, User Interface Design'. The main text of the article is present, followed by the 'Everyone loves paper' sidebar and the cartoon illustration. The sidebar topics are listed on the right side of the page.

Figura 12.2 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.

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) ")";  
}
```

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, ya que gte se interpreta como "greater than or equal" ("igual o mayor que"). Otros valores disponibles son gt ("greater than" o "mayor que"), lt ("less than" o "menor que") y lte ("less than or equal" o "igual o menor que").

```
<!--[if gt IE 7]>
  Mayor que Internet Explorer 7
<![endif]-->

<!--[if gte IE 7]>
  Mayor o igual que Internet Explorer 7
<![endif]-->

<!--[if lt IE 8]>
  Menor que Internet Explorer 8
<![endif]-->

<!--[if lte IE 7]>
  Igual o menor que Internet Explorer 7
<![endif]-->
```

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.

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>

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.)

Además de la hoja de estilos del navegador, cada usuario puede crear su propia hoja de estilos y aplicarla automáticamente a todas las páginas que visite con su navegador. Se trata de una opción muy útil para personas discapacitadas visualmente, ya que pueden aumentar el contraste y el tamaño del texto de todas las páginas para facilitar su lectura.

La forma en la que se indica la hoja de estilo del usuario es diferente en cada navegador. A continuación se muestra cómo se hace en los navegadores más populares:

Internet Explorer

1. Pincha sobre el menú Herramientas y después sobre la opción Opciones de Internet
2. En la pestaña General que se muestra, pulsa sobre el botón Accesibilidad que se encuentra dentro de la sección Apariencia

3. En la nueva ventana que aparece, activa la opción Formatear los documentos con mi hoja de estilos y selecciónala pulsando sobre el botón Examinar...
4. Pulsa Aceptar hasta volver al navegador

Firefox

1. Guarda tu hoja de estilos en un archivo llamado userContent.css
2. Entra en el directorio de tu perfil de usuario de Firefox. En los sistemas operativos Windows este directorio se encuentra normalmente en C:\Documents and Settings\[tu_usuario_de_windows]\Datos de programa\Mozilla\Firefox\Profiles\[cadena_aleatoria_de_letras_y_numeros].default
3. Copia la hoja de estilos userContent.css en el directorio chrome de tu perfil
4. Reinicia el navegador para que se apliquen los cambios

Safari

1. Pincha sobre el menú Editar y después sobre la opción Preferencias
2. Selecciona la sección Avanzado
3. Pincha sobre la lista desplegable llamada Hoja de estilos y selecciona la opción Otra...
4. En la ventana que aparece, selecciona tu hoja de estilos

Opera

1. Pincha sobre el menú Herramientas y después sobre la opción Preferencias
2. Selecciona la pestaña Avanzado y pulsa sobre el botón Opciones de estilo...
3. Pulsa sobre el botón Seleccionar... para seleccionar la hoja de estilos
4. Pulsa Aceptar hasta volver al navegador

El orden normal en el que se aplican las hojas de estilo es el siguiente:



Figura 12.3 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 {  
    color: red !important;  
    color: blue;  
}
```

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 12.4 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.

Si se aplica el valor !important a una propiedad de tipo "shorthand", se interpreta como si se hubiera aplicado el valor !important a cada una de las propiedades individuales.

Capítulo 13

En las próximas secciones se muestran las siguientes técnicas imprescindibles:

- Propiedades shorthand para crear hojas de estilos concisas.
- Limpiar floats, para trabajar correctamente con los elementos posicionados de forma flotante.
- Cómo crear elementos de la misma altura, imprescindible para el layout o estructura de las páginas.
- Rollovers y sprites CSS para mejorar el tiempo de respuesta de las páginas.

Algunas propiedades del estándar CSS 2.1 son especiales, ya que permiten establecer simultáneamente el valor de varias propiedades diferentes. Este tipo de propiedades se denominan "*propiedades shorthand*" y todos los diseñadores web profesionales las utilizan.

La gran ventaja de las *propiedades shorthand* es que permiten crear hojas de estilos mucho más concisas y por tanto, mucho más fáciles de leer. A continuación se incluye a modo de referencia la definición formal de las seis propiedades shorthand disponibles en el estándar CSS 2.1.

Si se considera la siguiente hoja de estilos:

```
p {  
    font-style: normal;  
    font-variant: small-caps;  
    font-weight: bold;  
    font-size: 1.5em;  
    line-height: 1.5;  
    font-family: Arial, sans-serif;  
}  
  
div {  
    margin-top: 5px;  
    margin-right: 10px;  
    margin-bottom: 5px;  
    margin-left: 10px;  
    padding-top: 3px;  
    padding-right: 5px;  
    padding-bottom: 10px;  
    padding-left: 7px;  
}  
  
h1 {  
    background-color: #FFFFFF;  
    background-image: url("imagenes/icono.png");  
    background-repeat: no-repeat;  
    background-position: 10px 5px;  
}
```

Utilizando las propiedades shorthand es posible convertir las 24 líneas que ocupa la hoja de estilos anterior en sólo 10 líneas, manteniendo los mismos estilos:

```
p {  
    font: normal small-caps bold 1.5em/1.5 Arial, sans-serif;  
}  
  
div {  
    margin: 5px 10px;  
    padding: 3px 5px 10px 7px;  
}  
  
h1 {
```

```
background: #FFF url("imagenes/icono.png") no-repeat 10px 5px;  
}
```

La principal característica de los elementos posicionados de forma flotante mediante la propiedad `float` es que desaparecen del flujo normal del documento. De esta forma, es posible que algunos o todos los elementos flotantes se salgan de su elemento contenedor.

La siguiente imagen muestra un elemento contenedor que encierra a dos elementos de texto. Como los elementos interiores están posicionados de forma flotante y el elemento contenedor no dispone de más contenidos, el resultado es el siguiente:

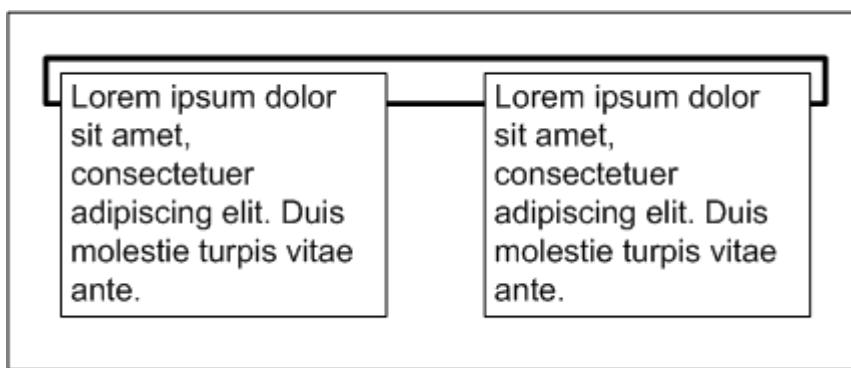


Figura 13.1 Los elementos posicionados de forma flotante se salen de su elemento contenedor

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

```
<div id="contenedor">  
  <div id="izquierda">Lorem ipsum dolor sit amet, consectetuer  
    adipiscing elit. Duis molestie turpis vitae ante.</div>  
  <div id="derecha">Lorem ipsum dolor sit amet, consectetuer  
    adipiscing elit. Nulla bibendum mi non lacus.</div>  
</div>  
  
#contenedor {  
  border: thick solid #000;  
}  
  
#izquierda {  
  float: left;  
  width: 40%;  
}
```

```
#derecha {  
    float: right;  
    width: 40%;  
}
```

La solución tradicional de este problema consiste en añadir un elemento invisible después de todos los elementos posicionados de forma flotante para forzar a que el elemento contenedor tenga la altura suficiente. Los elementos invisibles más utilizados son `<div>`, `
` y `<p>`.

De esta forma, si se añade un elemento `<div>` invisible con la propiedad `clear` de CSS en el ejemplo anterior:

```
<div id="contenedor">  
    <div id="izquierda">Lorem ipsum dolor sit amet, consectetuer  
    adipiscing elit. Duis molestie turpis vitae ante.</div>  
    <div id="derecha">Lorem ipsum dolor sit amet, consectetuer  
    adipiscing elit. Nulla bibendum mi non lacus.</div>  
  
    <div style="clear: both"></div>  
</div>
```

Ahora el elemento contenedor se visualiza correctamente porque encierra a todos sus elementos:

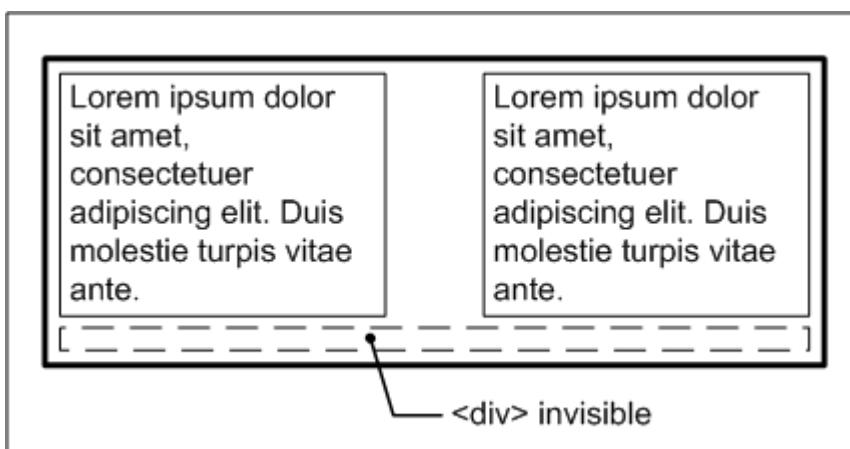


Figura 13.2 Solución tradicional al problema de los elementos posicionados de forma flotante

La técnica de corregir los problemas ocasionados por los elementos posicionados de forma flotante se suele denominar "*limpiar los float*".

Aunque añadir un elemento invisible corrige correctamente el problema, se trata de una solución poco elegante e incorrecta desde el punto de

vista semántico. No tiene ningún sentido añadir un elemento vacío en el código HTML, sobre todo si ese elemento se utiliza exclusivamente para corregir el aspecto de los contenidos.

Afortunadamente, existe una solución alternativa para *limpiar los float* que no obliga a añadir nuevos elementos HTML y que además es elegante y muy sencilla. La solución consiste en utilizar la propiedad `overflow` de CSS sobre el elemento contenedor.

Si se modifica el código CSS anterior y se incluye la siguiente regla:

```
#contenedor {  
    border: thick solid #000;  
    overflow: hidden;  
}
```

Ahora, el contenedor encierra correctamente a los dos elementos `<div>` interiores y no es necesario añadir ningún elemento adicional en el código HTML. Además del valor `hidden`, también es posible utilizar el valor `auto` obteniendo el mismo resultado.

Hasta hace unos años, la estructura de las páginas web complejas se creaba mediante tablas HTML. Aunque esta solución presenta muchos inconvenientes, su principal ventaja es que todas las columnas que forman la página son de la misma altura.

Normalmente, cuando se crea la estructura de una página compleja, se desea que todas las columnas que la forman tengan la misma altura. De hecho, cuando algunas o todas las columnas tienen imágenes o colores de fondo, esta característica es imprescindible para obtener un diseño correcto.

Sin embargo, como el contenido de cada columna suele ser variable, no es posible determinar la altura de la columna más alta y por tanto, no es posible hacer que todas las columnas tengan la misma altura directamente con la propiedad `height`.

Cuando se utiliza una tabla para crear la estructura de la página, este problema no existe porque cada columna de la estructura se corresponde con una celda de datos de la tabla. Sin embargo, cuando se diseña la estructura de la página utilizando sólo CSS, el problema no es tan fácil

de solucionar. Afortunadamente, existen varias soluciones para asegurar que dos elementos tengan la misma altura.

El truco consiste en añadir un espacio de relleno inferior (`padding-bottom`) muy grande a todas las columnas y después añadirles un margen inferior negativo (`margin-bottom`) del mismo tamaño.

```
#contenedor {  
    overflow: hidden;  
}  
  
#columna1, #columna2, #columna3 {  
    padding-bottom: 32767px;  
    margin-bottom: -32767px;  
}
```

El valor utilizado en el espacio de relleno y en el margen inferior de las columnas debe ser tan grande como la altura esperada para la columna más alta. Para evitar quedarse corto, se recomienda utilizar valores a partir de 10.000 píxeles.

Los dos principales problemas que presenta esta solución son los siguientes:

- Se pueden producir errores al imprimir la página con el navegador Internet Explorer.
- Si se utilizan enlaces de tipo ancla en cualquier columna, al pulsar sobre el enlace las columnas se desplazan de forma ascendente y desaparecen de la página.

Otra solución al problema de los elementos de la misma altura es la que presentó el diseñador Dan Cederholm en su célebre artículo Faux Columns. Si la solución anterior consistía en engañar al navegador, esta segunda solución se basa en engañar al ojo del usuario.

La solución de las columnas falsas consiste en establecer una imagen de fondo repetida verticalmente en el elemento contenedor. Como el contenedor es tan alto como la columna más alta, su imagen de fondo da la sensación de que todas las columnas son de la misma altura.

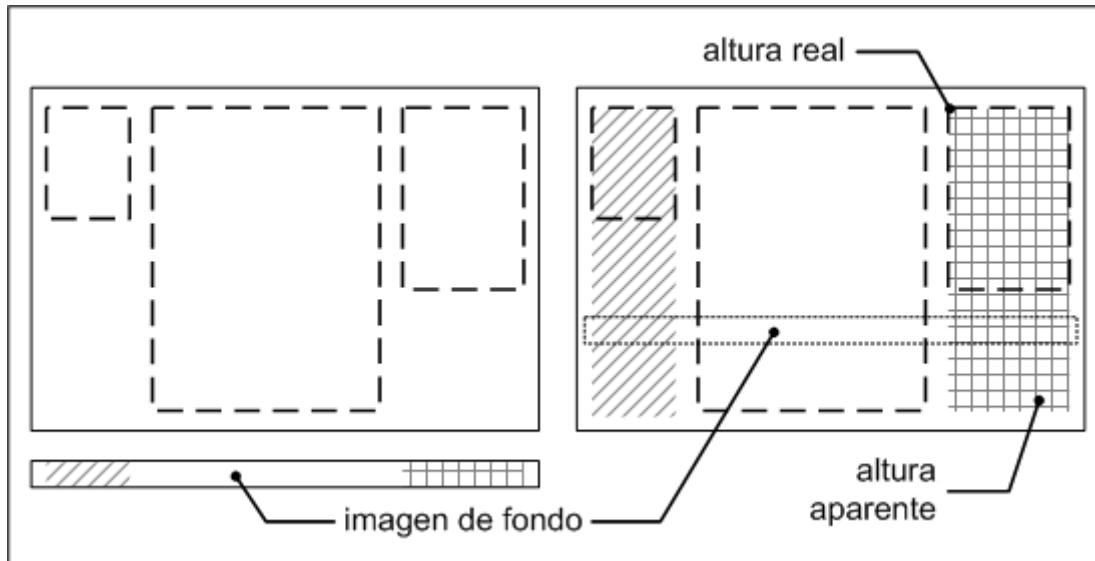


Figura 13.3 Las columnas parecen de la misma altura porque el elemento contenedor muestra una imagen de fondo repetida verticalmente

El principal inconveniente de esta técnica es que sólo se puede emplear cuando la estructura de la página es de anchura fija, es decir, cuando su diseño no es líquido y no se adapta a la anchura de la ventana del navegador.

Las dos soluciones planteadas hasta el momento consisten en trucos para engañar a los navegadores y a los usuarios. A continuación se presenta la única solución técnicamente correcta para forzar a que dos elementos muestren la misma altura.

La solución fue propuesta por el diseñador Roger Johansson en su artículo *Equal height boxes with CSS* y se basa en el uso avanzado de la propiedad `display` de CSS.

En primer lugar, es necesario añadir un elemento adicional (`<div id="contenedores">`) en el código HTML de la página:

```
<div id="contenedor">
  <div id="contenedores">
    <div id="columna1"></div>
    <div id="columna2"></div>
    <div id="columna3"></div>
  </div>
</div>
```

A continuación, se utiliza la propiedad `display` de CSS para mostrar los elementos `<div>` anteriores como si fueran celdas de una tabla de datos:

```
#contenedor {  
    display: table;  
}  
  
#contenidos {  
    display: table-row;  
}  
  
#columna1, #columna2, #columna3 {  
    display: table-cell;  
}
```

Gracias a la propiedad `display` de CSS, cualquier elemento se puede comportar como una tabla, una fila de tabla o una celda de tabla, independientemente del tipo de elemento que se trate.

De esta forma, los elementos `<div>` que forman las columnas de la página en realidad se comportan como celdas de tabla, lo que permite que el navegador las muestre con la misma altura.

Según varios estudios realizados por Yahoo!, hasta el 80% de la mejora en el rendimiento de la descarga de páginas web depende de la parte del cliente. En el artículo Performance Research, Part 1: What the 80/20 Rule Tells Us about Reducing HTTP Requests Yahoo! (<http://yuiblog.com/blog/2006/11/28/performance-research-part-1/>) explica que generar dinámicamente el código HTML de la página y servirla ocupa el 20% del tiempo total de descarga de la página. El 80% del tiempo restante los navegadores descargan las imágenes, archivos JavaScript, hojas de estilos y cualquier otro tipo de archivo enlazado.

Además, en la mayoría de páginas web *normales*, la mayor parte de ese 80% del tiempo se dedica a la descarga de las imágenes. Por tanto, aunque los mayores esfuerzos siempre se centran en reducir el tiempo de generación dinámica de las páginas, se consigue más y con menos esfuerzo mejorando la descarga de las imágenes.

La idea para mejorar el rendimiento de una página que descarga por ejemplo 15 imágenes consiste en crear una única imagen grande que incluya las 15 imágenes individuales y utilizar las propiedades CSS de las imágenes de fondo para mostrar cada imagen. Esta técnica se presentó en el artículo CSS Sprites: Image Slicing's Kiss of Death

(<http://www.alistapart.com/articles/sprites>) y desde entonces se conoce con el nombre de sprites CSS.

El siguiente ejemplo explica el uso de los *sprites* CSS en un sitio web que muestra la previsión meteorológica de varias localidades utilizando iconos:

Previsiones meteorológicas

 Localidad 1: soleado, máx: 35º, mín: 23º

 Localidad 2: nublado, máx: 25º, mín: 13º

 Localidad 3: muy nublado, máx: 22º, mín: 10º

 Localidad 4: tormentas, máx: 23º, mín: 11º

Figura 13.4 Aspecto de la previsión meteorológica mostrada con iconos

La solución tradicional para crear la página anterior consiste en utilizar cuatro elementos en el código HTML y disponer de cuatro imágenes correspondientes a los cuatro iconos:

```
<h3>Previsiones meteorológicas</h3>
<p id="localidad1">
    
    Localidad 1: soleado, máx: 35º, mín: 23º
</p>
<p id="localidad2">
    
    Localidad 2: nublado, máx: 25º, mín: 13º
</p>
<p id="localidad3">
    
    Localidad 3: muy nublado, máx: 22º, mín: 10º
</p>
<p id="localidad4">
    
    Localidad 4: tormentas, máx: 23º, mín: 11º
</p>
```

Aunque es una solución sencilla y que funciona muy bien, se trata de una solución completamente ineficiente. El navegador debe descargar cu-

tro imágenes diferentes para mostrar la página, por lo que debe realizar cuatro peticiones al servidor.

Después del tamaño de los archivos descargados, el número de peticiones realizadas al servidor es el factor que más penaliza el rendimiento en la descarga de páginas web. Utilizando la técnica de los sprites CSS se va a rehacer el ejemplo anterior para conseguir el mismo efecto con una sola imagen y por tanto, una sola petición al servidor.

El primer paso consiste en crear una imagen grande que incluya las cuatro imágenes individuales. Como los iconos son cuadrados de tamaño 32px, se crea una imagen de 32px x 128px:

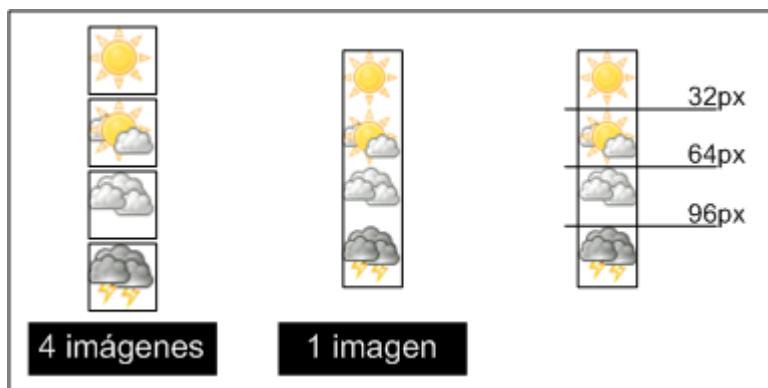


Figura 13.5 Creando un sprite de CSS a partir de varias imágenes individuales

Para facilitar el uso de esta técnica, todas las imágenes individuales ocupan el mismo sitio dentro de la imagen grande. De esta forma, los cálculos necesarios para desplazar la imagen de fondo se simplifican al máximo.

El siguiente paso consiste en simplificar el código HTML:

```
<h3>Previsiones meteorológicas</h3>
<p id="localidad1">
    Localidad 1: soleado, máx: 35º, mín: 23º
</p>
<p id="localidad2">
    Localidad 2: nublado, máx: 25º, mín: 13º
</p>
<p id="localidad3">
    Localidad 3: muy nublado, máx: 22º, mín: 10º
</p>
<p id="localidad4">
```

```
Localidad 4: tormentas, máx: 23º, mín: 11º  
</p>
```

La clave de la técnica de los sprites CSS consiste en mostrar las imágenes mediante la propiedad `background-image`. Para mostrar cada vez una imagen diferente, se utiliza la propiedad `background-position` que desplaza la imagen de fondo teniendo en cuenta la posición de cada imagen individual dentro de la imagen grande:

```
#localidad1, #localidad2, #localidad3, #localidad4 {  
    padding-left: 38px;  
    height: 32px;  
    line-height: 32px;  
    background-image: url("imagenes/sprite.png");  
    background-repeat: no-repeat;  
}  
  
#localidad1 {  
    background-position: 0 0;  
}  
#localidad2 {  
    background-position: 0 -32px;  
}  
#localidad3 {  
    background-position: 0 -64px;  
}  
#localidad4 {  
    background-position: 0 -96px;  
}
```

La siguiente imagen muestra lo que sucede con el segundo párrafo:

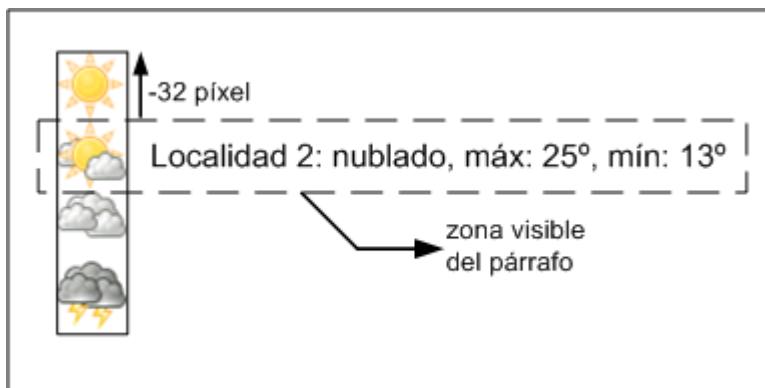


Figura 13.6 Funcionamiento de la técnica de los sprites CSS

El párrafo tiene establecida una altura de 32px, idéntica al tamaño de los iconos. Después de añadir un padding-left al párrafo, se añade la imagen de fondo mediante background-image. Para cambiar de una imagen a otra, sólo es necesario desplazar de forma ascendente o descendente la imagen grande.

Si se quiere mostrar la segunda imagen, se desplaza de forma ascendente la imagen grande. Para desplazarla en ese sentido, se utilizan valores negativos en el valor indicado en la propiedad background-position. Por último, como la imagen grande ha sido especialmente preparada, se sabe que el desplazamiento necesario son 32 píxel, por lo que la regla CSS de este segundo elemento resulta en:

```
#localidad2 {  
    padding-left: 38px;  
    height: 32px;  
    line-height: 32px;  
    background-image: url("imagenes/sprite.png");  
    background-repeat: no-repeat;  
    background-position: 0 -32px;  
}
```

La solución original utilizaba cuatro imágenes y realizaba cuatro peticiones al servidor. La solución basada en sprites CSS sólo realiza una conexión para descargar una sola imagen. Además, los iconos del proyecto Tango que se han utilizado en este ejemplo ocupan 7.441 bytes cuando se suman los tamaños de los cuatro iconos por separado. Por su parte, la imagen grande que contiene los cuatro iconos sólo ocupa 2.238 bytes.

Los *sprites* que incluyen todas sus imágenes verticalmente son los más fáciles de manejar. Si en el ejemplo anterior se emplea un sprite con las imágenes dispuestas también horizontalmente:

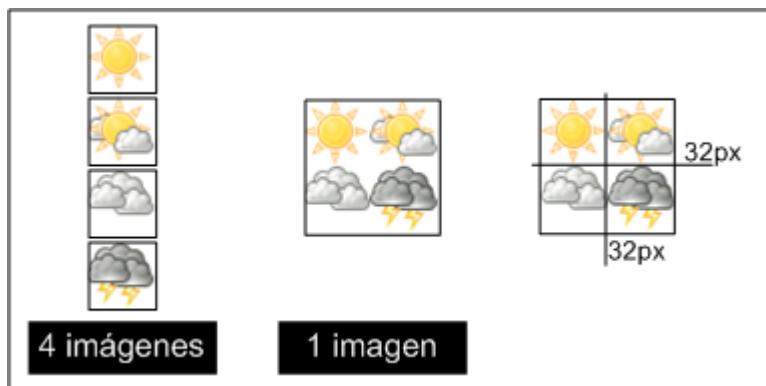


Figura 13.7 Sprite complejo que dispone las imágenes de forma vertical y horizontal

Aparentemente, utilizar este nuevo sprite sólo implica que la imagen de fondo se debe desplazar también horizontalmente:

```
#localidad1, #localidad2, #localidad3, #localidad4 {  
    padding-left: 38px;  
    height: 32px;  
    line-height: 32px;  
    background-image: url("imagenes/sprite.png");  
    background-repeat: no-repeat;  
}  
  
#localidad1 {  
    background-position: 0 0;  
}  
#localidad2 {  
    background-position: -32px 0;  
}  
#localidad3 {  
    background-position: 0 -32px;  
}  
#localidad4 {  
    background-position: -32px -32px;  
}
```

El problema del *sprite* anterior es que cuando una imagen tiene a su derecha o a su izquierda otras imágenes, estas también se ven:

Previsiones meteorológicas

 Localidad 1: soleado, máx: 35°, min: 23°

 Localidad 2: nublado, máx: 25°, min: 13°

 Localidad 3: muy nublado, máx: 22°, min: 10°

 Localidad 4: tormentas, máx: 23°, min: 11°

Figura 13.8 Errores producidos por utilizar un sprite complejo

La solución de este problema es sencilla, aunque requiere algún cambio en el código HTML:

```
<h3>Previsiones meteorológicas</h3>
<p id="localidad1">
    
    Localidad 1: soleado, máx: 35º, mín: 23º
</p>
<p id="localidad2">
    
    Localidad 2: nublado, máx: 25º, mín: 13º
</p>
<p id="localidad3">
    
    Localidad 3: muy nublado, máx: 22º, mín: 10º
</p>
<p id="localidad4">
    
    Localidad 4: tormentas, máx: 23º, mín: 11º
</p>
```

El código anterior muestra uno de los trucos habituales para manejar sprites complejos. En primer lugar se añade una imagen transparente de 1px x 1px a todos los elementos mediante una etiqueta ``. A continuación, desde CSS se establece una imagen de fondo a cada elemento `` y se limita su tamaño para que no deje ver las imágenes que se encuentran cerca:

```
#localidad1 img, #localidad2 img, #localidad3 img, #localidad4 img {
    height: 32px;
    width: 32px;
    background-image: url("imagenes/sprite2.png");
    background-repeat: no-repeat;
    vertical-align: middle;
}

#localidad1 img {
    background-position: 0 0;
}
#localidad2 img {
    background-position: -32px 0;
}
#localidad3 img {
    background-position: 0 -32px;
```

```
        }  
        #localidad4 img {  
            background-position: -32px -32px;  
        }
```

Utilizar *sprites* CSS es una de las técnicas más eficaces para mejorar los tiempos de descarga de las páginas web complejas. La siguiente imagen muestra un *sprite* complejo que incluye 241 iconos del proyecto Tango y sólo ocupa 42 KB:

Previsiones meteorológicas

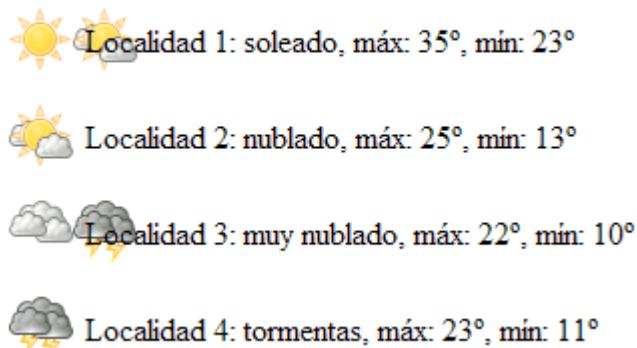


Figura 13.9 Sprite complejo que incluye 210 iconos en una sola imagen

La mayoría de sitios web profesionales utilizan *sprites* para mostrar sus imágenes de adorno. La siguiente imagen muestra un *sprite* del sitio web Flickr:



Figura 13.10 Un sprite utilizado por el sitio web Flickr

Los principales inconvenientes de los *sprites* CSS son la poca flexibilidad que ofrece (añadir o modificar una imagen individual no es inmediato) y el esfuerzo necesario para crear el *sprite*.

Afortunadamente, existen aplicaciones web como [CSS Sprite Generator](http://spritegen.website-performance.org/) (<http://spritegen.website-performance.org/>) que generan el *sprite* a partir de un archivo comprimido en formato ZIP con todas las imágenes individuales.

Conocer y dominar todos los selectores de CSS es imprescindible para crear diseños web profesionales. El estándar de CSS 2.1 incluye una docena de tipos diferentes de selectores, que permiten seleccionar de for-

ma muy precisa elementos individuales o conjuntos de elementos dentro de una página web.

Utilizando solamente los cinco selectores básicos de CSS 2.1 (universal, de tipo, descendente, de clase y de id) es posible diseñar cualquier página web. No obstante, los selectores avanzados de CSS 2.1 permiten simplificar las reglas CSS y también el código HTML.

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

Mientras que en el selector descendente sólo importa que un elemento esté dentro de otro, independientemente de lo profundo que se encuentre, en el selector de hijos el elemento debe ser *hijo directo* de otro elemento.

```
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>`.

Utilizando el mismo ejemplo anterior se pueden comparar 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 (`p a`) 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.

El segundo selector es de hijos (`p > a`) por lo que obliga a que el elemento `<a>` sea hijo directo de un elemento `<p>`. Por tanto, los estilos del selector `p > a` no se aplican al segundo enlace del ejemplo anterior.

El selector adyacente se emplea para seleccionar elementos que son hermanos (su elemento padre es el mismo) y están seguidos en el código HTML. Este selector emplea en su sintaxis el símbolo `+`. Si se considera el siguiente ejemplo:

```
h1 + h2 { color: red }

<body>
  <h1>Titulo1</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 puede ser útil para los textos que se muestran como libros:

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

En muchos libros es habitual que la primera línea de todos los párrafos esté indentada, salvo la primera línea del primer párrafo. El selector `p + p` selecciona todos los párrafos que están dentro de un mismo elemento padre y que estén precedidos por otro párrafo. En otras palabras, el selector `p + p` selecciona todos los párrafos de un elemento salvo el primer párrafo.

El selector adyacente requiere que los dos elementos sean hermanos, por lo que su elemento padre debe ser el mismo. Si se considera el siguiente ejemplo:

```
| p + p { color: red; }  
  
<p>Lorem ipsum dolor sit amet...</p>  
<p>Lorem ipsum dolor sit amet...</p>  
<div>  
  <p>Lorem ipsum dolor sit amet...</p>  
</div>
```

En el ejemplo anterior, solamente el segundo párrafo se ve de color rojo, ya que:

- El primer párrafo no va precedido de ningún otro párrafo, por lo que no cumple una de las condiciones de `p + p`
- El segundo párrafo va precedido de otro párrafo y los dos comparten el mismo parente, por lo que se cumplen las dos condiciones del selector `p + p` y el párrafo muestra su texto de color rojo.
- El tercer párrafo se encuentra dentro de un elemento `<div>`, por lo que no se cumple ninguna condición del selector `p + p` ya que ni va precedido de un párrafo ni comparte parente con ningún otro párrafo.

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 cuyo valor es una lista de palabras separadas por espacios en blanco en la que al menos una de ellas es exactamente igual a 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 }
```

La pseudo-clase :first-child selecciona el primer elemento hijo de un elemento. Si se considera el siguiente ejemplo:

```
p em:first-child {
  color: red;
}

<p>Lorem <span><em>ipsum dolor</em></span> sit amet, consectetuer
adipiscing elit. Praesent odio sem, tempor quis, <em>auctor eu</em>,
tempus at, enim. Praesent nulla ante, <em>ultricies</em> id, porttitor
ut, pulvinar quis, dui.</p>
```

El selector p em:first-child selecciona el primer elemento que sea hijo de un elemento y que se encuentre dentro de un elemento <p>. Por tanto, en el ejemplo anterior sólo el primer se ve de color rojo.

La pseudo-clase :first-child también se puede utilizar en los selectores simples, como se muestra a continuación:

```
p:first-child { ... }
```

La regla CSS anterior aplica sus estilos al primer párrafo de cualquier elemento. Si se modifica el ejemplo anterior y se utiliza un selector compuesto:

```
p:first-child em {
  color: red;
}

<body>
  <p>Lorem <span><em>ipsum dolor</em></span> sit amet, consectetuer
  adipiscing elit. Praesent odio sem, tempor quis, <em>auctor eu</em>,
  tempus at, enim.</p>
```

```
<p>Lorem <span><em>ipsum dolor</em></span> sit amet, consectetuer  
adipiscing elit. Praesent odio sem, tempor quis, <em>auctor eu</em>,  
tempus at, enim.</p>  
  
<div>  
    <p>Lorem <span><em>ipsum dolor</em></span> sit amet, consectetuer  
    adipiscing elit. Praesent odio sem, tempor quis, <em>auctor  
    eu</em>,  
    tempus at, enim.</p>  
</div>  
</body>
```

El selector `p:first-child em` selecciona todos aquellos elementos `` que se encuentren dentro de un elemento `<p>` que sea el primer hijo de cualquier otro elemento.

El primer párrafo del ejemplo anterior es el primer hijo de `<body>`, por lo que sus `` se ven de color rojo. El segundo párrafo de la página no es el primer hijo de ningún elemento, por lo que sus elementos `` interiores no se ven afectados. Por último, el tercer párrafo de la página es el primer hijo del elemento `<div>`, por lo que sus elementos `` se ven de la misma forma que los del primer párrafo.

Las pseudo-clases `:link` y `:visited` se pueden utilizar para aplicar diferentes estilos a los enlaces de una misma página:

- La pseudo-clase `:link` se aplica a todos los enlaces que todavía no han sido visitados por el usuario.
- La pseudo-clase `:visited` se aplica a todos los enlaces que han sido visitados al menos una vez por el usuario.

El navegador gestiona de forma automática el cambio de enlace no visitado a enlace visitado. Aunque el usuario puede borrar la cache y el historial de navegación de forma explícita, los navegadores también borran de forma periódica la lista de enlaces visitados.

Por su propia definición, las pseudo-clases `:link` y `:visited` son mutuamente excluyentes, de forma que un mismo enlace no puede estar en los dos estados de forma simultánea.

Como los navegadores muestran por defecto los enlaces de color azul y los enlaces visitados de color morado, es habitual modificar los estilos para adaptarlos a la guía de estilo del sitio web:

```
a:link { color: red; }
a:visited { color: green; }
```

Las pseudo-clases :hover, :active y :focus permiten al diseñador web variar los estilos de un elemento en respuesta a las acciones del usuario. Al contrario que las pseudo-clases :link y :visited que sólo se pueden aplicar a los enlaces, estas pseudo-clases se pueden aplicar a cualquier elemento.

A continuación se indican las acciones del usuario que activan cada pseudo-clase:

- :hover, se activa cuando el usuario pasa el ratón o cualquier otro elemento apuntador por encima de un elemento.
- :active, se activa cuando el usuario activa un elemento, por ejemplo cuando pulsa con el ratón sobre un elemento. El estilo se aplica durante un espacio de tiempo prácticamente imperceptible, ya que sólo dura desde que el usuario pulsa el botón del ratón hasta que lo suelta.
- :focus, se activa cuando el elemento tiene el foco del navegador, es decir, cuando el elemento está seleccionado. Normalmente se aplica a los elementos <input> de los formularios cuando están activados y por tanto, se puede escribir directamente en esos campos.

De las definiciones anteriores se desprende que un mismo elemento puede verse afectado por varias pseudo-clases diferentes de forma simultánea. Cuando se pulsa por ejemplo un enlace que fue visitado previamente, al enlace le afectan las pseudo-clases :visited, :hover y :active.

Debido a esta característica y al comportamiento en cascada de los estilos CSS, es importante cuidar el orden en el que se establecen las diferentes pseudo-clases. El siguiente ejemplo muestra el único orden correcto para establecer las cuatro pseudo-clases principales en un enlace:

```
a:link { ... }  
a:visited { ... }  
a:hover { ... }  
a:active { ... }
```

Por último, también es posible aplicar estilos combinando varias pseudo-clases compatibles entre sí. La siguiente regla CSS por ejemplo sólo se aplica a aquellos enlaces que están seleccionados y en los que el usuario pasa el ratón por encima:

```
a:focus:hover { ... }
```

La pseudo-clase `:lang` se emplea para seleccionar elementos en función de su idioma. Los navegadores utilizan los atributos `lang`, las etiquetas `<meta>` y la información de la respuesta del servidor para determinar el idioma de cada elemento.

Si se considera el siguiente ejemplo:

```
p { color: blue; }  
p:lang(es) { color: red; }
```

Los párrafos del ejemplo anterior se ven de color azul, salvo los párrafos cuyo contenido esté escrito en español, que se ven de color rojo.

Como los navegadores actuales no son capaces de inferir el idioma de un elemento a partir de su contenido, el uso de esta clase está muy limitado salvo que la página utilice de forma explícita los atributos `'lang'`:

```
<p lang="en">Lorem ipsum dolor sit amet...</p>  
<div lang="fr">  
    <p>Lorem ipsum dolor sit amet...</p>  
    <p lang="es_ES">Lorem ipsum dolor sit amet...</p>  
</div>  
<p lang="en">Lorem ipsum dolor sit amet...</p>  
<ul>  
    <li lang="fr">Lorem ipsum dolor sit amet...</li>  
</ul>
```

La pseudo-clase `:lang(xx)` es muy diferente al selector de atributos `'[lang]=xx'`, tal y como muestran las siguientes reglas:

```
*[lang|=es] { ... } /* selector de atributo */  
*:lang(es) { ... } /* pseudo-clase */  
  
<body lang="es">  
  <p>Lorem ipsum dolor sit amet...</p>  
</body>
```

El selector `*[lang|=es]` selecciona todos los elementos de la página que tengan un atributo llamado `lang` cuyo valor empiece por `es`. En el ejemplo anterior, solamente el elemento `<body>` cumple con la condición del selector.

Por otra parte, el selector `*:lang(es)` selecciona todos los elementos de la página cuyo idioma sea el español, sin tener en cuenta el método empleado por el navegador para averiguar el idioma de cada elemento. En este caso, tanto el elemento `<body>` como el elemento `<p>` cumplen esta condición.

Los selectores de CSS, las pseudo-clases y todos los elementos HTML no son suficientes para poder aplicar estilos a algunos elementos especiales. Si se desea por ejemplo cambiar el estilo de la primera línea de texto de un elemento, no es posible hacerlo con las utilidades anteriores.

La primera línea del texto normalmente es variable porque el usuario puede aumentar y disminuir la ventana del navegador, puede disponer de más o menos resolución en su monitor y también puede aumentar o disminuir el tamaño de letra del texto.

La única forma de poder seleccionar estos elementos especiales es mediante los pseudo-elementos definidos por CSS para este propósito.

El pseudo-elemento `:first-line` permite seleccionar la primera línea de texto de un elemento. Así, la siguiente regla CSS muestra en mayúsculas la primera línea de cada párrafo:

```
p:first-line { text-transform: uppercase; }
```

Este pseudo-elemento sólo se puede utilizar con los elementos de bloque y las celdas de datos de las tablas.

Se pueden combinar varios pseudo-elementos de tipo :first-line para crear efectos avanzados:

```
div:first-line { color: red; }
p:first-line { text-transform: uppercase; }

<div>
  <p>Lorem ipsum dolor sit amet...</p>
  <p>Lorem ipsum dolor sit amet...</p>
  <p>Lorem ipsum dolor sit amet...</p>
</div>
```

En el ejemplo anterior, la primera línea del primer párrafo también es la primera línea del elemento `<div>`, por lo que se le aplican las dos reglas CSS y su texto se ve en mayúsculas y de color rojo.

El pseudo-elemento :first-letter permite seleccionar la primera letra de la primera línea de texto de un elemento. De esta forma, la siguiente regla CSS muestra en mayúsculas la primera letra del texto de cada párrafo:

```
p:first-letter { text-transform: uppercase; }
```

Los signos de puntuación y los caracteres como las comillas que se encuentran antes y después de la primera letra también se ven afectados por este pseudo-elemento.

Este pseudo-elemento sólo se puede utilizar con los elementos de bloque y las celdas de datos de las tablas.

Los pseudo-elementos :before y :after se utilizan en combinación con la propiedad content de CSS para añadir contenidos antes o después del contenido original de un elemento.

Las siguientes reglas CSS añaden el texto Capítulo - delante de cada título de sección `<h1>` y el carácter . detrás de cada párrafo de la página:

```
h1:before { content: "Capítulo - "; }
p:after   { content: ". "; }
```

El contenido insertado mediante los pseudo-elementos :before y :after se tiene en cuenta en los otros pseudo-elementos :first-line y :first-letter.

El estándar CSS 2.1 incluye 115 propiedades que abarcan el modelo de cajas (box model), la tipografía, las tablas, las listas, el posicionamiento de los elementos, la generación de contenidos y los medios impresos y auditivos.

Aunque la mayoría de diseñadores web conocen y utilizan casi todas las propiedades de CSS 2.1, no siempre hacen uso de todas sus posibilidades. Algunas propiedades de CSS 2.1 han sido infroutilizadas hasta hace poco tiempo porque los navegadores no las soportaban.

display	
Valores	inline block 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 none inherit
Se aplica a	Todas los elementos
Valor inicial	inline
Descripción	Establece el tipo de caja generada por un elemento

La propiedad `display` es una de las propiedades CSS más infroutilizadas. Aunque todos los diseñadores conocen esta propiedad y utilizan sus valores `inline`, `block` y `none`, las posibilidades de `display` son mucho más avanzadas.

De hecho, la propiedad `display` es una de las más complejas de CSS 2.1, ya que establece el tipo de la caja que genera cada elemento. La propiedad `display` es tan compleja que casi ningún navegador es capaz de mostrar correctamente todos sus valores.

El valor más sencillo de `display` es `none` que hace que el elemento no genere ninguna caja. El resultado es que el elemento desaparece por completo de la página y no ocupa sitio, por lo que los elementos adyacentes ocupan su lugar. Si se utiliza la propiedad `display: none` sobre un ele-

mento, todos sus descendientes también desaparecen por completo de la página.

Si se quiere hacer un elemento invisible, es decir, que no se vea pero que siga ocupando el mismo sitio, se debe utilizar la propiedad `visibility`. La propiedad `display: none` se utiliza habitualmente en aplicaciones web dinámicas creadas con JavaScript y que muestran/ocultan contenidos cuando el usuario realiza alguna acción como pulsar un botón o un enlace.

Los otros dos valores más utilizados son `block` e `inline` que hacen que la caja de un elemento sea de bloque o en línea respectivamente. El siguiente ejemplo muestra un párrafo y varios enlaces a los que se les ha añadido un borde para mostrar el espacio ocupado por cada caja:

`Lorem ipsum dolor sit amet, consectetuer adipiscing elit.`

`[display: block] Lorem ipsum (#) Donec mollis nunc in leo (#) Vivamus fermentum (#)`

Como el párrafo es por defecto un elemento de bloque ("*block element*"), ocupa todo el espacio disponible hasta el final de su línea, aunque sus contenidos no ocupen todo el sitio. Por su parte, los enlaces por defecto son elementos en línea ("*inline element*"), por lo que su caja sólo ocupa el espacio necesario para mostrar sus contenidos.

Si se aplica la propiedad `display: inline` al párrafo del ejemplo anterior, su caja se convierte en un elemento en línea y por tanto sólo ocupa el espacio necesario para mostrar sus contenidos:

`[display: inline] Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Lorem ipsum (#) Donec mollis nunc in leo (#) Vivamus fermentum (#)`

Para visualizar más claramente el cambio en el tipo de caja, el siguiente ejemplo muestra un mismo párrafo largo con `display: block` y `display: inline`:

`[display: block] Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Sed non sem quis tellus vulputate lobortis. Vivamus fermentum, tortor id ornare ultrices, ligula ipsum tincidunt pede, et blandit sem pede suscipit pede. Nulla cursus porta sem. Donec mollis nunc in leo.`

[display: inline] Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Sed non sem quis tellus vulputate lobortis. Vivamus fermentum, tortor id ornare ultrices, ligula ipsum tincidunt pede, et blandit sem pede suscipit pede. Nulla cursus porta sem. Donec mollis nunc in leo.

De la misma forma, si en los enlaces del ejemplo anterior se emplea la propiedad `display: block` se transforman en elementos de bloque, por lo que siempre empiezan en una nueva línea y siempre ocupan todo el espacio disponible en la línea, aunque sus contenidos no ocupen todo el sitio:

Lorem ipsum dolor sit amet, consectetuer adipiscing elit.

[display: block] Lorem ipsum (#)

[display: block] Donec mollis nunc in leo (#)

[display: block] Vivamus fermentum (#)

Uno de los valores más curiosos de `display` es `inline-block`, que crea cajas que son de bloque y en línea de forma simultánea. Una caja de tipo `inline-block` se comporta como si fuera de bloque, pero respecto a los elementos que la rodean es una caja en línea.

El enlace del siguiente ejemplo es de tipo `inline-block`, lo que permite por ejemplo establecerle un tamaño mediante la propiedad `width`:

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Sed non sem quis tellus vulputate lobortis. Vivamus fermentum, tortor id ornare ultrices, ligula ipsum tincidunt pede, et blandit sem pede suscipit pede. Nu-

[display: inline-block, width: 25%] Quisque semper, magna sed pharetra tincidunt, quam urna dapibus dolor, a dignissim sem neque id purus. Etiam luctus vive-

lla cursus porta sem. Donec mollis nunc in leo. In-

teger lobortis accumsan felis. Cras venenatis. Morbi cursus, tellus vitae iaculis pulvinar, turpis nibh posuere nisl, sed vehicula massa orci at dui. Morbi pede ipsum, porta quis, venenatis et, ullamcorper in, metus. Nulla facilisi. Quisque laoreet molestie mi. Ut mollis elit eget urna.

Otro de los valores definidos por la propiedad display es `list-item`, que hace que cualquier elemento de cualquier tipo se muestre como si fuera un elemento de una lista (elemento ``). El siguiente ejemplo muestra tres párrafos que utilizan la propiedad display: `list-item` para simular que son una lista de elementos de tipo ``:

- Lorem ipsum dolor sit amet, consectetuer adipiscing elit.
- Sed non sem quis tellus vulputate lobortis.
- Vivamus fermentum, tortor id ornare ultrices, ligula ipsum tincidunt pede, et blandit sem pede suscipit pede.

A continuación se muestra el código HTML del ejemplo anterior:

```
<p style="display: list-item; margin-left: 2em">Lorem ipsum dolor sit  
amet, consectetuer adipiscing elit.</p>  
<p style="display: list-item; margin-left: 2em">Sed non sem quis tellus  
vulputate lobortis.</p>  
<p style="display: list-item; margin-left: 2em">Vivamus fermentum,  
tortor id ornare ultrices, ligula ipsum tincidunt pede, et blandit sem  
pede suscipit pede.</p>
```

Los elementos con la propiedad display: `list-item` son exactamente iguales que los elementos `` a efectos de su visualización, por lo que se pueden utilizar las propiedades de listas como `list-style-type`, `list-style-image`, `list-style-position` y `list-style`.

Otro de los valores curiosos de la propiedad display es `run-in`, que genera una caja de bloque o una caja en línea dependiendo del contexto, es decir, dependiendo de sus elementos adyacentes. El comportamiento de las cajas de tipo `run-in` se rige por las siguientes reglas:

- Si la caja `run-in` contiene una caja de bloque, la caja `run-in` se convierte en una caja de bloque.
- Si después de la caja `run-in` se encuentra una caja de bloque (que no esté posicionada de forma absoluta y tampoco esté posicionada

de forma flotante), la caja run-in se convierte en una caja en línea en el interior de la caja de bloque.

- En cualquier otro caso, la caja run-in se convierte en una caja de bloque.

[display: run-in] Lorem ipsum [display: block] dolor sit amet, consec-tetuer adipiscing elit.

[display: run-in] Lorem ipsum [display: inline] dolor sit amet, consec-tetuer adipiscing elit.

El estándar CSS 2.1 incluye un ejemplo del posible uso del valor run-in. En este ejemplo, un título de sección `<h3>` crea una caja run-in, de forma que cuando va seguido de un párrafo, el titular se mete dentro del párrafo:

```
<h3 style="display: run-in">Lorem ipsum dolor sit amet</h3>
<p>Sed non sem quis tellus vulputate lobortis. Vivamus fermentum, tortor
id ornare ultrices, ligula ipsum tincidunt pede, et blandit sem pede
suscipit pede. Nulla cursus porta sem. Donec mollis nunc in leo. Integer
lobortis accumsan felis.</p>
```

El resto de valores de la propiedad display están relacionados con las tablas y hacen que un elemento se muestre como si fuera una parte de una tabla: fila, columna, celda o grupos de filas/columnas. Los valores definidos por la propiedad display son `inline-table`, `table-row-group`, `table-header-group`, `table-footer-group`, `table-row`, `table-column-group`, `table-column`, `table-cell`, `table-caption`.

Aunque los valores relacionados con las tablas son los más avanzados, también son los que peor soportan los navegadores. A continuación se muestra un ejemplo con tres párrafos de texto que establecen la propiedad display: `table-cell`:

[display: table- [display: table- [display: table-cell] Morbi
cell] Lorem ipsum cell] In molestie sed nisl sed dui consequat
dolor sit amet, suscipit libero. Cras sodales. Vivamus ornare felis
consectetuer adi- sem. Nunc non te- nec est. Phasellus massa jus-
piscing elit. Mae- llus et urna mattis to, ornare sed, malesuada a,
cenas non tortor. tempor. Nulla nec dignissim a, nibh. Vestibulum
Vestibulum ante tellus a quam hen- vitae nunc at lectus euismod

ipsum primis in drerit venenatis. feugiat. Nullam eleifend. faucibus orci luc- Suspendisse pe- Class aptent taciti sociosqu tus et ultrices po- llentesque odio et ad litora torquent per conubia suere cubilia Cu- est. Morbi sed nisl nostra, per inceptos himerae; Sed fermen- sed dui consequat naeos. In ut ipsum. tum lorem a velit. sodales.

La propiedad `display: table-cell` hace que cualquier elemento se muestre como si fuera una celda de una tabla. Como en el ejemplo anterior los tres elementos `<p>` utilizan la propiedad `display: table-cell`, el resultado es visualmente idéntico a utilizar una tabla y tres elementos `<td>`.

	content
Valores	(string uri counter attr open-quote close-quote no-open-quote no-close-quote) normal none inherit
Se aplica a	Solamente a los pseudo-elementos <code>before</code> y <code>after</code>
Valor inicial	normal
Descripción	Genera contenido de forma dinámica

La propiedad `content` es una de las propiedades CSS más poderosas y a la vez más controvertidas. La propiedad `content` se emplea para generar nuevo contenido de forma dinámica e insertarlo en la página HTML. Como CSS es un lenguaje de hojas de estilos cuyo único propósito es controlar el aspecto o presentación de los contenidos, algunos diseñadores defienden que no es correcto generar nuevos contenidos mediante CSS.

En primer lugar, el estándar CSS 2.1 indica que la propiedad `content` sólo puede utilizarse en los pseudo-elementos `:before` y `:after`. Como su propio nombre indica, estos pseudo-elementos permiten seleccionar (y por tanto modificar) la parte anterior o posterior de un elemento de la página.

El siguiente ejemplo muestra cómo añadir la palabra `Capítulo` delante del contenido de cada título de sección `<h1>`:

```
h1:before {
    content: "Capítulo ";
}
```

Los pseudo-elementos :before y :after se pueden utilizar sobre cualquier elemento de la página. El siguiente ejemplo añade la palabra Nota: delante de cada párrafo cuya clase CSS sea nota:

```
p.nota:before {  
    content: "Nota: ";  
}
```

Combinando las propiedades content y quotes con los pseudo-elementos :before y :after, se pueden añadir de forma dinámica comillas de apertura y de cierre a todos los elementos <blockquote> de la página:

```
blockquote:before {  
    content: open-quote;  
}  
blockquote:after {  
    content: close-quote;  
}  
blockquote {  
    quotes: "«" "»";  
}
```

Los contenidos insertados dinámicamente en un elemento son a todos los efectos parte de ese mismo elemento, por lo que heredan el valor de todas sus propiedades CSS.

Los dos valores más sencillos de la propiedad content son none y normal. En la práctica, estos dos valores tienen el mismo efecto ya que hacen que el pseudo-elemento no se genere.

El siguiente valor que se puede indicar en la propiedad content es una cadena de texto. En el estándar CSS 2.1, una cadena de texto es un conjunto de uno o más caracteres encerrados por las comillas dobles ("") o las comillas simples (''). Si la cadena contiene comillas dobles, se encierra con las comillas simples y viceversa. Si una cadena de texto tiene tanto comillas simples como dobles, las comillas problemáticas se modifican y se les añade la barra invertida \ por delante:

```
p:before {  
    content: "Contenido generado \"dinámicamente\" mediante CSS. ";  
}  
#ultimo:after {
```

```
    content: " Fin de los 'contenidos' de la página.";  
}
```

Las cadenas de texto sólo permiten incluir texto básico. Si se incluye alguna etiqueta HTML en la cadena de texto, el navegador muestra la etiqueta tal y como está escrita, ya que no las interpreta. Para incluir un salto de línea en el contenido generado, se utiliza el carácter especial \A

El siguiente valor aceptado por la propiedad content es una URL, que suele utilizarse para indicar la URL de una imagen que se quiere añadir de forma dinámica al contenido. La sintaxis es idéntica al resto de URL que se pueden indicar en otras propiedades CSS:

```
span.especial:after {  
    content: url("imagenes/imagen.png");  
}
```

Otros valores que se pueden indicar en la propiedad content son open-quote, close-quote, no-open-quote y no-close-quote. Los dos primeros indican que se debe mostrar una comilla de apertura o de cierre respectivamente. Las comillas utilizadas se establecen mediante la propiedad quotes:

```
blockquote { quotes: "«" "»" '‘' '’' }  
blockquote:before {  
    content: open-quote;  
}  
blockquote:after {  
    content: close-quote;  
}
```

Los valores no-open-quote y no-close-quote se utilizan para no mostrar ninguna comilla en ese elemento, pero incrementando el nivel de anidamiento de las comillas. De esta forma se puede evitar mostrar una comilla en un determinado elemento mientras se mantiene la jerarquía de comillas establecida por la propiedad quotes.

Uno de los valores más avanzados de la propiedad content es attr(), que permite obtener el valor de un atributo del elemento sobre el que se utiliza la propiedad content. En el siguiente ejemplo, se modifican los elementos <abbr> y <acronym> para que muestren entre paréntesis el valor de sus atributos title:

```
abbr:after, acronym:after {  
    content: " (" attr(title) ")"  
}
```

El valor de la propiedad content anterior en realidad es la combinación de tres valores:

- Cadena de texto " (", que es el paréntesis de apertura tras el cual se muestra el valor del atributo title.
- Atributo title del elemento obtenido mediante attr(title)
- Cadena de texto ")", que es el paréntesis de cierre que se muestra detrás del valor del atributo title.

Si el elemento no dispone del atributo solicitado, la función attr(nombre_del_atributo) devuelve una cadena de texto vacía. Utilizando attr() solamente se puede obtener el valor de los atributos del elemento al que se aplica la propiedad content.

La función attr() es muy útil por ejemplo para mostrar la dirección a la que apuntan los enlaces de la página:

```
a:after {  
    content: " (" attr(href) ")";  
}
```

Capítulo 14

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=UTF-8" />
  <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 parrafos 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>
```

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 blandit
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 14.1 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, lacus 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>
    </tbody>
  
```

```
<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>
```

Los nombres de los colores 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.

A partir del código HTML y CSS proporcionados, determinar las reglas CSS necesarias para añadir los siguientes márgenes y rellenos:

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](#)

Ehiam 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...](#)

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 14.2 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

[Lorem Ipsum Dolor Sit Amet](#)
Buscar

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. 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 14.3 Página con márgenes y rellenos

[Descargar ficheros \(snippets/cap14/ej03.zip\)](#)

A partir del código HTML y CSS proporcionados, determinar las reglas CSS necesarias para añadir los siguientes bordes:

LOGOTIPO

[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



Nulum 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 14.4 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 [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 platerat](#)

[Nulla in felis](#)

[Nam luctus](#)

Publicidad

Etiam fermentum, misl 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 finibus 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 14.5 Página con bordes

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 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...](#)

Figura 14.6 Página original

- Los elementos #noticias y #pie tiene un color de fondo gris claro (#F8F8F8).
- El elemento #publicidad muestra un color de fondo amarillo claro (#FFF6CD).
- Los elementos <h2> del lateral #secundario muestran un color de fondo #DB905C y un pequeño padding de .2em.
- El fondo del elemento #menu se construye mediante una pequeña imagen llamada fondo_menu.gif.
- 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 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...](#)

Placeholder



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](#) | [Pron](#) | [Placerat](#)

© Copyright Lorem ipsum

Figura 14.7 Página con colores e imágenes de fondo

[Descargar ficheros \(snippets/cap14/ej05.zip\)](#)

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=UTF-8" />
<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:

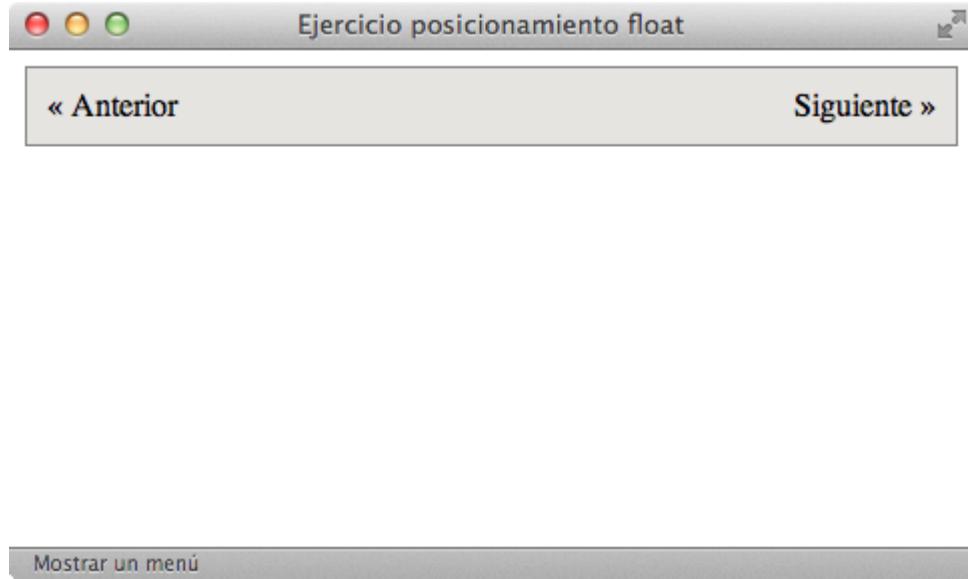


Figura 14.8 Elementos posicionados mediante float

A partir del código HTML y CSS proporcionados, determinar las reglas CSS necesarias para añadir las siguientes propiedades a la tipografía de la página:

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.

Determinar las reglas CSS necesarias para que el resultado sea similar al mostrado en la siguiente imagen:

The screenshot shows a web page with the following structure and content:

- Header:** A blue header bar with the word "Logotipo" in orange. To its right is a search input field labeled "Buscar". Below the header is a navigation menu with links: "Lorem", "Ipsum", "Dolor", "Sit", and "Amet".
- Left Sidebar (Noticias):** A sidebar with a yellow background containing a list of news items. Each item has a date (dd/mm/aaaa) followed by a link. The links are colored blue and underlined.
 - [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](#)
- Main Content Area:**
 - Section 1:** A large orange header "Lorem ipsum dolor sit amet, consectetuer adipiscing elit". Below it is a gray square with a black 'X' symbol. To the right is a text block with several paragraphs of placeholder text (Lorem ipsum). At the end of the text is a blue "Seguir leyendo..." link.
 - Section 2:** An orange header "Vivamus lobortis turpis ac ante fringilla faucibus". Below it is another gray square with a black 'X' symbol. To the right is a text block with placeholder text. At the end of the text is a blue "Seguir leyendo..." link.
- Right Sidebar (Phasellus blandit):** A sidebar with a yellow background containing a list of text snippets. Each snippet has a date (dd/mm/aaaa) followed by a link. The links are colored blue and underlined.
 - [dd/mm/aaaa](#)
 - [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...](#)
 - Section 3:** An orange header "Nullam vel turpis". Below it is a text block with placeholder text. At the end of the text is a blue "Seguir leyendo..." link.

At the bottom of the page, there is a footer with links: "Nulla", "Pharetra", "Luctus", "Ipsum", "Proin", and "Placerat". To the right of the footer is a copyright notice: "© Copyright Lorem ipsum".

Figura 14.9 Página con propiedades tipográficas

Descargar ficheros ([snippets/cap14/ej07.zip](#))

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.



Menú

- Inicio
- Noticias
 - [Recientes](#)
 - [Más leídas](#)
 - [Más valoradas](#)



Figura 14.10 Enlaces con estilos aplicados mediante CSS

Modificar el menú vertical sencillo para que muestre el siguiente comportamiento:

- 1) Los elementos deben mostrar una imagen de fondo (flecha_inactiva.png):

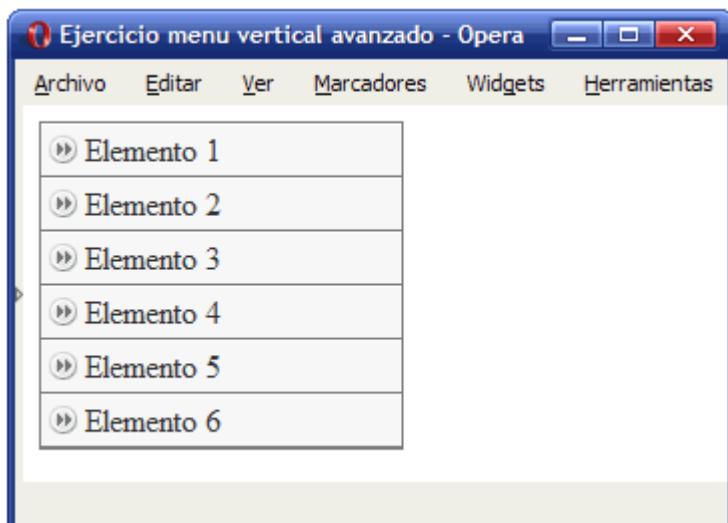


Figura 14.11 Menú vertical con imagen de fondo

2) Cuando se pasa el ratón por encima de un elemento, se debe mostrar una imagen alternativa (flecha_activa.png):

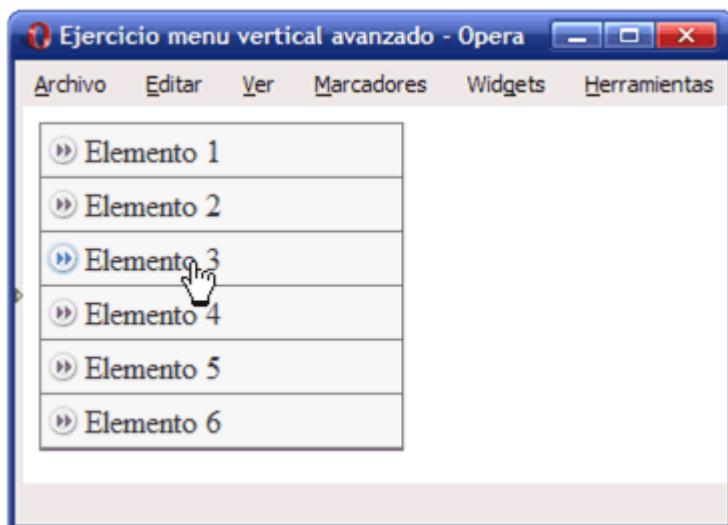


Figura 14.12 Menú vertical con imagen de fondo alternativa

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:

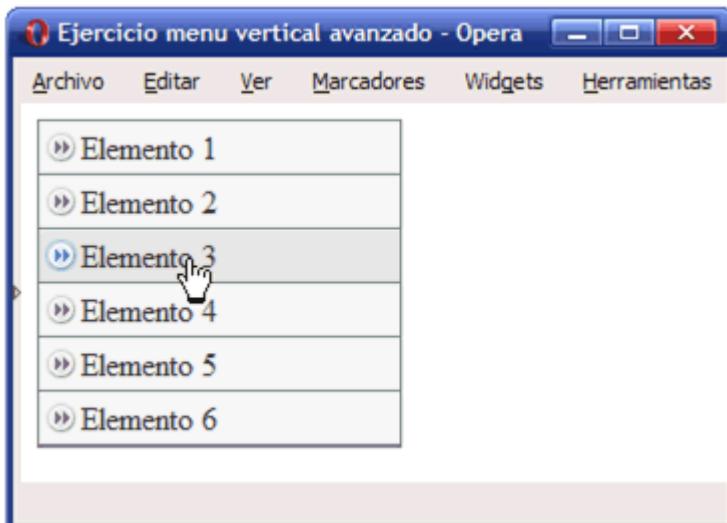


Figura 14.13 Menú vertical con imagen de fondo y color alternativos

[Descargar ficheros \(snippets/cap14/ej09.zip\)](#)

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	Provincia
	País
Email	Teléfono *
<input type="text"/>	<input type="text"/>
Fijo	Móvil
<hr/> <div style="text-align: right; padding-right: 10px;"> <input type="button" value="Darme de alta →"/> </div>	

Figura 14.14 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

The image shows a simple form with a yellow background. At the top, it says "Nombre y apellidos *". Below that is a yellow-bordered input field labeled "Nombre" with a cursor icon pointing to its right. Underneath it are two white input fields labeled "Primer apellido" and "Segundo apellido".

Figura 14.15 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 14.16 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
Código postal
Municipio
Provincia
País

El código postal es imprescindible para poder recibir los pedidos

Email

Teléfono *

Fijo Móvil

Darme de alta →

The form consists of several input fields and dropdown menus. The 'Nombre y apellidos' section has three input fields. The 'Dirección' section has five input fields and one note. The 'Email' and 'Teléfono' sections have single input fields. The 'Darme de alta' button is at the bottom right.

Figura 14.17 Aspecto final del formulario

[Descargar ficheros \(snippets/cap14/ej10.html\)](#)

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 14.18 Aspecto original de la tabla

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
¥ Yen/Euro	0.6711	0.6705	0.6676	0.6713

Figura 14.19 Aspecto final de la tabla

- 1) Alinear el texto de las celdas, cabeceras y título. Definir los bordes de la tabla, celdas y cabeceras (color gris oscuro #333):

The screenshot shows a table with five columns: Cambio, Compra, Venta, Máximo, and Mínimo. The rows represent currency pairs: Euro/Dolar, Dolar/Yen, Libra/Dolar, and Yen/Euro. The text within the cells is aligned to the right. The table has a border and a light gray 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 14.20 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 the same table as Figure 14.20, but with colored backgrounds and currency symbols. The header row has a light gray background. The data rows have alternating colors: white, light blue, and light green. The first column contains currency symbols: €, \$, £, and ¥.

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 14.21 Tabla con colores e imágenes de fondo

3) Mostrar un color alterno en las filas de datos (color amarillo claro #FFFFCC).

The screenshot shows a window titled "Ejercicio formatear tabla - Opera". The menu bar includes Archivo, Editar, Ver, Marcadores, Widgets, Herramientas, and Ayuda. Below the menu is 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 14.22 Tabla con colores de fila alternos

[Descargar ficheros \(snippets/cap14/ej11.zip\)](#)

Determinar las reglas CSS necesarias para mostrar la página HTML que se proporciona con el estilo que se muestra en la siguiente imagen:



Figura 14.23 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 14.24 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 blog article page. At the top, there is a header with the title "Lorem ipsum dolor sit amet". Below the title is a timestamp "dd-mm-aaaa 00:00" and author information "diseño Nombre Apellido". There is also a link to "Añadir comentario". The main content area contains placeholder text: "Lorem ipsum dolor sit amet, consectetuer 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 content area, there is a link "Seguir leyendo...".

Figura 14.25 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.



Figura 14.26 Aspecto de las secciones de la columna secundaria de contenidos

[Descargar ficheros \(snippets/final/final.zip\)](#)