



INTRODUCCIÓN A CSS

AUTOR: JAVIER EGUILUZ

Licencia


Creative Commons No comercial - Atribución - Compartir igual (CC BY-NC-SA) 3.0

Ciclo formativo de Desarrollo de aplicaciones Web

Módulo de Diseño de interfaces Web

Profesor: Eugenio Álvarez

CIFP La Laboral de Gijón



Contenido

Capítulo 1. Introducción	5
1.1. ¿Qué es CSS?	5
1.2. Breve historia de CSS.....	5
1.3. Soporte de CSS en los navegadores	6
1.4. Especificación oficial	7
1.5. Funcionamiento básico de CSS.....	8
1.6. Cómo incluir CSS en un documento XHTML.....	10
1.6.1. Incluir CSS en el mismo documento HTML	10
1.6.2. Definir CSS en un archivo externo.....	11
1.6.3. Incluir CSS en los elementos HTML	13
1.7. Glosario básico	14
1.8. Medios CSS	15
1.8.1. Medios definidos con las reglas de tipo @media	17
1.8.2. Medios definidos con las reglas de tipo @import	18
1.8.3. Medios definidos con la etiqueta <link>.....	18
1.8.4. Medios definidos mezclando varios métodos.....	18
1.9. Comentarios	19
1.10. Sintaxis de la definición de cada propiedad CSS	19
Capítulo 2. Selectores.....	21
2.1. Selectores básicos	21
2.1.1. Selector universal	21
2.1.2. Selector de tipo o etiqueta.....	21
2.1.3. Selector descendente	23
2.1.4. Selector de clase.....	25
2.1.5. Selectores de ID.....	29
2.1.6. Combinación de selectores básicos	30
2.2. Selectores avanzados	31
2.2.1. Selector de hijos	31
2.2.2. Selector adyacente	32
2.2.3. Selector de atributos	33
2.3. Agrupación de reglas.....	34
2.4. Herencia	35
2.5. Colisiones de estilos	37
Capítulo 3. Unidades de medida y colores.....	39

3.1. Unidades de medida	39
3.1.1. Unidades absolutas	39
3.1.2. Unidades relativas	40
3.1.3. Porcentajes.....	43
3.1.4. Recomendaciones	43
3.2. Colores.....	44
3.2.1. Palabras clave.....	44
3.2.2. RGB decimal	44
3.2.3. RGB porcentual	45
3.2.4. RGB hexadecimal.....	45
3.2.5. Colores del sistema	47
3.2.6. Colores web safe	47
Capítulo 4. Modelo de cajas	49
4.1. Anchura y altura	51
4.1.1. Anchura	51
4.1.2. Altura	52
4.2. Margen y relleno	53
4.2.1. Margen	53
4.2.2. Relleno.....	59
4.3. Bordes	60
4.3.1. Anchura	60
4.3.2. Color	63
4.3.3. Estilo	64
4.3.4. Propiedades shorthand	67
4.4. Margen, relleno, bordes y modelo de cajas.....	70
4.4.1. Los modos de compatibilidad de Internet Explorer 8.....	73
4.5. Fondos	75
Capítulo 5. Posicionamiento y visualización	86
5.1. Tipos de elementos	86
5.2. Posicionamiento	88
5.3. Posicionamiento normal	90
5.4. Posicionamiento relativo.....	91
5.5. Posicionamiento absoluto	94
5.6. Posicionamiento fijo.....	100
5.7. Posicionamiento flotante	101
5.8. Visualización	108

5.8.1. Propiedades display y visibility.....	108
5.8.2. Relación entre display, float y position	111
5.8.3. Propiedad overflow	112
5.8.4. Propiedad z-index.....	114
Capítulo 6. Texto	117
6.1. Tipografía	117
6.2. Texto.....	127
Capítulo 7. Enlaces	142
7.1. Estilos básicos.....	142
7.1.1. Tamaño, color y decoración	142
7.1.2. Pseudo-clases	143
7.2. Estilos avanzados.....	145
7.2.1. Decoración personalizada	145
7.2.2. Imágenes según el tipo de enlace	147
7.2.3. Mostrar los enlaces como si fueran botones	148
Capítulo 8. Imágenes.....	150
8.1. Estilos básicos.....	150
8.1.1. Establecer la anchura y altura de las imágenes	150
8.1.2. Eliminar el borde de las imágenes con enlaces.....	150
8.2. Estilos avanzados.....	151
8.2.1. Sombra (drop shadow).....	151
Capítulo 9. Listas	154
9.1. Estilos básicos.....	154
9.1.1. Viñetas personalizadas.....	154
9.1.2. Menú vertical	159
Capítulo 10. Tablas	163
10.1. Estilos básicos.....	163
10.2. Estilos avanzados.....	167
Capítulo 11. Formularios	171
11.1. Estilos básicos.....	171
11.1.1. Mostrar un botón como un enlace	171
11.1.2. Mejoras en los campos de texto	171
11.1.3. Labels alineadas y formateadas	172
11.2. Estilos avanzados.....	176
11.2.1. Formulario en varias columnas	176
11.2.2. Resaltar el campo seleccionado	178

Capítulo 12. Layout	179
12.1. Centrar una página horizontalmente	179
12.2. Centrar una página verticalmente	183
12.3. Estructura o layout	187
12.3.1. Diseño a 2 columnas con cabecera y pie de página	187
12.3.2. Diseño a 3 columnas con cabecera y pie de página	189
12.4. Alturas/anchuras máximas y mínimas	192
12.5. Estilos avanzados.....	195
Capítulo 13. Otros	197
13.1. Propiedades shorthand	197
13.2. Versión para imprimir	200
13.2.1. Imprimiendo los enlaces.....	202
13.3. Personalizar el cursor	203
13.4. Hacks y filtros	206
13.5. Prioridad de las declaraciones CSS.....	209
13.6. Validador	211
13.7. Recomendaciones generales sobre CSS.....	212
13.7.1. Atributos ID y class	212
13.7.2. CLASSitis y DIVitis	213
13.7.3. Estructuración del código CSS.....	214
13.7.4. División de los estilos en varios archivos CSS.....	217

Capítulo 1. Introducción

1.1. ¿Qué es CSS?

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

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

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

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

1.2. Breve historia de CSS

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

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

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

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

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

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

El 12 de Mayo de 1998, el grupo de trabajo de CSS publica su segunda recomendación oficial, conocida como "CSS nivel 2". La versión de CSS que utilizan todos los navegadores de hoy en día es CSS 2.1, una revisión de CSS 2 que aún se está elaborando (la última actualización es del 8 de septiembre de 2009). Al mismo tiempo, la siguiente recomendación de CSS, conocida como "CSS nivel 3", continúa en desarrollo desde 1998 y hasta el momento sólo se han publicado borradores.

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

1.3. Soporte de CSS en los navegadores

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

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

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

Navegador	Motor	CSS 1	CSS 2.1	CSS 3
Google Chrome	WebKit	Completo desde la versión 85 del motor	Completo	Todos los selectores, pseudo-clases y muchas propiedades

Navegador	Motor	CSS 1	CSS 2.1	CSS 3
Internet Explorer	Trident	Completo desde la versión 7.0 del navegador	Completo	Todos los selectores, pseudo-clases y muchas propiedades a partir de la versión 10.0 del navegador
Firefox	Gecko	Completo desde la versión 1.0 del navegador	Completo	Todos los selectores, pseudo-clases y muchas propiedades
Safari	WebKit	Completo desde la versión 85 del motor	Completo	Todos los selectores, pseudo-clases y muchas propiedades
Opera	Presto	Completo desde la versión 1.0 del navegador	Completo	Todos los selectores, pseudo-clases y muchas propiedades

Los navegadores Firefox, Chrome, Safari y Opera son los más avanzados en el soporte de CSS, ya que incluyen muchos elementos de la futura versión CSS 3 y un soporte casi perfecto de la actual versión 2.1.

Por su parte, el navegador Internet Explorer sólo puede considerarse adecuado desde el punto de vista de CSS a partir de su versión 7. Internet Explorer 6, utilizado todavía por un número no despreciable de usuarios, sufre carencias muy importantes y contiene decenas de errores en su soporte de CSS. Internet Explorer 8 soporta casi todas las propiedades y características de CSS 2.1.

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

1.4. Especificación oficial

La especificación o norma oficial que se utiliza actualmente para diseñar páginas web con CSS es la versión CSS 2.1, actualizada por última vez el 7 de junio de 2011 y que se puede consultar libremente en 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. También existe un [blog oficial en el que se publican todas las novedades relacionadas con el estándar CSS](#).

1.5. Funcionamiento básico de CSS

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=iso-8859-1"
    />

    <title>Ejemplo de estilos sin CSS</title>

</head>

<body>

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

    <p><font color="gray" face="Verdana" size="2">Un párrafo de texto no mu
y 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="text/html; charset=iso-8859-1" /
>

<title>Ejemplo de estilos con CSS</title>

<style type="text/css">

    h1 { color: red; font-family: Arial; font-size: large; }

    p { color: gray; font-family: Verdana; font-size: medium; }

</style>

</head>

<body>

    <h1>Titular de la página</h1>

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

</body>

</html>
```

CSS permite separar los contenidos de la página y 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.

1.6. Cómo incluir CSS en un documento XHTML

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

1.6.1. Incluir CSS en el mismo documento HTML

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

Ejemplo:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" /
>
<title>Ejemplo de estilos CSS en el propio documento</title>
<style type="text/css">
    p { color: black; font-family: Verdana; }
</style>
</head>

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

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

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

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.

1.6.2. Definir CSS en un archivo externo

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

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=iso-8859-1" /
>
<title>Ejemplo de estilos CSS en un archivo externo</title>
<link rel="stylesheet" type="text/css" href="/css/estilos.css" media="scr
een" />
</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=iso-8859-1" /
>

<title>Ejemplo de estilos CSS en un archivo externo</title>
```

```

<style type="text/css" media="screen">

    @import '/css/estilos.css';

</style>

</head>


<body>

<p>Un párrafo de texto.</p>

</body>

</html>

```

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

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

```

@import '/css/estilos.css';

@import "/css/estilos.css";

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

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

```

1.6.3. Incluir CSS en los elementos HTML

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

Ejemplo:

```

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"

    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

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

<head>

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

<title>Ejemplo de estilos CSS en el propio documento</title>

</head>

```

```

<body>

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

</body>

</html>

```

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

1.7. Glosario básico

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

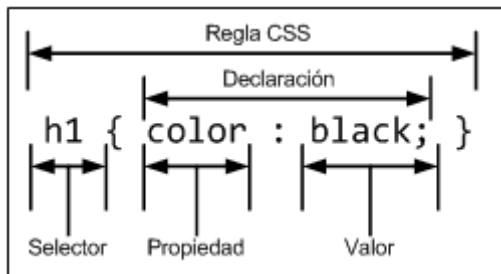


Figura 1.1 Componentes de un estilo CSS básico

Los diferentes términos se definen a continuación:

- **Regla:** cada uno de los estilos que componen una hoja de estilos CSS. Cada regla está compuesta de una parte de "selectores", un símbolo de "llave de apertura" (`{`), otra parte denominada "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.

1.8. Medios CSS

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

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

Medio	Descripción
<code>all</code>	Todos los medios definidos
<code>braille</code>	Dispositivos táctiles que emplean el sistema braille
<code>embosed</code>	Impresoras braille
<code>handheld</code>	Dispositivos de mano: móviles, PDA, etc.
<code>print</code>	Impresoras y navegadores en el modo <i>"Vista Previa para Imprimir"</i>
<code>projection</code>	Proyectores y dispositivos para presentaciones
<code>screen</code>	Pantallas de ordenador
<code>speech</code>	Sintetizadores para navegadores de voz utilizados por personas discapacitadas

Medio	Descripción
tty	Dispositivos textuales limitados como teletipos y terminales de texto
tv	Televisores y dispositivos con resolución baja

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

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

Medio	Continuo / Paginado	Visual / Auditivo / Táctil / Vocal	Mapa de bits / Caracteres	Interactivo / Estático
braille	continuo	táctil	caracteres	ambos
embossed	paginado	táctil	caracteres	estático
handheld	ambos	visual, auditivo, vocal	ambos	ambos
print	paginado	visual	mapa de bits	estático
projection	paginado	visual	mapa de bits	interactivo
screen	continuo	visual, auditivo	mapa de bits	ambos

Medio	Continuo / Paginado	Visual / Auditivo / Táctil / Vocal	Mapa de bits / Caracteres	Interactivo / Estático
speech	continuo	vocal	(no tiene sentido)	ambos
tty	continuo	visual	caracteres	ambos
tv	ambos	visual, auditivo	mapa de bits	ambos

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

1.8.1. Medios definidos con las reglas de tipo @media

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

A continuación se muestra un ejemplo sencillo:

```
@media print {
    body { font-size: 10pt }
}

@media screen {
    body { font-size: 13px }
}

@media screen, print {
    body { line-height: 1.2 }
}
```

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

imprimen sus contenidos, el interlineado del texto debe ser de 1.2 veces el tamaño de letra del texto.

1.8.2. Medios definidos con las reglas de tipo @import

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

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

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

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

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

1.8.3. Medios definidos con la etiqueta <link>

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

```
<link rel="stylesheet" type="text/css" media="screen" href="basico.css" />

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

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

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".

1.8.4. Medios definidos mezclando varios métodos

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

```
<link rel="stylesheet" type="text/css" media="screen" href="basico.css" />

@import url("estilos_seccion.css") screen;

@media print {

    /* Estilos específicos para impresora */
```

```
}
```

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

1.9. Comentarios

CSS permite incluir comentarios entre sus reglas y estilos. Los comentarios son contenidos de texto que el diseñador incluye en el archivo CSS para su propia información y utilidad. Los navegadores ignoran por completo cualquier comentario de los archivos CSS, por lo que es común utilizarlos para estructurar de forma clara los archivos CSS complejos.

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

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

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

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

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  
  
líneas -->
```

1.10. Sintaxis de la definición de cada propiedad CSS

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

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

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

Si el valor permitido se indica como una sucesión de valores separados por una barra simple (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.

Capítulo 2. Selectores

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

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

A un mismo elemento HTML se le pueden 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.

2.1. Selectores básicos

2.1.1. Selector universal

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

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

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

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

2.1.2. Selector de tipo o etiqueta

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

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

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

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

```
h1 {  
    color: red;  
}
```

```
h2 {  
    color: blue;  
}
```

```
p {  
    color: black;  
}
```

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

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

```
font-weight: normal;

font-family: Arial, Helvetica, sans-serif;

}
```

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

```
h1, h2, h3 {

color: #8A8E27;

font-weight: normal;

font-family: Arial, Helvetica, sans-serif;

}
```

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

```
h1, h2, h3 {

color: #8A8E27;

font-weight: normal;

font-family: Arial, Helvetica, sans-serif;

}
```

```
h1 { font-size: 2em; }

h2 { font-size: 1.5em; }

h3 { font-size: 1.2em; }
```

2.1.3. Selector descendente

Selecciona los elementos que se 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`.

2.1.4. Selector de clase

Si se considera el siguiente código HTML de ejemplo:

```
<body>
```

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

<p>Nunc sed lacus et est adipiscing accumsan...</p>

<p>Class aptent taciti sociosqu ad litora...</p>
```

```
</body>
```

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

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

```
<body>

  <p class="destacado">Lorem ipsum dolor sit amet...</p>

  <p>Nunc sed lacus et est adipiscing accumsan...</p>

  <p>Class aptent taciti sociosqu ad litora...</p>

</body>
```

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

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

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

Este tipo de selectores se llaman selectores de clase y son los más utilizados junto con los selectores de ID que se verán a continuación. La principal característica de este selector es que en una misma página HTML varios elementos diferentes pueden 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> a
ccumsan...</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: #f6feda;  
}  
  
.error {  
    color: #930;  
    font-weight: bold;  
}  
  
<span class="error">...</span>  
  
<div class="aviso">...</div>
```

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

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

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

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

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

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

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

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

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

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

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

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

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

```
p, .aviso { ... }
```

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

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

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

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

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

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

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

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

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

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

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

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

2.1.5. Selectores de ID

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

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

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

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

```
<p>Primer párrafo</p>
```

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

```
<p>Tercer párrafo</p>
```

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

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

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

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

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

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

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

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

```
/* Todos los elementos de tipo "p" con atributo id="aviso" */
```

```
p#aviso { ... }
```

```
/* Todos los elementos con atributo id="aviso" que estén dentro  
de cualquier elemento de tipo "p" */
```

```
p #aviso { ... }
```

```
/* Todos los elementos "p" de la página y todos los elementos con  
atributo id="aviso" de la página */
```

```
p, #aviso { ... }
```

2.1.6. Combinación de selectores básicos

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

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

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

Si se modifica el anterior selector:

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

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

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

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

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

2.2. Selectores avanzados

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

2.2.1. Selector de hijos

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

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

```
<p><span>Texto1</span></p>
```

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

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

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

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

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

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

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

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

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

2.2.2. Selector adyacente

El selector adyacente 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>Titulo1</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.

2.2.3. Selector de atributos

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

Los cuatro tipos de selectores de atributos son:

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

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

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

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

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

```
/* Se muestran de color azul todos los enlaces que tengan
```

```

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

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

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

```

2.3. Agrupación de reglas

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

```

h1 { color: red; }

...

h1 { font-size: 2em; }

...

h1 { font-family: Verdana; }

```

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

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

```

h1 {

    color: red;

    font-size: 2em;

    font-family: Verdana;

}

```

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

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

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

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

2.4. Herencia

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=iso-8859-1" /
>
<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=iso-8859-1" /
>

<title>Ejemplo de herencia de estilos</title>

<style type="text/css">

    body { font-family: Arial; color: black; }

    h1 { font-family: Verdana; }

    p { color: red; }

</style>

</head>


<body>

    <h1>Titular de la página</h1>

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

</body>

</html>
```

En el ejemplo anterior, se 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.

2.5. Colisiones de estilos

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

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

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

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

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

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

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

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:

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

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

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

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

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

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

Capítulo 3. Unidades de medida y colores

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.

3.1. Unidades de medida

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

CSS divide 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`).

3.1.1. Unidades absolutas

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.

3.1.2. Unidades relativas

La unidad relativa, 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 ($32\text{px} \times 0.5 = 16\text{px}$). De la misma forma, si se quiere mostrar un margen de `8px` de anchura, se debería utilizar el valor `0.25em`, ya que $32\text{px} \times 0.25 = 8\text{px}$.

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

Como se vio en los capítulos anteriores, 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 $3\text{em} \times 15\text{px} = 45\text{px}$. No obstante, como se ha comentado, los valores que se heredan no son los relativos, sino los valores ya calculados.

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

3.1.3. Porcentajes

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% x 600px = 480px`.

3.1.4. Recomendaciones

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

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.

3.2. Colores

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

3.2.1. Palabras clave

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

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

maroon #800000	red #ff0000	orange #ffa500	yellow #ffff00	olive #808000
purple #800080	fuchsia #ff00ff	white #ffffff	lime #00ff00	green #008000
navy #000080	blue #0000ff	aqua #00ffff	teal #008080	
black #000000	silver #c0c0c0	gray #808080		

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](#).

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/Web_safe).

3.2.2. RGB decimal

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.

3.2.3. RGB porcentual

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%.

3.2.4. RGB hexadecimal

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; }
```

Recuerda que, aunque es el método más complicado para definir un color, se trata del método que utilizan la inmensa mayoría de sitios web, por lo que es imprescindible dominarlo. Afortunadamente, todos los programas de diseño gráfico convierten de forma automática los valores RGB decimales a sus valores RGB hexadecimales, por lo que no tienes que hacer ninguna operación matemática:

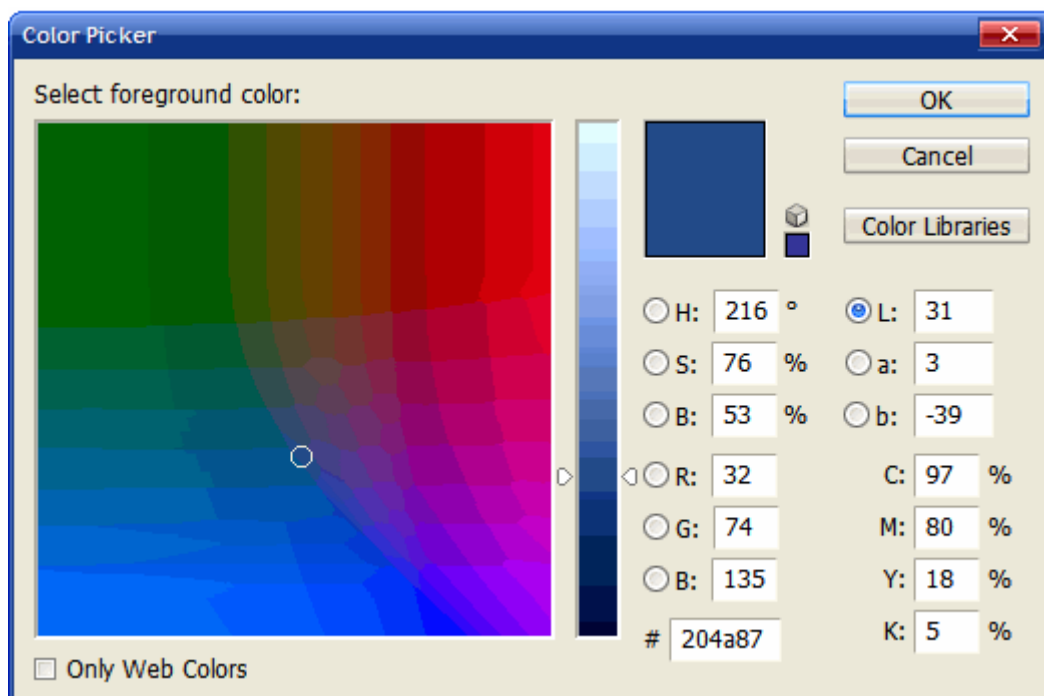


Figura 3.2 Herramienta de color de Photoshop para definir los colores según los modelos RGB, CMYK, Lab, HSB y RGB hexadecimal

El formato RGB hexadecimal es la forma más compacta de indicar un color, ya que incluso es posible comprimir sus valores cuando todas sus componentes son iguales dos a dos:

#AAA = #AAAAAA

#FFF = #FFFFFF

#A0F = #AA00FF

#369 = #336699

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

```
body { background-color: #FFF; color: #000; }
```

```
h1, h2, h3, h4, h5, h6 { color: #C00; }
```

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.

3.2.5. Colores del sistema

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

Existen varios colores definidos, como por ejemplo `ActiveBorder`, que hace referencia al color del borde de las ventanas activas. Consulta la [lista completa de colores del sistema](#).

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

3.2.6. Colores web safe

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

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

Hoy en día, su importancia ha descendido notablemente, ya que prácticamente todos los usuarios utilizan dispositivos con una profundidad de color de 16 y 32 bits. No obstante, el auge en el uso de los dispositivos móviles hace que siga

siendo un tema a considerar, ya que las pantallas de muchos móviles sólo pueden representar un número reducido de colores.

Capítulo 4. Modelo de cajas

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:

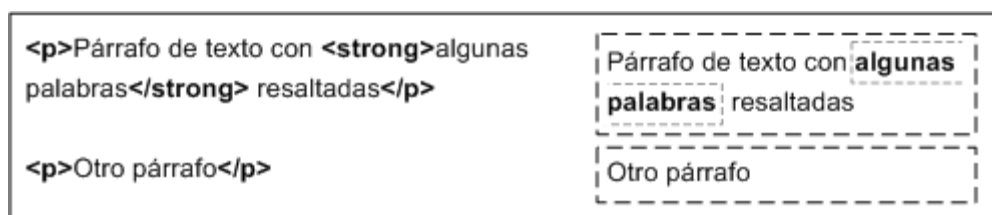


Figura 4.1 Las cajas se crean automáticamente al definir cada elemento HTML

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](http://www.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:

THE CSS BOX MODEL HIERARCHY

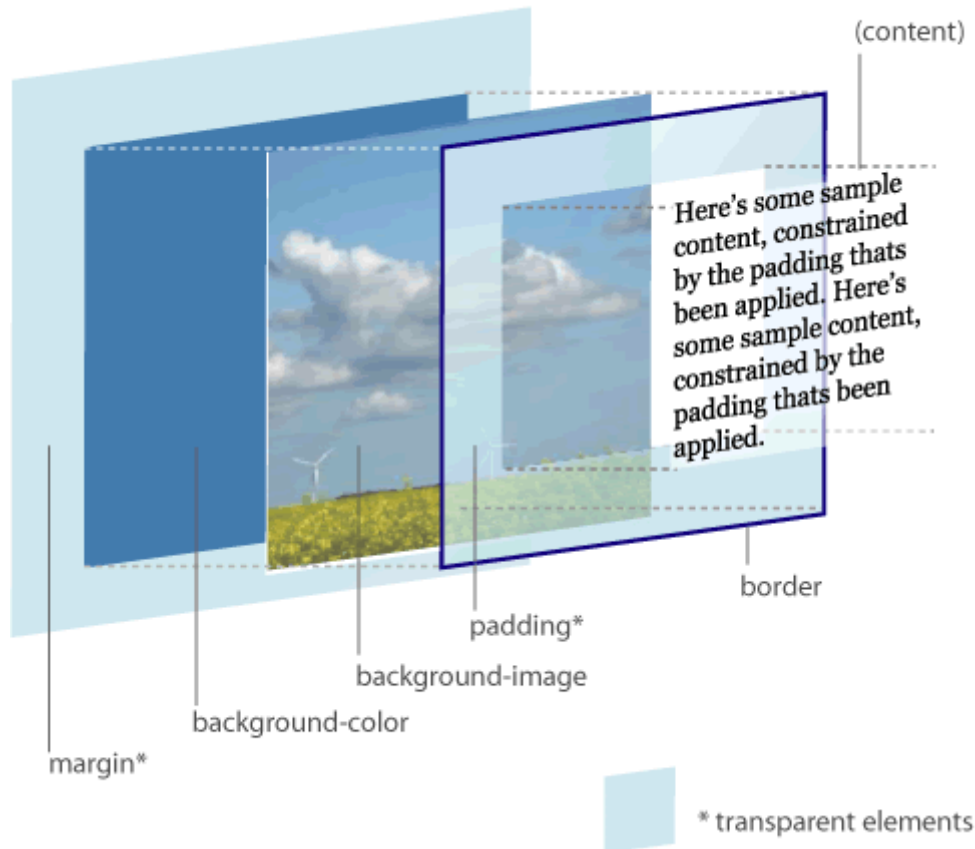


Figura 4.3 Representación tridimensional del box model de CSS

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

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

- Contenido (*content*): se trata del contenido HTML del elemento (las palabras de un párrafo, una imagen, el texto de una lista de elementos, etc.)
- Relleno (*padding*): espacio libre opcional 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.

4.1. Anchura y altura

4.1.1. Anchura

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

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

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

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

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

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

```
...
```

```
</div>
```

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

4.1.2. Altura

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

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

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

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

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

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

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

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

4.2. Margen y relleno

4.2.1. Margen

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

Propiedades	<code>margin-top</code> , <code>margin-right</code> , <code>margin-bottom</code> , <code>margin-left</code>
Valores	<code>unidad de medida</code> <code>porcentaje</code> <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
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:

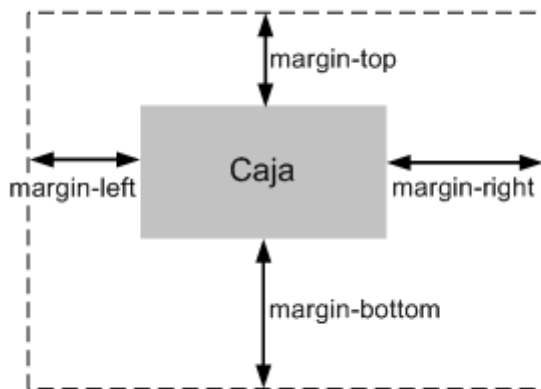


Figura 4.4 Las cuatro propiedades relacionadas con los márgenes

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, consectetur 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 ultricies,

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 consectetur tincidunt,

risus turpis laoreet elit, ut tincidunt risus sem et nunc.`</p>`

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

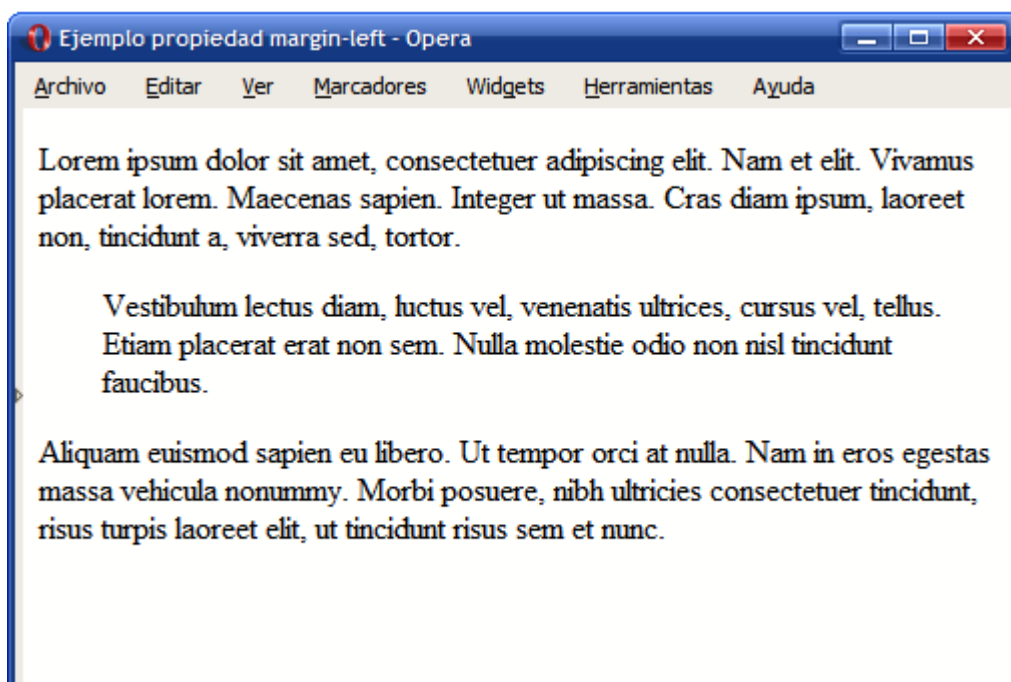


Figura 4.5 Ejemplo de propiedad margin-left

Algunos diseñadores web utilizan la etiqueta `<blockquote>` para tabular los contenidos de los párrafos. Se trata de un error grave porque HTML no debe utilizarse para controlar el aspecto de los elementos. CSS es el único responsable de establecer el estilo de los elementos, por lo que en vez de utilizar la etiqueta `<blockquote>` de HTML, debería utilizarse la propiedad `margin-left` de CSS.

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

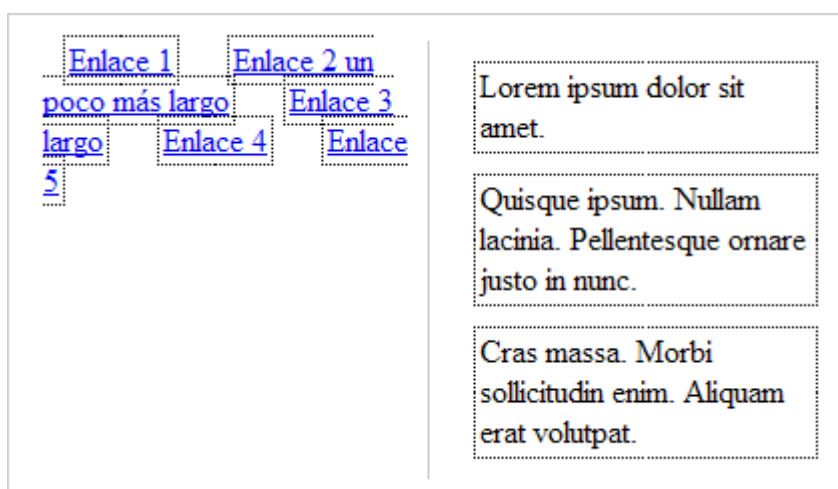


Figura 4.6 Los márgenes verticales sólo se aplican a los elementos de bloque e imágenes

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

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`.

Propiedad	<code>margin</code>
Valores	(<code>unidad de medida</code> <code>porcentaje</code> <code>auto</code>) {1, 4} <code>inherit</code>
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.

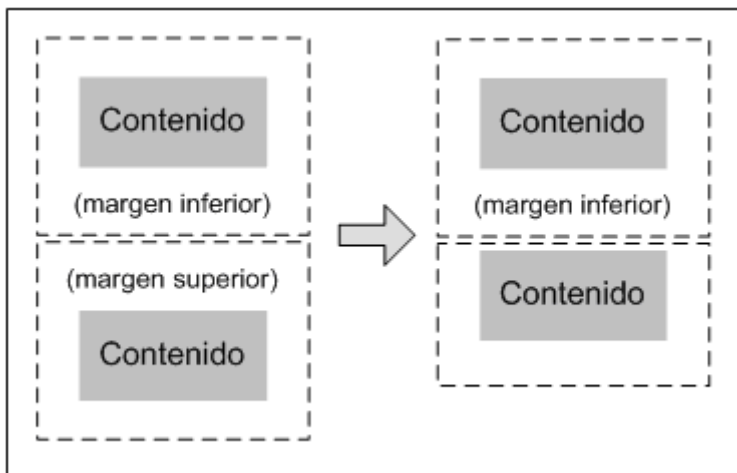


Figura 4.7 Fusión automática de los márgenes verticales

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:

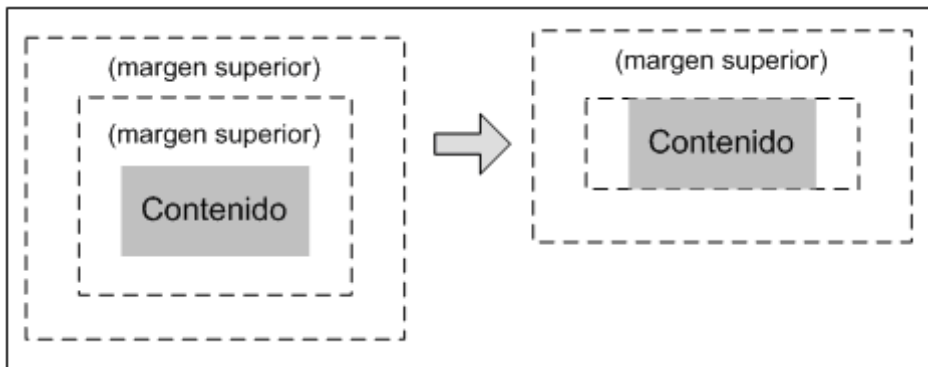


Figura 4.8 Fusión de los márgenes de los elementos interiores

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

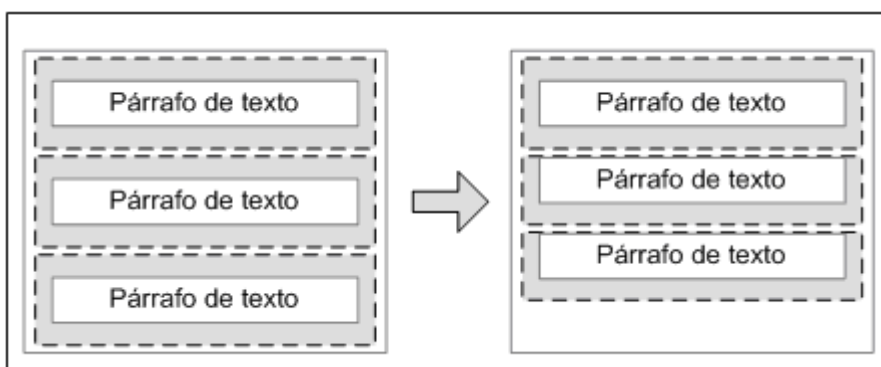


Figura 4.9 Motivo por el que se fusionan automáticamente los márgenes verticales

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.

4.2.2. Relleno

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

Propiedades	padding-top, padding-right, padding-bottom, padding-left
Valores	unidad de medida porcentaje inherit
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:

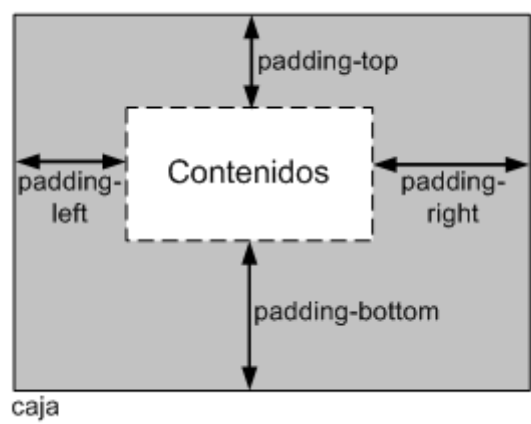


Figura 4.10 Las cuatro propiedades relacionadas con los rellenos

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.

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

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

```
body {padding: 2em} /* Todos los rellenos valen 2em */
```

```
body {padding: 1em 2em} /* Superior e inferior = 1em, Izquierdo y derecho = 2em */
```

```
body {padding: 1em 2em 3em} /* Superior = 1em, derecho = 2em, inferior = 3em, izquierdo = 2em */
```

```
body {padding: 1em 2em 3em 4em} /* Superior = 1em, derecho = 2em, inferior = 3em, izquierdo = 4em */
```

4.3. Bordes

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.

4.3.1. Anchura

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

Propiedades	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:

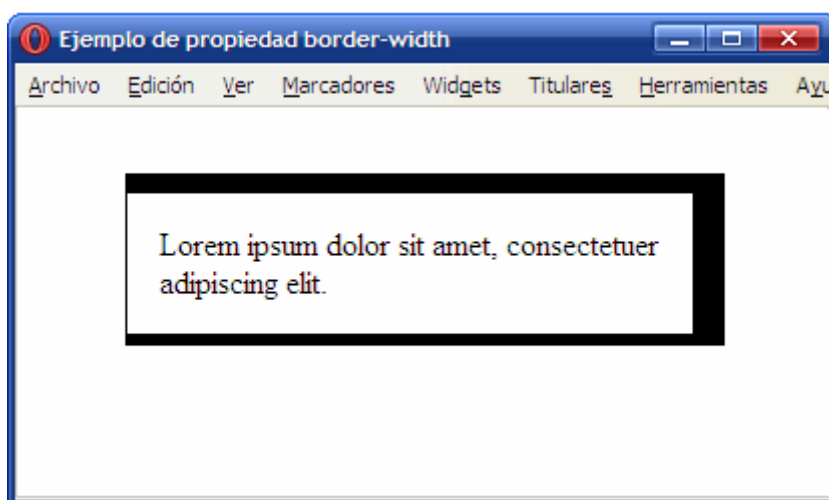


Figura 4.11 Ejemplo de propiedad border-width

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`:

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

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

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

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

Si se indican tres valores, el primero se aplica al borde superior, el segundo se aplica al borde izquierdo y derecho y el tercer valor se aplica al borde inferior.

Si se indican los cuatro valores, el orden de aplicación es superior, derecho, inferior e izquierdo.

4.3.2. Color

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

Propiedades	border-top-color, border-right-color, border-bottom-color, border-left-color
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:

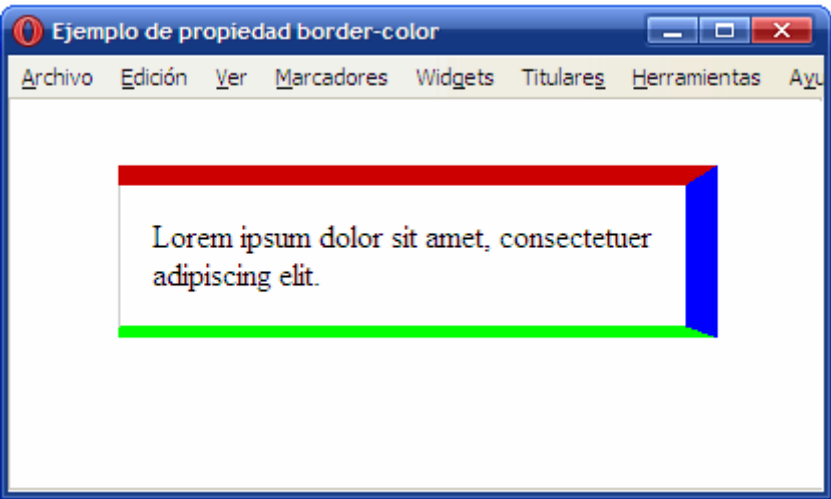


Figura 4.12 Ejemplo de propiedad border-color

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

```
div {  
    border-top-color: #CC0000;
```



```
border-right-color: blue;

border-bottom-color: #00FF00;

border-left-color: #CCC;

}
```

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

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

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

4.3.3. Estilo

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

Propiedades	<code>border-top-style</code> , <code>border-right-style</code> , <code>border-bottom-style</code> , <code>border-left-style</code>
Valores	<code>none</code> <code>hidden</code> <code>dotted</code> <code>dashed</code> <code>solid</code> <code>double</code> <code>groove</code> <code>ridge</code> <code>inset</code> <code>outset</code> <code>inherit</code>
Se aplica a	Todos los elementos

Propiedades	border-top-style, border-right-style, border-bottom-style, border-left-style
Valor inicial	none
Descripción	Establece el estilo de cada uno de los cuatro bordes de los elementos

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

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

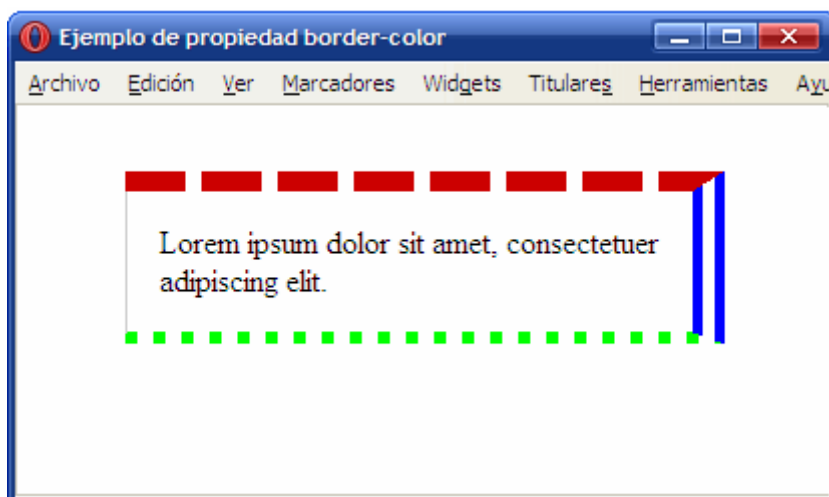


Figura 4.13 Ejemplo de propiedad border-style

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;
}
```

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

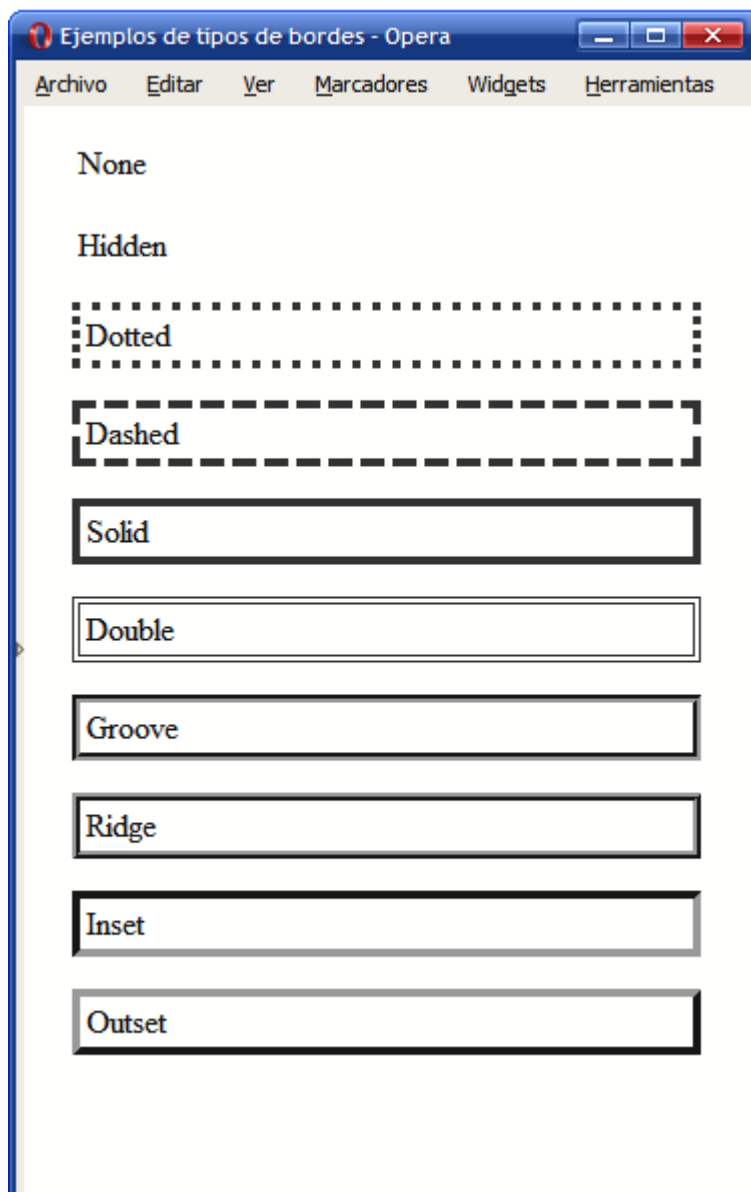


Figura 4.14 Tipos de bordes definidos por CSS

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`:

Propiedad	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".

4.3.4. Propiedades shorthand

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

Propiedades	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	-

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

Propiedad	<code>border</code>
Valores	(unidad de medida_borde color_borde <code>estilo_borde</code>) <code>inherit</code>

Propiedad	border
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;
```

4.4. Margen, relleno, bordes y modelo de cajas

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

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

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

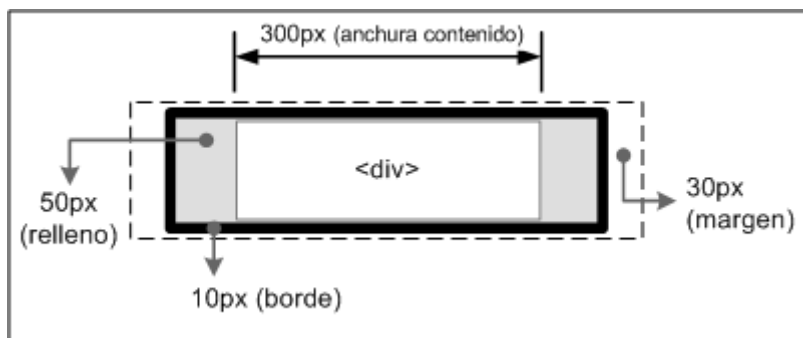


Figura 4.15 La anchura total de un elemento tiene en cuenta los márgenes, rellenos y bordes

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:

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

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

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

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

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

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

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

```
<?xml version="1.0" encoding="UTF-8"?>

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

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

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

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

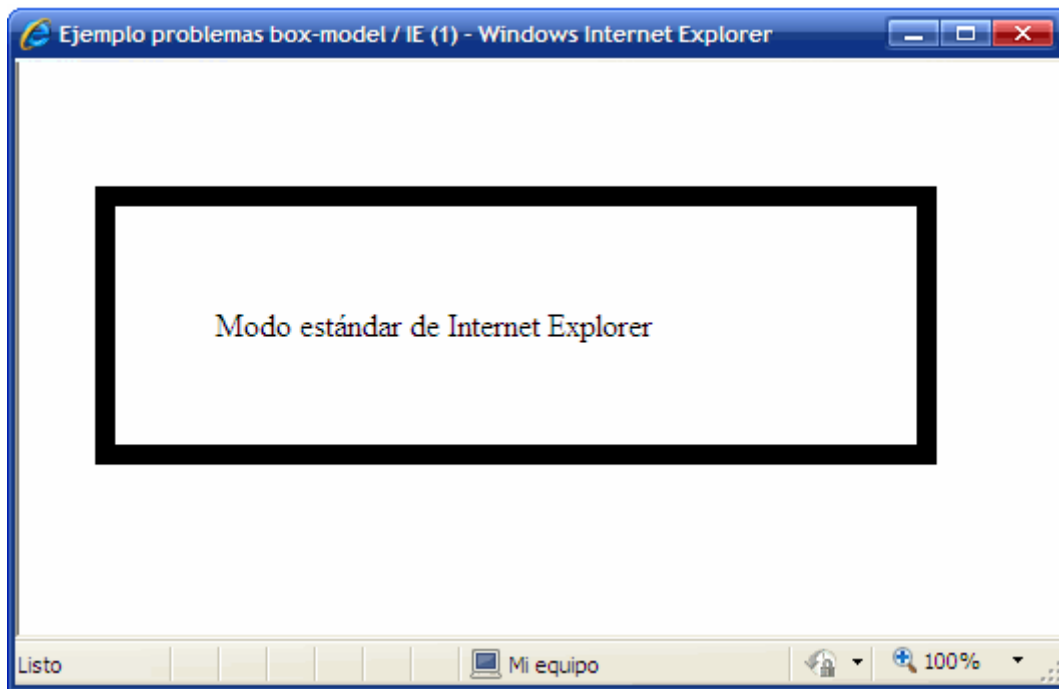


Figura 4.16 Internet Explorer 6 en modo estándar

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

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

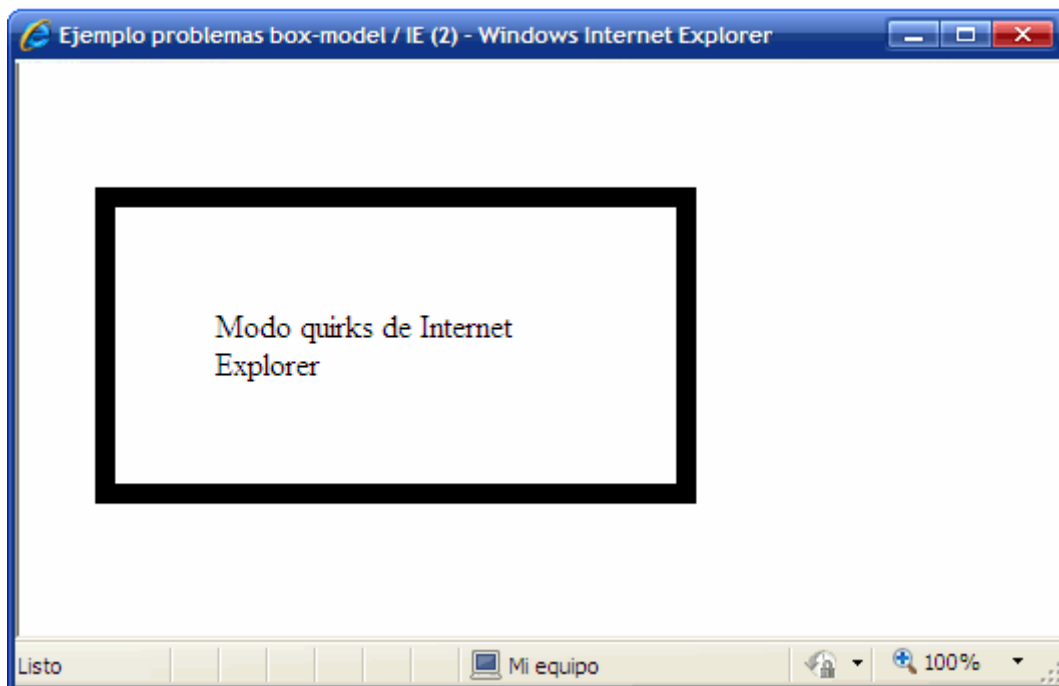


Figura 4.17 Internet Explorer 6 en modo quirks

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

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

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

4.4.1. Los modos de compatibilidad de Internet Explorer 8

El navegador Internet Explorer 8 introduce el concepto de "*compatibilidad de la página*" para asegurar que todas las páginas HTML se vean correctamente en cualquier versión de ese navegador. En realidad, esta nueva característica es una mejora del *modo quirks* explicado anteriormente.

Internet Explorer 8, a diferencia de sus versiones anteriores, soporta completamente el estándar CSS 2.1. Sin embargo, muchos sitios web se diseñaron para Internet Explorer 6 y 7, por lo que incluyen trucos, *hacks* y filtros que arreglan los errores y carencias de esas versiones del navegador.

Para evitar que las páginas diseñadas para navegadores anteriores se vean mal en esta nueva versión, Internet Explorer 8 incluye la opción de "compatibilidad de la página", que permite indicar la versión de Internet Explorer para la que la página ha sido diseñada.

De esta forma, si la página no se visualiza correctamente en Internet Explorer 8, se puede indicar al navegador que la muestre como si fuera Internet Explorer 6 o 7. En realidad, Internet Explorer 8 incluye seis modos de funcionamiento:

- **Modo IE5:** la página se muestra según el modo *quirks* de Internet Explorer 7, que es casi idéntico a como se veían las páginas en el navegador Internet Explorer 5.
- **Modo IE7:** la página se muestra en el modo estándar de Internet Explorer 7, sin importar si la página contiene o no la directiva `<!DOCTYPE>`.
- **Modo IE8:** los contenidos se muestran en el modo estándar de Internet Explorer 8, que es el más parecido al del resto de navegadores que soportan los estándares (Firefox, Opera, Safari y Google Chrome).
- **Emular el modo IE7:** el navegador decide cómo mostrar los contenidos a partir de la directiva `<!DOCTYPE>` de la página. Si esa directiva es una de las que activan el modo estándar, la página se muestra en el modo estándar de Internet Explorer 7. En otro caso, se muestra en el modo *quirks* de Internet Explorer 5. Este modo es el más útil para la mayoría de sitios web.

- **Emular el modo IE8:** el navegador decide cómo mostrar los contenidos a partir de la directiva `<!DOCTYPE>` de la página. Si esa directiva es una de las que activan el modo estándar, la página se muestra en el modo estándar de Internet Explorer 8. En otro caso, se muestra en el modo *quirks* de Internet Explorer 5.
- **Modo límite ("edge mode"):** indica a Internet Explorer que los contenidos se deben mostrar en el modo de compatibilidad más avanzado disponible. Actualmente, este modo es equivalente al modo IE8. Si las futuras versiones Internet Explorer 9 y 10 incluyeran mejor compatibilidad, las páginas se visualizarían en ese modo avanzado de compatibilidad.

El modo de compatibilidad de la página se indica mediante una nueva etiqueta `<meta>` con la propiedad `X-UA-Compatible` y cuyo valor es el que utiliza Internet Explorer 8 para determinar el modo que se utiliza:

```
<!-- Modo IE5 -->
```

```
<head>
```

```
    <meta http-equiv="X-UA-Compatible" content="IE=5" />
```

```
    ...
```

```
</head>
```

```
<!-- Modo IE7 -->
```

```
<head>
```

```
    <meta http-equiv="X-UA-Compatible" content="IE=7" />
```

```
    ...
```

```
</head>
```

```
<!-- Modo IE8 -->
```

```
<head>
```

```
    <meta http-equiv="X-UA-Compatible" content="IE=8" />
```

```
    ...
```

```
</head>
```

```
<!-- Emular el modo IE7 -->
```

```
<head>
```

```

<meta http-equiv="X-UA-Compatible" content="IE=EmulateIE7" />

...

</head>

<!-- Emular el modo IE8 -->

<head>

  <meta http-equiv="X-UA-Compatible" content="IE=EmulateIE8" />

  ...

</head>

<!-- Modo límite -->

<head>

  <meta http-equiv="X-UA-Compatible" content="IE=edge" />

  ...

</head>

```

No obstante, esta opción de compatibilidad de la página debe entenderse como una solución temporal que evita que los sitios web se vean mal en Internet Explorer 8. La única solución correcta a largo plazo consiste en actualizar las páginas para que sus diseños sigan los estándares web.

4.5. Fondos

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

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

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

La propiedad `background-color` permite mostrar un color de fondo sólido en la caja de un elemento. Esta propiedad no permite crear degradados ni ningún otro efecto avanzado.

Propiedad	background-color
Valores	color transparent inherit
Se aplica a	Todos los elementos
Valor inicial	transparent
Descripción	Establece un color de fondo para los elementos

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

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

Para crear efectos gráficos avanzados, es necesario utilizar la propiedad `background-image`, que permite mostrar una imagen como fondo de la caja de cualquier elemento:

Propiedad	background-image
Valores	url none inherit
Se aplica a	Todos los elementos
Valor inicial	none

Propiedad	background-image
Descripción	Establece una imagen como fondo para los elementos

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

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

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

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

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

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

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

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

Imagen original



Figura 4.18 Imagen original utilizada para el fondo de la página

Reglas CSS

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

Resultado

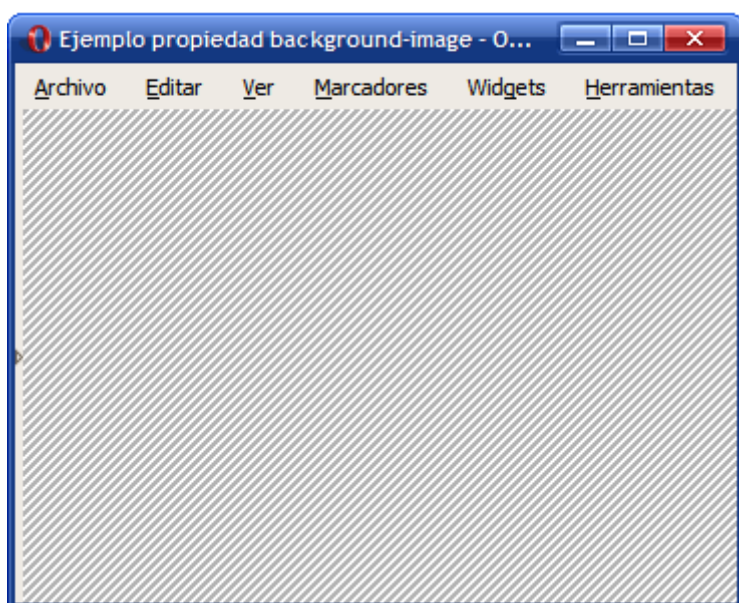


Figura 4.19 Página con una imagen de fondo

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

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

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

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

imagen sólo horizontalmente y el valor `repeat-y` repite la imagen solamente de forma vertical.

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

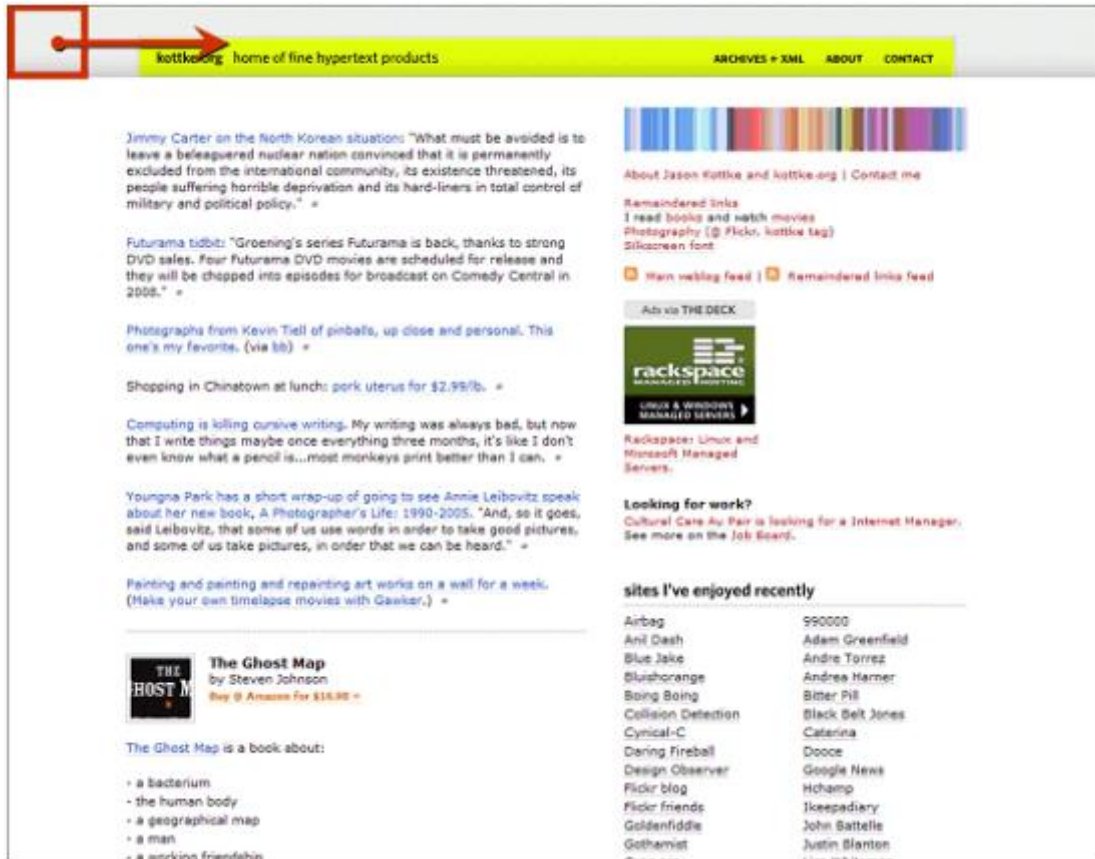


Figura 4.20 Uso de `repeat-x` en la página de Kottke.org

Las reglas CSS definidas para la cabecera son:

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

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

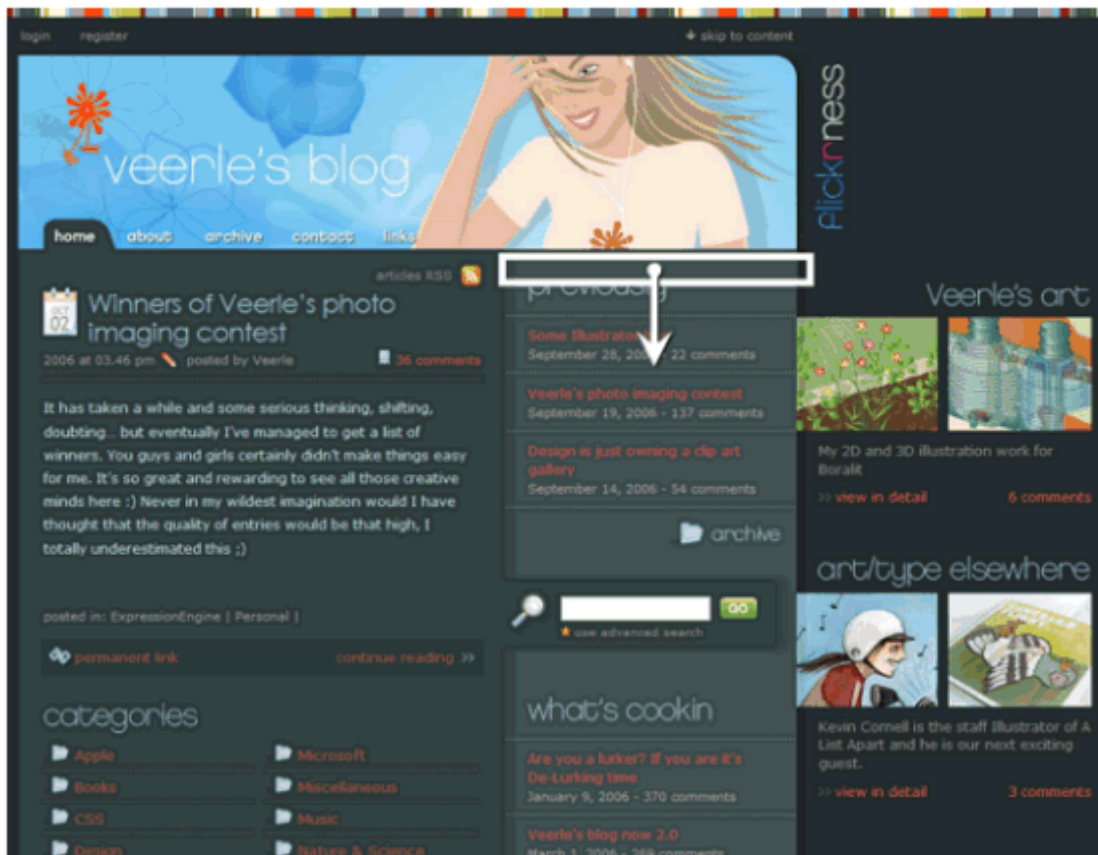


Figura 4.21 Uso de repeat-y en la página de Veerle.duoh.com

Las reglas CSS definidas para esa columna de contenidos son:

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

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

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

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

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

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

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

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

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

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

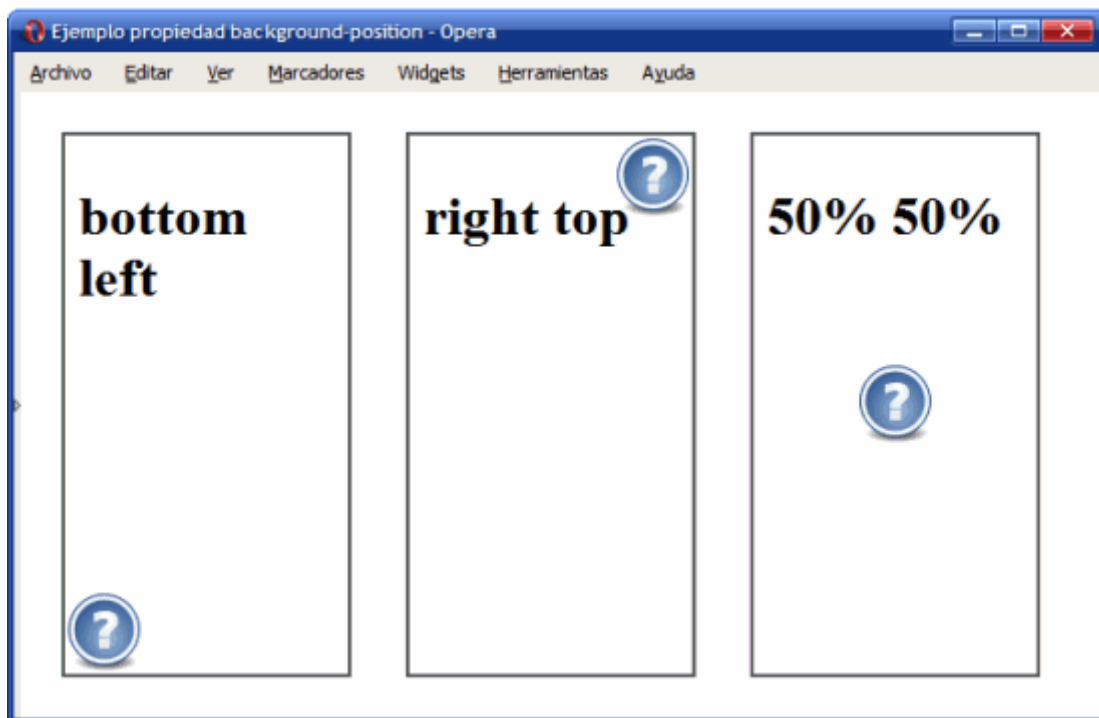


Figura 4.22 Ejemplo de propiedad background-position

Las reglas CSS del ejemplo anterior se muestran a continuación:

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

```
<div id="caja1"><h1>bottom left</h1></div>
```

```
<div id="caja2"><h1>right top</h1></div>
```

```
<div id="caja3"><h1>50% 50%</h1></div>
```

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

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

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

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

Propiedad	<code>background</code>
Valores	(<code>background-color</code> <code>background-image</code> <code>background-repeat</code> <code>background-attachment</code> <code>background-position</code>) <code>inherit</code>

Propiedad	background
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 propie
dades */
```

```
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;
```

Capítulo 5. Posicionamiento y visualización

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

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

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

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

5.1. Tipos de elementos

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

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

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

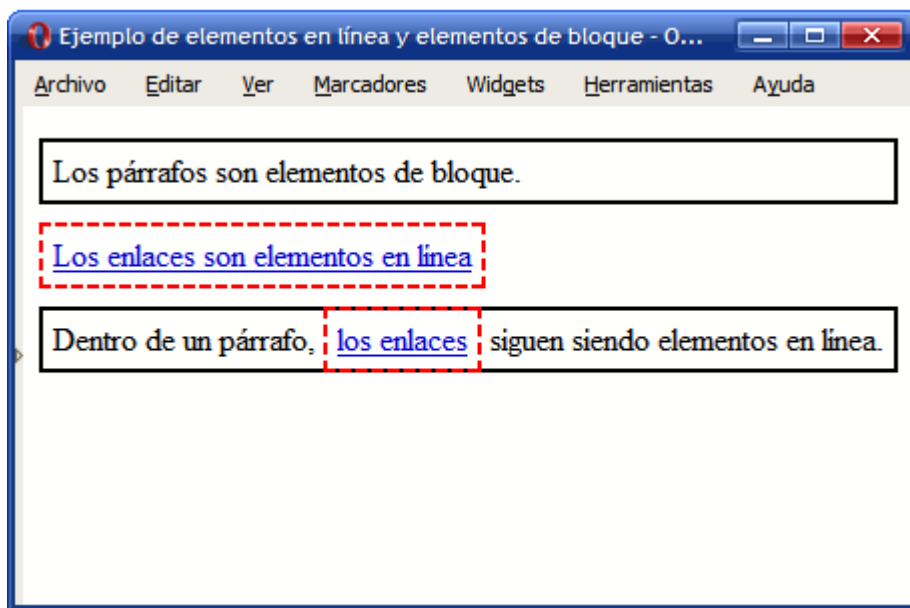


Figura 5.1 Cajas creadas por los elementos de línea y los elementos de bloque

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

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

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

Por sus características, los elementos de bloque no pueden insertarse dentro de elementos en línea y tan sólo pueden aparecer dentro de otros elementos de bloque. En cambio, un elemento en línea puede aparecer tanto dentro de un elemento de bloque como dentro de otro elemento en línea.

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

Los elementos de bloque definidos por HTML son: `address`, `blockquote`, `center`, `dir`, `div`, `dl`, `fieldset`, `form`, `h1`, `h2`, `h3`, `h4`, `h5`, `h6`, `hr`, `isindex`, `menu`, `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`.

5.2. Posicionamiento

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

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

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

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

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

Propiedad	<code>position</code>
Valores	<code>static</code> <code>relative</code> <code>absolute</code> <code>fixed</code> <code>inherit</code>
Se aplica a	Todos los elementos

Propiedad	position
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:

Propiedades	top, right, bottom, left
Valores	unidad de medida porcentaje auto inherit

Propiedades	top, right, bottom, left
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 padre posicionado.

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

5.3. Posicionamiento normal

El posicionamiento normal o estático es el modelo que utilizan por defecto los navegadores para mostrar los elementos de las páginas. En este modelo, 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.

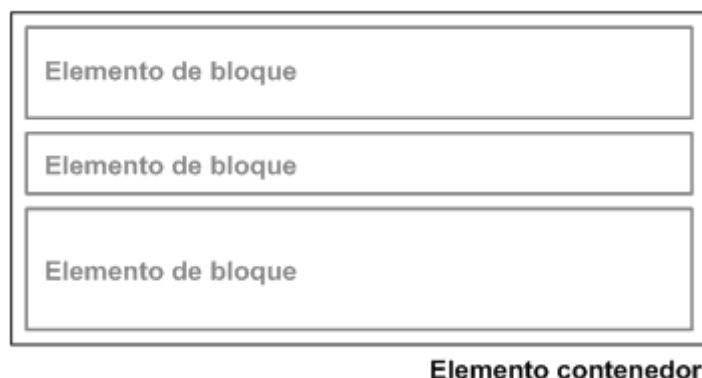


Figura 5.2 Posicionamiento normal de los elementos de bloque

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.

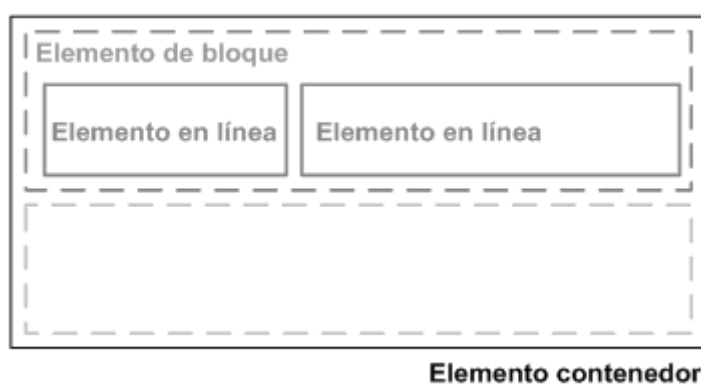


Figura 5.3 Posicionamiento normal de los elementos en línea

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.

5.4. Posicionamiento relativo

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 diseñar 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.

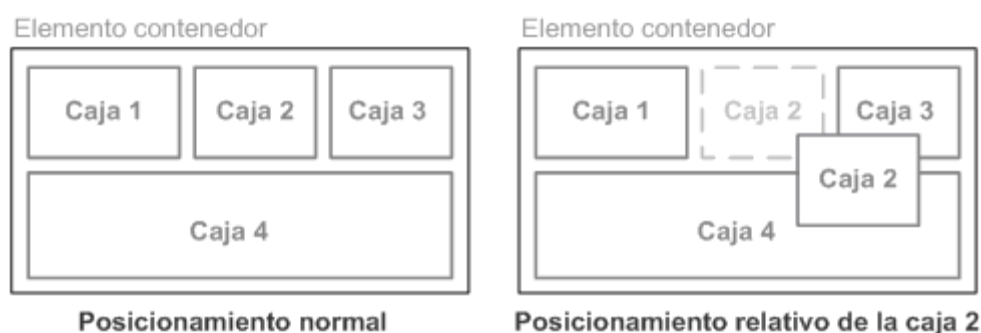


Figura 5.4 Ejemplo de posicionamiento relativo de un elemento

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 $-\text{right}$. Igualmente, si sólo el valor de `right` es `auto`, su valor real es $-\text{left}$.

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

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

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

de la propiedad `left`. De la misma forma, si el valor de `direction` es `rtl`, se ignora el valor de `left` y sólo se tiene en cuenta el valor de `right`.

El siguiente ejemplo muestra tres imágenes posicionadas de forma normal:



Figura 5.5 Elementos posicionados de forma normal

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

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

```

```

```

```

```

```

El aspecto que muestran ahora las imágenes es el siguiente:

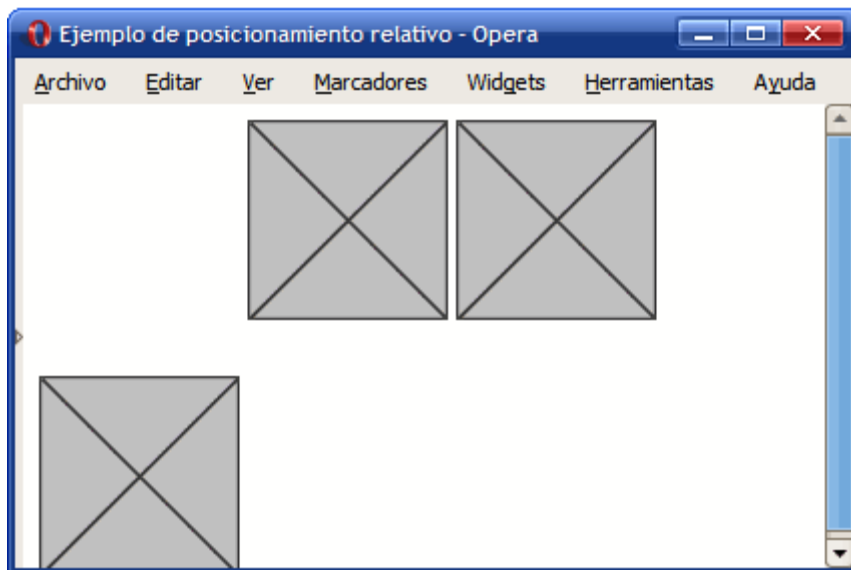


Figura 5.6 Elemento posicionado de forma relativa

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

5.5. Posicionamiento absoluto

El posicionamiento absoluto se emplea para establecer de forma 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:

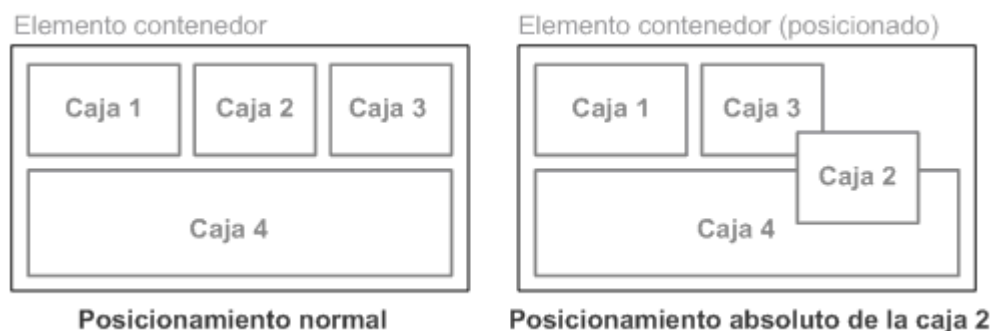


Figura 5.7 Ejemplo de posicionamiento absoluto de un elemento

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:

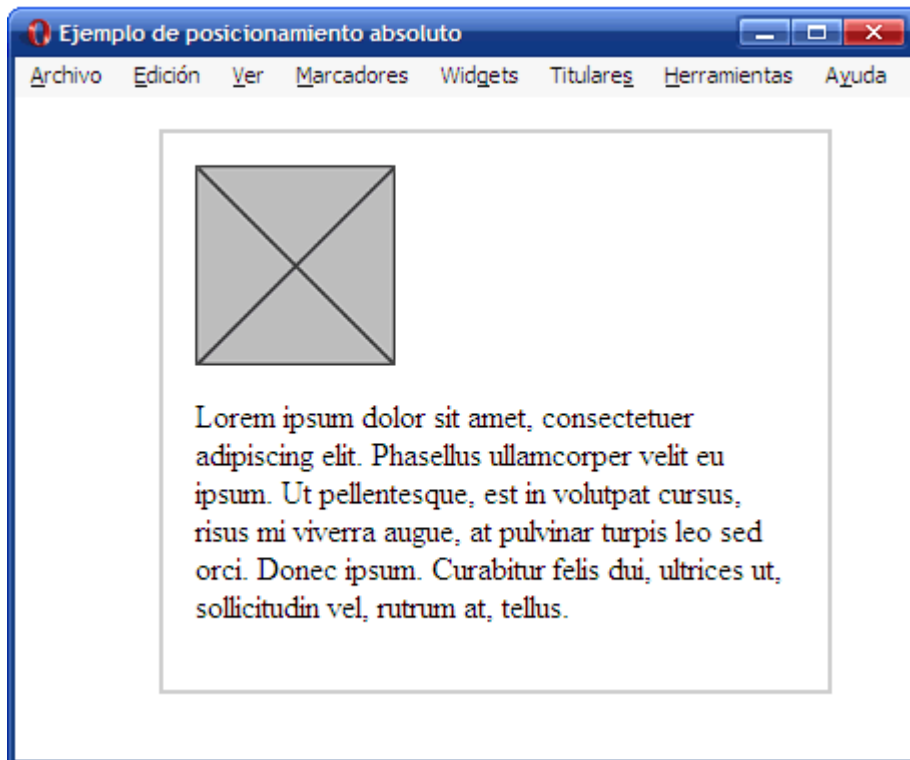


Figura 5.8 Situación original antes de modificar el posicionamiento

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

```
div {  
    border: 2px solid #CCC;  
    padding: 1em;  
    margin: 1em 0 1em 4em;  
    width: 300px;  
}
```

```
<div>  
      
    <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Phasellus  
    ullamcorper velit eu ipsum. Ut pellentesque, est in volutpat cursus, risu  
    s  
    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:

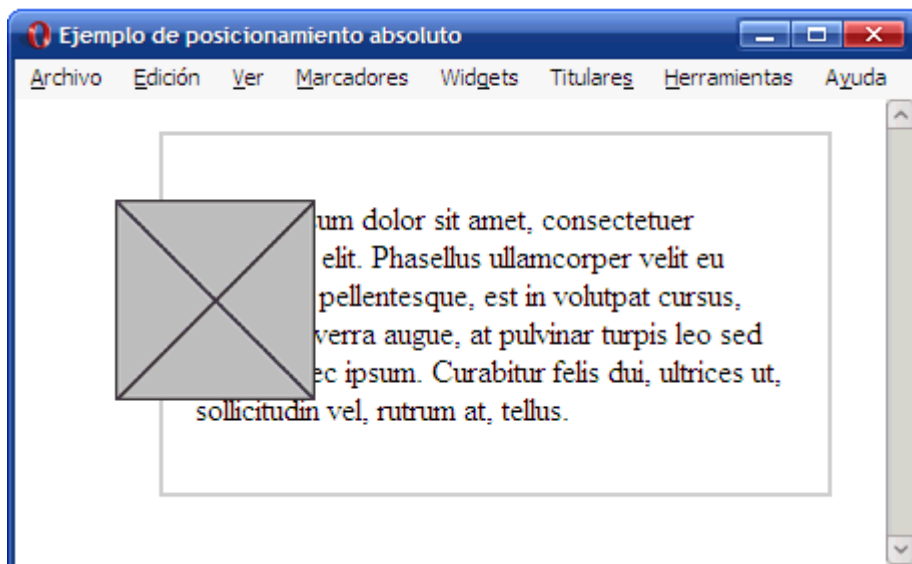


Figura 5.9 Imagen posicionada de forma absoluta

La imagen posicionada de forma absoluta no toma como referencia su elemento contenedor `<div>`, sino la ventana del navegador, tal y como demuestra la siguiente imagen:

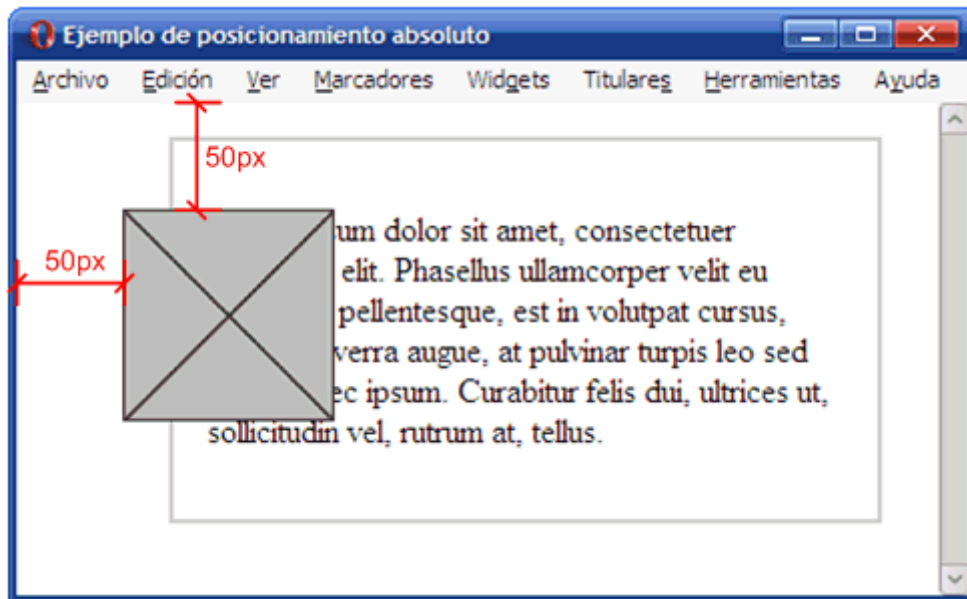


Figura 5.10 La referencia del posicionamiento absoluto es la ventana del navegador

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;

}
```

La siguiente imagen muestra el resultado obtenido:

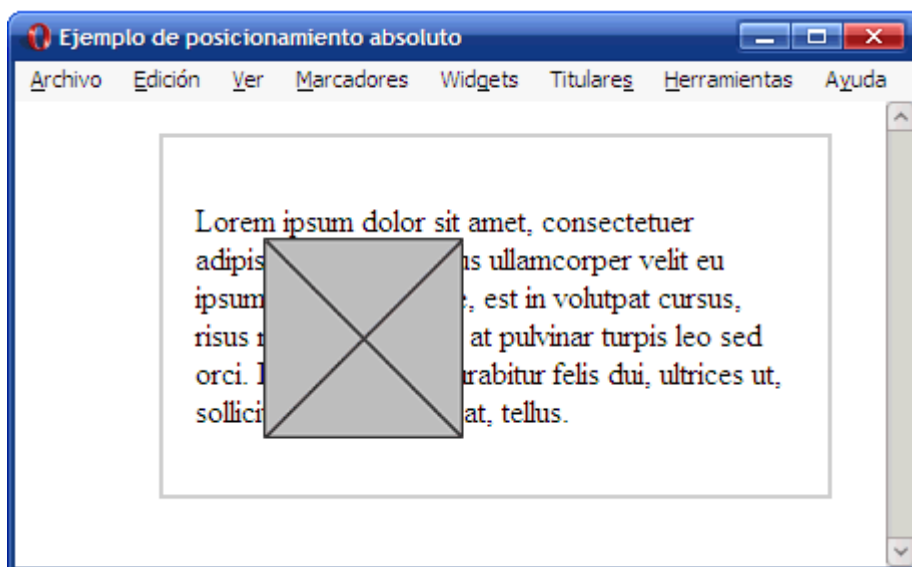


Figura 5.11 Imagen posicionada de forma absoluta

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:

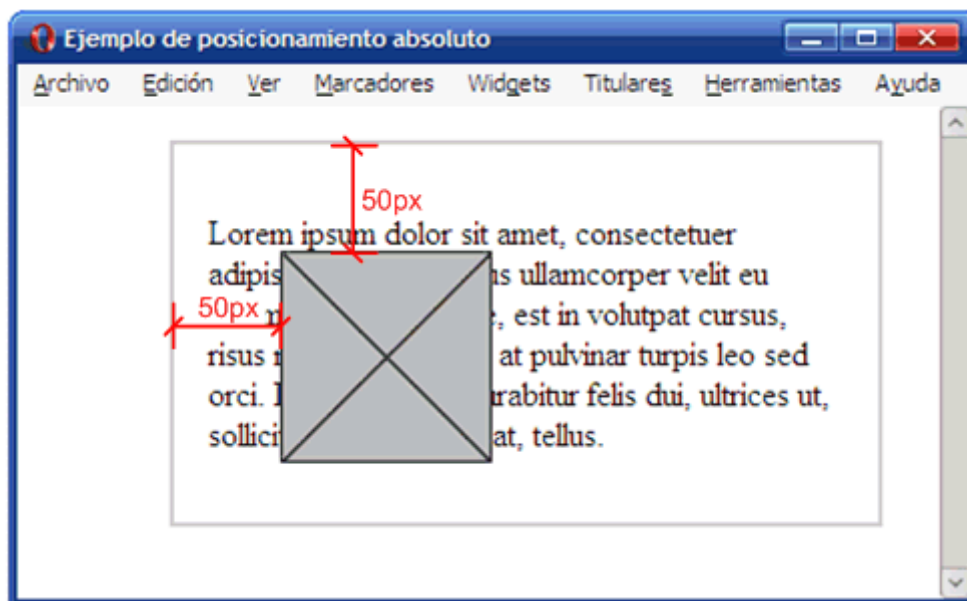


Figura 5.12 La referencia del posicionamiento absoluto es el elemento contenedor de la imagen

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.

5.6. Posicionamiento fijo

El estándar CSS considera que el posicionamiento fijo es un caso particular del posicionamiento absoluto, ya que sólo se diferencian en el comportamiento de las cajas posicionadas.

Cuando una caja se posiciona de forma fija, la forma de obtener el origen de coordenadas para interpretar su desplazamiento es idéntica al posicionamiento absoluto. De hecho, si el usuario no mueve la página HTML en la ventana del navegador, no existe ninguna diferencia entre estos dos modelos de posicionamiento.

La principal característica de una caja posicionada de forma fija es que su posición es inamovible dentro de la ventana del navegador. El posicionamiento fijo hace que las cajas no modifiquen su posición ni aunque el usuario suba o baje la página en la ventana de su navegador.

Si la página se visualiza en un medio paginado (por ejemplo en una impresora) las cajas posicionadas de forma fija se repiten en todas las páginas. Esta característica puede ser útil para crear encabezados o pies de página en páginas HTML preparadas para imprimir.

El posicionamiento fijo apenas se ha utilizado en el diseño de páginas web hasta hace poco tiempo porque el navegador Internet Explorer 6 y las versiones anteriores no lo soportan.

5.7. Posicionamiento flotante

El posicionamiento flotante es el más difícil de comprender pero al mismo tiempo es el más utilizado. La mayoría de estructuras de las páginas web complejas están diseñadas con el posicionamiento flotante, como se verá más adelante.

Cuando una caja se posiciona con el modelo de posicionamiento flotante, automáticamente se convierte en una *caja flotante*, lo que significa que se desplaza hasta la zona más a la izquierda o más a la derecha de la posición en la que originalmente se encontraba.

La siguiente imagen muestra el resultado de posicionar de forma flotante hacia la derecha la caja 1:

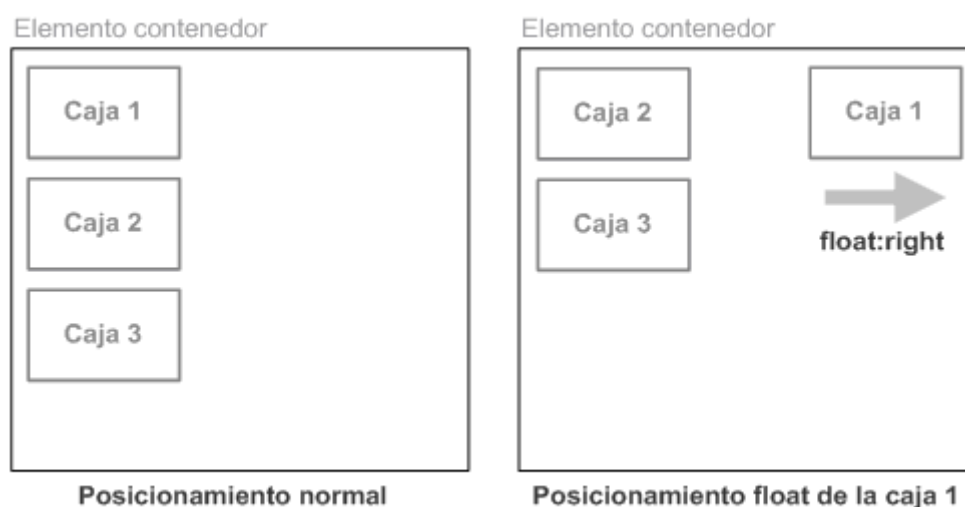


Figura 5.13 Ejemplo de posicionamiento float de una caja

Cuando se posiciona una caja de forma flotante: * La caja deja de pertenecer al flujo normal de la página, lo que significa que el resto de cajas ocupan el lugar dejado por la caja flotante. * La caja flotante se posiciona lo más a la izquierda o lo más a la derecha posible de la posición en la que se encontraba originalmente.

Si en el anterior ejemplo la caja 1 se posiciona de forma flotante hacia la izquierda, el resultado es el que muestra la siguiente imagen:

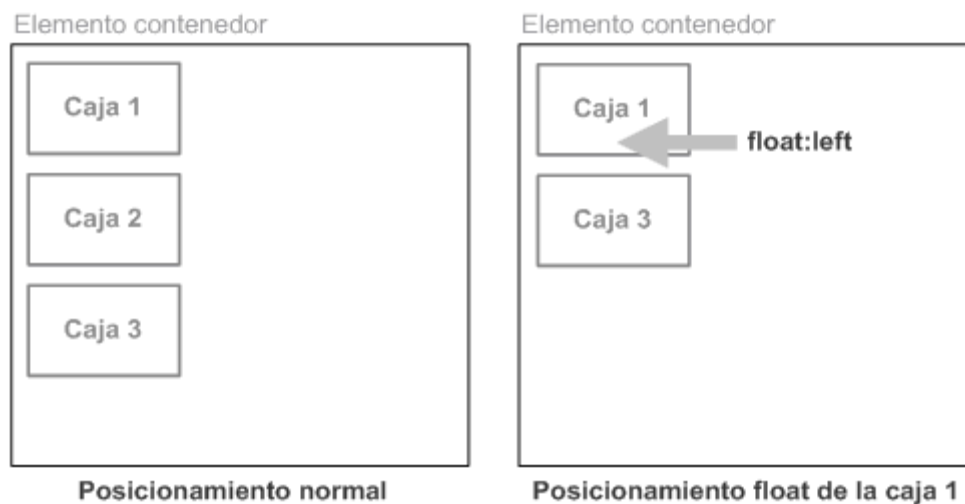


Figura 5.14 Ejemplo de posicionamiento float de una caja

La caja 1 es de tipo flotante, por lo que *desaparece del flujo normal* de la página y el resto de cajas ocupan su lugar. El resultado es que la caja 2 ahora se muestra donde estaba la caja 1 y la caja 3 se muestra donde estaba la caja 2.

Al mismo tiempo, la caja 1 se desplaza todo lo posible hacia la izquierda de la posición en la que se encontraba. El resultado es que la caja 1 se muestra encima de la nueva posición de la caja 2 y tapa todos sus contenidos.

Si existen otras cajas flotantes, al posicionar de forma flotante otra caja, se tiene en cuenta el sitio disponible. En el siguiente ejemplo se posicionan de forma flotante hacia la izquierda las tres cajas:

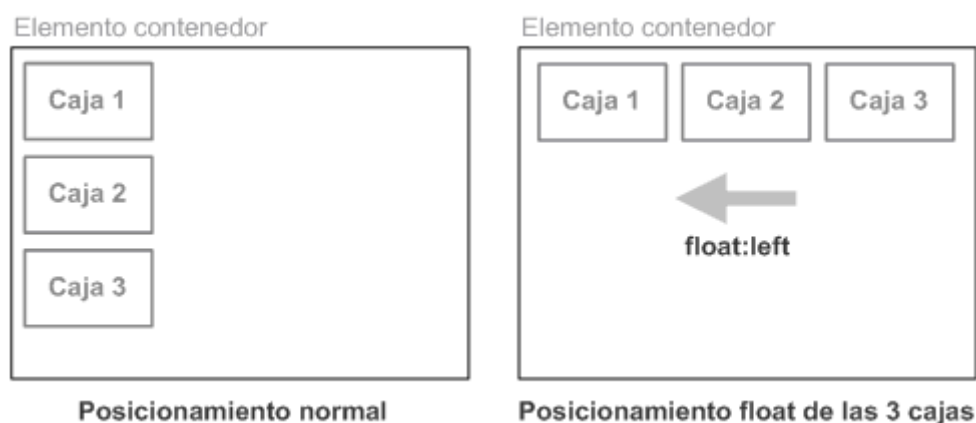


Figura 5.15 Ejemplo de posicionamiento float de varias cajas

En el ejemplo anterior, las cajas no se superponen entre sí porque las cajas flotantes tienen en cuenta las otras cajas flotantes existentes. Como la caja 1 ya estaba posicionada lo más a la izquierda posible, la caja 2 sólo puede colocarse al lado del borde derecho de la caja 1, que es el sitio más a la izquierda posible respecto de la zona en la que se encontraba.

Si no existe sitio en la línea actual, la caja flotante baja a la línea inferior hasta que encuentra el sitio necesario para mostrarse lo más a la izquierda o lo más a la derecha posible en esa nueva línea:

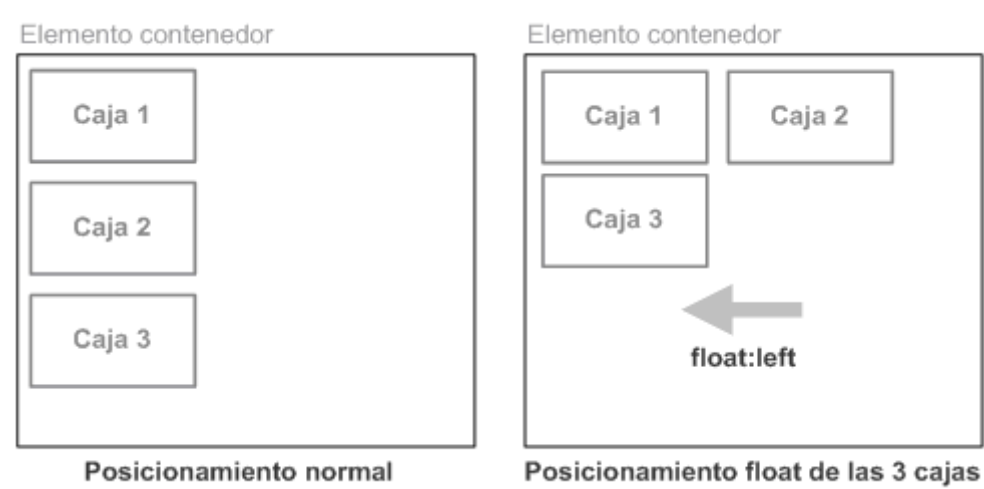


Figura 5.16 Ejemplo de posicionamiento float cuando no existe sitio suficiente

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

Propiedad	float
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.

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

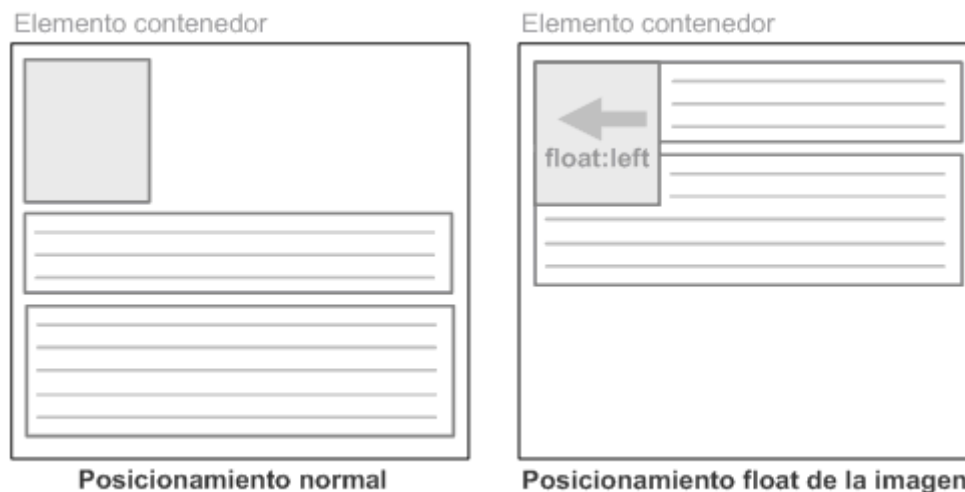


Figura 5.17 Elementos que fluyen alrededor de un elemento posicionado mediante float

La regla CSS que se aplica en la imagen del ejemplo anterior es:

```
img {  
    float: left;  
}
```

Uno de los principales motivos para la creación del posicionamiento `float` fue precisamente la posibilidad de colocar imágenes alrededor de las cuales fluye el texto.

CSS permite controlar la forma en la que los contenidos fluyen alrededor de los contenidos posicionados mediante `float`. De hecho, en muchas ocasiones es admisible que algunos contenidos fluyan alrededor de una imagen, pero el resto de contenidos deben mostrarse en su totalidad sin fluir alrededor de la imagen:

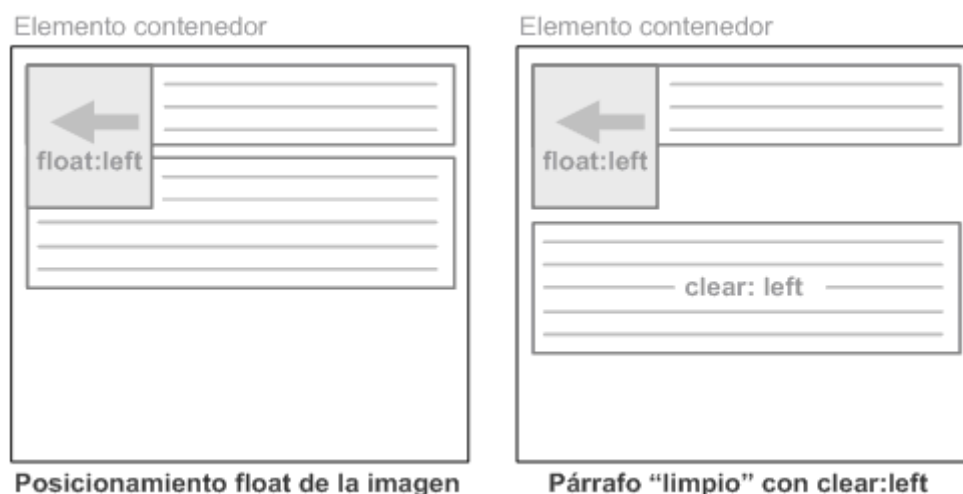


Figura 5.18 Forzando a que un elemento no fluya alrededor de otro elemento posicionado mediante float

La propiedad `clear` permite modificar el comportamiento por defecto del posicionamiento flotante para forzar a un elemento a mostrarse debajo de cualquier caja flotante. La regla CSS que se aplica al segundo párrafo del ejemplo anterior es la siguiente:

```
<p style="clear: left;">...</p>
```

La definición formal de la propiedad `clear` se muestra a continuación:

Propiedad	<code>clear</code>
Valores	<code>none</code> <code>left</code> <code>right</code> <code>both</code> <code>inherit</code>
Se aplica a	Todos los elementos de bloque
Valor inicial	<code>none</code>
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">&laquo; Anterior</span>  
    <span class="derecha">Siguiendo &raquo;</span>  
</div>
```

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

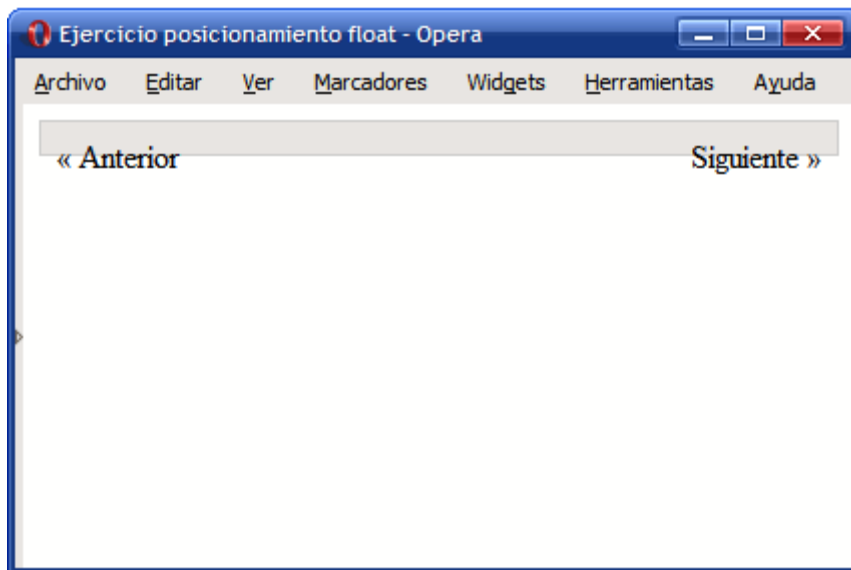


Figura 5.19 Visualización incorrecta de dos elementos posicionados mediante float

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:

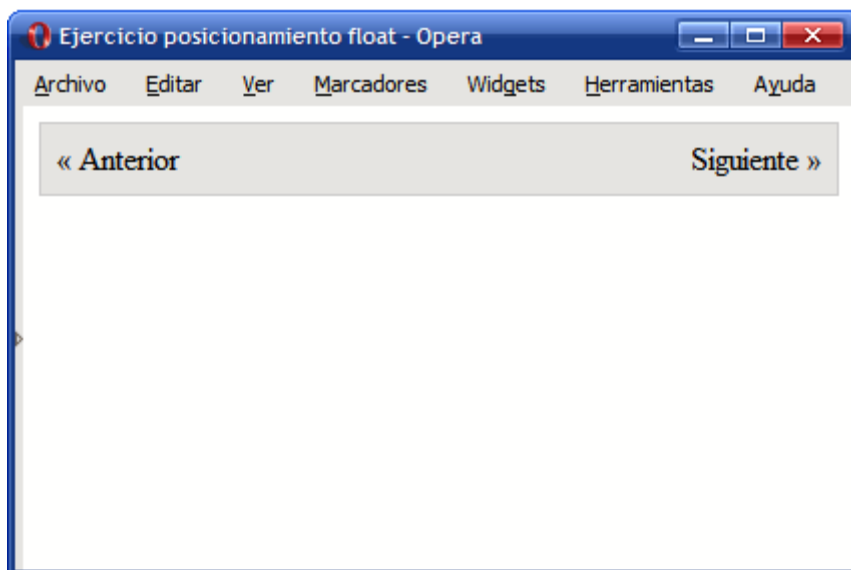


Figura 5.20 Visualización correcta de dos elementos posicionados mediante float

5.8. Visualización

Además de las propiedades que controlan el posicionamiento de los elementos, CSS define otras cuatro propiedades para controlar su visualización: `display`, `visibility`, `overflow` y `z-index`.

Utilizando algunas de estas propiedades es posible ocultar y/o hacer invisibles las cajas de los elementos, por lo que son imprescindibles para realizar efectos avanzados y animaciones.

5.8.1. Propiedades `display` y `visibility`

Las propiedades `display` y `visibility` controlan la visualización de los elementos. Las dos propiedades permiten ocultar 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`:

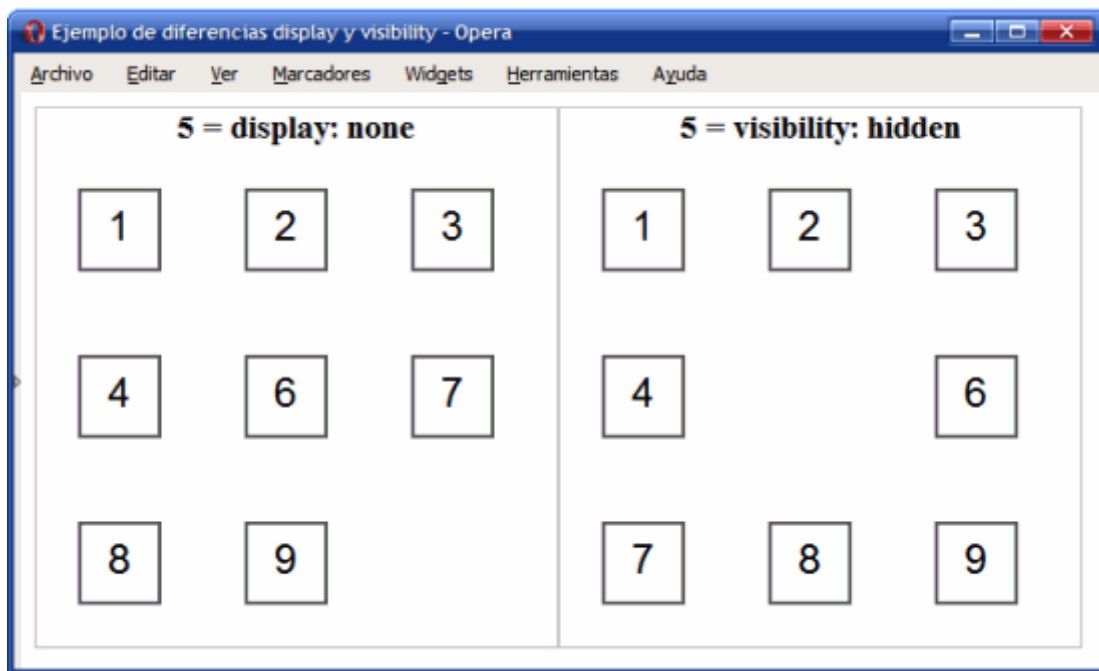


Figura 5.21 Diferencias visuales entre las propiedades display y visibility

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`:

Propiedad	<code>display</code>
Valores	<code>inline</code> <code>block</code> <code>none</code> <code>list-item</code> <code>run-in</code> <code>inline-block</code> <code>table</code> <code>inline-table</code> <code>table-row-group</code> <code>table-header-group</code> <code>table-footer-group</code> <code>table-row</code> <code>table-column-group</code> <code>table-column</code> <code>table-cell</code> <code>table-caption</code> <code>inherit</code>
Se aplica a	Todos los elementos
Valor inicial	<code>inline</code>
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.

El siguiente ejemplo muestra el uso de la propiedad `display` para mostrar un elemento de bloque como si fuera un elemento en línea y para mostrar un elemento en línea como si fuera un elemento de bloque:

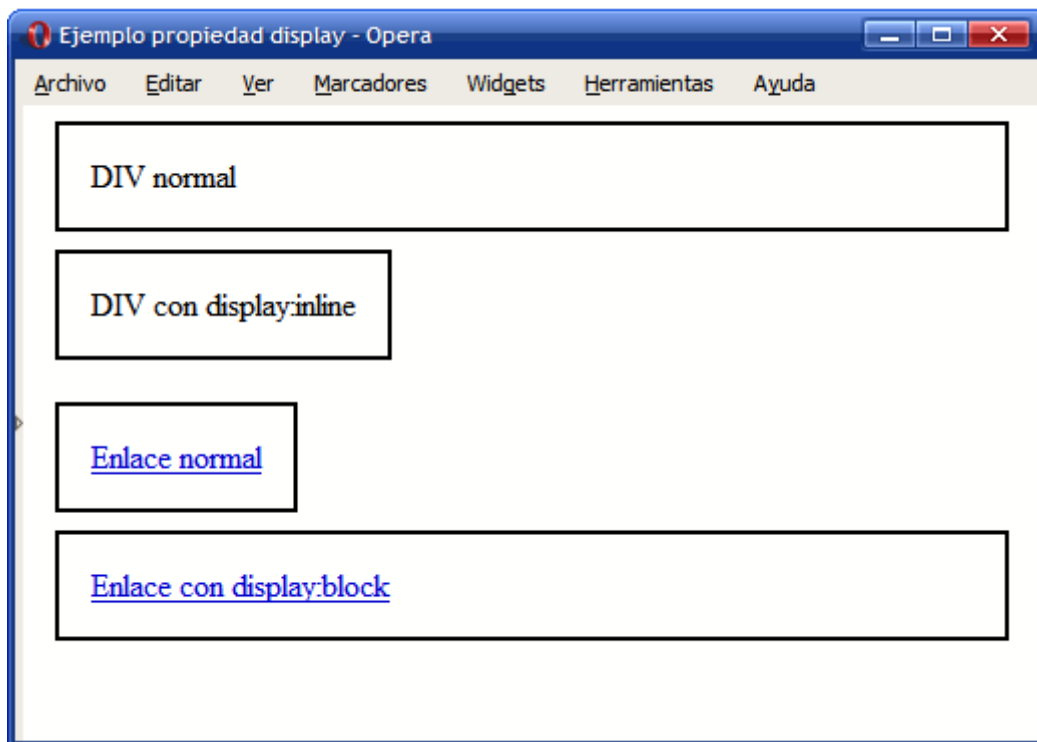


Figura 5.22 Ejemplo de propiedad display

Las reglas CSS del ejemplo anterior son las siguientes:

```
<div>DIV normal</div>
```

```
<div style="display:inline">DIV con display:inline</div>
```

```
<a href="#">Enlace normal</a>
```

```
<a href="#" style="display:block">Enlace con display:block</a>
```

Como se verá más adelante, la propiedad `display: inline` se 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:

Propiedad	<code>visibility</code>
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`.

5.8.2. Relación entre `display`, `float` y `position`

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

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

2. Si `position` vale `absolute` o `fixed`, la caja se posiciona de forma absoluta, se considera que `float` vale `none` y la propiedad `display` vale `block` tanto para los elementos en línea como para los elementos de bloque. La posición de la caja se determina mediante el valor de las propiedades `top`, `right`, `bottom` y `left`.
3. En cualquier otro caso, si `float` tiene un valor distinto de `none`, la caja se posiciona de forma flotante y la propiedad `display` vale `block` tanto para los elementos en línea como para los elementos de bloque.

5.8.3. Propiedad overflow

Normalmente, los contenidos de un elemento se pueden mostrar en el espacio reservado para ese elemento. Sin embargo, en algunas ocasiones el contenido de un elemento no cabe en el espacio reservado para ese elemento y se desborda.

La situación más habitual en la que el contenido sobresale de su espacio reservado es cuando se establece la anchura y/o altura de un elemento mediante la propiedad `width` y/o `height`. Otra situación habitual es la de las líneas muy largas contenidas dentro de un elemento `<pre>`, que hacen que la página entera sea demasiado ancha.

CSS define la propiedad `overflow` para controlar la forma en la que se visualizan los contenidos que sobresalen de sus elementos.

Propiedad	<code>overflow</code>
Valores	<code>visible</code> <code>hidden</code> <code>scroll</code> <code>auto</code> <code>inherit</code>
Se aplica a	Elementos de bloque y celdas de tablas
Valor inicial	<code>visible</code>
Descripción	Permite controlar los contenidos sobrantes de un elemento

Los valores de la propiedad `overflow` tienen el siguiente significado:

- `visible`: el contenido no se corta y se muestra sobresaliendo la zona reservada para visualizar el elemento. Este es el comportamiento por defecto.
- `hidden`: el contenido sobrante se oculta y sólo se visualiza la parte del contenido que cabe dentro de la zona reservada para el elemento.

- **scroll**: solamente se visualiza el contenido que cabe dentro de la zona reservada para el elemento, pero también se muestran barras de *scroll* que permiten visualizar el resto del contenido.
- **auto**: el comportamiento depende del navegador, aunque normalmente es el mismo que la propiedad **scroll**.

La siguiente imagen muestra un ejemplo de los tres valores típicos de la propiedad **overflow**:

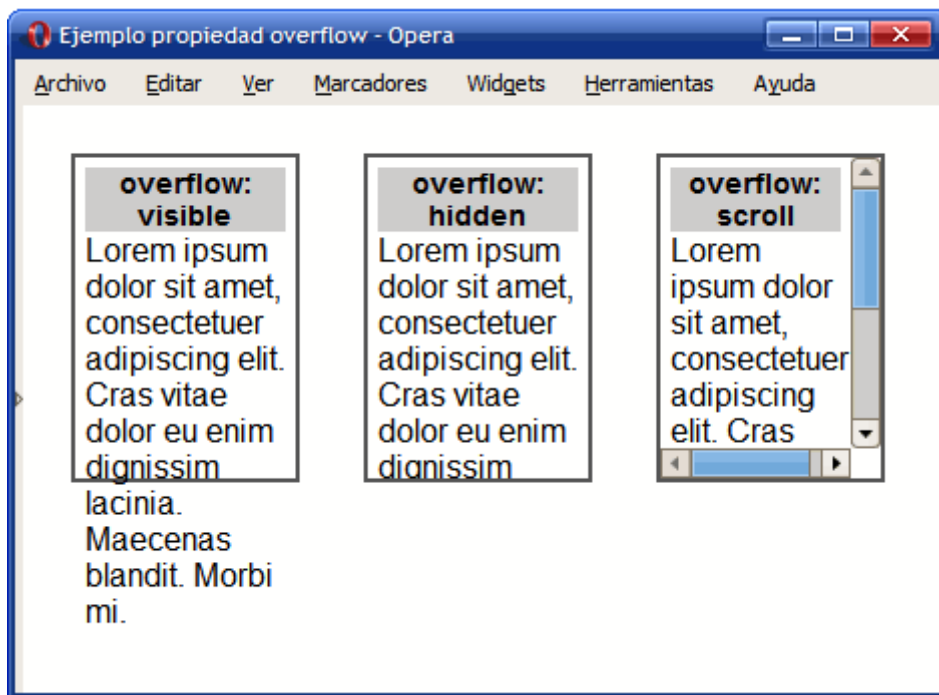


Figura 5.23 Ejemplo de propiedad overflow

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

```
div {  
  
    display: inline;  
  
    float: left;  
  
    margin: 1em;  
  
    padding: .3em;  
  
    border: 2px solid #555;  
  
    width: 100px;  
  
    height: 150px;  
  
    font: 1em Arial, Helvetica, sans-serif;  
  
}
```

```
<div><h1>overflow: visible</h1> Lorem ipsum dolor sit amet, consectetur  
adipiscing elit. Cras vitae dolor eu enim dignissim lacinia. Maecenas  
blandit. Morbi mi.</div>
```

```
<div style="overflow:hidden"><h1>overflow: hidden</h1> Lorem ipsum dolor  
sit amet, consectetur adipiscing elit. Cras vitae dolor eu enim dignissi  
m  
lacinia. Maecenas blandit. Morbi mi.</div>
```

```
<div style="overflow:scroll"><h1>overflow: scroll</h1> Lorem ipsum dolor  
sit  
amet, consectetur adipiscing elit. Cras vitae dolor eu enim dignissim la  
cinia.  
Maecenas blandit. Morbi mi.</div>
```

5.8.4. Propiedad z-index

Además de posicionar una caja de forma horizontal y vertical, CSS permite controlar la posición tridimensional de las cajas posicionadas. De esta forma, es posible 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`:

Propiedad	<code>z-index</code>
Valores	auto numero <code>inherit</code>
Se aplica a	Elementos que han sido posicionados explícitamente
Valor inicial	auto

Propiedad	z-index
Descripción	Establece el nivel tridimensional en el que se muestra el elemento

El valor más común de la propiedad **z-index** es un número entero. Aunque la especificación oficial permite los números negativos, en general se considera el número **0** como el nivel más bajo.

Cuanto más alto sea el valor numérico, más cerca del usuario se muestra la caja. Un elemento con **z-index: 10** se muestra por encima de los elementos con **z-index: 8** o **z-index: 9**, pero por debajo de elementos con **z-index: 20** o **z-index: 50**.

La siguiente imagen muestra un ejemplo de uso de la propiedad **z-index**:

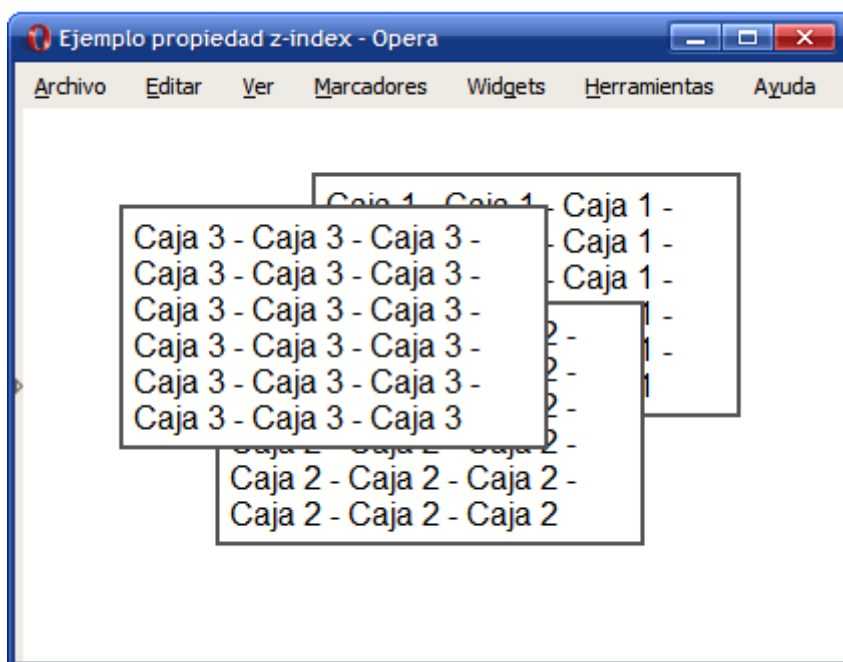


Figura 5.24 Ejemplo de propiedad z-index

El código HTML y CSS del ejemplo anterior es el siguiente:

```
div { position: absolute; }

#caja1 { z-index: 5; top: 1em; left: 8em;}

#caja2 { z-index: 15; top: 5em; left: 5em;}

#caja3 { z-index: 25; top: 2em; left: 2em;}
```

```
<div id="caja1">Caja 1 - Caja 1 - Caja 1 - Caja 1 - Caja 1 - Caja 1 -  
Caja 1 - Caja 1 - Caja 1 - Caja 1 - Caja 1 - Caja 1 - Caja 1 - Caja 1 -  
Caja 1 - Caja 1 - Caja 1 - Caja 1</div>
```

```
<div id="caja2">Caja 2 - Caja 2 - Caja 2 - Caja 2 - Caja 2 - Caja 2 -  
Caja 2 - Caja 2 - Caja 2 - Caja 2 - Caja 2 - Caja 2 - Caja 2 - Caja 2 -  
Caja 2 - Caja 2 - Caja 2 - Caja 2</div>
```

```
<div id="caja3">Caja 3 - Caja 3 - Caja 3 - Caja 3 - Caja 3 - Caja 3 -  
Caja 3 - Caja 3 - Caja 3 - Caja 3 - Caja 3 - Caja 3 - Caja 3 - Caja 3 -  
Caja 3 - Caja 3 - Caja 3 - Caja 3</div>
```

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. Texto

6.1. Tipografía

CSS define numerosas propiedades para modificar la apariencia del texto. A pesar de que no dispone de tantas posibilidades como los lenguajes y programas específicos para crear documentos impresos, CSS permite aplicar estilos complejos y muy variados al texto de las páginas web.

La propiedad básica que define CSS relacionada con la tipografía se denomina **color** y se utiliza para establecer el color de la letra.

Propiedad	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:

```
body { color: #777; }
```

En el ejemplo anterior, todos los elementos de la página muestran el mismo color de letra salvo que establezcan de forma explícita otro color. La única excepción de este comportamiento son los enlaces que se crean con la etiqueta `<a>`. Aunque el color de la letra se hereda de los elementos padre a los elementos hijo, con los enlaces no sucede lo mismo, por lo que es necesario indicar su color de forma explícita:

```
/* Establece el mismo color a todos los elementos de  
   la página salvo los enlaces */  
  
body { color: #777; }
```

```
/* Establece el mismo color a todos los elementos de  
   la página, incluyendo los enlaces */  
  
body, a { color: #777; }
```

La otra propiedad básica que define CSS relacionada con la tipografía se denomina `font-family` y se utiliza para indicar el tipo de letra con el que se muestra el texto.

Propiedad	font-family
Valores	((nombre_familia familia_generica) (,nombre_familia familia_generica)*) <code>inherit</code>
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`.

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

Además de todas las unidades de medida relativas y absolutas y el uso de porcentajes, CSS permite utilizar una serie de palabras clave para indicar el tamaño de letra del texto:

- `tamaño_absoluto`: indica el tamaño de letra de forma absoluta mediante alguna de las siguientes palabras clave: `xx-small`, `x-small`, `small`, `medium`, `large`, `x-large`, `xx-large`.
- `tamaño_relativo`: indica de forma relativa el tamaño de letra del texto mediante dos palabras clave (`larger`, `smaller`) que toman como referencia el tamaño de letra del elemento padre.

La siguiente imagen muestra una comparación entre los tamaños típicos del texto y las unidades que más se utilizan:

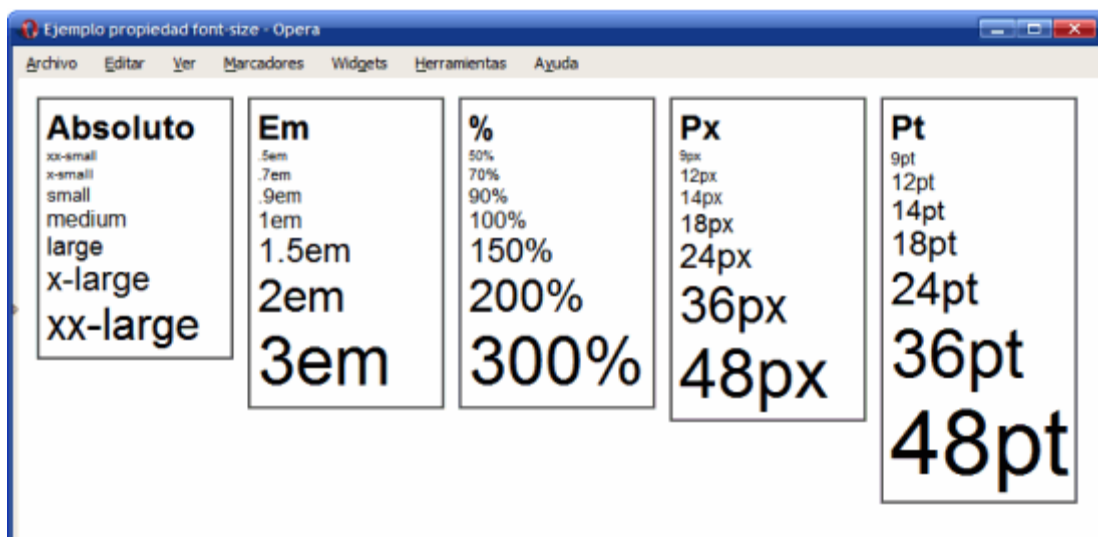


Figura 6.1 Comparación visual de las distintas unidades para indicar el tamaño del texto

CSS recomienda indicar el tamaño del texto en la unidad **em** o en porcentaje (%). Además, es habitual indicar el tamaño del texto en puntos (**pt**) cuando el documento está específicamente diseñado para 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**.

Propiedad	font-weight
Valores	normal bold bolder lighter 100 200 300 400 500 600 700 800 900 inherit
Se aplica a	Todos los elementos
Valor inicial	normal
Descripción	Establece la anchura de la letra utilizada para el texto

Los valores que normalmente se utilizan son `normal` (el valor por defecto) y `bold` para los textos en negrita. El valor `normal` equivale al valor numérico 400 y el valor `bold` al valor numérico 700.

El siguiente ejemplo muestra una aplicación práctica de la propiedad `font-weight`:

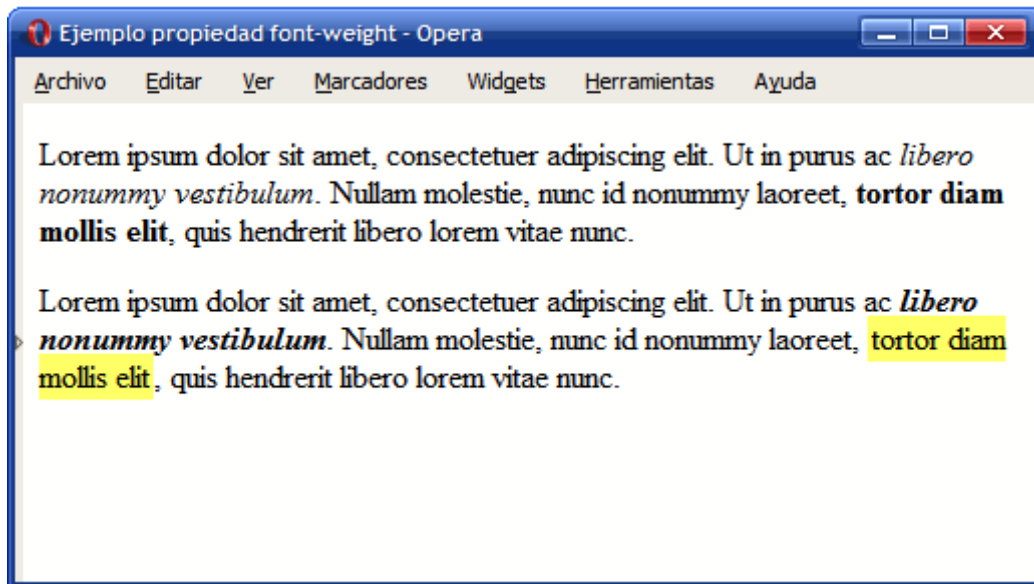


Figura 6.2 Ejemplo de propiedad font-weight

Por defecto, los navegadores muestran el texto de los elementos `` en cursiva y el texto de los elementos `` en negrita. La propiedad `font-weight` permite alterar ese aspecto por defecto y mostrar por ejemplo los elementos `` como cursiva y negrita y los elementos `` destacados mediante un color de fondo y sin negrita.

Las reglas CSS del ejemplo anterior se muestran a continuación:

```
#especial em {  
    font-weight: bold;  
}  
  
#especial strong {  
    font-weight: normal;  
    background-color: #FFFF66;  
    padding: 2px;  
}
```

`<p>` Lorem ipsum dolor sit amet, consectetur adipiscing elit. Ut in

purus ac ``libero nonummy vestibulum``. Nullam molestie, nunc id nonummy laoreet, ``tortor diam mollis elit``, quis hendrerit libero lorem vitae nunc.`</p>`

`<p id="especial">`Lorem ipsum dolor sit amet, consectetur 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.`</p>`

Además de la anchura de la letra, CSS permite variar su estilo mediante la propiedad `font-style`.

Propiedad	font-style
Valores	normal italic oblique <code>inherit</code>
Se aplica a	Todos los elementos
Valor inicial	normal
Descripción	Establece el estilo de la letra utilizada para el texto

Normalmente la propiedad `font-style` se emplea para mostrar un texto en cursiva mediante el valor `italic`.

El ejemplo anterior se puede modificar para personalizar aun más el aspecto por defecto de los elementos `` y ``:

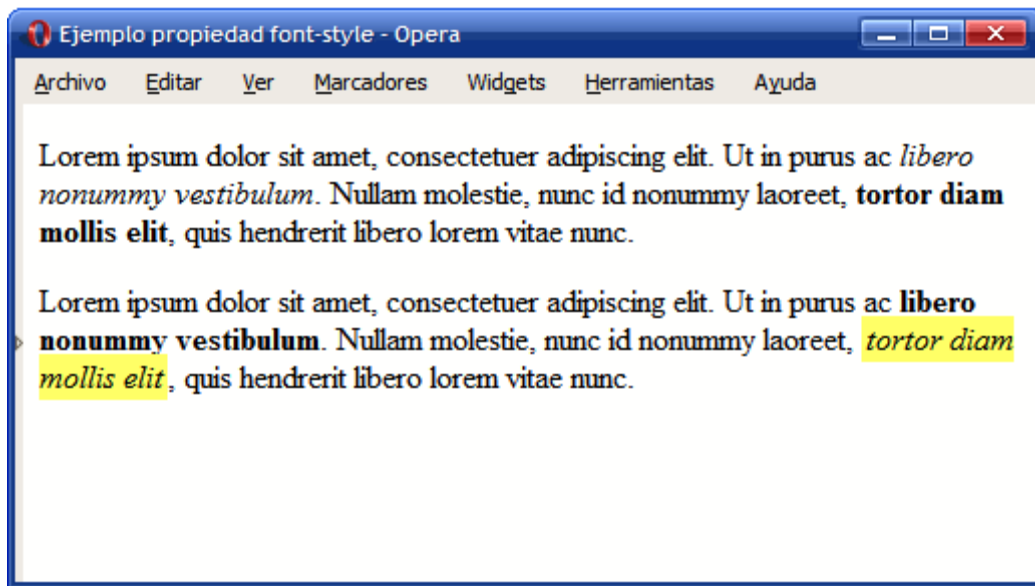


Figura 6.3 Ejemplo de propiedad font-style

Ahora, el texto del elemento `` se muestra como un texto en cursiva y el texto del elemento `` se muestra como un texto en negrita 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`.

Propiedad	font-variant
Valores	normal small-caps inherit
Se aplica a	Todos los elementos
Valor inicial	normal
Descripción	Establece el estilo alternativo de la letra utilizada para el texto

La propiedad `font-variant` no se suele emplear habitualmente, ya que sólo permite mostrar el texto con *letra versal* (mayúsculas pequeñas).

Siguiendo con el ejemplo anterior, se ha aplicado la propiedad `font-variant: small-caps` al segundo párrafo de texto:

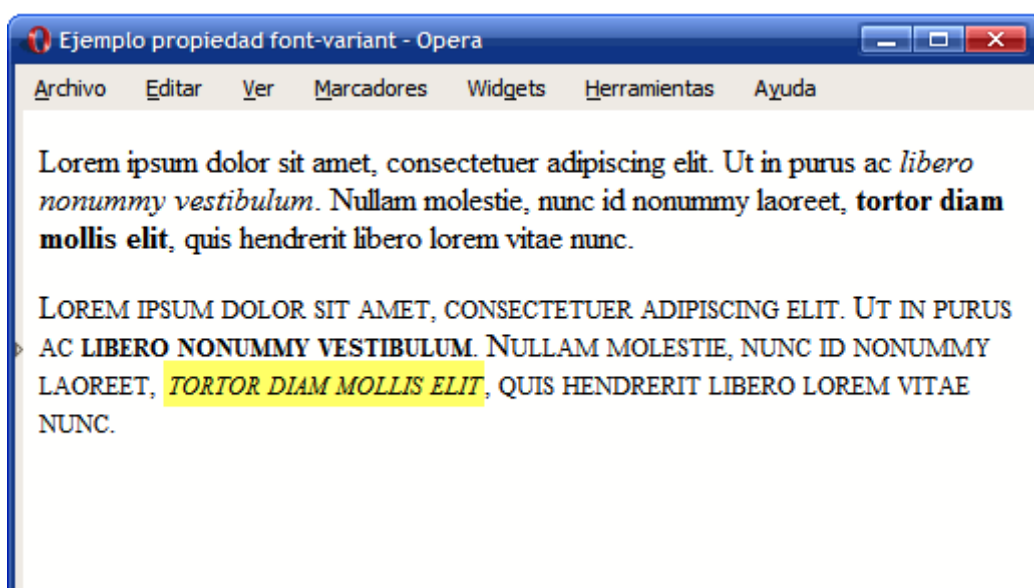


Figura 6.4 Ejemplo de propiedad font-variant

Para este último ejemplo, solamente es necesario añadir una regla a los estilos CSS:

```
#especial {
    font-variant: small-caps;
}
```

Por otra parte, CSS proporciona una propiedad tipo *"shorthand"* denominada **font** y que permite indicar de forma directa algunas o todas las propiedades de la tipografía de un texto.

Propiedad	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.

6.2. Texto

Además de las propiedades relativas a la tipografía del texto, CSS define numerosas propiedades que determinan la apariencia del texto en su conjunto. Estas propiedades adicionales permiten controlar al alineación del texto, el interlineado, la separación entre palabras, etc.

La propiedad que define la alineación del texto se denomina `text-align`.

Propiedad	<code>text-align</code>
Valores	<code>left</code> <code>right</code> <code>center</code> <code>justify</code> <code>inherit</code>
Se aplica a	Elementos de bloque y celdas de tabla
Valor inicial	<code>left</code>
Descripción	Establece la alineación del contenido del elemento

Los valores definidos por CSS permiten alinear el texto según los valores tradicionales: a la izquierda (`left`), a la derecha (`right`), centrado (`center`) y justificado (`justify`).

La siguiente imagen muestra el efecto de establecer el valor `left`, `right`, `center` y `justify` respectivamente a cada uno de los párrafos de la página.

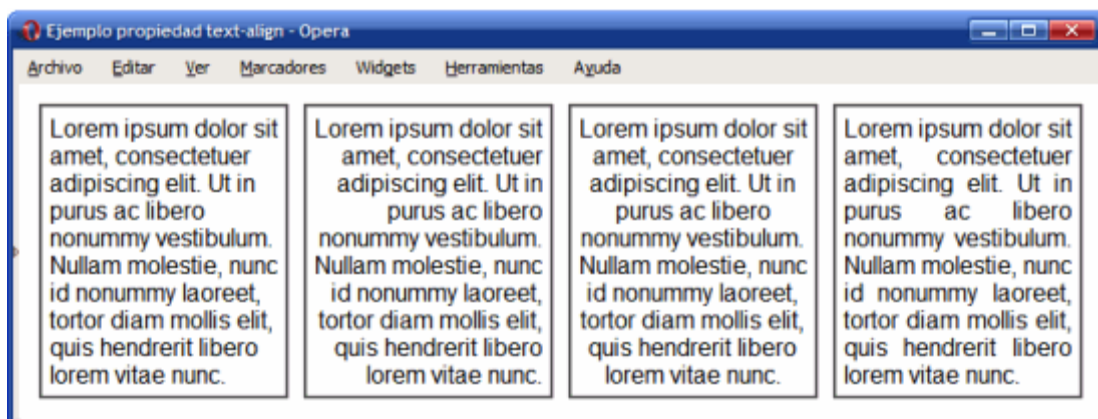


Figura 6.5 Ejemplo de propiedad text-align

La propiedad `text-align` no sólo alinea el texto que contiene un elemento, sino que también alinea todos sus contenidos, como por ejemplo las imágenes.

El interlineado de un texto se controla mediante la propiedad `line-height`, que permite controlar la altura ocupada por cada línea de texto:

Propiedad	<code>line-height</code>
Valores	normal numero unidad de medida porcentaje inherit
Se aplica a	Todos los elementos
Valor inicial	normal
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**.

Propiedad	text-decoration
Valores	none (underline overline line-through blink) inherit
Se aplica a	Todos los elementos
Valor inicial	none

Propiedad	<code>text-decoration</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.

Propiedad	<code>text-transform</code>
Valores	<code>capitalize</code> <code>uppercase</code> <code>lowercase</code> <code>none</code> <code>inherit</code>
Se aplica a	Todos los elementos
Valor inicial	<code>none</code>
Descripción	Transforma el texto original (lo transforma a mayúsculas, a minúsculas, etc.)

La propiedad `text-transform` permite mostrar el texto original transformado en un texto completamente en mayúsculas (`uppercase`), en minúsculas (`lowercase`) o con la primera letra de cada palabra en mayúscula (`capitalize`).

La siguiente imagen muestra cada uno de los posibles valores:

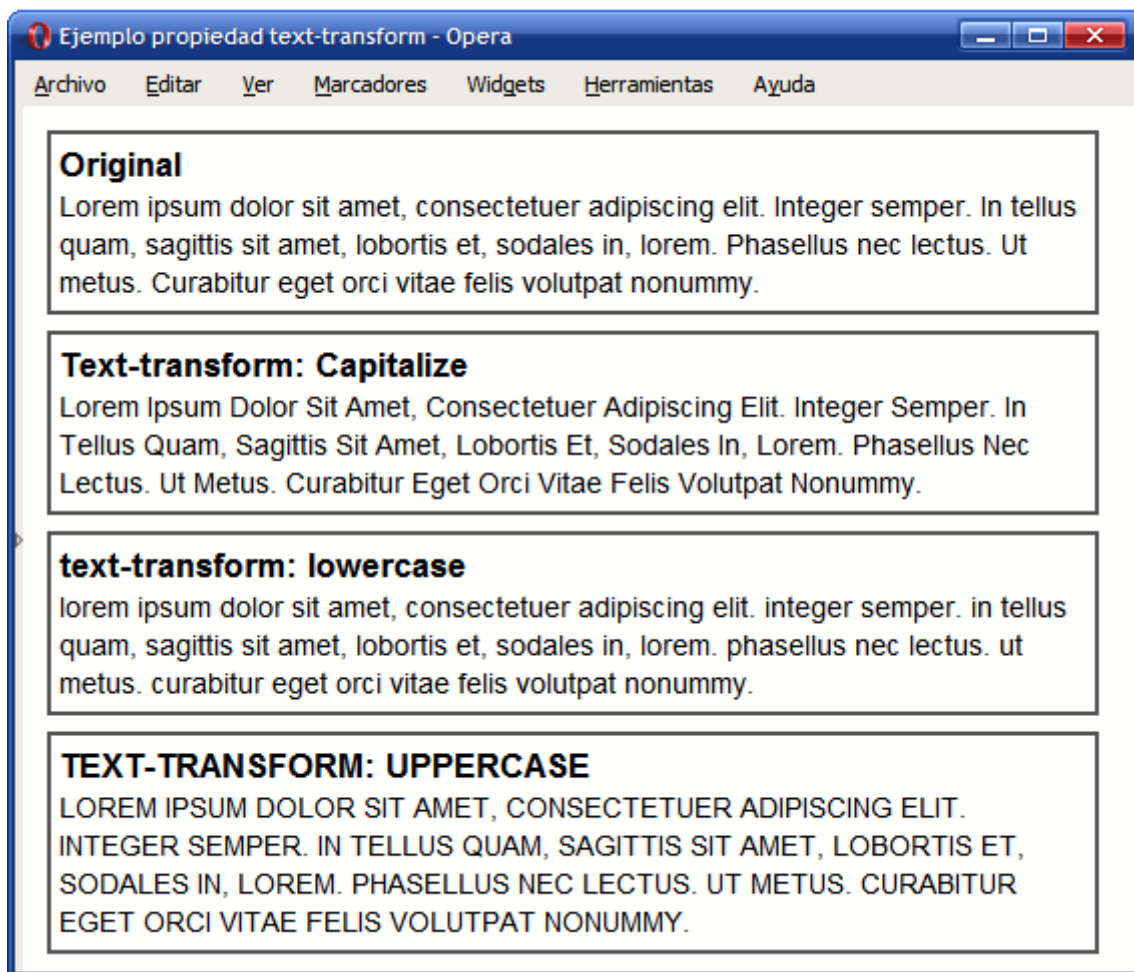


Figura 6.7 Ejemplo de propiedad text-transform

Las reglas CSS del ejemplo anterior se muestran a continuación:

```
<div style="text-transform: none"><h1>Original</h1>Lorem ipsum dolor  
sit amet...</div>
```

```
<div style="text-transform: capitalize"><h1>text-transform: capitalize</h1>  
1>  
Lorem ipsum dolor sit amet...</div>
```

```
<div style="text-transform: lowercase"><h1>text-transform: lowercase</h1>  
1>  
Lorem ipsum dolor sit amet...</div>
```

```
<div style="text-transform: uppercase"><h1>text-transform: uppercase</h1>  
1>  
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`.

Propiedad	<code>vertical-align</code>
Valores	baseline sub super top text-top middle bottom text-bottom porcentaje unidad de medida <code>inherit</code>
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`.

Propiedad	<code>text-indent</code>
Valores	<code>unidad de medida</code> <code>porcentaje</code> <code>inherit</code>
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:

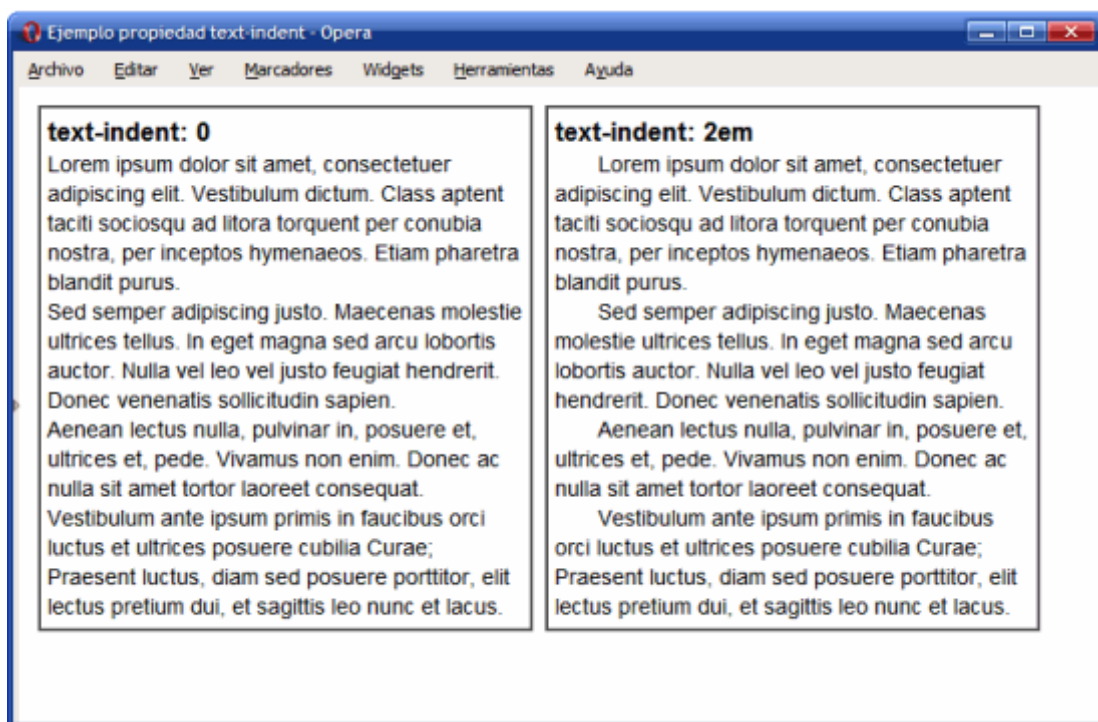


Figura 6.9 Ejemplo de propiedad `text-indent`

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`.

Propiedad	letter-spacing
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
Propiedad	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:

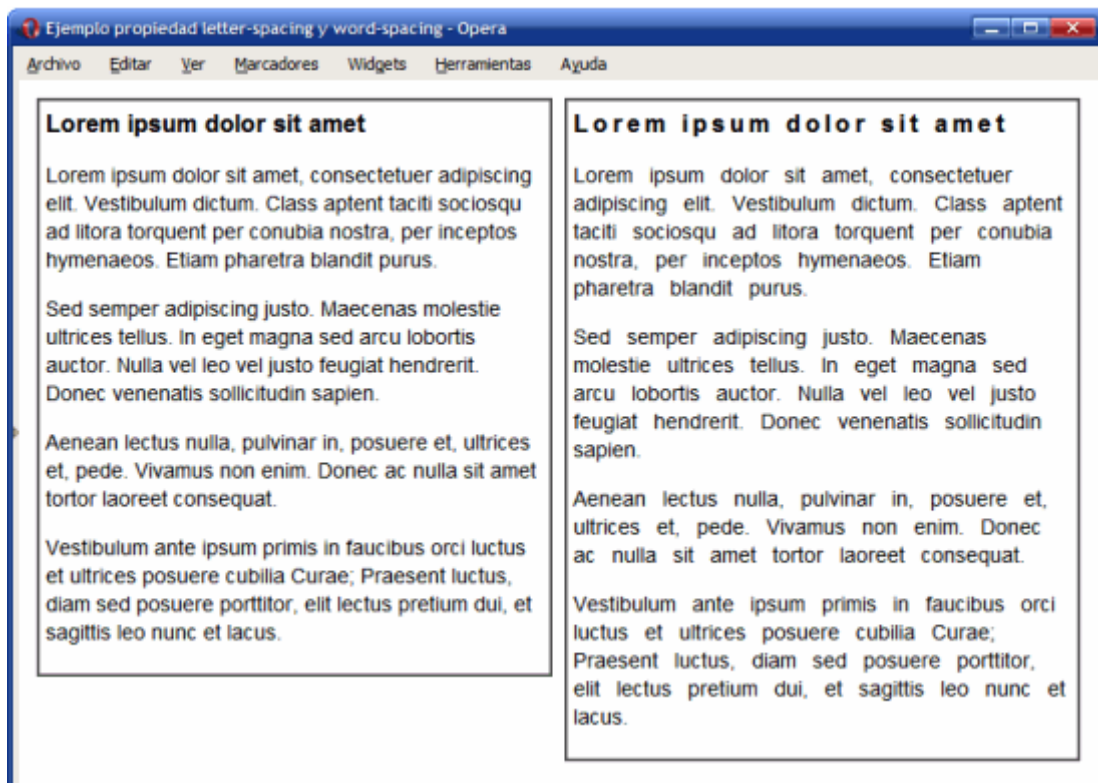


Figura 6.10 Ejemplo de propiedades letter-spacing y word-spacing

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

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

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


...

</div>

Cuando se utiliza un valor numérico en las propiedades `letter-spacing` y `word-spacing`, se interpreta como la separación adicional que se añade (si el valor es positivo) o se quita (si el valor es negativo) a la separación por defecto entre letras y palabras respectivamente.

Como ya se sabe, el tratamiento que hace HTML de los espacios en blanco es uno de los aspectos más difíciles de comprender cuando se empiezan a crear las primeras páginas web. Básicamente, HTML elimina todos los espacios en blanco sobrantes, es decir, todos salvo un espacio en blanco entre cada palabra.

Para forzar los espacios en blanco adicionales se debe utilizar la entidad HTML ` `; y para forzar nuevas líneas, se utiliza el elemento `
`. Además, HTML proporciona el elemento `<pre>` que muestra el contenido tal y como se escribe, respetando todos los espacios en blanco y todas las nuevas líneas.

CSS también permite controlar el tratamiento de los espacios en blanco de los textos mediante la propiedad `white-space`.

Propiedad	<code>white-space</code>
Valores	<code>normal</code> <code>pre</code> <code>nowrap</code> <code>pre-wrap</code> <code>pre-line</code> <code>inherit</code>
Se aplica a	Todos los elementos
Valor inicial	<code>normal</code>
Descripción	Establece el tratamiento de los espacios en blanco del texto

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

- `normal`: comportamiento por defecto de HTML.
- `pre`: se respetan los espacios en blanco y las nuevas líneas (exactamente igual que la etiqueta `<pre>`). Si la línea es muy larga, se sale del espacio asignado para ese contenido.
- `nowrap`: elimina los espacios en blanco y las nuevas líneas. Si la línea es muy larga, se sale del espacio asignado para ese contenido.

- **pre-wrap**: se respetan los espacios en blanco y las nuevas líneas, pero ajustando cada línea al espacio asignado para ese contenido.
- **pre-line**: elimina los espacios en blanco y respeta las nuevas líneas, pero ajustando cada línea al espacio asignado para ese contenido.

En la siguiente tabla se resumen las características de cada valor:

Valor	Respetar espacios en blanco	Respetar saltos de línea	Ajusta las líneas
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:

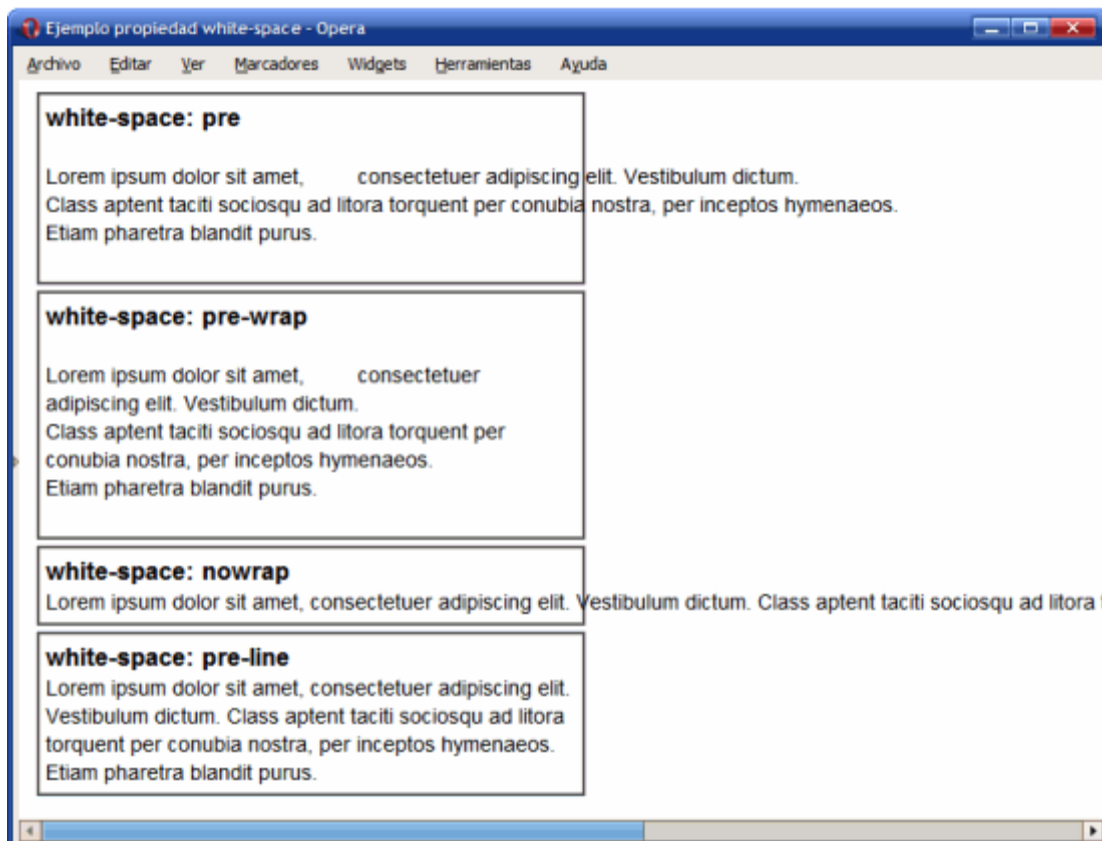


Figura 6.11 Ejemplo de propiedad white-space

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:

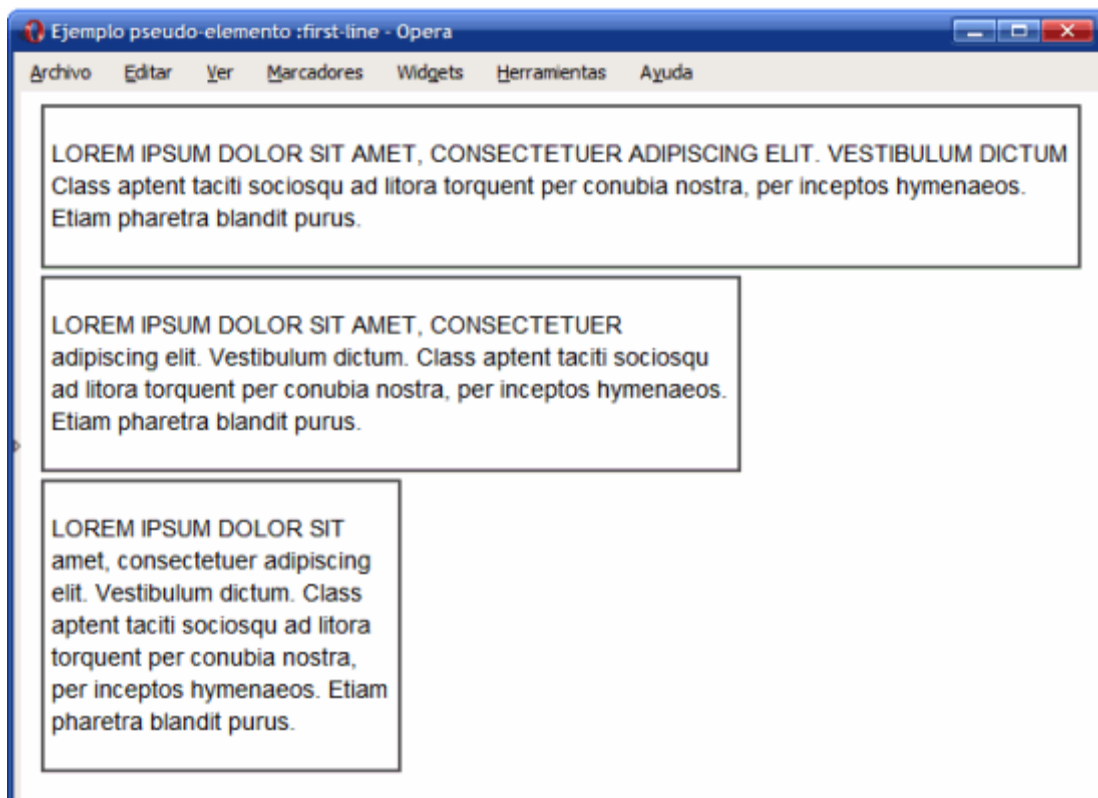


Figura 6.12 Ejemplo de pseudo-elemento first-line

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:

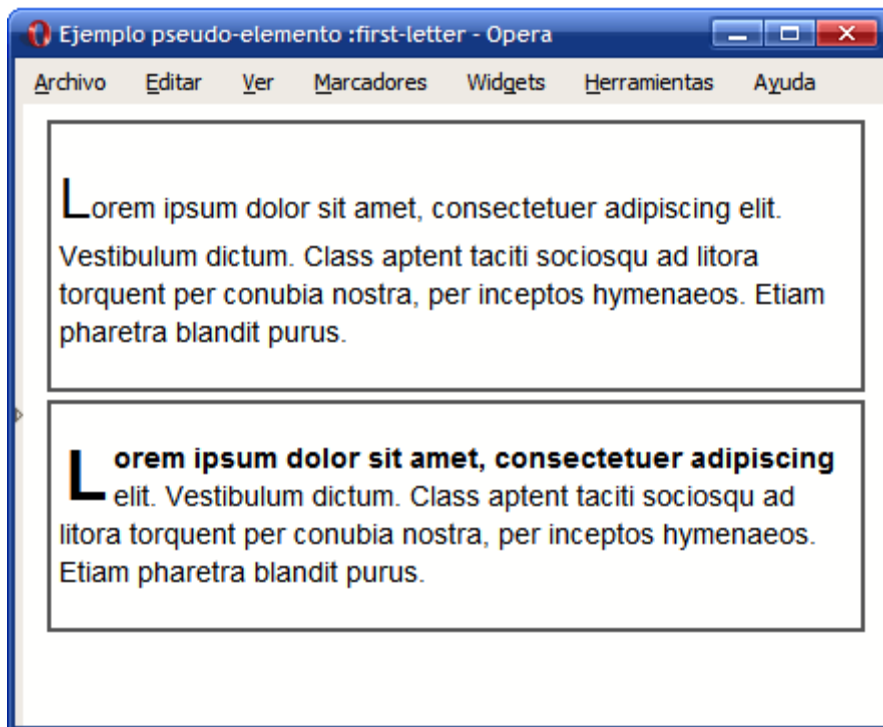


Figura 6.13 Ejemplo de pseudo-elemento first-letter

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>

Capítulo 7. Enlaces

7.1. Estilos básicos

7.1.1. Tamaño, color y decoración

Los estilos más sencillos que se pueden aplicar a los enlaces son los que modifican su tamaño de letra, su color y la decoración del texto del enlace. Utilizando las propiedades `text-decoration` y `font-weight` se pueden conseguir estilos como los que se muestran en la siguiente imagen:

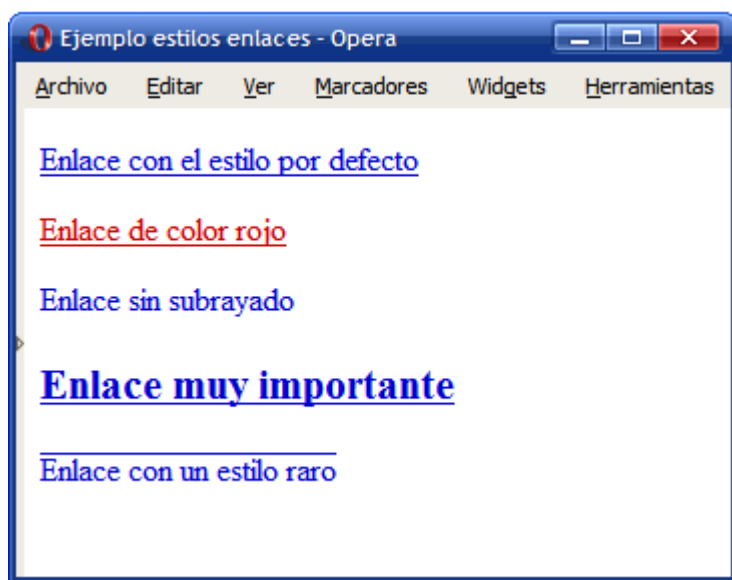


Figura 7.1 Ejemplo de enlaces con estilos diferentes

A continuación se muestran las reglas CSS del ejemplo anterior:

```
a { margin: 1em 0; display: block;}
```

```
.alternativo {color: #CC0000;}
```

```
.simple {text-decoration: none;}
```

```
.importante {font-weight: bold; font-size: 1.3em;}
```

```
.raro {text-decoration: overline;}
```

```
<a href="#">Enlace con el estilo por defecto</a>
```

```
<a class="alternativo" href="#">Enlace de color rojo</a>
```

```
<a class="simple" href="#">Enlace sin subrayado</a>
```

```
<a class="importante" href="#">Enlace muy importante</a>
```

```
<a class="raro" href="#">Enlace con un estilo raro</a>
```

7.1.2. Pseudo-clases

CSS 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 posicionado 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.primer: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 { ... }
```

7.2. Estilos avanzados

7.2.1. Decoración personalizada

La propiedad `text-decoration` permite añadir o eliminar el subrayado de los enlaces. Sin embargo, el aspecto del subrayado lo controla automáticamente el navegador, por lo que su color siempre es el mismo que el del texto del enlace y su anchura es proporcional al tamaño de letra.

No obstante, utilizando la propiedad `border-bottom` es posible añadir cualquier tipo de subrayado para los enlaces de las páginas. El siguiente ejemplo muestra algunos enlaces con el subrayado personalizado:

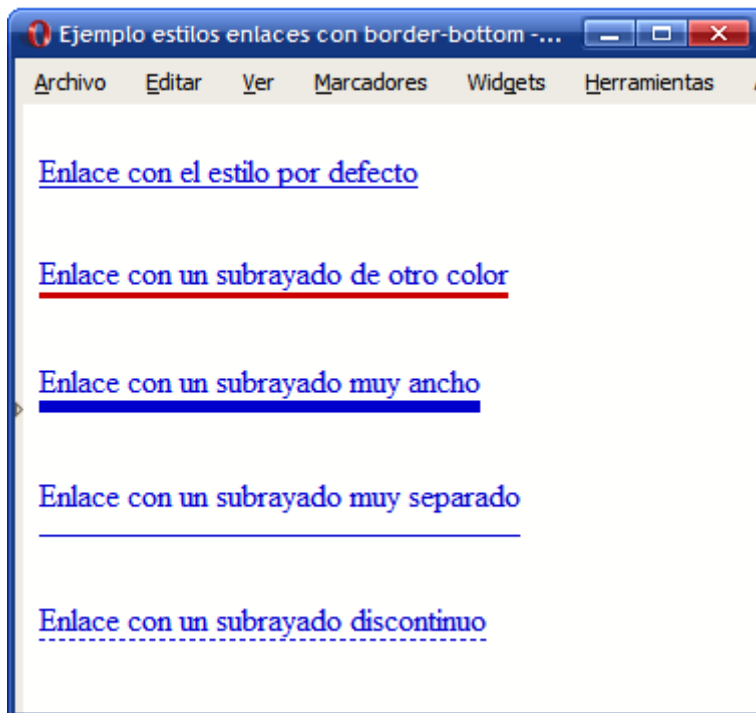


Figura 7.3 Enlaces con subrayado personalizado mediante la propiedad `border`

Las reglas CSS definidas en el ejemplo anterior se muestran a continuación:

```
a { margin: 1em 0; float: left; clear: left; text-decoration: none;}
.simple {text-decoration: underline;}
.color { border-bottom: medium solid #CC0000;}
.ancha {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="ancha" href="#">Enlace con un subrayado muy ancho</a>
```

```
<a class="separado" href="#">Enlace con un subrayado muy separado</a>
```

```
<a class="discontinuo" href="#">Enlace con un subrayado discontinuo</a>
```

7.2.2. Imágenes según el tipo de enlace

En ocasiones, puede resultar útil incluir un pequeño icono 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 icono a su lado.

La técnica consiste en mostrar una imagen de fondo sin repetición en el enlace y añadir el `padding` necesario al texto del enlace para que no se solape con la imagen de fondo.

El siguiente ejemplo personaliza el aspecto de dos enlaces añadiéndoles una imagen de fondo:

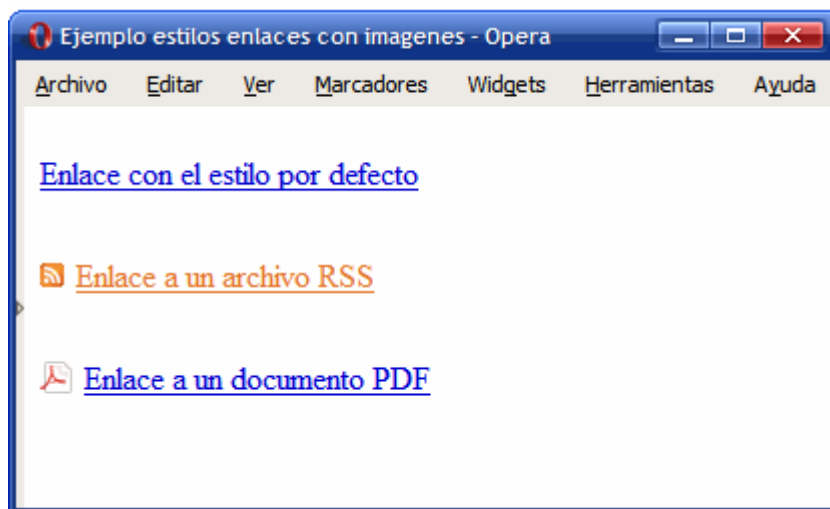


Figura 7.4 Personalizando el aspecto de los enlaces en función de su tipo

Las reglas CSS necesarias se muestran a continuación:

```
a { margin: 1em 0; float: left; clear: left; }
```

```
.rss {
```

```
    color: #E37529;
```

```
    padding: 0 0 0 18px;
```

```
    background: #FFF url(imagenes/rss.gif) no-repeat left center;
```

```
}
```

```
.pdf {  
    padding: 0 0 0 22px;  
    background: #FFF url(imagenes/pdf.png) no-repeat left center;  
}
```

```
<a href="#">Enlace con el estilo por defecto</a>
```

```
<a class="rss" href="#">Enlace a un archivo RSS</a>
```

```
<a class="pdf" href="#">Enlace a un documento PDF</a>
```

7.2.3. Mostrar los enlaces como si fueran botones

Las propiedades definidas por CSS permiten mostrar los enlaces con el aspecto de un botón, lo que puede ser útil en formularios en los que no se utilizan elementos específicos para crear botones.

El siguiente ejemplo muestra un enlace simple formateado como un botón:

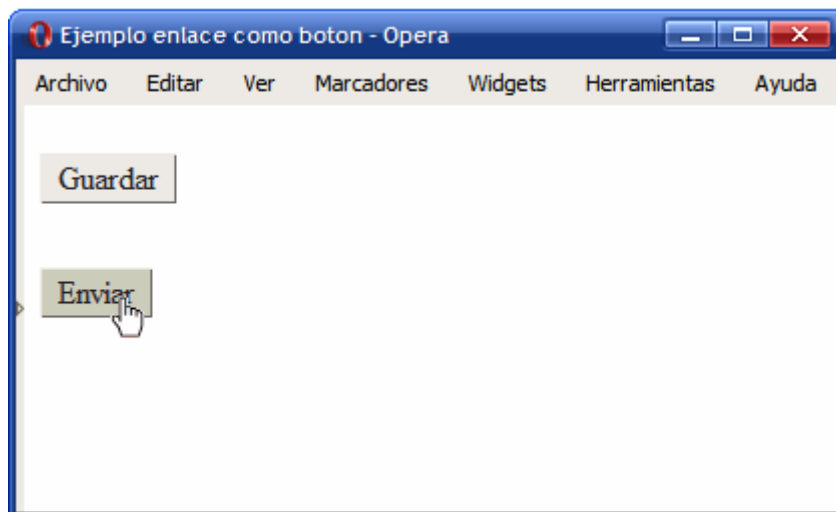


Figura 7.5 Mostrando un enlace como si fuera un botón

Las reglas CSS utilizadas en el ejemplo anterior son las siguientes:

```
a { margin: 1em 0; float: left; clear: left; }  
a.boton {  
    text-decoration: none;
```

```
background: #EEE;

color: #222;

border: 1px outset #CCC;

padding: .1em .5em;
}

a.boton:hover {

background: #CCB;

}

a.boton:active {

border: 1px inset #000;

}


<a class="boton" href="#">Guardar</a>
<a class="boton" href="#">Enviar</a>
```

Capítulo 8. Imágenes

8.1. Estilos básicos

8.1.1. Establecer la anchura y altura de las imágenes

Utilizando las propiedades `width` y `height`, es posible mostrar las imágenes con cualquier altura/anchura, independientemente de su altura/anchura real:

```
#destacada {  
  
    width: 120px;  
  
    height: 250px;  
  
}
```

```

```

No obstante, si se utilizan alturas/anchuras diferentes de las reales, el navegador deforma las imágenes y el resultado estético es muy desagradable.

Por otra parte, establecer la altura/anchura de todas las imágenes mediante CSS es una práctica poco recomendable. El motivo es que normalmente dos imágenes diferentes no comparten la misma altura/anchura. Por lo tanto, se debe crear una nueva regla CSS (y un nuevo selector) para cada una de las diferentes imágenes del sitio web.

En la práctica, esta forma de trabajo produce una sobrecarga de estilos que la hace inviable. Por este motivo, aunque es una solución que no respeta la separación completa entre contenidos (XHTML) y presentación (CSS), se recomienda establecer la altura/anchura de las imágenes mediante los atributos `width` y `height` de la etiqueta ``:

```

```

8.1.2. Eliminar el borde de las imágenes con enlaces

Cuando una imagen forma parte de un enlace, los navegadores muestran por defecto un borde azul grueso alrededor de las imágenes. Por tanto, una de las reglas más utilizadas en los archivos CSS es la que elimina los bordes de las imágenes con enlaces:

```
img {  
  
    border: none;  
  
}
```

8.2. Estilos avanzados

8.2.1. Sombra (drop shadow)

La mayoría de aplicaciones de diseño gráfico permiten añadir una sombra (llamada *drop shadow* en inglés) a las imágenes. CSS no incluye propiedades que permitan mostrar de forma sencilla sombras en los elementos.

No obstante, existen varias técnicas sencillas y otras más avanzadas que permiten crear sombras que se adapten a cualquier imagen o elemento. A continuación se muestra una técnica sencilla para añadir sombra a las imágenes.

El estilo final de las sombras creadas con esta técnica se muestra a continuación:

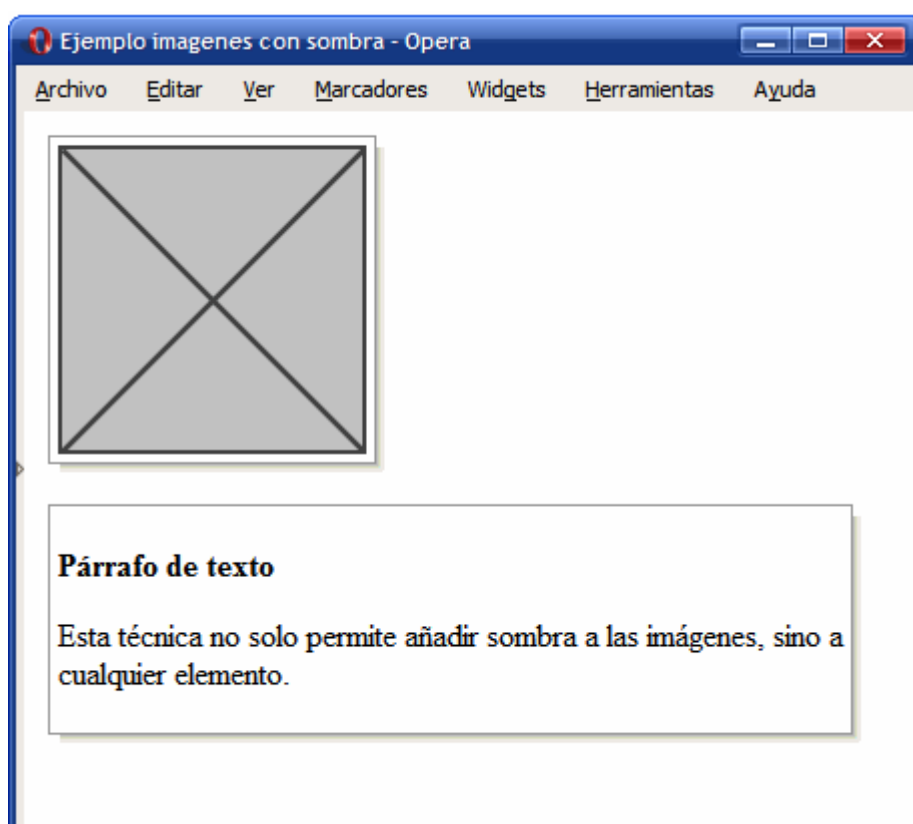


Figura 8.1 Sombra aplicada a las imágenes y otros elementos mediante CSS

La técnica mostrada se presentó por primera vez en la página web <http://wubbleyew.com/tests/dropshadows.htm> y consiste en la utilización de un par de elementos `<div>` alrededor del elemento que va a mostrar una sombra. La sombra se consigue mediante una imagen muy grande que contiene una sombra orientada hacia el lado derecho e inferior.

La ventaja de este método es que es sencillo y solamente requiere añadir un par de `<div>` por cada elemento. Además, como la sombra es una imagen muy grande, el efecto funciona con elementos de cualquier tamaño.

El mayor inconveniente de este método es que la sombra siempre se muestra en la misma dirección (derecha inferior) y que en Internet Explorer 6 y

versiones anteriores sólo muestran la sombra correctamente cuando el color de fondo de la página es blanco (ya que Internet Explorer 6 y versiones anteriores no soportan imágenes en formato PNG con transparencias).

El código HTML y CSS del ejemplo anterior es bastante sencillo:

```
.dropshadow {  
    float:left;  
    clear:left;  
    background: url(imagenes/shadowAlpha.png) no-repeat bottom right !important;  
    background: url(imagenes/shadow.gif) no-repeat bottom right;  
    margin: 10px 0 10px 10px !important;  
    margin: 10px 0 10px 5px;  
    padding: 0px;  
}
```

```
.innerbox {  
    position:relative;  
    bottom:6px;  
    right: 6px;  
    border: 1px solid #999999;  
    padding:4px;  
    margin: 0px 0px 0px 0px;  
}
```

```
<div class="dropshadow">  
  <div class="innerbox">  
      
  </div>  
</div>  
  
<div class="dropshadow">
```

```
<div class="innerbox">  
  <h4>Párrafo de texto</h4>  
  <p>Esta técnica no sólo permite añadir sombra a las imágenes, sino a cualquier elemento.</p>  
</div>  
</div>
```

Capítulo 9. Listas

9.1. Estilos básicos

9.1.1. Viñetas personalizadas

Por defecto, los navegadores muestran los elementos de las listas no ordenadas con una viñeta formada por un pequeño círculo de color negro. Los elementos de las listas ordenadas se muestran por defecto con la numeración decimal utilizada en la mayoría de países.

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

Propiedad	<code>list-style-type</code>
Valores	<code>disc</code> <code>circle</code> <code>square</code> <code>decimal</code> <code>decimal-leading-zero</code> <code>lower-roman</code> <code>upper-roman</code> <code>lower-greek</code> <code>lower-latin</code> <code>upper-latin</code> <code>armenian</code> <code>georgian</code> <code>lower-alpha</code> <code>upper-alpha</code> <code>none</code> <code>inherit</code>
Se aplica a	Elementos de una lista
Valor inicial	<code>disc</code>
Descripción	Permite establecer el tipo de viñeta mostrada para una lista

En primer lugar, el valor `none` permite mostrar una lista en la que sus elementos no contienen viñetas, números o letras. Se trata de un valor muy utilizado, ya que es imprescindible para los menús de navegación creados con listas, como se verá más adelante.

El resto de valores de la propiedad `list-style-type` se dividen en tres tipos: gráficos, numéricos y alfabéticos.

- Los valores gráficos son `disc`, `circle` y `square` y muestran como viñeta un círculo relleno, un círculo vacío y un cuadrado relleno respectivamente.
- Los valores numéricos están formados por `decimal`, `decimal-leading-zero`, `lower-roman`, `upper-roman`, `armenian` y `georgian`.

- Por último, los valores alfanuméricos se controlan mediante `lower-latin`, `lower-alpha`, `upper-latin`, `upper-alpha` y `lower-greek`.

La siguiente imagen muestra algunos de los valores definidos por la propiedad `list-style-type`:

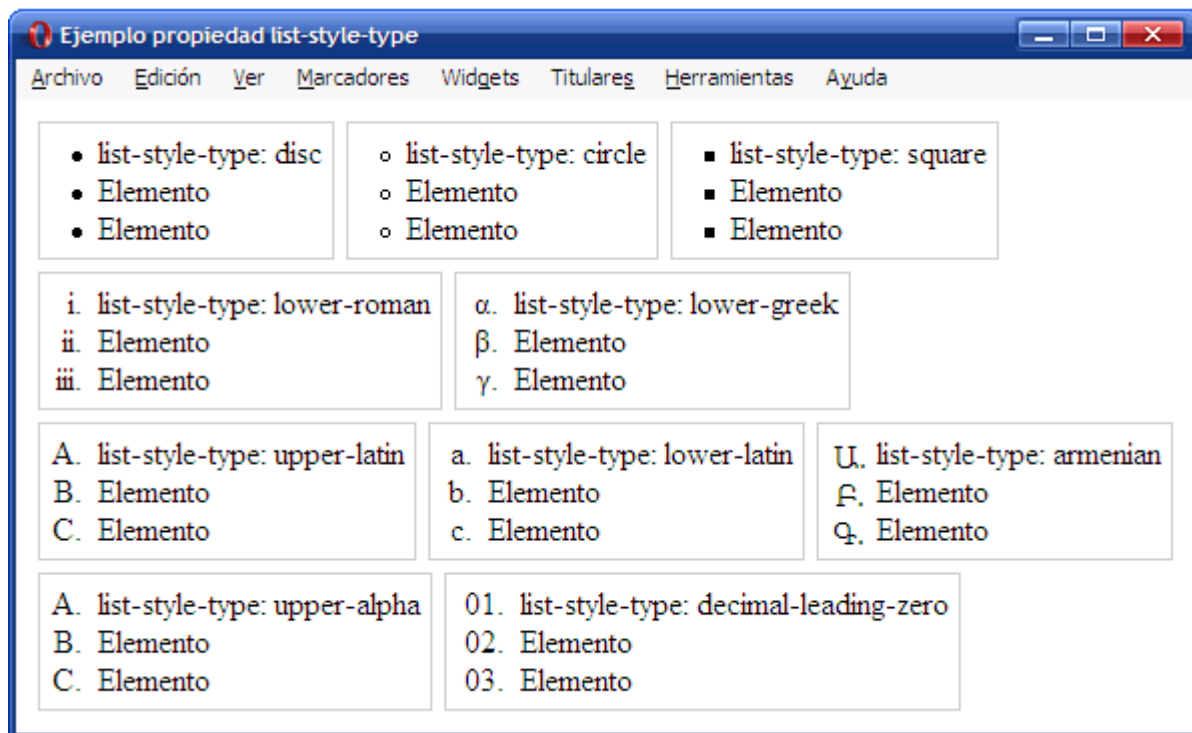


Figura 9.1 Ejemplo de propiedad `list-style-type`

El código CSS de algunas de las listas del ejemplo anterior se muestra a continuación:

```
<ul style="list-style-type: square">
  <li>list-style-type: square</li>
  <li>Elemento</li>
  <li>Elemento</li>
</ul>
```

```
<ol style="list-style-type: lower-roman">
  <li>list-style-type: lower-roman</li>
  <li>Elemento</li>
  <li>Elemento</li>
</ol>
```

```
<ol style="list-style-type: decimal-leading-zero; padding-left: 2em;">
  <li>list-style-type: decimal-leading-zero</li>
  <li>Elemento</li>
  <li>Elemento</li>
</ol>
```

La propiedad `list-style-position` permite controlar la colocación de las viñetas.

Propiedad	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:

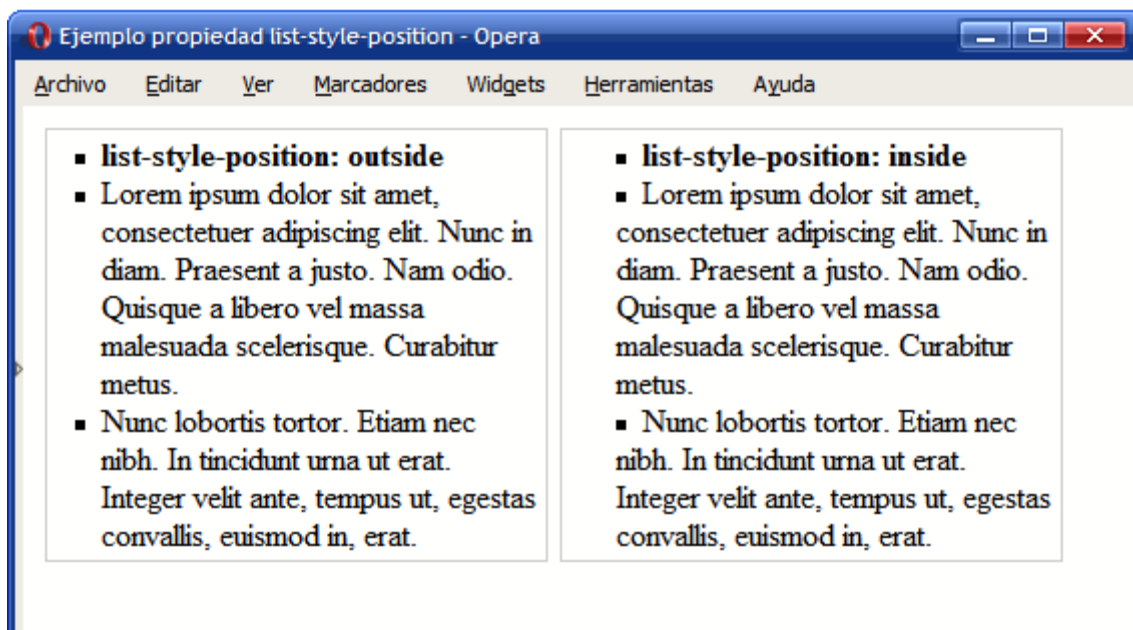


Figura 9.2 Ejemplo de propiedad list-style-position

Utilizando las propiedades anteriores ([list-style-type](#) y [list-style-position](#)), se puede seleccionar el tipo de viñeta y su posición, pero no es posible personalizar algunas de sus características básicas como su color y tamaño.

Cuando se requiere personalizar el aspecto de las viñetas, se debe emplear la propiedad [list-style-image](#), que permite mostrar una imagen propia en vez de una viñeta automática.

Propiedad	<code>list-style-image</code>
Valores	<code>url</code> <code>none</code> <code>inherit</code>
Se aplica a	Elementos de una lista
Valor inicial	<code>none</code>
Descripción	Permite reemplazar las viñetas automáticas por una imagen personalizada

Las imágenes personalizadas se indican mediante la URL de la imagen. Si no se encuentra la imagen o no se puede cargar, se muestra la viñeta automática correspondiente (salvo que explícitamente se haya eliminado mediante la propiedad [list-style-type](#)).

La siguiente imagen muestra el uso de la propiedad `list-style-image` mediante tres ejemplos sencillos de listas con viñetas personalizadas:

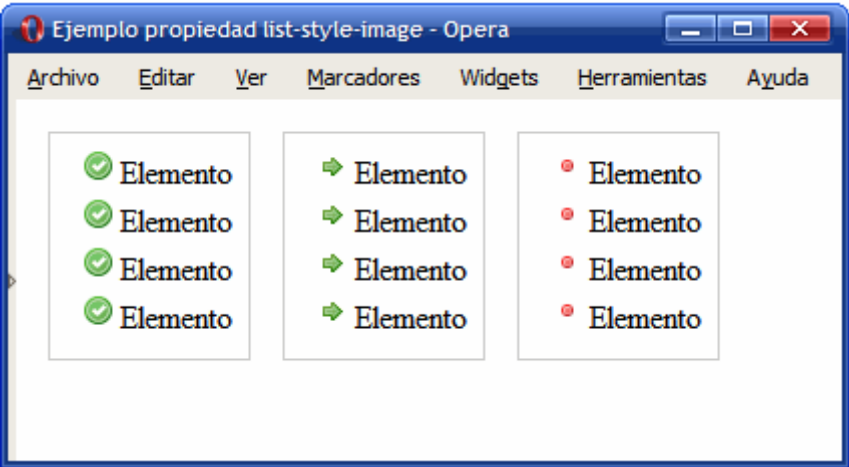


Figura 9.3 Ejemplo de propiedad `list-style-image`

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

```
ul.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`.

Propiedad	<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
Valor inicial	-
Descripción	Propiedad que permite establecer de forma simultánea todas las opciones de una lista

En la definición anterior, la notación `||` significa que el orden en el que se indican los valores de la propiedad es indiferente. El siguiente ejemplo indica que no se debe mostrar ni viñetas automáticas ni viñetas personalizadas:

```
ul { list-style: none }
```

Cuando se utiliza una viñeta personalizada, es conveniente indicar la viñeta automática que se mostrará cuando no se pueda cargar la imagen:

```
ul { list-style: url("imagenes/cuadrado_rojo.gif") square; }
```

9.1.2. Menú vertical

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:

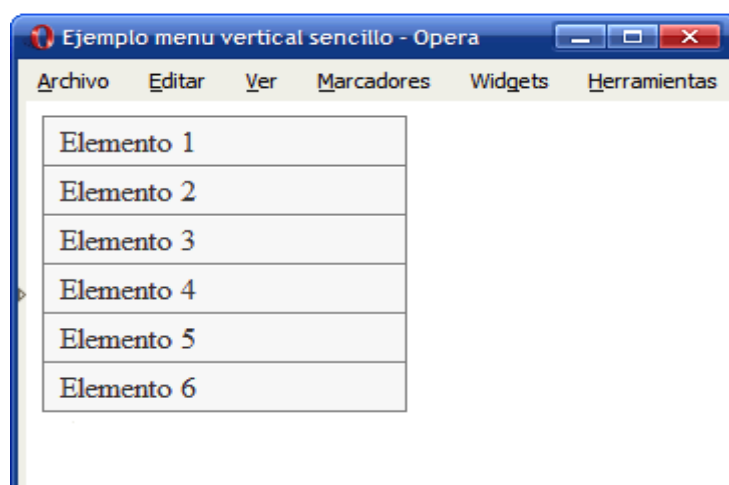


Figura 9.4 Menú vertical sencillo creado con CSS

El proceso de transformación de la lista en un menú requiere de los siguientes pasos:

1) Definir la anchura del menú:

```
ul.menu { width: 180px; }
```

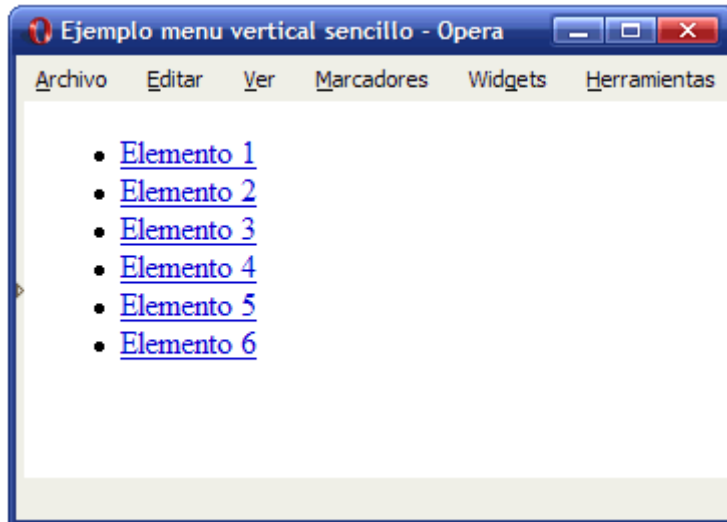


Figura 9.5 Menú vertical: definiendo su anchura

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

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

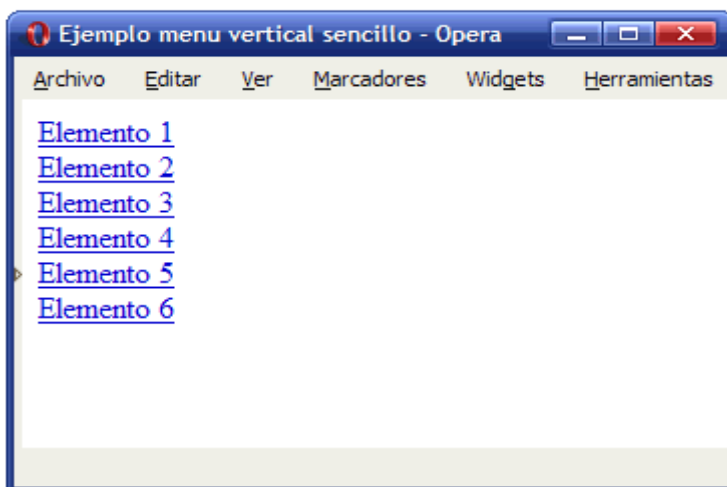


Figura 9.6 Menú vertical: eliminar viñetas por defecto

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

```
ul.menu {  
    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;  
}
```



Figura 9.7 Menú vertical: añadiendo bordes

4) Aplicar estilos a los enlaces: mostrarlos como un elemento de bloque para que ocupen todo el espacio de cada `` del menú, añadir un espacio de relleno y modificar los colores y la decoración por defecto:

```
ul.menu li a {  
    color: #333;  
    display: block;
```

```
padding: .2em 0 .2em .5em;  
text-decoration: none;  
}
```

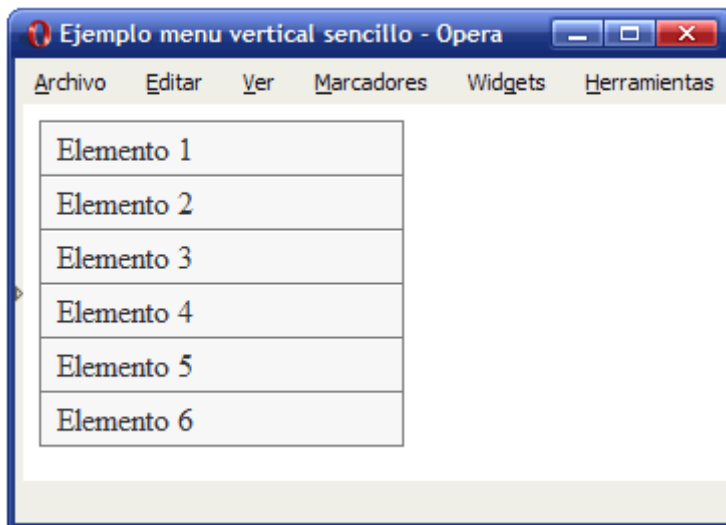


Figura 9.8 Aspecto final del menú vertical sencillo creado con CSS

Capítulo 10. Tablas

10.1. Estilos básicos

Cuando se aplican bordes a las celdas de una tabla, el aspecto por defecto con el que se muestra en un navegador es el siguiente:

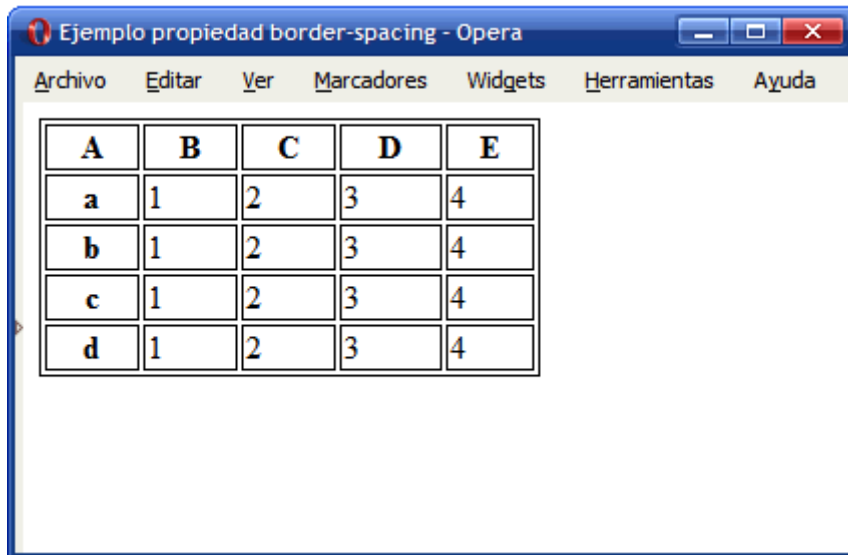


Figura 10.1 Aspecto por defecto de los bordes de una tabla

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

```
.normal {  
    width: 250px;  
    border: 1px solid #000;  
}  
  
.normal th, .normal td {  
    border: 1px solid #000;  
}
```

```
<table class="normal" summary="Tabla genérica">  
    <tr>  
        <th scope="col">A</th>  
        <th scope="col">B</th>  
        <th scope="col">C</th>  
        <th scope="col">D</th>
```

```
<th scope="col">E</th>

</tr>

...

</table>
```

El estándar CSS 2.1 define dos modelos diferentes para el tratamiento de los bordes de las celdas. La propiedad que permite seleccionar el modelo de bordes es `border-collapse`:

Propiedad	<code>border-collapse</code>
Valores	<code>collapse</code> <code>separate</code> <code>inherit</code>
Se aplica a	Todas las tablas
Valor inicial	<code>separate</code>
Descripción	Define el mecanismo de fusión de los bordes de las celdas adyacentes de una tabla

El modelo `collapse` fusiona de forma automática los bordes de las celdas adyacentes, mientras que el modelo `separate` fuerza a que cada celda muestre sus cuatro bordes. Por defecto, los navegadores utilizan el modelo `separate`, tal y como se puede comprobar en el ejemplo anterior.

En general, los diseñadores prefieren el modelo `collapse` porque estéticamente resulta más atractivo y más parecido a las tablas de datos tradicionales. El ejemplo anterior se puede rehacer para mostrar la tabla con bordes sencillos y sin separación entre celdas:

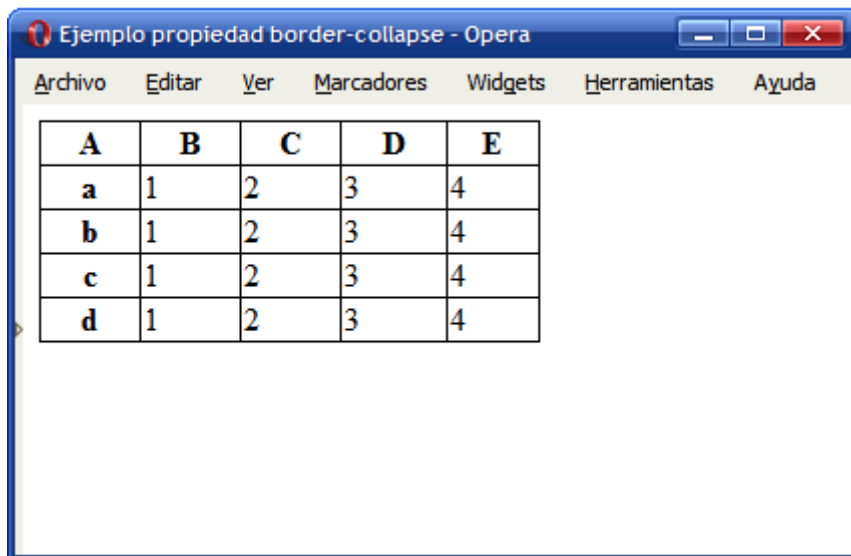


Figura 10.2 Ejemplo de propiedad border-collapse

El código CSS completo del ejemplo anterior se muestra a continuación:

```
.normal {
    width: 250px;
    border: 1px solid #000;
    border-collapse: collapse;
}

.normal th, .normal td {
    border: 1px solid #000;
}

<table class="normal" summary="Tabla genérica">
    <tr>
        <th scope="col">A</th>
        <th scope="col">B</th>
        <th scope="col">C</th>
        <th scope="col">D</th>
        <th scope="col">E</th>
    </tr>
    ...
</table>
```

Aunque parece sencillo, el mecanismo que utiliza el modelo `collapse` es muy complejo, ya que cuando los bordes que se fusionan no son exactamente iguales, debe tener en cuenta la anchura de cada borde, su estilo y el tipo de celda que contiene el borde (columna, fila, grupo de filas, grupo de columnas) para determinar la prioridad de cada borde.

Si se opta por el modelo `separate` (que es el 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.

Propiedad	<code>border-spacing</code>
Valores	<code>unidad de medida unidad de medida?</code> <code>inherit</code>
Se aplica a	Todas las tablas
Valor inicial	0
Descripción	Establece la separación entre los bordes de las celdas adyacentes de una tabla

Si solamente se indica como valor una medida, se asigna ese valor como separación horizontal y vertical. Si se indican dos medidas, la primera es la separación horizontal y la segunda es la separación vertical entre celdas.

La propiedad `border-spacing` sólo controla la separación entre celdas y por tanto, no se puede utilizar para modificar el tipo de modelo de bordes que se utiliza. En concreto, si se establece un valor igual a 0 para la separación entre los bordes de las celdas, el resultado es muy diferente al modelo `collapse`:

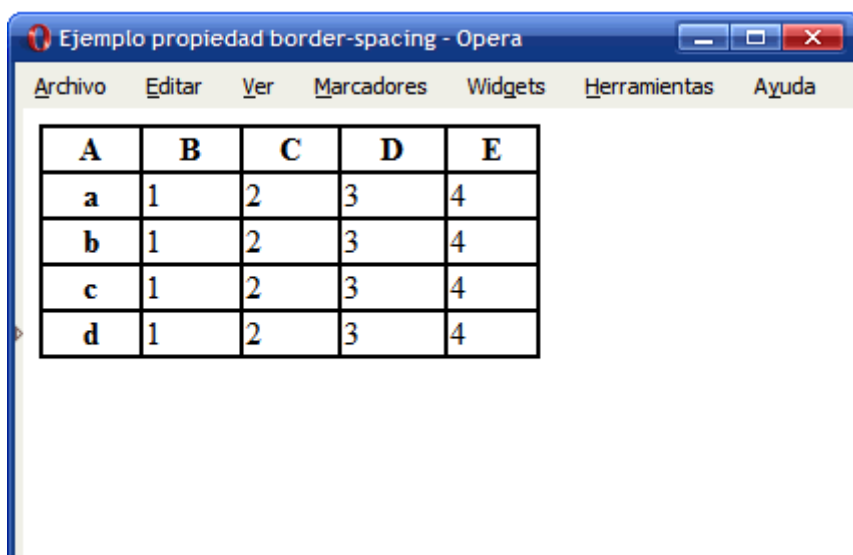


Figura 10.3 Ejemplo de propiedad border-spacing

En la tabla del ejemplo anterior, se ha establecido la propiedad `border-spacing: 0`, por lo que el navegador no introduce ninguna separación entre los bordes de las celdas. Además, como no se ha establecido de forma explícita ningún modelo de bordes, el navegador utiliza el modelo `separate`.

El resultado es que `border-spacing: 0` muestra los bordes con una anchura doble, ya que en realidad se trata de dos bordes iguales sin separación entre ellos. Por tanto, las diferencias visuales con el modelo `border-collapse: collapse` son muy evidentes.

10.2. Estilos avanzados

CSS define otras propiedades específicas para el control del aspecto de las tablas. Una de ellas es el tratamiento que reciben las celdas vacías de una tabla, que se controla mediante la propiedad `empty-cells`. Esta propiedad sólo se aplica cuando el modelo de bordes de la tabla es de tipo `separate`.

Propiedad	<code>empty-cells</code>
Valores	<code>show</code> <code>hide</code> <code>inherit</code>
Se aplica a	Celdas de una tabla
Valor inicial	<code>show</code>

Propiedad	<code>empty-cells</code>
Descripción	Define el mecanismo utilizado para el tratamiento de las celdas vacías de una tabla

El valor `hide` indica que las celdas vacías no se deben mostrar. Una celda vacía es aquella que no tiene ningún contenido, ni siquiera un espacio en blanco o un ` `.

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

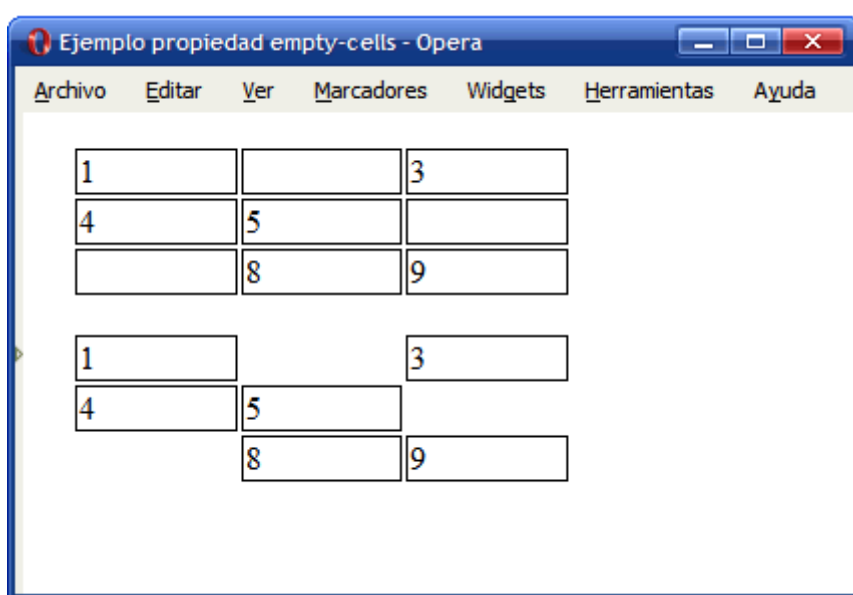


Figura 10.4 Ejemplo de propiedad `empty-cells`

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

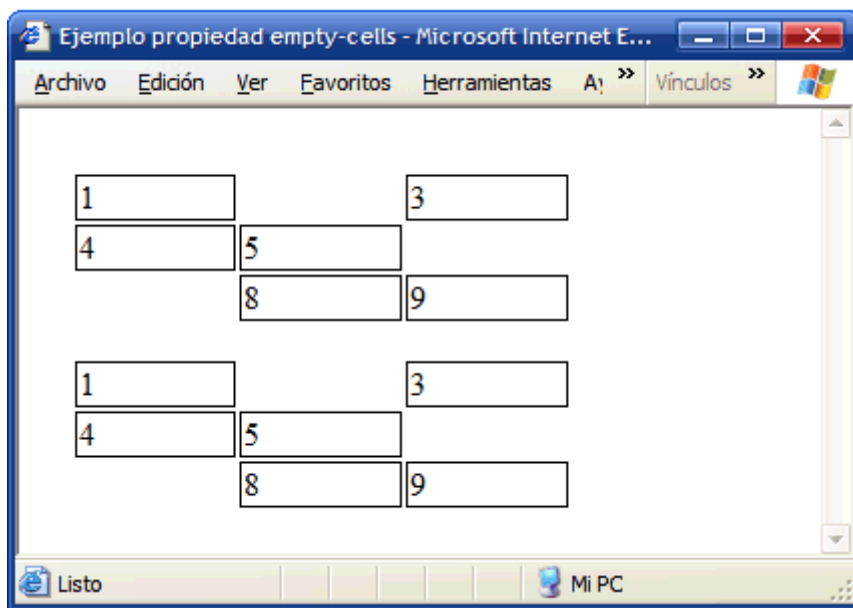


Figura 10.5 Internet Explorer no soporta la propiedad empty-cells

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.

Propiedad	<code>caption-side</code>
Valores	<code>top</code> <code>bottom</code> <code>inherit</code>
Se aplica a	Los elementos <code>caption</code>
Valor inicial	<code>top</code>
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:

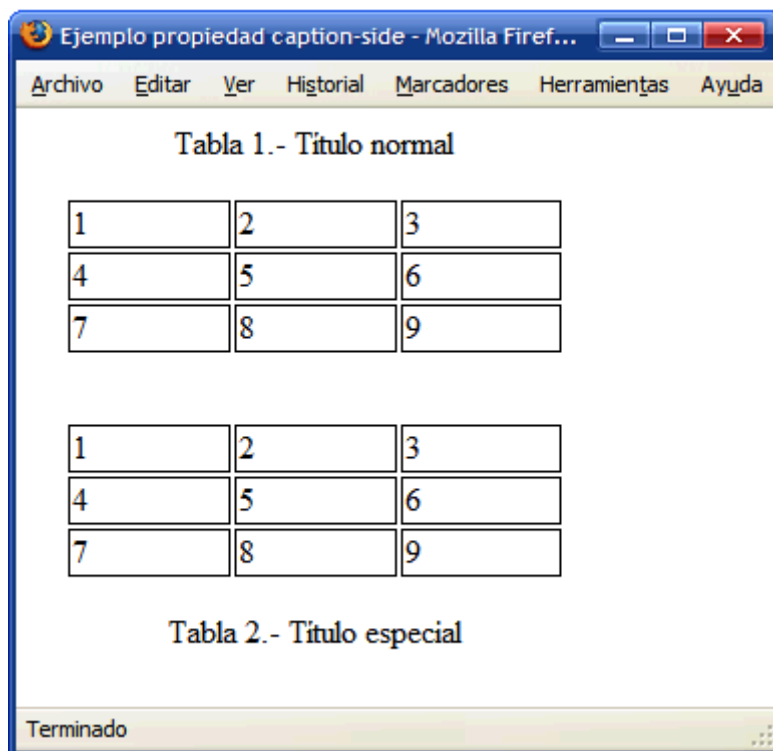


Figura 10.6 Ejemplo de propiedad caption-side

Capítulo 11. Formularios

11.1. Estilos básicos

11.1.1. Mostrar un botón como un enlace

Como ya se vio anteriormente, el estilo por defecto de los enlaces se puede modificar para que se muestren como botones de formulario. Ahora, los botones de formulario también se pueden modificar para que parezcan enlaces.

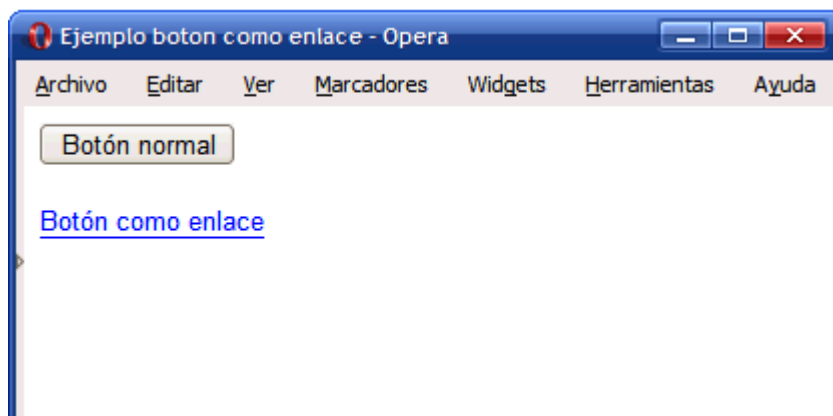


Figura 11.1 Mostrando un botón como si fuera un enlace

Las reglas CSS del ejemplo anterior son las siguientes:

```
.enlace {  
    border: 0;  
    padding: 0;  
    background-color: transparent;  
    color: blue;  
    border-bottom: 1px solid blue;  
}
```

```
<input type="button" value="Botón normal" />
```

```
<input class="enlace" type="button" value="Botón como enlace" />
```

11.1.2. Mejoras en los campos de texto

Por defecto, los campos de texto de los formularios no incluyen ningún espacio de relleno, por lo que el texto introducido por el usuario aparece *pegado* a los bordes del cuadro de texto.

Añadiendo un pequeño `padding` a cada elemento `<input>`, se mejora notablemente el aspecto del formulario:

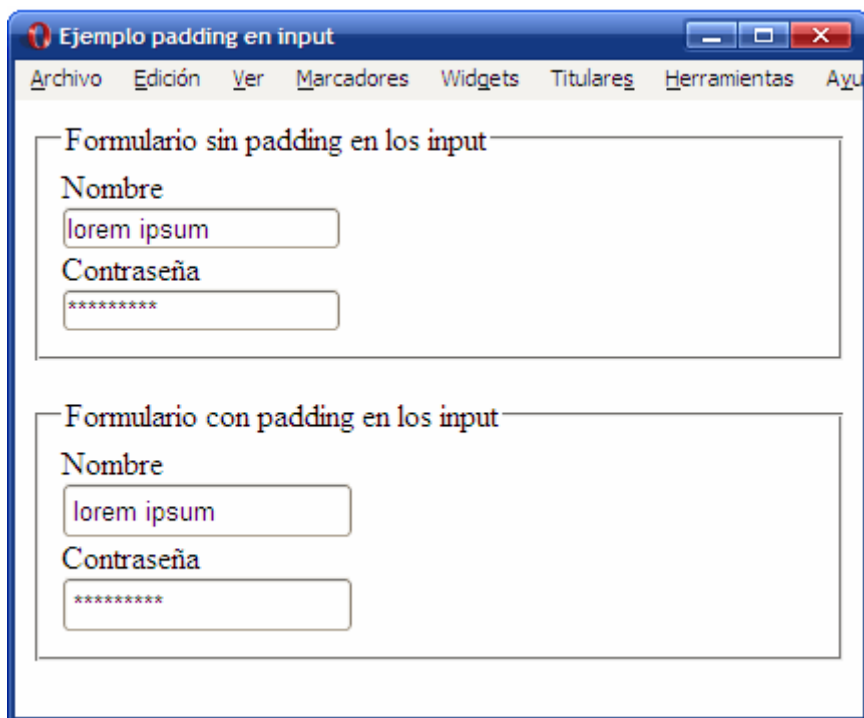


Figura 11.2 Mejorando el aspecto de los formularios gracias al padding

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

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

11.1.3. Labels alineadas y formateadas

Los elementos `<input>` y `<label>` de los formularios son elementos en línea, por lo que el aspecto que muestran los formularios por defecto, es similar al de la siguiente imagen:

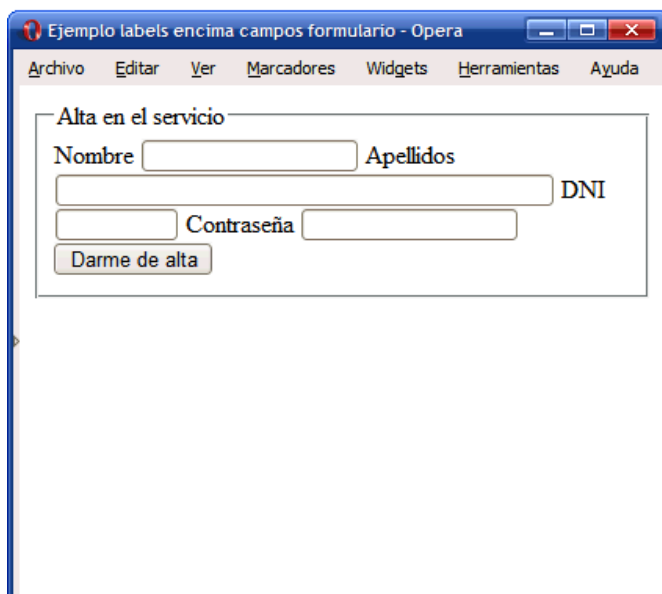


Figura 11.3 Aspecto por defecto que muestran los formularios

El código HTML del ejemplo anterior es el siguiente:

```
<form>

<fieldset>

  <legend>Alta en el servicio</legend>


  <label for="nombre">Nombre</label>
  <input type="text" id="nombre" />


  <label for="apellidos">Apellidos</label>
  <input type="text" id="apellidos" size="50" />


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


  <label for="contrasena">Contraseña</label>
  <input type="password" id="contrasena" />


  <input class="btn" type="submit" value="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:

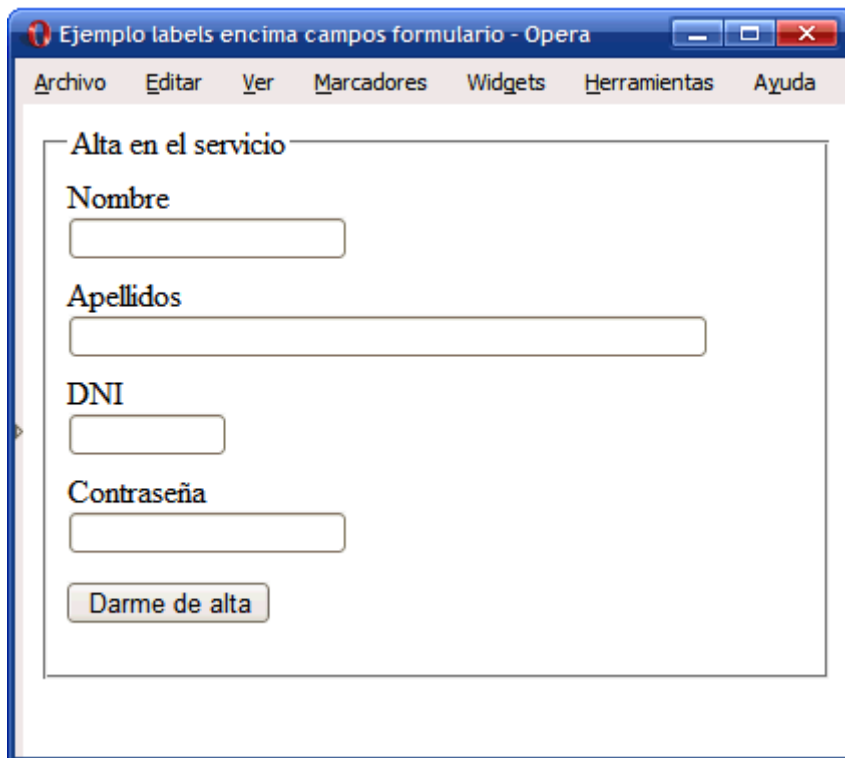


Figura 11.4 Mostrando las etiquetas label encima de los campos del formulario

En primer lugar, se muestran los elementos `<label>` como elementos de bloque, para que añadan una separación para cada campo del formulario. Además, se añade un margen superior para no mostrar juntas todas las filas del formulario:

```
label {  
    display: block;  
    margin: .5em 0 0 0;  
}
```

El botón del formulario también se muestra como un elemento de bloque y se le añade un margen para darle el aspecto final deseado:

```
.btn {  
    display: block;  
    margin: 1em 0;  
}
```

En ocasiones, es más útil mostrar todos los campos del formulario con su `<label>` alineada a la izquierda y el campo del formulario a la derecha de cada `<label>`, como muestra la siguiente imagen:

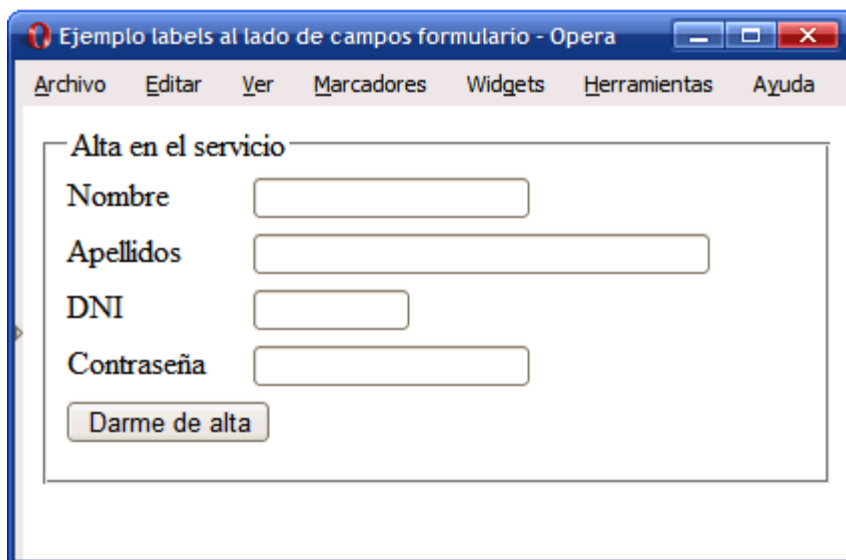


Figura 11.5 Mostrando las etiquetas label alineadas con los campos del formulario

Para mostrar un formulario tal y como aparece en la imagen anterior no es necesario crear una tabla y controlar la anchura de sus columnas para conseguir una alineación perfecta. Sin embargo, sí que es necesario añadir un nuevo elemento (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:

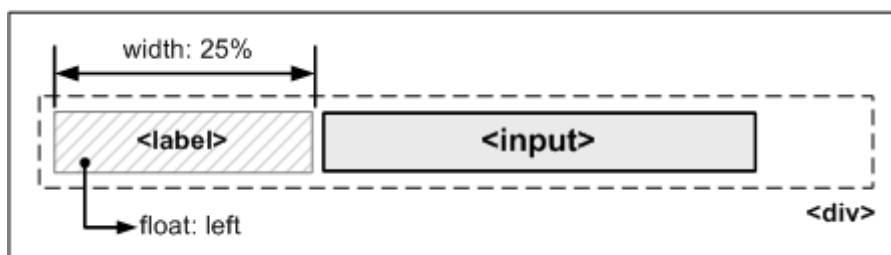


Figura 11.6 Esquema de la técnica de alineación de etiquetas label y campos de formulario

Por tanto, en el código HTML del formulario anterior se añaden los elementos `<div>`:

```
<form>

  <fieldset>

    <legend>Alta en el servicio</legend>

    <div>

      <label for="nombre">Nombre</label>

      <input type="text" id="nombre" />
```



```

</div>

<div>
    <label for="apellidos">Apellidos</label>
    <input type="text" id="apellidos" size="35" />
</div>

...

</fieldset>

</form>

```

Y en el código CSS se añaden las reglas necesarias para alinear los campos del formulario:

```

div {
    margin: .4em 0;
}

div label {
    width: 25%;
    float: left;
}

```

11.2. Estilos avanzados

11.2.1. Formulario en varias columnas

Los formularios complejos con decenas de campos pueden ocupar mucho espacio en la ventana del navegador. Además del uso de pestañas para agrupar los campos relacionados en un formulario, también es posible mostrar el formulario a dos columnas, para aprovechar mejor el espacio.

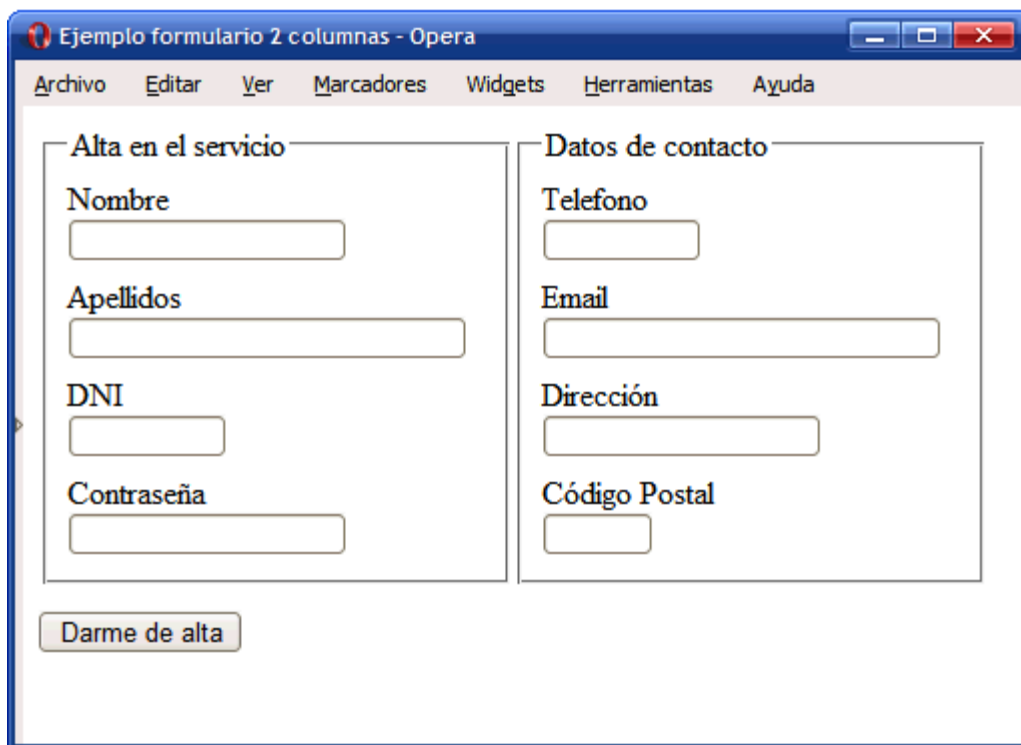


Figura 11.7 Ejemplo de formulario a dos columnas

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>`.

11.2.2. Resaltar el campo seleccionado

Una de las mejoras más útiles para los formularios HTML consiste en resaltar de alguna forma especial el campo en el que el usuario está introduciendo datos. Para ello, CSS define la pseudo-clase `:focus`, que permite aplicar estilos especiales al elemento que en ese momento tiene el *foco* o atención del usuario.

La siguiente imagen muestra un formulario que resalta claramente el campo en el que el usuario está introduciendo la información:

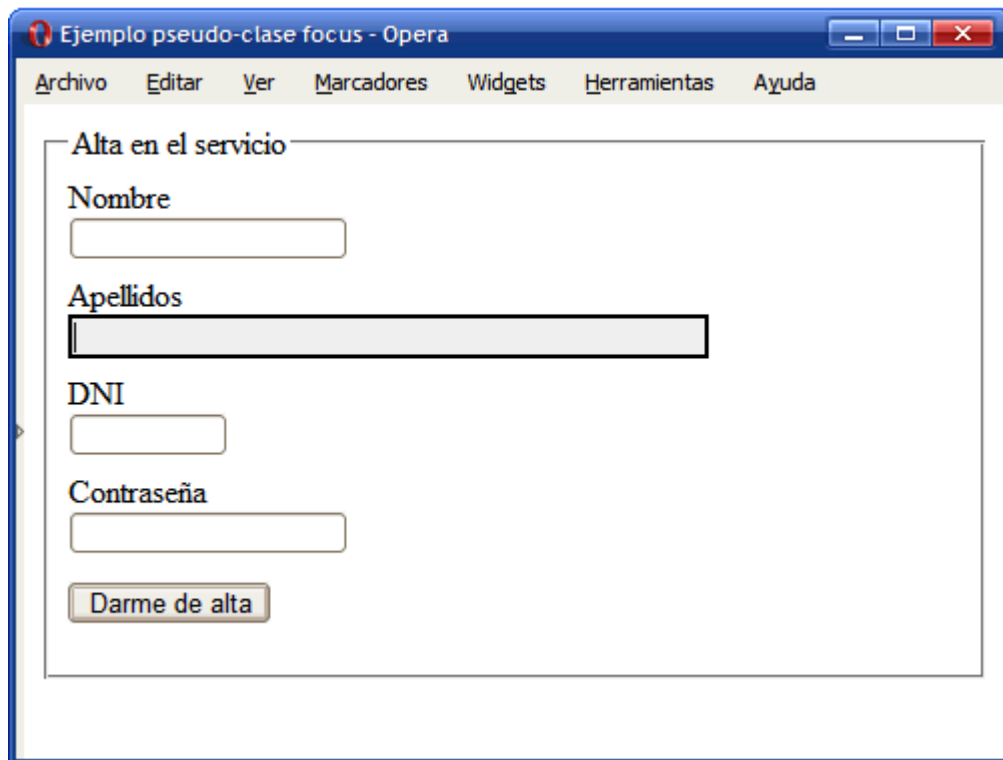


Figura 11.8 Ejemplo de pseudo-clase `:focus`

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

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

Desafortunadamente, la pseudo-clase `:focus` no funciona en navegadores obsoletos como Internet Explorer 6, por lo que si la página debe visualizarse de la misma forma en todos los navegadores, es preciso recurrir a soluciones con JavaScript.

Capítulo 12. Layout

El diseño de las páginas web habituales se divide en bloques: cabecera, menú, contenidos y pie de página. Visualmente, los bloques se disponen en varias filas y columnas. Por este motivo, hace varios años la estructura de las páginas HTML se definía mediante tablas.

El desarrollo de CSS ha permitido que se puedan realizar los mismos diseños sin utilizar tablas HTML. Las principales ventajas de diseñar la estructura de las páginas web con CSS en vez de con tablas HTML son las siguientes:

- **Mantenimiento:** una página diseñada exclusivamente con CSS es mucho más fácil de mantener que una página diseñada con tablas. Cambiar el aspecto de una página creada con CSS es tan fácil como modificar unas pocas reglas en las hojas de estilos. Sin embargo, realizar la misma modificación en una página creada con tablas supone un esfuerzo muy superior y es más probable cometer errores.
- **Accesibilidad:** las páginas creadas con CSS son más accesibles que las páginas diseñadas con tablas. De hecho, los navegadores que utilizan las personas discapacitadas (en especial las personas invidentes) pueden tener dificultades con la estructura de las páginas complejas creadas con tablas HTML. No obstante, diseñar una página web exclusivamente con CSS no garantiza que la página sea accesible.
- **Velocidad de carga:** el código HTML de una página diseñada con tablas es 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 `<u1>` para crear párrafos de texto.

Por estos motivos, la estructura basada en tablas ha dado paso a la estructura basada exclusivamente en CSS. Aunque crear la estructura de las páginas sólo con CSS presenta en ocasiones retos importantes, en general es más sencilla y flexible.

En este capítulo se muestra cómo crear algunas de las estructuras o *layouts* más habituales de los diseños web utilizando exclusivamente CSS.

12.1. Centrar una página horizontalmente

A medida que aumenta el tamaño y la resolución de las pantallas de ordenador, se hace más difícil diseñar páginas que se adapten al tamaño de la ventana del navegador. El principal reto que se presenta con resoluciones superiores a 1024 x 768 píxel, es que las líneas de texto son demasiado largas como para leerlas con comodidad. Por ese motivo, normalmente se opta por diseños con

una anchura fija limitada a un valor aceptable para mantener la legibilidad del texto.

Por otra parte, los navegadores alinean por defecto las páginas web a la izquierda de la ventana. Cuando la resolución de la pantalla es muy grande, la mayoría de páginas de anchura fija alineadas a la izquierda parecen muy estrechas y provocan una sensación de vacío.

La solución más sencilla para evitar los grandes espacios en blanco consiste en crear páginas con una anchura fija adecuada y centrar la página horizontalmente respecto de la ventana del navegador. Las siguientes imágenes muestran el aspecto de una página centrada a medida que aumenta la anchura de la ventana del navegador.



Figura 12.1 Página de anchura fija centrada mediante CSS



Figura 12.2 Página de anchura fija centrada mediante CSS



Figura 12.3 Página de anchura fija centrada mediante CSS

Utilizando la propiedad `margin` de CSS, es muy sencillo centrar una página web horizontalmente. La solución consiste en agrupar todos los contenidos de la página en un elemento `<div>` y asignarle a ese `<div>` unos márgenes laterales automáticos. El `<div>` que encierra los contenidos se suele llamar `contenedor` (en inglés se denomina `wrapper` o `container`):

```
#contenedor {  
    width: 300px;  
    margin: 0 auto;  
}  
  
<body>  
    <div id="contenedor">  
        <h1>Lorem ipsum dolor sit amet</h1>  
        ...  
    </div>  
</body>
```

Como se sabe, el valor `0 auto` significa que los márgenes superior e inferior son iguales a `0` y los márgenes laterales toman un valor de `auto`. Cuando se asignan márgenes laterales automáticos a un elemento, los navegadores centran ese elemento respecto de su elemento padre. En este ejemplo, el elemento padre del `<div>` es la propia página (el elemento `<body>`), por lo que se consigue centrar el elemento `<div>` respecto de la ventana del navegador.

Modificando ligeramente el código CSS anterior se puede conseguir un diseño dinámico o *líquido* (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 12.4 Página de anchura variable centrada mediante CSS



Figura 12.5 Página de anchura variable centrada mediante CSS



Figura 12.6 Página de anchura variable centrada mediante CSS

12.2. Centrar una página verticalmente

Cuando se centra una página web de forma horizontal, sus márgenes laterales se adaptan dinámicamente de forma que la página siempre aparece en el centro de la ventana del navegador, independientemente de la anchura de la ventana. De la misma forma, cuando se centra una página web verticalmente, el objetivo es que sus contenidos aparezcan en el centro de la ventana del navegador y por tanto, que sus márgenes verticales se adapten de forma dinámica en función del tamaño de la ventana del navegador.

Aunque centrar una página web horizontalmente es muy sencillo, centrarla verticalmente es mucho más complicado. Afortunadamente, no es muy común que una página web aparezca centrada de forma vertical. El motivo es que la mayoría de páginas web son más altas que la ventana del navegador, por lo que no es posible centrarlas verticalmente.

A continuación se muestra la forma de centrar una página web respecto de la ventana del navegador, es decir, centrarla tanto horizontalmente como verticalmente.

Siguiendo el mismo razonamiento que el planteado para centrar la página horizontalmente, se podrían utilizar las siguientes reglas CSS para centrar la página respecto de la ventana del navegador:

```
#contenedor {  
    width: 300px;  
    height: 250px;  
    margin: auto;  
}
```

```
<body>  
    <div id="contenedor">
```



```

    <h1>Lorem ipsum dolor sit amet</h1>

    ...

    </div>

</body>

```

Si el valor `auto` se puede utilizar para que los márgenes laterales se adapten dinámicamente, también debería ser posible utilizar el valor `auto` para los márgenes verticales. Desafortunadamente, la propiedad `margin: auto` no funciona tal y como se espera para los márgenes verticales y la página no se muestra centrada.

La solución correcta para centrar verticalmente una página web se basa en el posicionamiento absoluto e implica realizar un cálculo matemático sencillo. A continuación se muestra el esquema gráfico de los cuatro pasos necesarios para centrar una página web en la ventana del navegador:

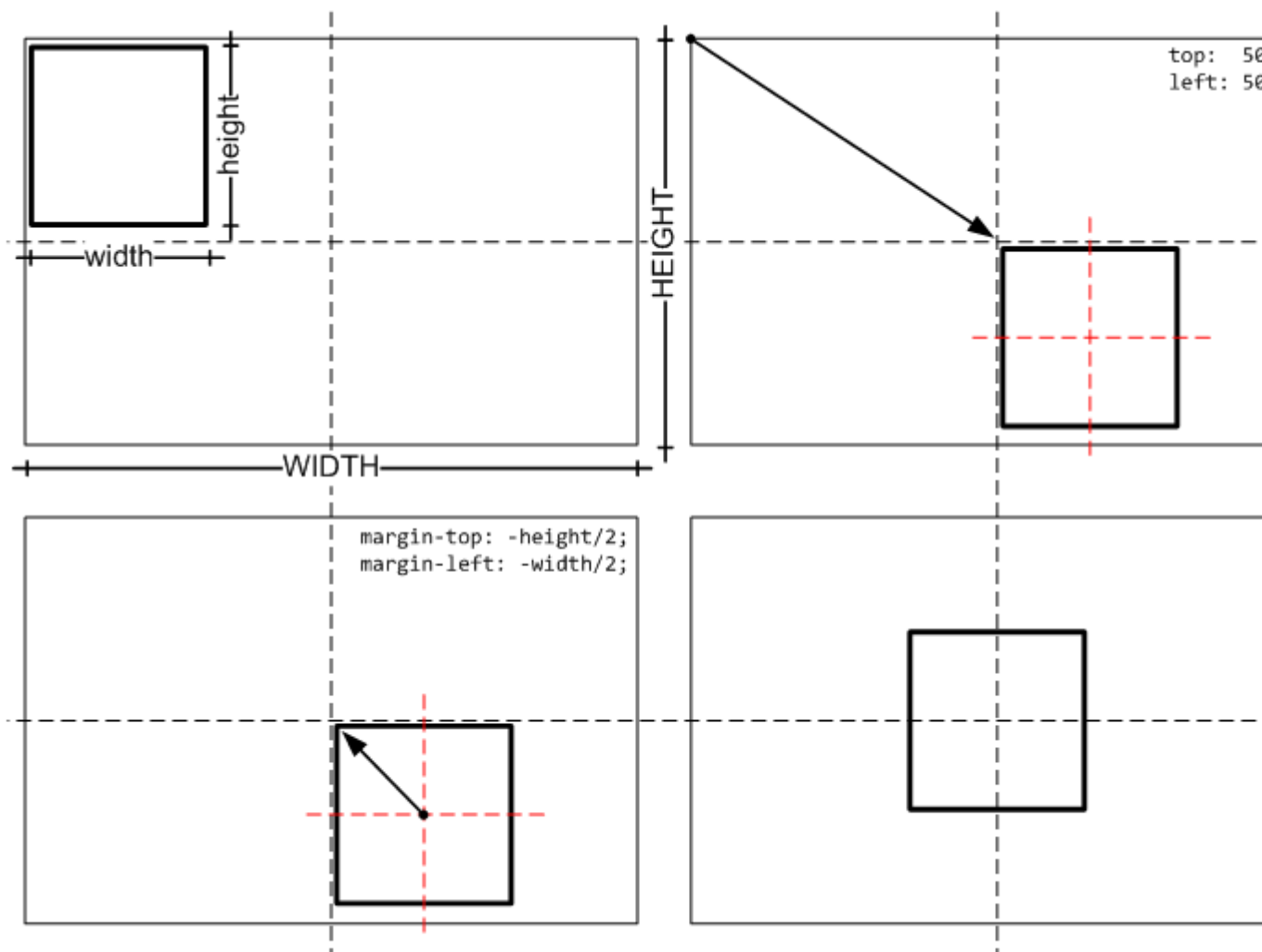


Figura 12.7 Pasos necesarios para centrar verticalmente una página web

En primer lugar, se asigna una altura y una anchura al elemento que encierra todos los contenidos de la página. En la primera figura, los contenidos de la página tienen una anchura llamada `width` y una altura llamada `height` que son

menores que la anchura y altura de la ventana del navegador. En el siguiente ejemplo, se supone que tanto la anchura como la altura de la página es igual a 500px:

```
#contenedor {  
    width: 500px;  
    height: 500px;  
}
```

```
<body>  
    <div id="contenedor">  
        <h1>Lorem ipsum dolor sit amet</h1>  
        ...  
    </div>  
</body>
```

A continuación, se posiciona de forma absoluta el elemento `contenedor` y se asigna un valor de 50% tanto a la propiedad `top` como a la propiedad `left`. El resultado es que la esquina superior izquierda del elemento `contenedor` se posiciona en el centro de la ventana del navegador:

```
#contenedor {  
    width: 500px;  
    height: 500px;  
  
    position: absolute;  
    top: 50%;  
    left: 50%;  
}
```

Si la página se debe mostrar en el centro de la ventana del navegador, es necesario desplazar hacia arriba y hacia la izquierda los contenidos de la página web. Para determinar el desplazamiento necesario, se realiza un cálculo matemático sencillo. Como se ve en la tercera figura del esquema anterior, el punto central de la página debe desplazarse hasta el centro de la ventana del navegador.

Como se desprende de la imagen anterior, la página web debe moverse hacia arriba una cantidad igual a la mitad de su altura y debe desplazarse hacia la

izquierda una cantidad equivalente a la mitad de su anchura. Utilizando las propiedades `margin-top` y `margin-left` con valores negativos, la página se desplaza hasta el centro de la ventana del navegador.

```
#contenedor {  
  
    width: 500px;  
  
    height: 500px;  
  
    position: absolute;  
  
    top: 50%;  
  
    left: 50%;  
  
    margin-top: -250px;    /* height/2 = 500px / 2 */  
    margin-left: -250px;   /* width/2 = 500px / 2 */  
}
```

Con las reglas CSS anteriores, la página web siempre aparece centrada verticalmente y horizontalmente respecto de la ventana del navegador. El motivo es que la anchura/altura de la página son fijas (propiedades `width` y `height`), el desplazamiento necesario para centrarla también es fijo (propiedades `margin-top` y `margin-left`) y el desplazamiento hasta el centro de la ventana del navegador se calcula dinámicamente gracias al uso de porcentajes en las propiedades `top` y `left`.

Para centrar una página sólo verticalmente, se debe prescindir tanto del posicionamiento horizontal como del desplazamiento horizontal:

```
#contenedor {  
  
    width: 500px;  
  
    height: 500px;  
  
    position: absolute;  
  
    top: 50%;  
  
    margin-top: -250px;    /* height/2 = 500px / 2 */  
}
```

12.3. Estructura o layout

12.3.1. Diseño a 2 columnas con cabecera y pie de página

El objetivo de este diseño es definir una estructura de página con cabecera y pie, un menú lateral de navegación y una zona de contenidos. La anchura de la página se fija en **700px**, la anchura del menú es de **150px** y la anchura de los contenidos es de **550px**:

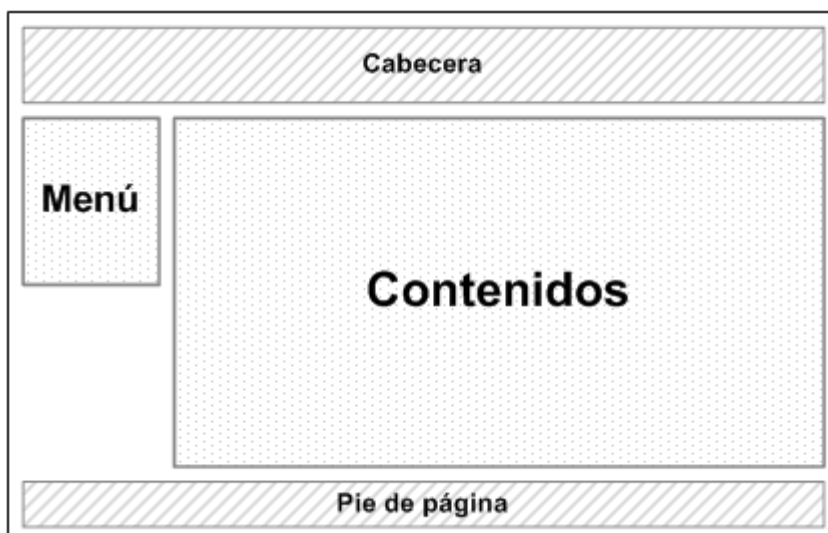


Figura 12.8 Esquema del diseño a 2 columnas con cabecera y pie de página

La solución CSS se basa en el uso de la propiedad **float** para los elementos posicionados como el menú y los contenidos y el uso de la propiedad **clear** en el pie de página para evitar los solapamientos ocasionados por los elementos posicionados con **float**.

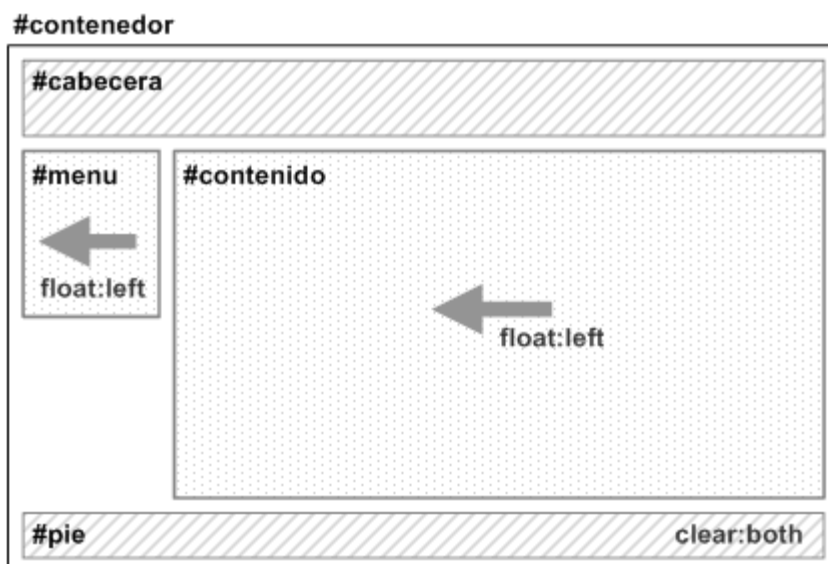


Figura 12.9 Propiedades CSS necesarias en el diseño a dos columnas con cabecera y pie de página

El código HTML y CSS mínimos para definir la estructura de la página sin aplicar ningún estilo adicional son los siguientes:

```
#contenedor {  
    width: 700px;  
}  
#cabecera {  
}  
#menu {  
    float: left;  
    width: 150px;  
}  
#contenido {  
    float: left;  
    width: 550px;  
}  
#pie {  
    clear: both;  
}
```

```
<body>  
<div id="contenedor">  
    <div id="cabecera">  
    </div>  
  
    <div id="menu">  
    </div>  
  
    <div id="contenido">  
    </div>  
  
    <div id="pie">
```

```
</div>

</div>

</body>
```

En los estilos CSS anteriores se ha optado por desplazar tanto el menú como los contenidos hacia la izquierda de la página (`float: left`). Sin embargo, en este caso también se podría desplazar el menú hacia la izquierda (`float:left`) y los contenidos hacia la derecha (`float: right`).

El diseño anterior es de anchura fija, lo que significa que no se adapta a la anchura de la ventana del navegador. Para conseguir una página de anchura variable y que se adapte de forma dinámica a la ventana del navegador, se deben aplicar las siguientes reglas CSS:

```
#contenedor {

}

#cabecera {

}

#menu {

    float: left;

    width: 15%;

}

#contenido {

    float: left;

    width: 85%;

}

#pie {

    clear: both;

}
```

Si se indican la anchuras de los bloques que forman la página en porcentajes, el diseño final es dinámico. Para crear diseños de anchura fija, basta con establecer las anchuras de los bloques en píxel.

12.3.2. Diseño a 3 columnas con cabecera y pie de página

Además del diseño a dos columnas, el diseño más utilizado es el de tres columnas con cabecera y pie de página. En este caso, los contenidos se dividen en dos zonas diferenciadas: zona principal de contenidos y zona lateral de contenidos auxiliares:

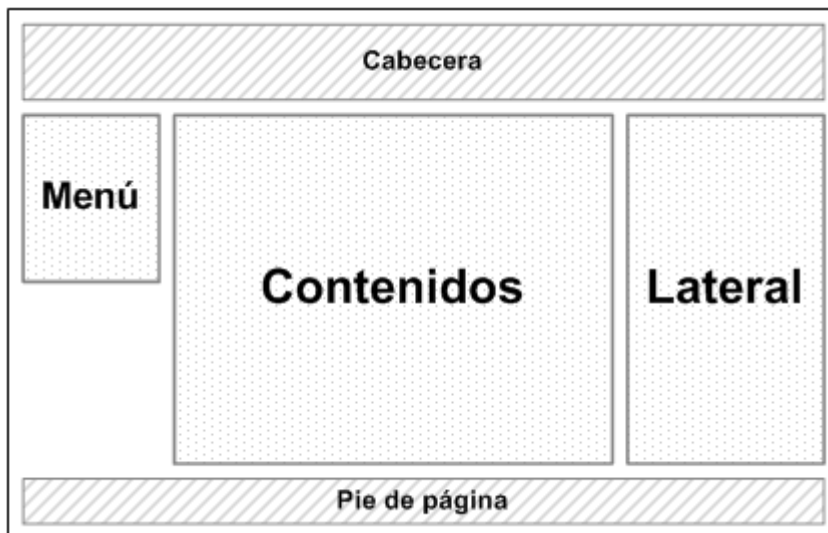


Figura 12.10 Esquema del diseño a tres columnas con cabecera y pie de página

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

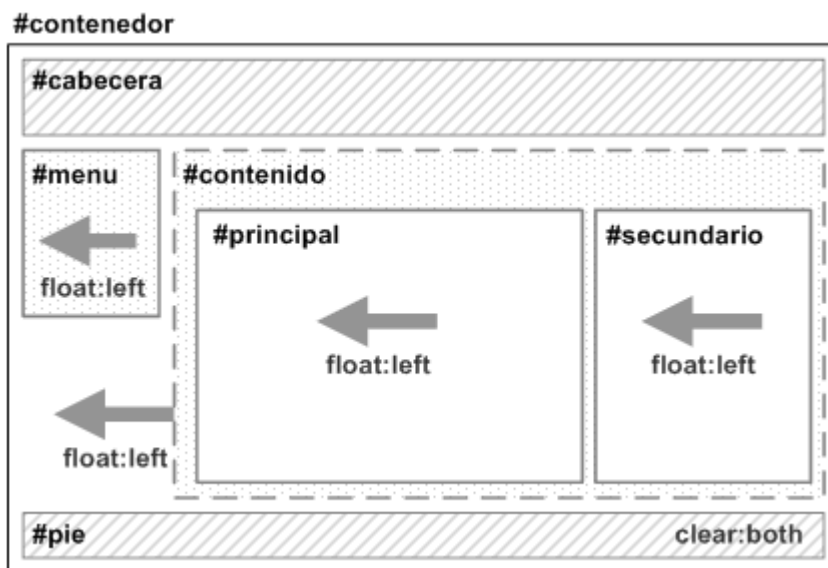


Figura 12.11 Propiedades CSS necesarias en el diseño a 3 columnas con cabecera y pie de página

El código HTML y CSS mínimos para definir la estructura de la página sin aplicar ningún estilo adicional son los siguientes:

```
#contenedor {  
  
}  
  
#cabecera {  
  
}  
  
#menu {
```

```

    float: left;

    width: 15%;
}

#contenido {

    float: left;

    width: 85%;
}

#contenido #principal {

    float: left;

    width: 80%;
}

#contenido #secundario {

    float: left;

    width: 20%;
}


#pie {

    clear: both;
}


<body>

<div id="contenedor">

    <div id="cabecera">

        </div>


        <div id="menu">

            </div>


            <div id="contenido">

```



```
<div id="principal">

</div>

<div id="secundario">

</div>

</div>

<div id="pie">

</div>

</div>

</body>
```

El código anterior crea una página con anchura variable que se adapta a la ventana del navegador. Para definir una página con anchura fija, solamente es necesario sustituir las anchuras en porcentajes por anchuras en píxel.

Al igual que sucedía en el diseño a dos columnas, se puede optar por posicionar todos los elementos mediante `float: left` o se puede utilizar `float: left` para el menú y la zona principal de contenidos y `float: right` para el contenedor de los contenidos y la zona secundaria de contenidos.

12.4. Alturas/anchuras máximas y mínimas

Cuando se diseña la estructura de una página web, se debe tomar la decisión de optar por un diseño de anchura fija o un diseño cuya anchura se adapta a la anchura de la ventana del navegador.

Sin embargo, la mayoría de las veces sería conveniente una solución intermedia: que la anchura de la página sea variable y se adapte a la anchura de la ventana del navegador, pero respetando ciertos límites. En otras palabras, que la anchura de la página no sea tan pequeña como para que no se puedan mostrar correctamente los contenidos y tampoco sea tan ancha como para que las líneas de texto no puedan leerse cómodamente.

CSS define cuatro propiedades que permiten limitar la anchura y altura mínima y máxima de cualquier elemento de la página. Las propiedades son `max-width`, `min-width`, `max-height` y `min-height`.

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

Propiedad	max-height
Valor inicial	none
Descripción	Permite definir la altura máxima de un elemento
Propiedad	min-height
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" );  
}
```

12.5. Estilos avanzados

En general, la columna de los contenidos es la más larga y la columna de navegación es la más corta. El principal inconveniente de los diseños mostrados anteriormente es que no se puede garantizar que todas las columnas se muestren con la misma altura.

Si las columnas tienen algún color o imagen de fondo, este comportamiento no es admisible, ya que se vería que alguna columna no llega hasta el final de la columna más larga y el diseño final parecería inacabado.

Desde la aparición de este problema se han presentado numerosas soluciones. La más conocida es la técnica *faux columns* ("columnas falsas") y que simula el

color/imagen de fondo de las columnas laterales mediante la imagen de fondo de la columna central de contenidos.

La técnica fue presentada originalmente por Dan Cederholm en su célebre artículo *"Faux Columns"* (<http://alistapart.com/articles/fauxcolumns/>).

Más recientemente se ha presentado el proyecto *"In Search of the One True Layout"* que busca definir una serie de técnicas que permitan crear de forma sencilla cualquier estructura de página basada en columnas.

La página principal del proyecto se puede encontrar en:
<http://www.positioniseverything.net/articles/onetruelayout/>

Además, está disponible una herramienta interactiva para crear diseños basados en columnas con la posibilidad de definir el número de columnas, su anchura y obligar a que todas las columnas muestren la misma altura:

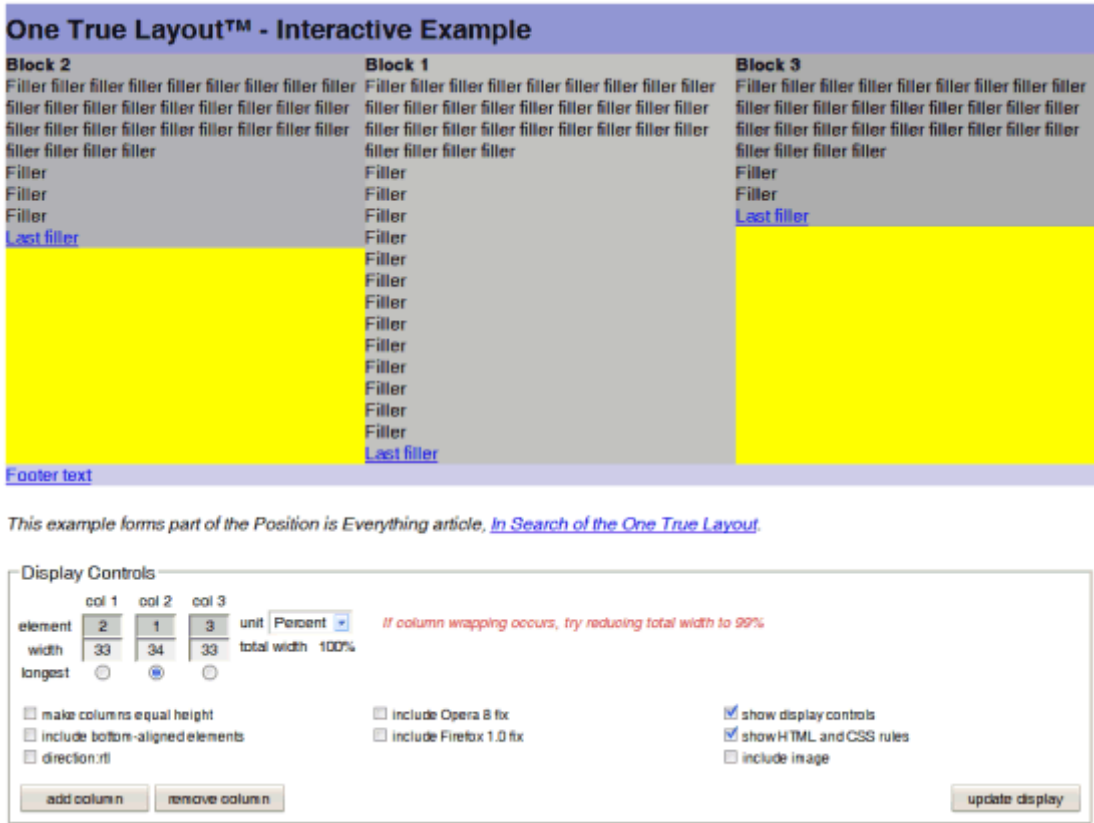


Figura 12.12 Herramienta online para diseñar layouts de varias columnas con CSS

Capítulo 13. Otros

13.1. Propiedades shorthand

Las propiedades de tipo "*shorthand*" son propiedades de CSS que permiten establecer de forma simultánea el valor de varias propiedades diferentes pero relacionadas. El uso de las propiedades "*shorthand*" es muy extendido, ya que permiten crear hojas de estilos más compactas.

A continuación se incluye a modo de referencia todas las propiedades de tipo "*shorthand*" que se han mostrado anteriormente.

Propiedad	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
Propiedad	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

Propiedad	font
Valor inicial	-
Descripción	Establece de forma directa todos los márgenes de un elemento
Propiedad	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
Propiedad	border
Valores	(unidad de medida _borde color _borde estilo_borde) inherit
Se aplica a	Todos los elementos
Valor inicial	-

Propiedad	padding
Descripción	Establece el estilo completo de todos los bordes de los elementos
Propiedad	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
Propiedad	list-style
Valores	(list-style-type list-style-position list-style-image) inherit
Se aplica a	Elementos de una lista
Valor inicial	-
Descripción	Propiedad que permite establecer de forma simultanea todas las opciones de una lista

13.2. Versión para imprimir

La mayoría de sitios web de calidad ofrecen al usuario la posibilidad de imprimir sus contenidos mediante una versión específica para impresora de cada página.

Estas versiones optimizadas para impresora eliminan los contenidos superfluos, modifican o eliminan las imágenes y colores de fondo y sobre todo, optimizan los contenidos de texto para facilitar su lectura.

CSS simplifica al máximo la creación de una versión para imprimir gracias al concepto de los medios CSS. Como se sabe, los estilos CSS que se aplican a los contenidos pueden variar en función del medio a través del que se acceden (pantalla, televisor, móvil, impresora, etc.)

De esta forma, realizar una versión para imprimir de una página HTML es tan sencillo como crear unas cuantas reglas CSS que optimicen sus contenidos para conseguir la mejor impresión.

El sitio web [A List Apart](#) es un excelente ejemplo de cómo los sitios web de calidad crean versiones específicas para impresora de las páginas web originales. Cuando se visualiza un artículo de ese sitio web en una pantalla normal, su aspecto es el siguiente:



Figura 13.1 Aspecto de un artículo de A List Apart como se ve en la pantalla

Además de sus contenidos, las páginas de los artículos muestran el logotipo del sitio, el menú principal de navegación y una barra lateral con varias utilidades como un buscador.

Sin embargo, cuando se imprime la página del mismo artículo, su aspecto es el que muestra la siguiente imagen:



Figura 13.2 Aspecto de un artículo de A List Apart como se ve cuando se imprime

La página impresa elimina todos los contenidos superfluos como los menús de navegación, el buscador y el formulario para añadir comentarios en el artículo. Además, modifica la estructura de la página para que la zona de contenidos ocupe toda la anchura de la página y el espacio se aproveche mejor.

Crear una versión para imprimir similar a la mostrada en el ejemplo anterior es una tarea que no lleva más de unos pocos minutos.

Las reglas CSS de la versión para imprimir se aplican solamente al medio `print`. Por lo tanto, en primer lugar se crea una nueva hoja de estilos y al enlazarla se especifica que sólo debe aplicarse en las impresoras:

```
<link rel="stylesheet" type="text/css" href="/css/imprimir.css" media="print" />
```

Normalmente, las hojas de estilos para la pantalla se aplican a todos los medios (por utilizar el valor `media="all"` al enlazarla o por no indicar ningún valor en el atributo `media`). Por este motivo, cuando se imprime una página se aplican los mismos estilos que se aplican al visualizarla en la pantalla.

Aprovechando este comportamiento, las hojas de estilos para impresoras son muy sencillas, ya que sólo deben modificar algunos estilos aplicados en el resto de hojas de estilos.

Por este motivo, normalmente las hojas de estilos para impresora se construyen siguiendo los pasos que se muestran a continuación:

1) Ocultar los elementos que no se van a imprimir:

```
#cabecera, #menu, #lateral, #comentarios {
    display: none !important;
```

```
}
```

Los bloques (normalmente encerrados en elementos de tipo `<div>`) que no se van a imprimir se ocultan con la propiedad `display` y su valor `none`. La palabra clave `!important` aumenta la prioridad de esta regla CSS y más adelante se explica su significado.

2) Corregir la estructura de la página:

```
body, #contenido, #principal, #pie {  
  
    float: none !important;  
  
    width: auto !important;  
  
    margin: 0 !important;  
  
    padding: 0 !important;  
  
}
```

Normalmente, las páginas web complejas están formadas por varias columnas posicionadas mediante la propiedad `float`. Si al imprimir la página se eliminan las columnas laterales, es conveniente reajustar la anchura y el posicionamiento de la zona de contenidos y de otras zonas que sí se van a imprimir.

3) Modificar los colores y tipos de letra:

```
body { color: #000; font: 100%/150% Georgia, "Times New Roman", Times, serif; }
```

Aunque el uso de impresoras en color es mayoritario, suele ser conveniente imprimir todo el texto de las páginas de color negro, para ahorrar costes y para aumentar el contraste cuando se imprime sobre hojas de color blanco. También suele ser conveniente modificar el tipo de letra y escoger uno que facilite al máximo la lectura del texto.

13.2.1. Imprimiendo los enlaces

Uno de los principales problemas de las páginas HTML impresas es la pérdida de toda la información relativa a los enlaces. En principio, imprimir los enlaces de una página es absurdo porque no se pueden utilizar en el medio impreso.

Sin embargo, lo que puede ser realmente útil es mostrar al lado de un enlace la dirección a la que apunta. De esta forma, al imprimir la página no se pierde la información relativa a los enlaces.

CSS incluye una propiedad llamada `content` que permite crear nuevos contenidos de texto para añadirlos a la página HTML. Si se combina la propiedad `content` y el *pseudo-elemento* `:after`, es posible insertar de forma dinámica la dirección a la que apunta un enlace justo después de su texto:

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

```
}
```

El código CSS anterior añade después de cada enlace de la página un texto formado por la dirección a la que apunta el enlace mostrada entre paréntesis. Si se quiere añadir las direcciones antes de cada enlace, se puede utilizar el *pseudo-elemento* `:before`:

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

13.3. Personalizar el cursor

CSS no permite modificar los elementos propios del navegador o de la interfaz de usuario del sistema operativo. Sin embargo, el puntero del ratón es una excepción muy importante, ya que se puede modificar mediante la propiedad `cursor`.

Propiedad	<code>cursor</code>
Valores	((url ,)* (auto crosshair default pointer move e-resize ne-resize nw-resize n-resize se-resize sw-resize s-resize w-resize text wait help progress)) <code>inherit</code>
Se aplica a	Todos los elementos
Valor inicial	auto
Descripción	Permite personalizar el puntero del ratón

La propiedad `cursor` no sólo permite seleccionar un puntero entre los disponibles en el sistema operativo (flecha, mano, reloj de arena, redimensionar, etc.) sino que incluso permite indicar la URL de una imagen que se quiere mostrar como puntero personalizado.









Se pueden indicar varias URL para que CSS intente cargar varias imágenes: si la primera imagen del puntero no se carga o no la soporta el navegador, se pasa a la siguiente imagen y así sucesivamente hasta que se pueda cargar alguna imagen.



El siguiente ejemplo muestra el caso de un puntero definido con varias URL y un valor estándar:



```
:link, :visited { cursor: url(puntero.svg), url(puntero.cur), pointer }
```

Si el navegador soporta las imágenes en formato SVG, el puntero del ratón cambia su aspecto por la imagen `puntero.svg`. Si el navegador no soporta el formato SVG, intenta cargar la siguiente URL que define un puntero en formato `.cur`. Si no se puede cargar correctamente, se mostraría el valor preestablecido `pointer`.

Los valores preestablecidos para el puntero se muestran a continuación:

Puntero	Navegadores que lo soportan
 <code>cursor: default</code>	Todos
 <code>cursor: crosshair</code>	Todos
 <code>cursor: hand</code>	Solo Internet Explorer
 <code>cursor: pointer</code>	Todos salvo Internet Explorer
 <code>cursor:pointer; cursor: hand</code>	Todos
 <code>cursor: move</code>	Todos
 <code>cursor: text</code>	Todos
 <code>cursor: wait</code>	Todos

Puntero	Navegadores que lo soportan
 <code>cursor: help</code>	Todos
 <code>cursor: n-resize</code>	Todos
 <code>cursor: ne-resize</code>	Todos
 <code>cursor: e-resize</code>	Todos
 <code>cursor: se-resize</code>	Todos
 <code>cursor: s-resize</code>	Todos
 <code>cursor: sw-resize</code>	Todos
 <code>cursor: w-resize</code>	Todos
 <code>cursor: nw-resize</code>	Todos
 <code>cursor: progress</code>	Solo Internet Explorer
 <code>cursor: not-allowed</code>	Solo Internet Explorer

Puntero	Navegadores que lo soportan
 <code>cursor: no-drop</code>	Solo Internet Explorer
 <code>cursor: vertical-text</code>	Solo Internet Explorer
 <code>cursor: all-scroll</code>	Solo Internet Explorer
 <code>cursor: col-resize</code>	Solo Internet Explorer
 <code>cursor: row-resize</code>	Solo Internet Explorer
 <code>cursor: url(...)</code>	Solo Internet Explorer

El puntero personalizado más utilizado es la opción `cursor: pointer` y `cursor: hand` que muestra en el puntero una mano que puede pinchar sobre el elemento. Otro puntero muy utilizado es `cursor: move` que permite indicar en las aplicaciones web dinámicas los elementos de la página que se pueden mover.

13.4. Hacks y filtros

Los diferentes navegadores y las diferentes versiones de cada navegador incluyen defectos y carencias en su implementación del estándar CSS 2.1. Algunos navegadores no soportan ciertas propiedades, otros las soportan a medias y otros ignoran el estándar e incorporan su propio comportamiento.

De esta forma, diseñar una página compleja que presente un aspecto homogéneo en varios navegadores y varias versiones diferentes de cada navegador es una tarea que requiere mucho esfuerzo. Para facilitar la creación de hojas de estilos homogéneas, se han introducido los filtros y los *hacks*.

A pesar de que utilizar filtros y *hacks* es una solución poco ortodoxa, en ocasiones es la única forma de conseguir que una página web muestre un aspecto idéntico en cualquier navegador.

En primer lugar, los filtros permiten definir u ocultar ciertas reglas CSS para algunos navegadores específicos. Los filtros se definen aprovechando los errores de algunos navegadores (sobre todo los antiguos) a la hora de procesar las hojas de estilos.

Un caso especial de filtro son los comentarios condicionales, que es un mecanismo propietario del navegador Internet Explorer. Los comentarios condicionales permiten incluir hojas de estilos o definir reglas CSS específicamente para una versión de Internet Explorer.

El siguiente ejemplo carga la hoja de estilos `basico_ie.css` solamente para los navegadores de tipo Internet Explorer:

```
<!--[if IE]>

  <style type="text/css">

    @import ("basico_ie.css");

  </style>

<![endif]-->
```

Los navegadores que no son Internet Explorer ignoran las reglas CSS anteriores ya que interpretan el código anterior como un comentario de HTML (gracias a los caracteres `<!--` y `-->`) mientras que los navegadores de la familia Internet Explorer lo interpretan como una instrucción propia y válida.

El filtro `[if IE]` indica que esos estilos CSS sólo deben tenerse en cuenta si el navegador es cualquier versión de Internet Explorer. Utilizando comentarios condicionales, también es posible incluir reglas CSS para versiones específicas de Internet Explorer:

```
<!--[if gte IE 6]>

  <style type="text/css">

    @import ("basico_ie6.css");

  </style>

<![endif]-->
```

El anterior ejemplo solamente carga la hoja de estilos `basico_ie6.css` si el navegador es la versión 6 o superior de Internet Explorer, 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 *patch*es y *chapuzas* que permiten que la hoja de estilos completa se muestre tal y como se espera.

Uno de los *hacks* más conocidos y utilizados es el llamado ** html*. Todas las propiedades CSS que se establezcan mediante el selector ** html* son interpretadas exclusivamente por el navegador Internet Explorer 6 y sus versiones anteriores:

```
div {  
    border-bottom: 1px dotted #000;  
}  
  
* html div {  
    border-bottom: 1px solid #000;  
}
```

El ejemplo anterior utiliza el *hack* ** html* para mostrar un borde inferior punteado en los `<div>` en todos los navegadores salvo Internet Explorer 6. Como en este navegador no se muestran correctamente los bordes punteados de 1 píxel de anchura, se decide mostrar un borde formado por una línea continua.

El otro *hack* más conocido y utilizado por su sencillez es el "*underscore hack*". Las propiedades cuyos nombres se indiquen con un guión bajo por delante, sólo son interpretadas por el navegador Internet Explorer 6 y sus versiones anteriores:

```
#menu {  
  
    position: fixed;  
  
    _position: static;  
  
}
```

Los navegadores más modernos soportan el valor `fixed` para la propiedad `position`, pero Internet Explorer 6 no la soporta. Por este motivo, la regla CSS anterior establece el valor de la propiedad `position` y seguidamente define la propiedad `_position`.

Los navegadores que siguen los estándares rechazan la propiedad `_position`, ya que su nombre no se corresponde con ninguna propiedad válida de CSS. Internet Explorer 6 y las versiones anteriores, consideran correcta tanto `position` como `_position`, por lo que el valor utilizado será el que se haya definido en último lugar (`static` en este caso).

13.5. Prioridad de las declaraciones CSS

Además de las hojas de estilos definidas por los diseñadores, los navegadores aplican a cada página otras dos hojas de estilos: la del navegador y la del usuario.

La hoja de estilos del navegador es la primera que se aplica y se utiliza para establecer el estilo inicial por defecto a todos los elementos HTML (tamaños de letra iniciales, decoración del texto, márgenes entre elementos, etc.)

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 13.3 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 13.4 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 sobrescribir 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.

13.6. Validador

La validación del código CSS y de las reglas que lo forman es un concepto similar a la validación de documentos XHTML.

La validación es un mecanismo que permite comprobar que el código CSS creado cumple las reglas de la sintaxis del lenguaje CSS y que por tanto es una hoja de estilos válida para aplicarla a cualquier documento XHTML.

La validación suele ser útil cuando se producen errores en los estilos definidos o comportamientos no deseados al aplicar las reglas CSS. En muchas ocasiones, los errores se producen porque el navegador está ignorando algunas reglas que contienen propiedades mal definidas o errores de sintaxis.

El W3C (*World Wide Web Consortium*) dispone de un validador online que permite validar reglas CSS sueltas, páginas XHTML con CSS incluido y archivos CSS independientes. El validador se puede acceder en <http://jigsaw.w3.org/css-validator/>

13.7. Recomendaciones generales sobre CSS

13.7.1. Atributos ID y class

El atributo `id` se emplea para identificar a cada elemento HTML, por lo que los identificadores deben ser únicos en una misma página. En otras palabras, dos elementos HTML diferentes de una misma página no pueden tener un mismo valor en el atributo `id`.

Por otra parte, el atributo `class` se emplea para indicar la clase o clases a las que pertenece el elemento. Una misma clase se puede aplicar a varios elementos diferentes y un único elemento puede tener asignadas varias clases (se indican separadas por espacios en blanco).

Aunque los dos atributos tienen muchos otros propósitos (sobre todo el atributo `id`), CSS los emplea principalmente con los selectores para indicar los elementos sobre los que se aplican los diferentes estilos.

En el siguiente ejemplo, las dos listas están formadas por un mismo elemento HTML ``, pero sus atributos `id` las distinguen de forma adecuada:

```
<ul id="menu">
```

```
...
```

```
</ul>
```

```
<ul id="enlaces">
```

```
...
```

```
</ul>
```

Una de las principales recomendaciones del diseño de páginas XHTML y hojas de estilos CSS está relacionada con los valores asignados a los atributos `id` y `class`. Siempre que sea posible, estos atributos se deben utilizar para mejorar la semántica del documento, es decir, para añadir significado a cada elemento de la página.

Por este motivo, se recomienda que los valores asignados a `id` y `class` indiquen la función del elemento y no estén relacionados con su aspecto o su posición:

Valores no recomendados	Valores recomendados
negrita	importante
arial15	titular
verdanaPequena	normal
menuIzquierdo	menuSecundario
letraRoja	error

Elegir el valor adecuado para los atributos `id` o `class` es sencillo: si el aspecto de un elemento cambia, el valor de `id` o `class` debe seguir siendo adecuado. Por tanto, evita utilizar valores relacionados con su posición (`izquierdo`, `derecho`, `primero`, `segundo`, `superior`, etc.), color (`textoRojo`, `subrayadoGrisClaro`, etc.) o tipo de letra (`verdana10`, `arial15px`, etc.)

Técnicamente, los valores de los atributos `id` y `class` deben cumplir las siguientes restricciones:

- Sólo pueden empezar por un guión medio (-), un guión bajo (_) o una letra.
- El resto de caracteres, pueden ser números, guiones medios (-), guiones bajos (_) y letras.
- Los navegadores distinguen entre mayúsculas y minúsculas.
- Aunque es posible utilizar letras como ñ y acentos, no se recomienda hacerlo porque no es seguro que funcione correctamente en todas las versiones de todos los navegadores.

13.7.2. CLASSitis y DIVitis

Un error común al comenzar a desarrollar páginas con estilos CSS es la utilización excesiva de etiquetas `<div>` y atributos `class`.

Ejemplo de *divitis* y *classitis*:

```
<div id="menu">
<ul class="menu">
```

```

    <li class="elemento_menu"><span class="texto_elemento_menu">...</span><
  /li>

    <li class="elemento_menu"><span class="texto_elemento_menu">...</span><
  /li>

    <li class="elemento_menu"><span class="texto_elemento_menu">...</span><
  /li>

    <li class="elemento_menu"><span class="texto_elemento_menu">...</span><
  /li>

</ul>

</div>

```

Los selectores de CSS permiten prescindir de la mayoría de etiquetas `<div>` y atributos `id` y `class`. Diseñar páginas con exceso de etiquetas `<div>` no mejora la semántica del documento y sólo consigue complicar el código HTML.

13.7.3. Estructuración del código CSS

La posibilidad de incluir todo el código CSS en archivos externos exclusivamente dedicados a contener las reglas CSS, permite ordenar de forma lógica las reglas, mejorando su legibilidad y facilitando su actualización.

Las reglas CSS de las hojas de estilos complejas se suelen agrupar según su funcionalidad y se suelen incluir en el siguiente orden:

- Estilos básicos (`<body>`, tipo de letra por defecto, márgenes de ``, `` y ``, etc.)
- Estilos de la estructura o layout (anchura, altura y posición de la cabecera, pie de página, zonas de contenidos, menús de navegación, etc.)
- Enlaces (estilos normales, estilos `:hover`, etc.)
- Estilos de cada una de las zonas (elementos de la cabecera, titulares y texto de la zona de contenidos, enlaces, listas e imágenes de las zonas laterales, etc.)

Otra característica común de los mejores sitios web es el uso de comentarios CSS para mejorar la estructura de las hojas de estilos muy largas.

Ejemplo de código CSS estructurado de <http://veerle.duoh.com/>

```

/* Veerle's blog Main stylesheet
-----*/

/* Wide browser windows
-----*/

#wrap {

```

```

    width:995px;
}

/* Global
-----*/

html, body, form, h1, h2, h3, h4, h5, h6, p, pre, blockquote, ul, ol, dl
{
    margin:0;
    padding:0;
}
ul,li {
    list-style-type:none;
}
...

/* Wide layout
-----*/

.wide #wrap-main {
    ...
}
...

/* Links
-----*/

a:link,
a:visited {
    text-decoration:none;
    color:#e45a49;
}

```



```
}
```

```
/* Header
```

```
-----*/
```

```
#header {
```

```
...
```

```
}
```

```
...
```

```
/* Main navigation
```

```
-----*/
```

```
ul#nav {
```

```
...
```

```
}
```

```
...
```

Ejemplo de código CSS estructurado de <http://www.uxmag.com/>

```
/* -----
```

```
UX Magazine
```

```
Design | Technology | Strategy | Common Sense
```

```
-----
```

```
Description:   Base setup styles
```

```
Filename:      uxm.base.css
```

```
Version:       1.9
```

```
Date:          Feb 9, 2006
```

```
----- */
```

```
/* -----
```

```
Base Body Styles
```

```
----- */
```

```

/* -----
Print Styles
----- */

/* -----
Top Bar Styles
----- */

/* Slogan
----- */

/* Search Form
----- */

/* Channels
----- */

```

13.7.4. División de los estilos en varios archivos CSS

Normalmente, los estilos de una página compleja se dividen en varios archivos CSS diferentes para hacerlos más manejables. En primer lugar, se suele utilizar un archivo común que contiene todos los estilos básicos de las páginas HTML del sitio web.

Además, si existe alguna sección especial del sitio web que requiera nuevos estilos, se crea un archivo CSS con todos esos estilos. También es habitual preparar una hoja de estilos específica para impresora y otra preparada para los dispositivos móviles.

Una vez creados los archivos CSS, existen dos estrategias para enlazar varios archivos CSS en las páginas HTML:

Si se puede modificar fácilmente la cabecera del documento (por ejemplo porque las páginas se generan dinámicamente) lo habitual es incluir tantos elementos `<link>` como archivos CSS se enlazan:

```

<head>
...
<link rel="stylesheet" type="text/css" href="/css/basico.css" media="screen" />

```

```
<link rel="stylesheet" type="text/css" href="/css/seccion.css" media="screen" />
```

```
<link rel="stylesheet" type="text/css" href="/css/impresora.css" media="print" />
```

```
<link rel="stylesheet" type="text/css" href="/css/movil.css" media="handheld" />
```

```
...
```

```
</head>
```

Si no se puede modificar de forma sencilla la cabecera de los documentos para añadir, eliminar y modificar los archivos CSS que se enlazan, lo habitual es enlazar un único archivo CSS que se encarga de importar todos los demás:

```
<head>
```

```
...
```

```
<link rel="stylesheet" type="text/css" href="/css/estilos.css" media="all" />
```

```
...
```

```
</head>
```

El contenido del archivo `estilos.css` debería ser el siguiente para ser equivalente al ejemplo anterior:

```
@import url("basico.css") screen;  
@import url("seccion.css") screen;  
@import url("impresora.css") print;  
@import url("movil.css") handheld;
```