Introducción a CSS

Javier Eguíluz Pérez

Sobre este libro...

- **Esta versión impresa se creó el 20 de febrero de 2008 y todavía está incompleta**. La versión más actualizada de los contenidos de este libro se puede encontrar en `http://www.librosweb.es/css`
- Si quieres aportar sugerencias, comentarios, críticas o informar sobre errores, puedes contactarnos en contacto@librosweb.es

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.

La separación de los contenidos y su presentación 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.

Mientras que el lenguaje HTML/XHTML se utiliza para *marcar* los contenidos, es decir, para designar lo que es un párrafo, lo que es un titular o lo que es una lista de elementos, el lenguaje CSS se utiliza para definir el aspecto de todos los contenidos, es decir, el color, tamaño y tipo de letra de los párrafos de texto, la separación entre titulares y párrafos, la tabulación con la que se muestran los elementos de una lista, 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) 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 siguiente recomendación, 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.

De hecho, uno de los navegadores más utilizados, Internet Explorer 6, tiene un soporte limitado de CSS 2 y decenas de errores conocidos en la parte de CSS 2 que implementa, lo que dificulta la creación de páginas con un aspecto homogéneo entre diferentes navegadores.

Los navegadores con mejor soporte de CSS 2 (incluso con soporte de algunas características de CSS 3) son Firefox (con su motor Gecko), Opera (con su motor Presto) y Safari/ Konqueror (con su motor KHTML).

Desde la publicación de la versión CSS 2, se han añadido pequeñas correcciones de errores y algunas variaciones en el estándar, hasta llegar a la actual versión CSS 2.1.

1.3. Especificación oficial

Actualmente, la última especificación oficial de CSS es la versión 2.1, actualizada por última vez el 19 de julio de 2007:

`http://www.w3.org/TR/CSS21/`

La versión CSS 3 supone un gran cambio de organización respecto de la versión 2.1 e incorporará muchas novedades. Actualmente se encuentra en proceso de elaboración:

`http://www.w3.org/Style/CSS/current-work#CSS3`

1.4. Funcionamiento básico de CSS

Antes de la adopción de CSS, los diseñadores de páginas HTML debían definir el estilo y el aspecto de los elementos HTML en el propio documento.

Ejemplo de un documento HTML sencillo sin 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="grey" face="Verdana" size="2">Un párrafo de texto no muy
largo.</font></p>

</body>
</html>
```

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

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

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

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

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
<title>Ejemplo de estilos con CSS</title>
<style type="text/css">
h1 { color: red;  font-family: Arial;   font-size: large;  }
p  { color: grey; 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 y su presentación, ya que por un lado se definen los contenidos HTML y por otro se definen los estilos de la página en una zona del documento específicamente reservada para CSS.

En el ejemplo anterior, no importa el número de elementos `<p>` que existan en la página, ya que todos tendrán el mismo aspecto definido mediante CSS. No obstante, esta forma de trabajar con CSS no es ideal, ya que si el sitio web dispone de 10.000 páginas, habría que definir un mismo estilo CSS 10.000 veces.

1.5. Cómo incluir CSS en un documento XHTML

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

1.5.1. 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 un determinado documento HTML que completen los estilos que se incluyen por defecto en todos los documentos del sitio web.

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

1.5.2. Definir CSS en un archivo externo

Todos los estilos CSS se pueden incluir en un archivo de tipo CSS que los documentos HTML enlazan mediante la etiqueta `<link>`. Se pueden crear todos los archivos CSS que sean necesarios y cada documento HTML puede enlazar tantos archivos CSS como necesite.

Ejemplo:

Archivo `estilos.css`

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

Documento HTML

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
```

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

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

Un archivo de tipo CSS no es más que un archivo de texto normal y corriente cuya extensión es `.css`. Aunque generalmente se emplea la etiqueta `<link>` para enlazar archivos CSS externos, también se puede emplear 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 el propio documento</title>
<style type="text/css" media="screen">
  @import '/css/estilos.css';
</style>
</head>

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

De todas las formas de aplicar CSS a los documentos HTML, esta es la más utilizada (sobre todo mediante la etiqueta `<link>`). 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 todos los documentos que forman un sitio web.

La principal razón por la que es el método más utilizado es que es el más sencillo de mantener, ya que un solo cambio en un solo archivo CSS permite variar de forma instantánea los estilos de todos los documentos HTML que enlazan ese archivo.

1.5.3. Incluir CSS en los elementos HTML

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

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.

1.6. Glosario básico

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

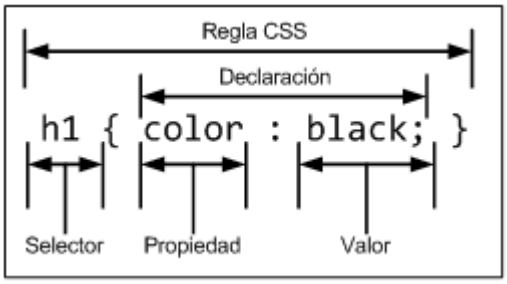

Figura 1.1. Componentes de un estilo CSS básico

Los diferentes términos se definen a continuación:

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

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

1.7. Medios CSS

Una de las características más importantes de las hojas de estilos CSS es que permiten indicar diferentes reglas para diferentes medios o dispositivos: pantalla, impresora, móviles, proyector, 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:

Tabla 1-1. Medios definidos por CSS

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

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

CSS permite definir reglas específicas para cada medio y reglas compartidas por varios o todos los medios. Existen varios métodos para indicar el medio en el que se deben aplicar las reglas.

Medios definidos con las reglas de tipo `@media`

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

El ejemplo anterior asigna un tamaño de letra de 13 píxel para la página que se muestra en pantalla y un tamaño de letra de 10 puntos para la página que se imprime. Por último, se indica un interlineado igual al 120% del tamaño de letra tanto para la página en pantalla como para la página impresa.

Medios definidos con las reglas de tipo `@import`

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

Cuando la página se visualiza por pantalla, se cargan los estilos definidos en el primer archivo CSS. Cuando la página se imprime, se tienen en cuenta los estilos que define el segundo archivo CSS.

Medios definidos con la etiqueta `<link>`

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

Este ejemplo es muy similar al anterior: 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.

Medios definidos mezclando varios métodos

Este último ejemplo mezcla los métodos descritos anteriormente:

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

La primera regla de tipo `@import`, indica el archivo CSS que se debe cargar para obtener los estilos con los que se muestra la página en pantalla. La segunda regla de tipo `@media` especifica directamente los estilos que se deben utilizar al imprimir la página web.

1.8. Comentarios

Como es habitual en cualquier lenguaje de programación, de marcado o de hojas de estilos, CSS también dispone de la posibilidad de incluir comentarios. 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 utilizar los comentarios para estructurar correctamente los archivos CSS complejos.

La sintaxis de los comentarios CSS se muestra a continuación:

- Los comentarios comienzan con los caracteres `/*`.
- Los comentarios finalizan con los caracteres `*/`.
- No se pueden anidar comentarios (es decir, no se puede incluir un comentario dentro de otro comentario).
- Los comentarios pueden ocupar tantas líneas como sea necesario.

Aunque los navegadores ignoren los comentarios, su contenido se envía junto con el resto de estilos, por lo que no se debería 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:

Comentarios en HTML:

```
<!-- Este es un comentario en HTML -->

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

Comentarios en CSS:

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

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

1.9. 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 se utiliza para decir *"qué hay que hacer"* y el selector es lo que dice *"a quién hay que hacérselo"*. La declaración de una regla sencilla puede indicar por ejemplo que el color de la letra debe ser rojo, y el selector de esa regla sencilla puede indicar por ejemplo que los elementos a los que se aplica ese estilo son todos los párrafos de la página.

A un mismo elemento HTML se le pueden definir *infinitas* reglas CSS y cada regla puede tener un número *infinito* de selectores sobre los que se aplica.

Aunque CSS 2.1 define una docena de tipos de selectores, la mayoría de las páginas web se pueden definir utilizando solamente los cinco selectores básicos. Además, Internet Explorer 6, uno de los navegadores que más utilizan los usuarios, no soporta los selectores avanzados, por lo que es casi *obligatorio* utilizar solamente los 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 la etiqueta HTML correspondiente a los elementos que se quieren seleccionar. El siguiente ejemplo aplica diferentes estilos a los titulares y a las listas de una página HTML:

```
h1 {
  margin: 0 0 0.5em;
  font-size: 1.8em;
  line-height: 1.2;
}

ul {
  padding: 0 0 0 2em;
  list-style: square;
}

ol { padding: 0 0 0 2.5em }
li { margin: 0 0 0.5em }
```

Los selectores se pueden encadenar para aplicar una misma regla a varios elementos diferentes. Si se considera el siguiente ejemplo, en el que 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;
}
```

CSS permite aplicar directamente los mismos estilos a varios selectores de forma simultánea. Para ello, se indican todos los selectores diferentes separados por una coma (`,`). La siguiente regla CSS es equivalente al ejemplo anterior:

```
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 y a continuación, establece el tamaño 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

Permite seleccionar los elementos que se encuentran dentro de otros elementos. El siguiente ejemplo se emplea para visualizar en negrita el texto de cualquier elemento `<span>` contenido dentro de un elemento `<p>`:

```
p span { font-weight: bold; }
```

Si el código HTML es el siguiente:

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

Aplicando la regla CSS anterior, tanto `texto1` como `texto2` se verán en negrita. La razón es que con el selector descendente, un elemento no tiene que ser *"hijo directo"* de otro, sino que la única condición es que esté dentro de ese elemento, sin importar lo profundo que se encuentre.

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

Los selectores descendentes permiten aplicar de forma sencilla diferentes estilos a los elementos del mismo tipo. El siguiente ejemplo amplía el anterior y muestra de color rojo todo el texto de los `<span>` contenidos dentro de un `<h1>`:

```
p span { font-weight: bold; }
h1 span { color: red; }
```

Con las reglas CSS anteriores, los elementos `<span>` que se encuentran dentro de un elemento `<p>` se muestran en negrita. Los elementos `<span>` dentro de un elemento `<h1>` se muestran de color rojo y el resto de elementos `<span>` de la página, se muestran con el aspecto por defecto aplicado por el navegador.

Se pueden utilizar varios selectores descendentes seguidos:

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

La anterior regla CSS solamente se aplica al texto contenido en los elementos `<em>` que formen parte de un `<span>` contenido en cualquier enlace de un párrafo. Aunque puede parecer difícil de entender, el razonamiento siempre es el mismo: el elemento al que se aplica el estilo siempre es el último selector indicado (`em` en el ejemplo anterior) y todos los selectores anteriores indican dónde debe encontrarse ese elemento para que se le apliquen los estilos (`p a span` en el ejemplo anterior).

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

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

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

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

```
p a { color: red; }

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

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

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

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

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

2.1.4. Selector de clase

Los selectores de clase son los selectores más utilizados junto con los selectores de ID (que se verán a continuación). Utilizando este selector, se pueden seleccionar todos los elementos de la página cuyo atributo `class` coincida con el selector.

Este tipo de selector es imprescindible para poder seleccionar elementos específicos de la página. ¿Cómo se pueden seleccionar tres párrafos concretos de una página HTML en la que cada párrafo se encuentra en una zona diferente?

- Con el selector universal se seleccionarían todos los elementos de la página, por lo que no se puede utilizar.
- El selector de tipo seleccionaría todos los párrafos de la página, por lo que tampoco es útil.
- El selector descendente tampoco es apropiado ya que cada párrafo puede encontrarse dentro de diferentes elementos.

En este tipo de situaciones, se asigna un atributo `class` específico a los elementos que se quieren seleccionar y en la hoja de estilos CSS se utiliza el selector de clase. Este selector está formado por un signo de punto (`.`) y el nombre del atributo `class` que se quiere seleccionar. Por tanto, en el siguiente ejemplo, solamente el segundo párrafo se mostrará de color rojo:

```
.especial { color: red; }

<p>Primer párrafo</p>
<p class="especial">Segundo párrafo</p>
<p>Tercer párrafo</p>
```

El selector `.especial` se traduce como *"cualquier elemento cuyo atributo `class` sea igual a `especial`"*, por lo que solamente el segundo párrafo cumple la condición.

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 `<span>` tiene un atributo `class="error"`, por lo que se le aplicarán las reglas CSS indicadas por el selector `.error`. Por su parte, el elemento `<div>` tiene un atributo `class="aviso"`, por lo que su estilo será el que definan las reglas CSS del selector `.aviso`.

Combinando este selector con los anteriores, se puede restringir el alcance de los selectores. El siguiente ejemplo aplica la regla CSS solamente a los elementos de tipo `<p>` que tengan un atributo `class` igual al indicado:

```
p.aviso {
  padding: 0.5em;
  border: 1px solid #98be10;
  background: #f6feda;
}
```

Para seleccionar solamente los elementos de un tipo y un atributo `class` determinado, se indica la etiqueta del elemento y sin dejar ningún espacio, se indica el selector de clase. De esta forma, la regla anterior ya no se aplicaría al elemento `<span class="aviso">...</span>`.

No debe confundirse este selector con los 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 { ... }
```

2.1.5. Selectores de ID

CSS también permite seleccionar elementos HTML en función del valor de su atributo `id`. La explicación es la misma que para el selector de clase. La sintaxis utilizada también es la misma, salvo que en este caso se utiliza el símbolo de la almohadilla (`#`) en vez del símbolo del punto (`.`).

```
#especial { color: red; }

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

En el ejemplo anterior, solamente el segundo párrafo (cuyo atributo `id` es igual a `especial`) será seleccionado por el selector `#especial`.

La principal diferencia entre este 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`.

Por tanto, cuando se quiere aplicar un estilo a un solo elemento específico, se utiliza el selector de `id`. Si se quiere aplicar un estilo a varios elementos diferentes, se utiliza el selector de clase.

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 {
  padding: 0.5em;
  border: 1px solid #98be10;
  background: #f6feda;
}
```

El ejemplo anterior puede parecer absurdo y redundante. Si el sitio web solamente contiene una página, el selector anterior es redundante, ya que `#aviso` sólo puede hacer referencia a un único elemento de la página (no sería necesario especificar que el elemento es de tipo `<p>`). Sin embargo, si la anterior regla forma parte de un archivo CSS utilizado en multitud de páginas diferentes, puede ser necesario restringir el alcance del selector `#aviso` sólo para los elementos de tipo `<p>`.

Tampoco debe confundirse este selector con los 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 `<span>` 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 `<li>` con un atributo `class` igual a `destacado`, que forma parte de una lista `<ul>` con un atributo `id` igual a `menuPrincipal`.

Ejercicio 1 Ver enunciado en la página 180

2.2. Selectores avanzados

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

Desafortunadamente, el navegador Internet Explorer 6 y sus versiones anteriores no soportan este tipo de selectores avanzados, por lo que su uso no está muy extendido. En la siguiente página se muestra el soporte de todos los selectores de CSS 1, CSS 2 y CSS 3 en los diferentes navegadores: `http://dev.l-c-n.com/CSS3-selectors/browser-support.php`

Además, existe un test que permite probar los selectores que soporta el navegador con el que se realiza la prueba: `http://www.css3.info/selectors-test/`

2.2.1. Selector de hijos

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

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

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

En el ejemplo anterior, el selector `p > span` se traduce como *"cualquier elemento span que se encuentre directamente dentro de un elemento p"*. El `texto1` pertenece a un elemento `span` directamente hijo de un elemento `p`, por lo que se cumple la condición del selector.

En cambio, el `texto2` se encuentra dentro de un elemento `span` que a su vez está dentro de un elemento `a` que es el que está directamente dentro del elemento `p`. Como entre el elemento `p` y el elemento `span` existen otros elementos, no se cumplen las condiciones del selector y no se le aplicarían los estilos.

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 encuentren dentro de elementos `p`. En este caso, los estilos de este selector se aplicarían a los dos enlaces. Por otra parte, el selector de hijos obliga a que el elemento `a` se encuentre directamente en un elemento `p`, sin ningún otro elemento intermedio. Por tanto, en este caso, los estilos del selector `p > a` no se aplicarían al segundo enlace.

2.2.2. Selector adyacente

El selector adyacente es uno de los más avanzados y su explicación no es sencilla. El selector utiliza el signo + y su sintaxis es:

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

El selector adyacente selecciona todos los elementos de tipo `elemento2` cuyo elemento padre sea el mismo que el de los elementos `elemento1`, con la condición adicional de que `elemento2` debe estar inmediatamente después que el `elemento1`.

En el siguiente ejemplo:

```
h1 + h2 { color: red }

<body>
<h1>Titulo1</h1>

<h2>Subtítulo</h2>
...

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

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

- El elemento padre del primer `h2` es `body`, y también tiene el mismo elemento padre el elemento `h1`. Como la primera condición se cumple (que los dos elementos tengan el mismo padre) se comprueba la segunda condición: que `h2` esté seguido de `h1`. En este caso, la segunda condición también se cumple, por lo que el primer elemento `h2` cumple con el selector `h1 + h2`.
- En el segundo elemento `h2`, la primera condición también se cumple: que los dos elementos tengan el mismo elemento padre. Sin embargo, el segundo elemento `h2` no va inmediatamente después de un elemento `h1`, por lo que la segunda condición del selector no se cumple y no se le aplicarán los estilos.

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

2.2.3. Selector de atributos

Por último, dentro de los selectores avanzados, CSS define los selectores de atributos. Este tipo de selectores permiten seleccionar elementos HTML en función de sus atributos y/o valores de esos atributos. Los 4 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`.

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 diferentes estilos de 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:

```
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

La herencia de los estilos definidos mediante CSS es uno de los conceptos más característicos de este lenguaje de hojas de estilos. Muchas de las propiedades que se aplican a los elementos, son heredadas por los elementos que se encuentran dentro de esos elementos.

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

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
<title>Ejemplo de herencia de estilos</title>
<style type="text/css">
body { font-family: Arial; color: black; }
h1 { font-family: Verdana; }
p { color: red; }
</style>
</head>

<body>

<h1>Titular de la página</h1>

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

</body>
</html>
```

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

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

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 propiedades múltiples es que se pueden dar colisiones como la del siguiente ejemplo:

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

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

¿De qué color se mostrará el párrafo anterior? CSS tiene un mecanismo de resolución de colisiones muy complejo y que tiene en cuenta el tipo de hoja de estilo que se trate (de

navegador, de usuario o de diseñador), la importancia de cada regla y lo específico que sea el selector.

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

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

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

1. Cuanto más específico sea un selector, más importancia tiene.
2. A igual *especificidad*, se considera el último selector indicado

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

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

3.1. Unidades de medida

Las unidades de medida en CSS se emplean para definir la altura, anchura y márgenes de los elementos y para establecer el tamaño de letra del texto.

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

Las medidas, absolutas y relativas, 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).

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

3.1.1. Unidades relativas

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

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

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

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

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

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

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

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

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

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

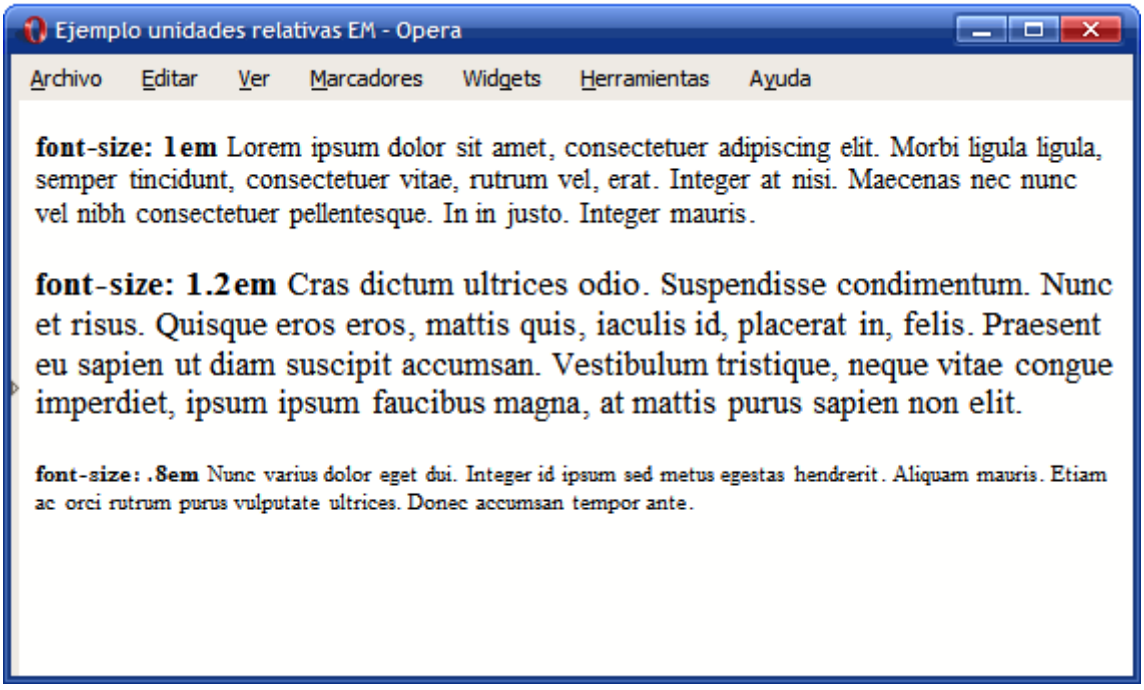

Figura 3.1. Ejemplo de tamaño de letra definido con la unidad relativa em

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

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

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

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

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

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

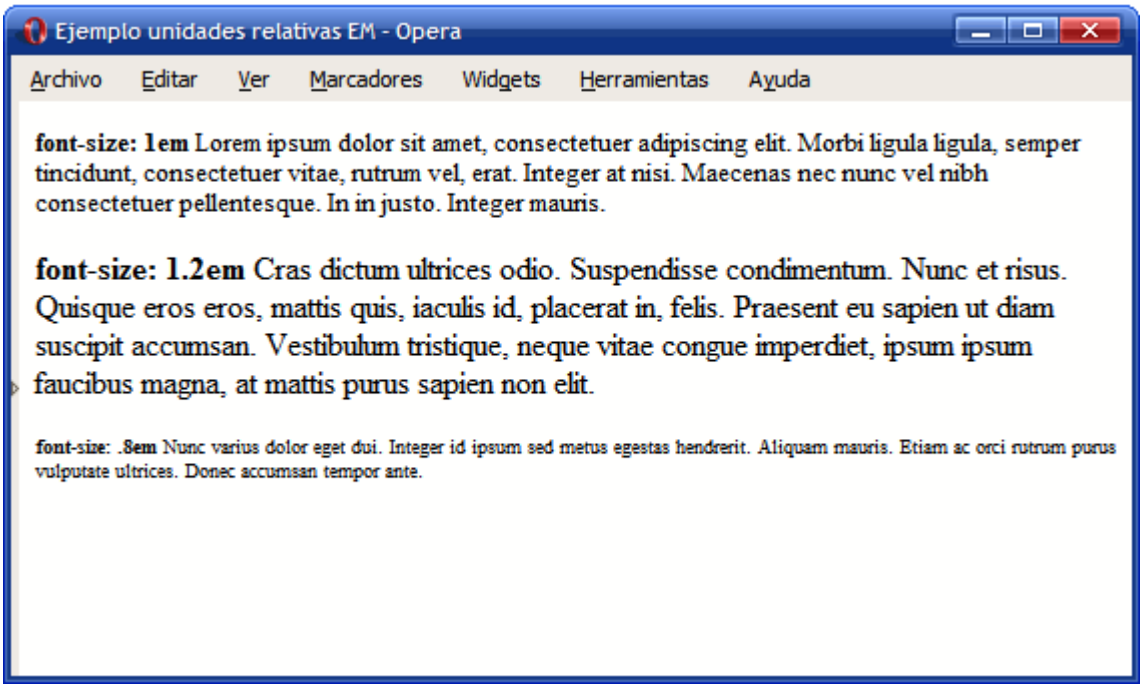

Figura 3.2. Ejemplo de tamaño de letra definido con la unidad relativa em

Al haber reducido el tamaño de letra que era la referencia del tamaño de letra de los tres párrafos, su texto se ve con una letra más pequeña, aunque manteniendo las proporciones: el primer párrafo se ve con un tamaño de letra normal, el segundo se ve con un tamaño grande y el tercero se visualiza con un tamaño de letra más pequeño de lo normal.

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

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

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

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

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

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

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

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

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

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

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

3.1.2. Unidades absolutas

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

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

A continuación se muestran ejemplos de utilización de unidades absolutas:

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

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

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

3.1.3. Porcentajes

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

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

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

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

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

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

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

En el ejemplo anterior, la referencia del valor `80%` es la anchura de su elemento padre. Por tanto, el elemento `<div>` cuyo atributo `class` vale `principal` tiene una anchura de `80% 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.

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

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

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

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

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

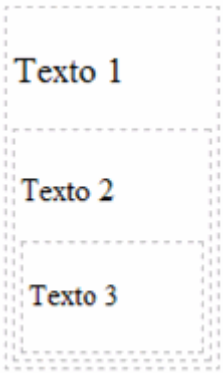

Figura 3.3. El texto cada vez se ve más pequeño

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

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

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

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

3.2. Colores

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

3.2.1. Palabras clave

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

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

Figura 3.4. Colores definidos mediante las palabras clave de CSS

La imagen anterior ha sido extraída de la especificación oficial de CSS y se puede acceder en `http://www.w3.org/TR/CSS21/syndata.html#value-def-color`.

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

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

3.2.2. RGB decimal

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

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

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

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

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

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

3.2.3. RGB porcentual

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

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

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

3.2.4. RGB hexadecimal

Aunque es el método más complicado de indicar los colores, se trata del método más utilizado con mucha diferencia. De hecho, prácticamente todos los sitios web reales utilizan exclusivamente este método. Para definir un color con RGB hexadecimal se siguen los siguientes pasos:

1. Se toman las componentes RGB del color original, por ejemplo R = `71`, G = `98`, B = `176`
2. El valor numérico de cada componente se transforma del sistema decimal al sistema hexadecimal. Esta operación es exclusivamente matemática. En el sistema decimal, se utilizan 10 símbolos para representar los números: del `0` al `9`. En el sistema hexadecimal se utilizan 16 símbolos (de ahí su nombre): del `0` al `9` y de la `A` a la `F`. Así, en el sistema hexadecimal, después del `9` no va el `10`, sino la `A`. La letra `B` sería `11`, la `C` sería `12`, etc.
3. Si se realiza la conversión hexadecimal de las componentes numéricas anteriores, se obtienen unos nuevos valores: R = `47`, G = `62`, B = `B0`.
4. Una vez obtenidas sus componentes hexadecimales, el color se indica concatenando el valor de las componentes y añadiendo el prefijo `#`. Así, el color anterior en la notación RGB hexadecimal de CSS sería `#4762B0`.

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

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

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

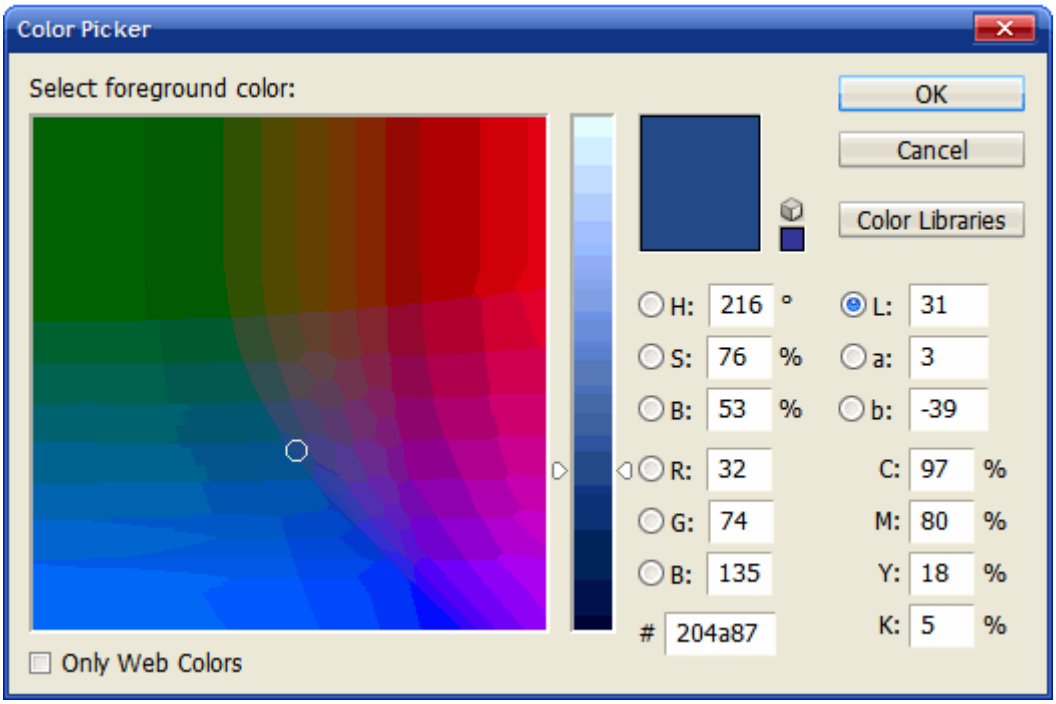

Figura 3.5. Herramienta de color de Photoshop para definir los colores según los modelos RGB, CMYK, Lab, HSB y RGB hexadecimal

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

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

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

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

3.2.5. Colores del sistema

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

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

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

3.2.6. Colores web safe

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

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

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

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

Capítulo 4. Box model

El *"box model"* es seguramente la característica más importante del lenguaje de hojas de estilos CSS, ya que condiciona el diseño de todas las páginas web. El *"box model"* (en ocasiones traducido como *"modelo de cajas"*) es el comportamiento de CSS que provoca que todos los elementos incluidos en una página HTML se representan mediante cajas rectangulares. Mediante CSS se controlan las propiedades de las cajas y también su representación visual.

El diseño de cualquier página web está compuesto por cajas rectangulares. CSS permite definir la altura y anchura de cada caja, el margen que se deja entre cada caja y el espacio de relleno interior que muestra cada caja.

Además, CSS permite controlar la forma en la que se visualizan las cajas: se pueden ocultar, desplazar respecto de su posición original, fijarlas en una posición concreta dentro del documento, etc.

Como la mayoría de cajas de las páginas web no muestran ningún color de fondo ni ningún borde, no son visibles a primera vista. La siguiente imagen muestra las cajas que forman la página web de `http://www.456bereastreet.com/` después de forzar a que todas las cajas muestren un borde punteado:

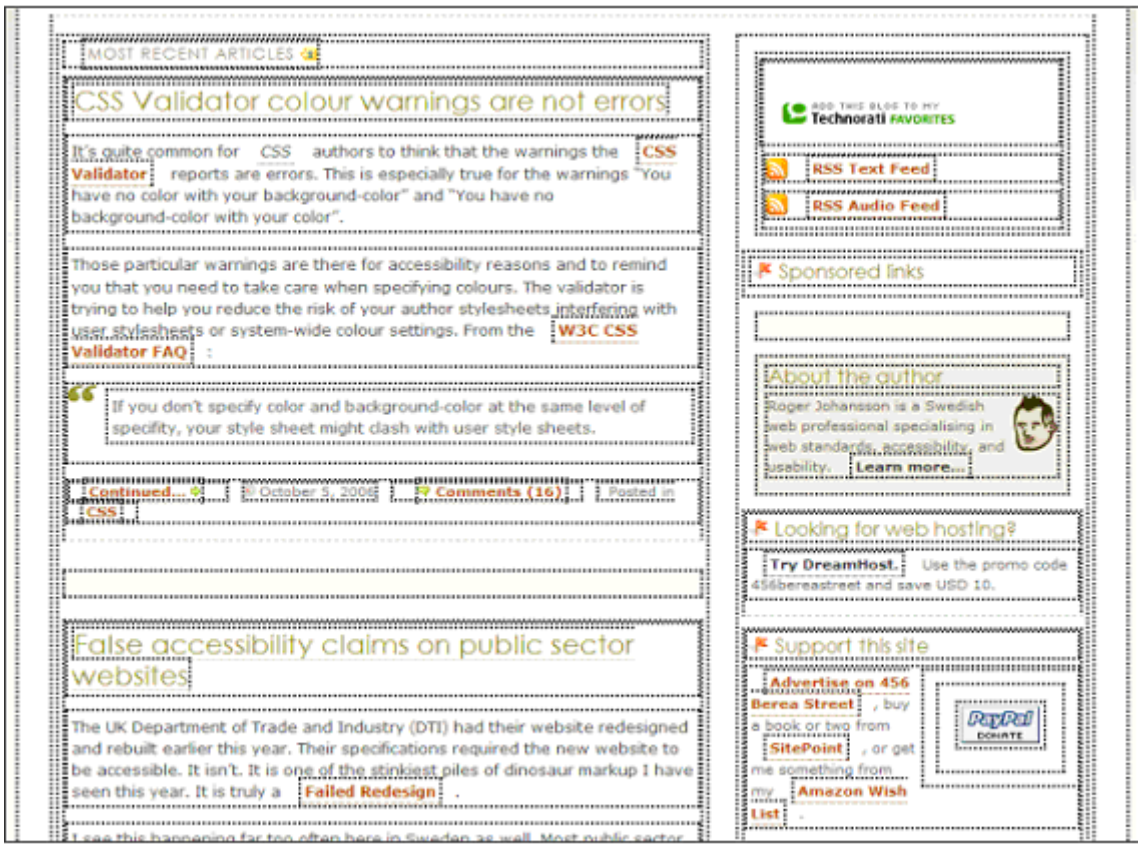

Figura 4.1. Cajas que forman el box model de la página 456bereastreet.com

Las cajas de la página se crean automáticamente. Cada vez que se inserta una etiqueta o elemento en la página, se crea una nueva caja rectangular que encierra los contenidos del elemento. El siguiente esquema muestra la creación automática de cajas por parte de HTML para cada elemento definido en el código HTML de la página:

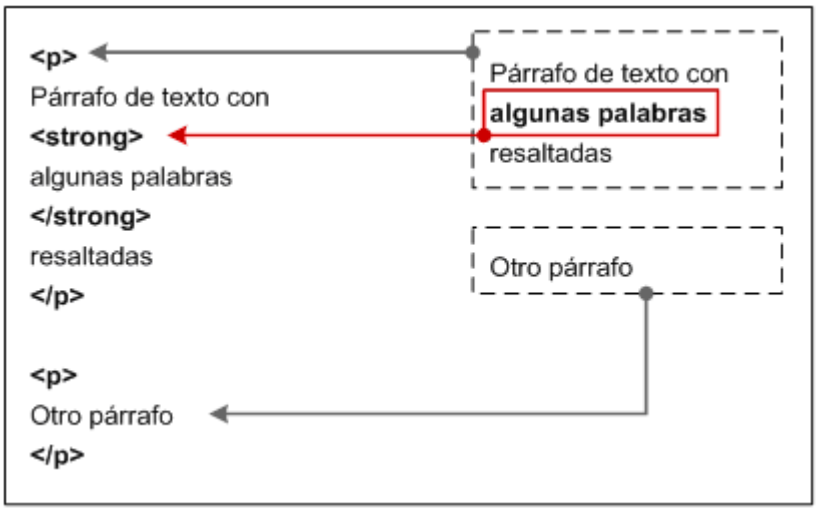

Figura 4.2. Las cajas se crean automáticamente al definir cada elemento HTML

Cada una de las cajas está formada por varias partes. La siguiente imagen muestra un esquema tridimensional de las partes que componen una caja en el *"box model"* de CSS:

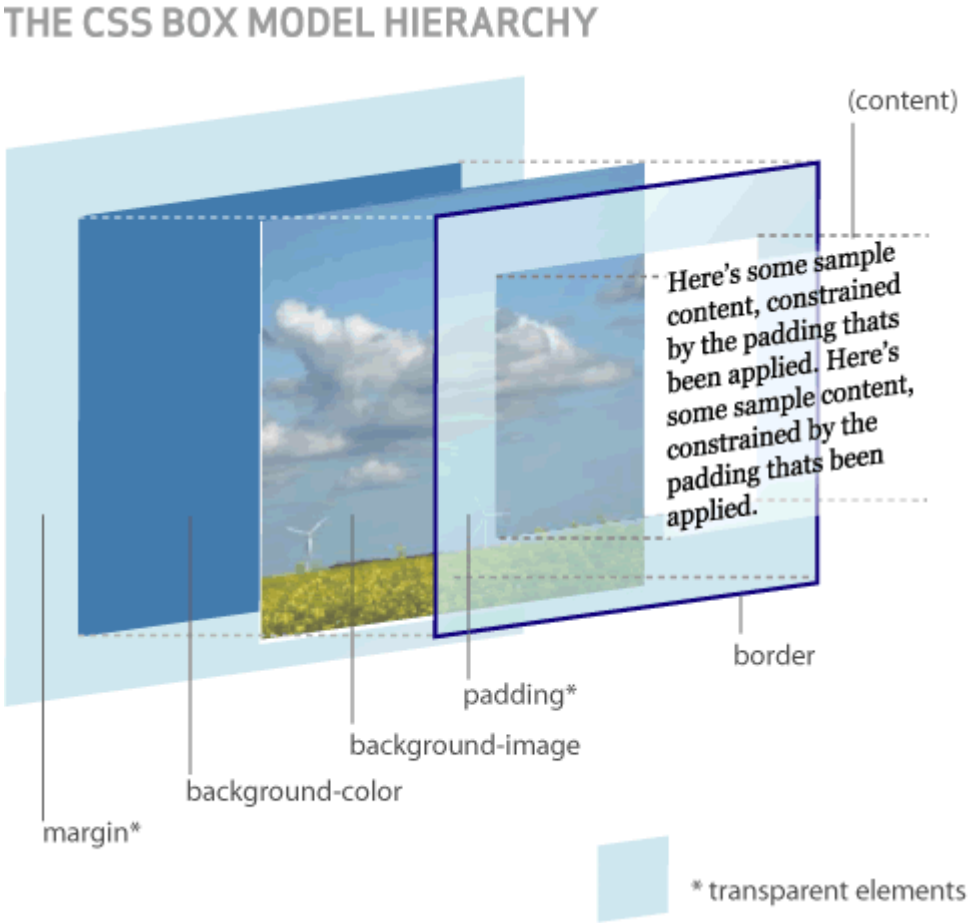

Figura 4.3. Representación tridimensional del box model de CSS

(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*): está formado por el espacio libre entre el contenido y el borde que lo encierra.
- Borde (*border*): línea que encierra completamente el contenido y su relleno.
- Imagen de fondo (*background image*): imagen que se muestra por debajo del contenido. Si se define un color y una imagen de fondo, la imagen tiene más prioridad y es la que se visualiza.
- Color de fondo (*background color*): color que rellena el espacio ocupado por el contenido y su posible relleno. Si se define un color y una imagen de fondo, el color tiene menos prioridad y por tanto se visualiza la imagen.
- Margen (*margin*): espacio libre entre la caja y las posibles cajas adyacentes.

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

4.1. Anchura y altura

4.1.1. Anchura

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

Tabla 4.1. Propiedad width

`width`	**Anchura**
Valores	`<medida>` | `<porcentaje>` | `auto` | `inherit`
Se aplica a	Todos los elementos salvo las filas (y grupos de filas) de una tabla
Valor inicial	auto
Descripción	Establece la anchura de un elemento

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

calcular automáticamente la anchura del elemento, teniendo en cuenta sus contenidos y el sitio disponible en la página.

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

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

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

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

4.1.2. Altura

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

Tabla 4.2. Propiedad height

`height`	**Altura**
Valores	`<medida>` | `<porcentaje>` | `auto` | `inherit`
Se aplica a	Todos los elementos salvo las columnas (y grupos de columnas) de una tabla
Valor inicial	auto
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.

Tabla 4.3. Propiedad margin-top, margin-right, margin-bottom, margin-left

margin-top **margin-right** **margin-bottom** **margin-left**	**Margen superior** **Margen derecho** **Margen inferior** **Margen izquierdo**
Valores	(<medida> | <porcentaje> | auto) | inherit
Se aplica a	Todos los elementos salvo algunos casos especiales de elementos mostrados como tablas
Valor inicial	0
Descripción	Establece cada uno de los márgenes horizontales y verticales de un elemento

Aunque no es habitual, el valor de los márgenes puede ser negativo. El siguiente ejemplo añade un margen izquierdo al segundo párrafo:

```
<!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 propiedad margin-left</title>
<style type="text/css">
.destacado {
  margin-left: 2em;
}
</style>
</head>

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

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

<p>Aliquam euismod sapien eu libero. Ut tempor orci at nulla. Nam in eros
egestas massa vehicula nonummy. Morbi posuere, nibh ultricies consectetuer tincidunt,
```

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

El aspecto que muestra el ejemplo anterior en cualquier navegador se muestra a continuación:

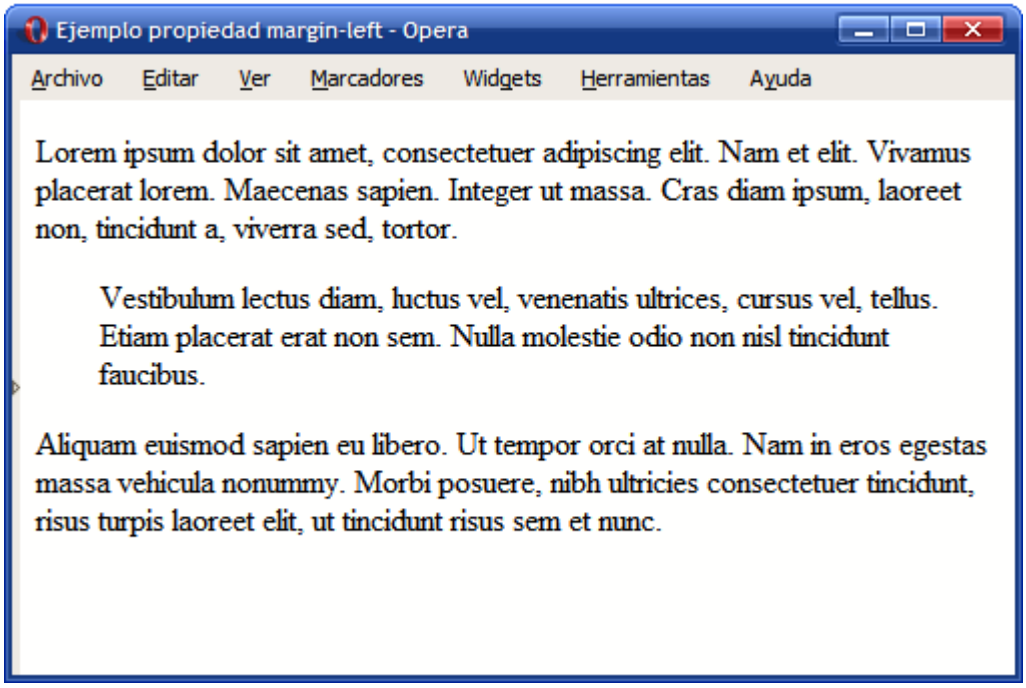

Figura 4.4. Ejemplo de propiedad margin-left

El siguiente ejemplo añade un margen a todos los iconos para facilitar su identificación y mejorar el diseño general de la página:

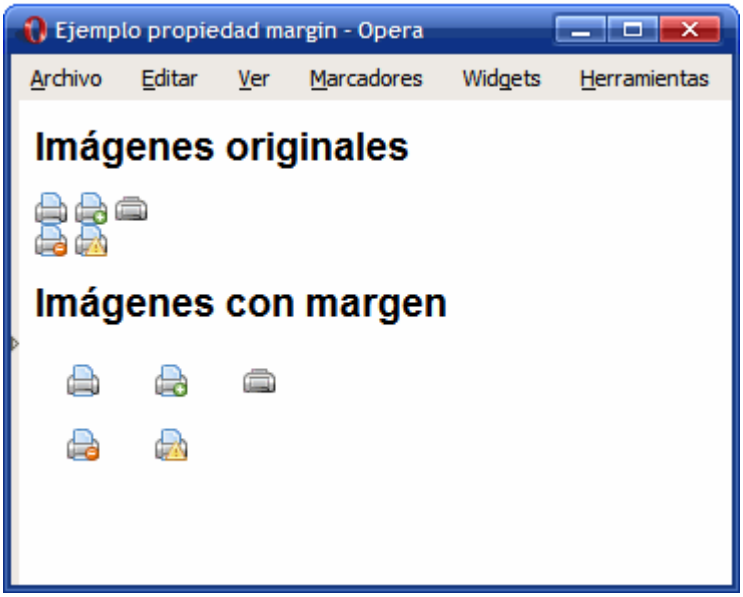

Figura 4.5. Ejemplo de propiedad margin

El código HTML y CSS del ejemplo de la imagen anterior 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 propiedad margin</title>
<style type="text/css">
div img {
  margin-top: .5em;
  margin-bottom: .5em;
  margin-left: 1em;
  margin-right: .5em;
}
</style>
</head>

<body>

<h1>Imágenes originales</h1>
<img src="imagenes/printer.png" alt="Impresora" />
<img src="imagenes/printer_add.png" alt="Impresora nueva" />
<img src="imagenes/printer_empty.png" alt="Impresora sin papel" />
<br/>
<img src="imagenes/printer_delete.png" alt="Impresora eliminada" />
<img src="imagenes/printer_error.png" alt="Impresora con error" />

<h1>Imágenes con margen</h1>
<div>
  <img src="imagenes/printer.png" alt="Impresora" />
  <img src="imagenes/printer_add.png" alt="Impresora nueva" />
  <img src="imagenes/printer_empty.png" alt="Impresora sin papel" />
  <br/>
  <img src="imagenes/printer_delete.png" alt="Impresora eliminada" />
  <img src="imagenes/printer_error.png" alt="Impresora con error" />
</div>

</body>
</html>
```

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

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

Tabla 4.4. Propiedad margin

margin	**Margen**
Valores	(<medida> | <porcentaje> | auto) {1, 4} | inherit

`margin`	**Margen**
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:

<code css>
div img {
  margin: .5em .5em .5m 1em;
}
```

Otra alternativa:

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

Ejercicio 2 Ver enunciado en la página 181

El resultado del ejercicio 2 es que, en ambos casos, el navegador mostrará los elementos `<div>` con el siguiente aspecto:

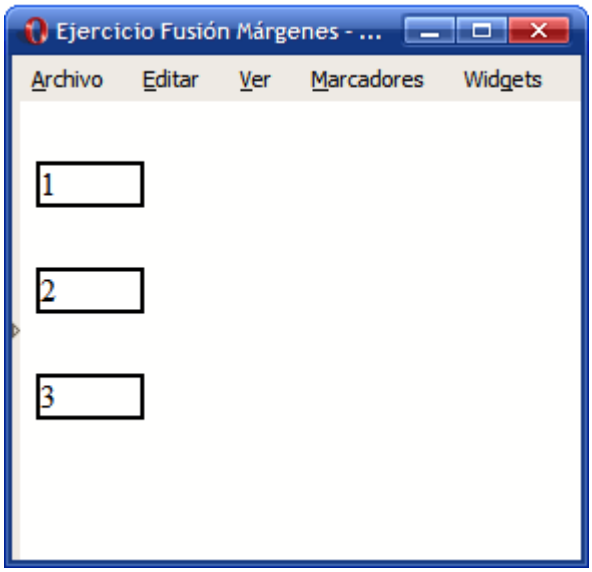

Figura 4.6. CSS fusiona de forma automática los márgenes verticales

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.

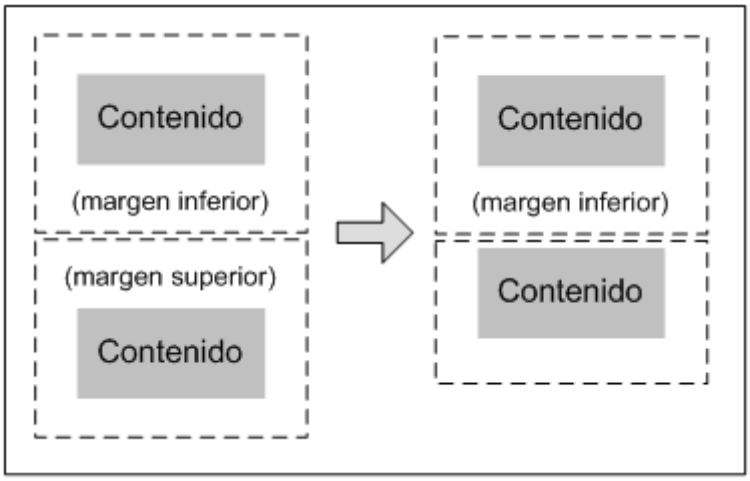

Figura 4.7. Fusión automática de los márgenes verticales

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:

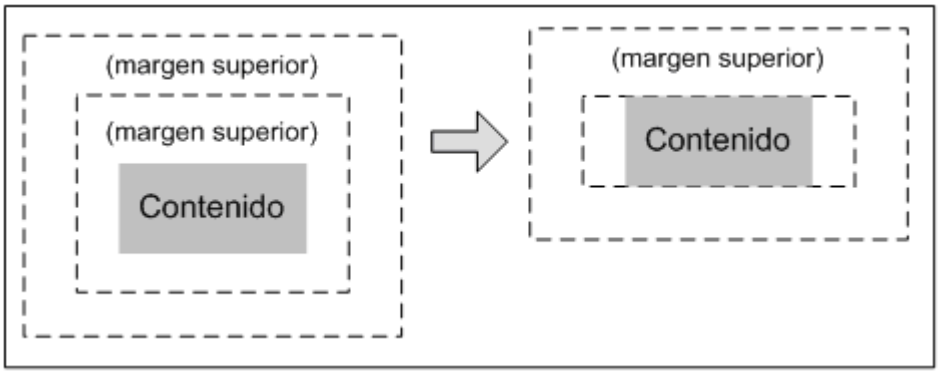

Figura 4.8. Fusión de los márgenes de los elementos interiores

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.

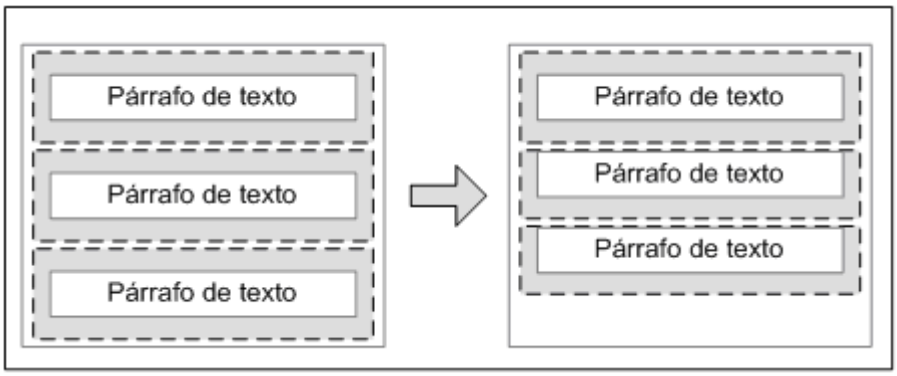

Figura 4.9. Motivo por el que se fusionan automáticamente los márgenes verticales

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.

Tabla 4.5. Propiedad padding-top, padding-right, padding-bottom, padding-left

padding-top **padding-right** **padding-bottom** **padding-left**	**Relleno superior** **Relleno derecho** **Relleno inferior** **Relleno izquierdo**
Valores	`( <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

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

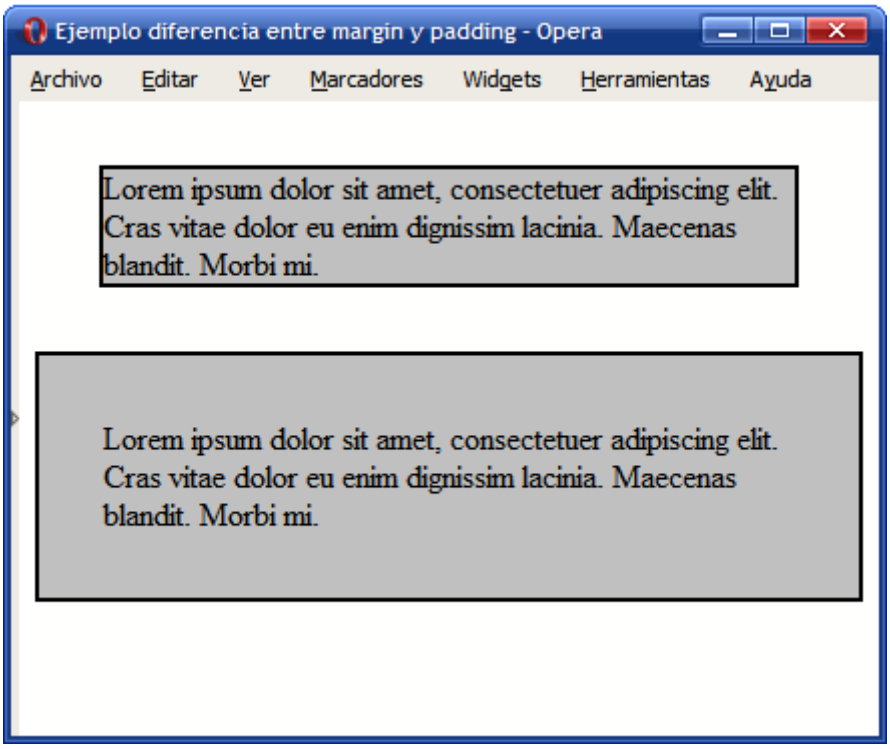

Figura 4.10. Diferencia entre relleno (padding) y margen (margin)

El código HTML y CSS del ejemplo 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 diferencia entre margin y padding</title>
<style type="text/css">
.margen {
  margin-top: 2em; margin-right: 2em; margin-bottom: 2em; margin-left: 2em;
```

```
}
.relleno {
  padding-top: 2em; padding-right: 2em; padding-bottom: 2em; padding-left: 2em;
}
</style>
</head>

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

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

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

Tabla 4.6. Propiedad padding

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

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

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

Ejercicio 3 Ver enunciado en la página 181

4.3. Bordes

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

4.3.1. Anchura

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

Tabla 4.7. Propiedad border-top-width, border-right-width, border-bottom-width, border-left-width

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

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

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

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

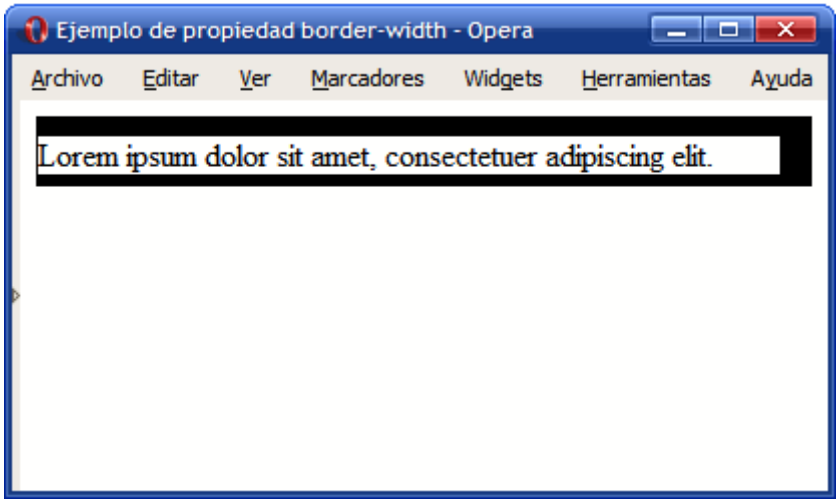

Figura 4.11. Ejemplo de propiedad border-width

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

Tabla 4.8. Propiedad border-width

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

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

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

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

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

4.3.2. Color

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

Tabla 4.9. Propiedad border-top-color, border-right-color, border-bottom-color, border-left-color

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

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

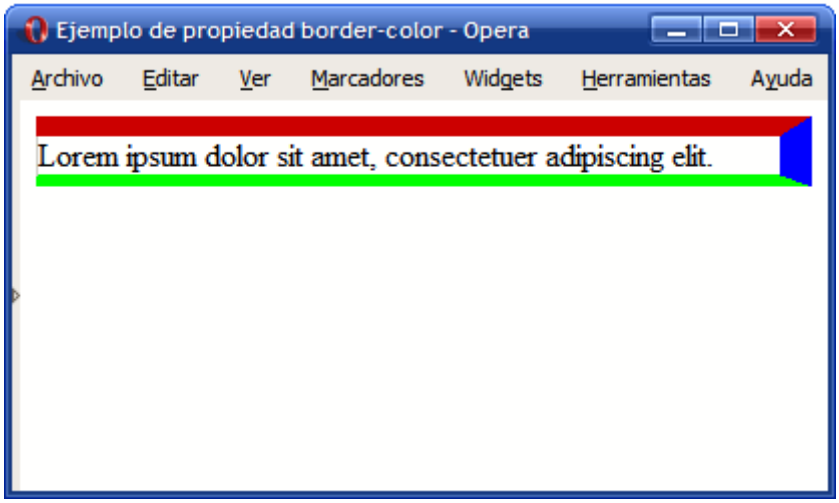

Figura 4.12. Ejemplo de propiedad border-color

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;
}
```

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

Tabla 4.10. Propiedad border-color

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

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

4.3.3. Estilo

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

Tabla 4.11. Propiedad border-top-style, border-right-style, border-bottom-style, border-left-style

`border-top-style`
`border-right-style`
`border-bottom-style`
`border-left-style`
	Estilo del borde superior
	Estilo del borde derecho
	Estilo del borde inferior
	Estilo del borde izquierdo

Valores	`none` | `hidden` | `dotted` | `dashed` | `solid` | `double` | `groove` | `ridge` | `inset` | `outset` | `inherit`
Se aplica a	Todos los elementos
Valor inicial	none
Descripción	Establece el estilo de cada uno de los cuatro bordes de los elementos

El estilo de los bordes sólo se puede indicar mediante alguna de las palabras reservadas definidas por CSS. Siguiendo el ejemplo anterior, se puede modificar el estilo de cada uno de los bordes:

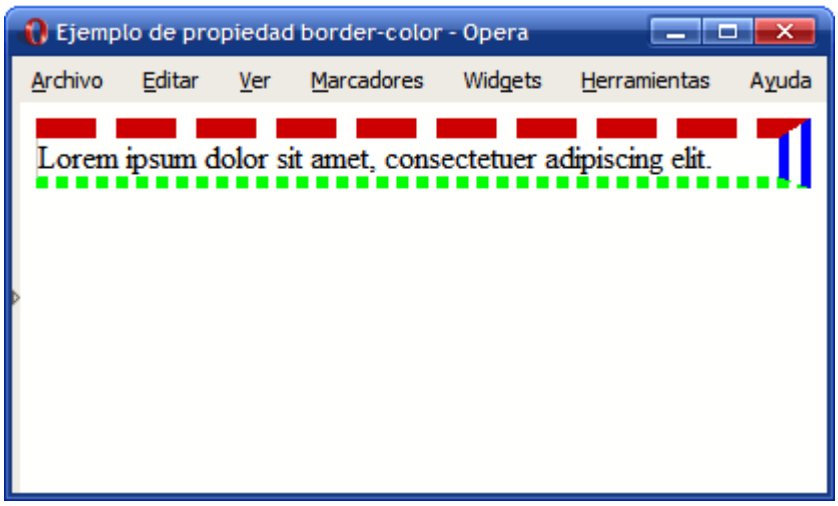

Figura 4.13. Ejemplo de propiedad border-style

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:

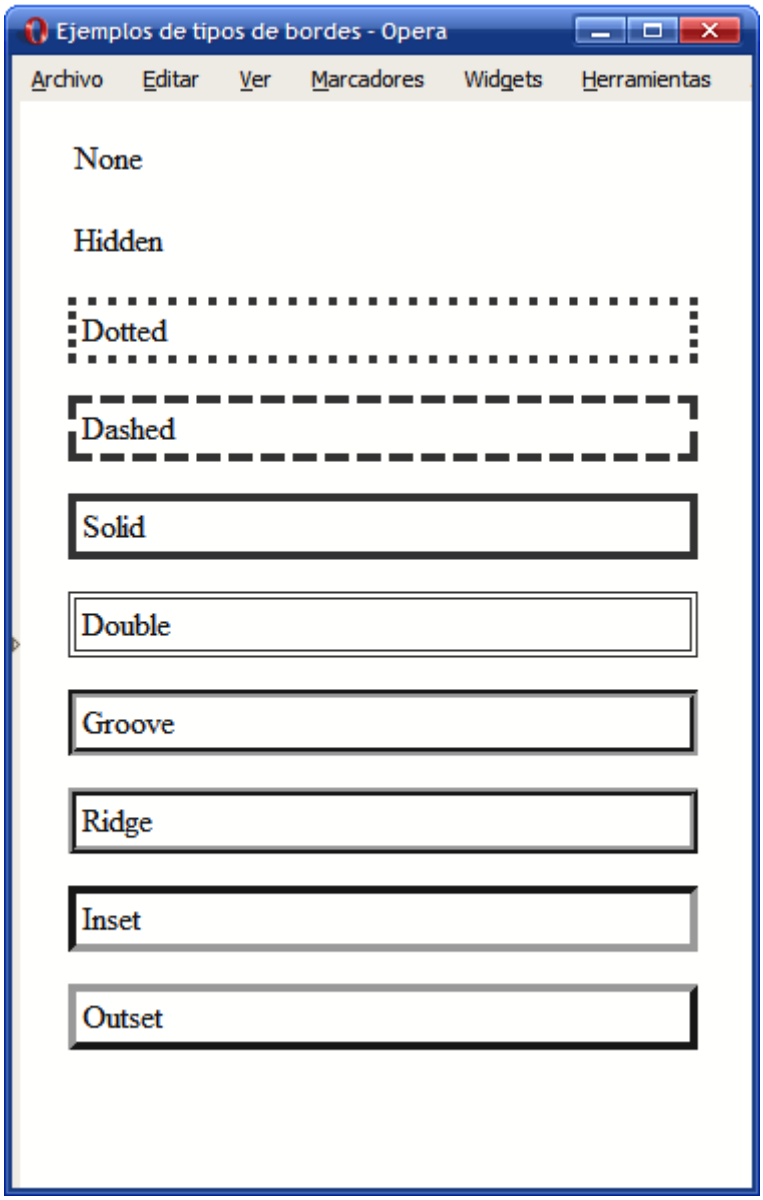

Figura 4.14. Tipos de bordes definidos por CSS

Los bordes más utilizados en los diseños habituales son `solid` y `dashed`, seguidos de `double` y `dotted`.

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

Tabla 4.12. Propiedad border-style

`border-style`	**Estilo del borde**
Valores	`(none | hidden | dotted | dashed | solid | double | groove | ridge | inset | outset ) {1, 4} | inherit`
Se aplica a	Todos los elementos
Valor inicial	-

border-style	Estilo del borde
Descripción	Establece el estilo de todos los bordes del elemento

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

4.3.4. Propiedades shorthand

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

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

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

Tabla 4.13. Propiedad border-top, border-right, border-bottom, border-left

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

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

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

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

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

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

Tabla 4.14. Propiedad border

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

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; }
```

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 específicas:

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

Ejercicio 4 Ver enunciado en la página 183

4.4. Margen, relleno, bordes y box model

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

```
div {
  width: 300px;
  padding-left: 50px; padding-right: 50px;
  margin-left: 30px; margin-right: 30px;
```

```
  border: 10px solid black;
}
```

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

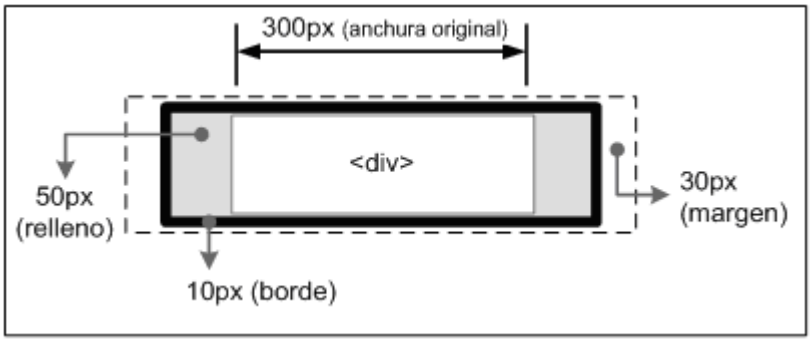

Figura 4.15. La anchura total de un elemento tiene en cuenta los márgenes, rellenos y bordes

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 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 encontraron fue la de dividir el comportamiento del navegador en dos modos diferentes: 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:

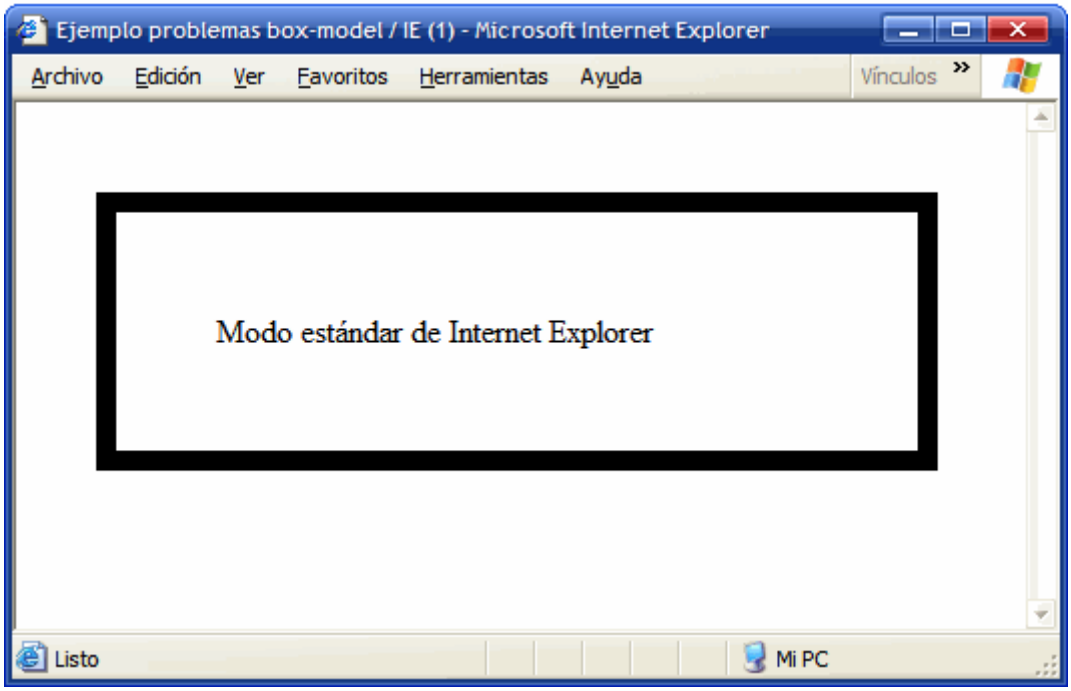

Figura 4.16. Internet Explorer 6 en modo estándar

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

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

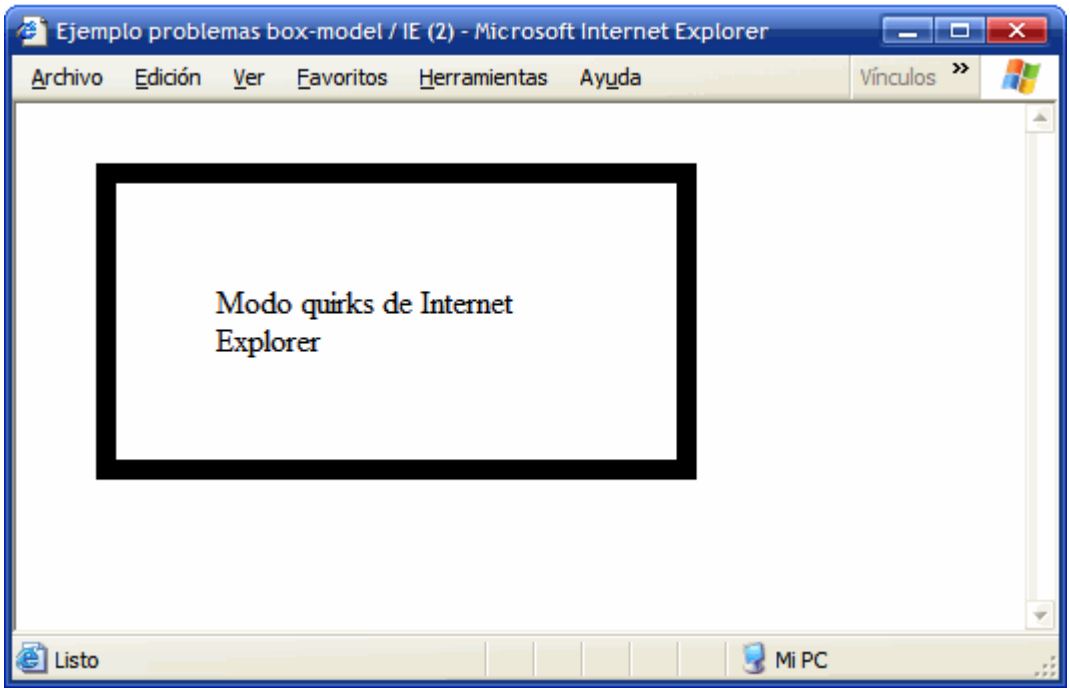

Figura 4.17. Internet Explorer 6 en modo quirks

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 deberían incluir la declaración apropiada de `DOCTYPE`.

4.5. Fondos

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

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

Tabla 4.15. Propiedad background-color

background-color	Color de fondo
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;
}
```

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

Tabla 4.16. Propiedad background-image

background-image	Imagen de fondo
Valores	<url> | none | inherit
Se aplica a	Todos los elementos
Valor inicial	none
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(images/fondo.gif);
}
```

Resultado

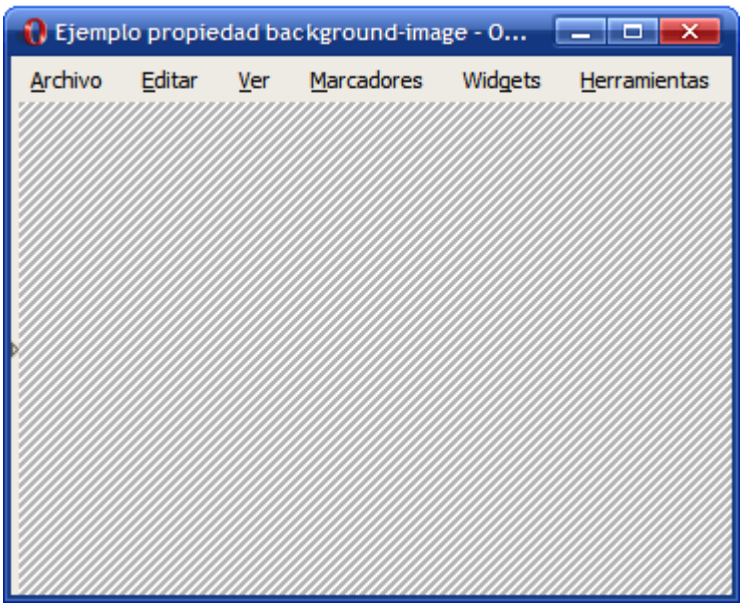

Figura 4.19. Página con una imagen de fondo

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.

Tabla 4.17. Propiedad background-repeat

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

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

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

Figura 4.20. 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:

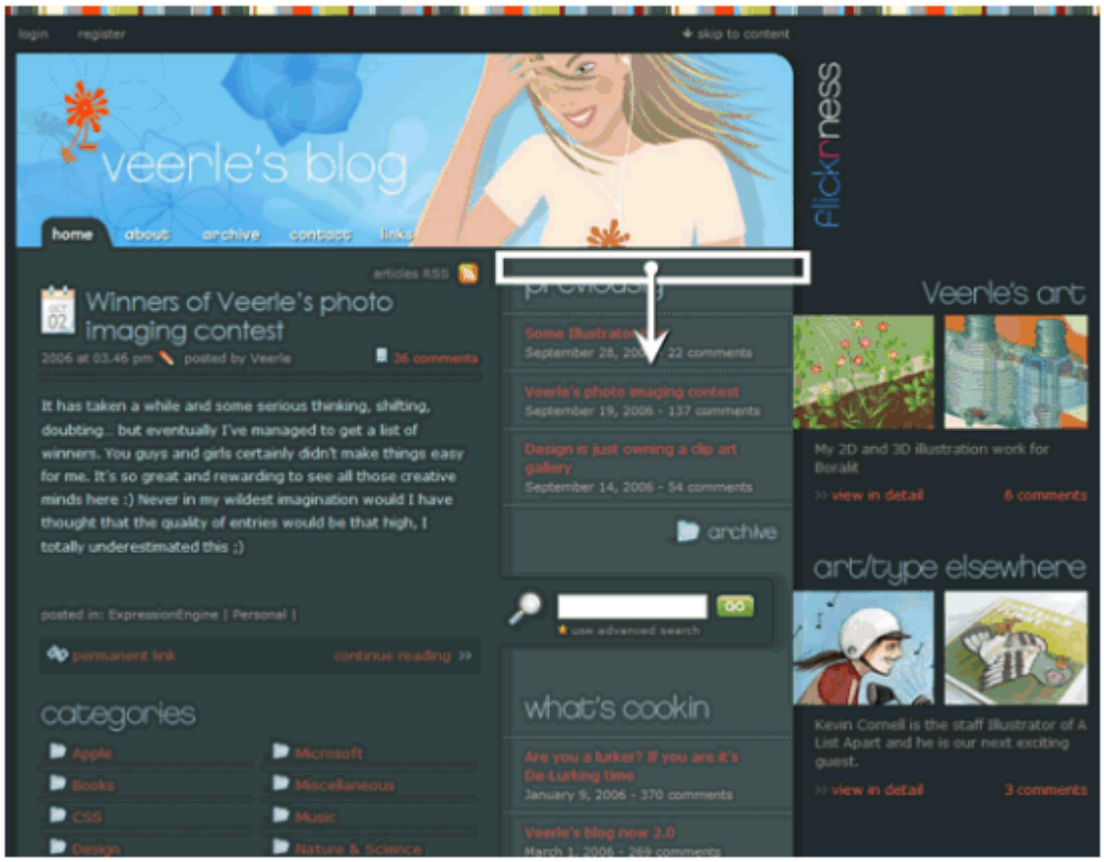

Figura 4.21. Uso de repeat-y en la página de Veerle.duoh.com

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`.

Tabla 4.18. Propiedad background-position

`background-position`	**Posición de la imagen de fondo**
Valores	`( ( <porcentaje> | <medida> | left | center | right ) ( <porcentaje> | <medida> | top | center | bottom )? ) | ( ( left | center | right ) || ( top | center | bottom ) ) | inherit`

background-position	Posición de la imagen de fondo
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 qué distancia 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%.

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:

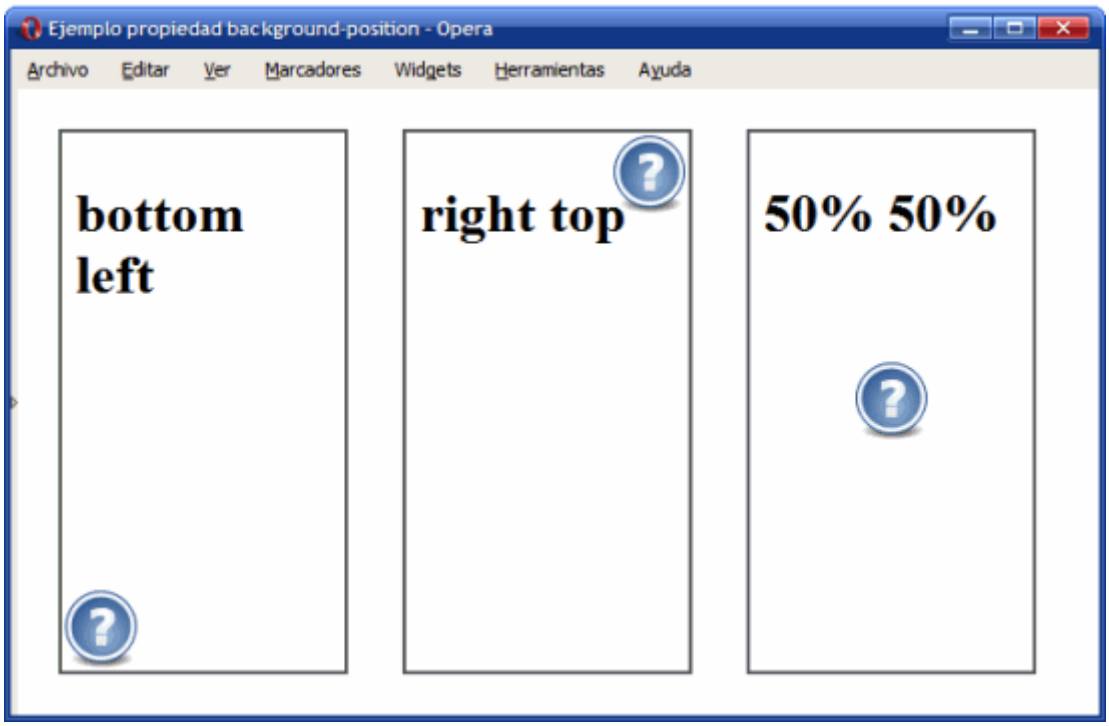

Figura 4.22. Ejemplo de propiedad background-position

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`.

Tabla 4.19. Propiedad background-attachment

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

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

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

Tabla 4.20. Propiedad background

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

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

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

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

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

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

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

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

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

background-position: 0px -27px;

background:none;

background: #293838 url(./graphics/icons/icon-permalink-big.gif) no-repeat center left;
```

Ejercicio 5 Ver enunciado en la página 184