Introducción a CSS

Javier Eguíluz Pérez

Sobre este libro...

- Los contenidos de este libro están bajo una licencia Creative Commons Reconocimiento - No Comercial - Sin Obra Derivada 3.0 (`http://creativecommons.org/licenses/by-nc-nd/3.0/deed.es`)
- **Esta versión impresa se creó el 21 de octubre de 2007 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 7
1.1. ¿Qué es CSS? 7
1.2. Breve historia de CSS 7
1.3. Especificación oficial 8
1.4. Funcionamiento básico de CSS 8
1.5. Cómo incluir CSS en un documento XHTML 10
1.5.1. Incluir CSS en el mismo documento HTML 10
1.5.2. Definir CSS en un archivo externo 10
1.5.3. Incluir CSS en los elementos HTML 11
1.6. Glosario básico 12
1.7. Medios CSS 12
1.8. Comentarios 14
1.9. Sintaxis de la definición de cada propiedad CSS 15
Capítulo 2. Selectores 17
2.1. Selectores básicos 17
2.1.1. Selector universal 17
2.1.2. Selector de tipo o etiqueta 17
2.1.3. Selector descendente 19
2.1.4. Selector de clase 20
2.1.5. Selectores de ID 22
2.1.6. Combinación de selectores básicos 23
2.2. Selectores avanzados 23
2.2.1. Selector de hijos 23
2.2.2. Selector adyacente 24
2.2.3. Selector de atributos 25
2.3. Agrupación de reglas 26
2.4. Herencia 27
2.5. Colisiones de estilos 27
Capítulo 3. Unidades de medida y colores 29
3.1. Unidades de medida 29
3.1.1. Unidades relativas 29
3.1.2. Unidades absolutas 33
3.1.3. Porcentajes 33
3.1.4. Recomendaciones 34
3.2. Colores 35
3.2.1. Palabras clave 35
3.2.2. RGB decimal 36
3.2.3. RGB porcentual 36
3.2.4. RGB hexadecimal 37
3.2.5. Colores del sistema 38
3.2.6. Colores web safe 39
Capítulo 4. Box model 40

4.1. Anchura y altura 42
4.1.1. Anchura 42
4.1.2. Altura 43
4.2. Margen y relleno 44
4.2.1. Margen 44
4.2.2. Relleno 49
4.3. Bordes 51
4.3.1. Anchura 51
4.3.2. Color 53
4.3.3. Estilo 54
4.3.4. Propiedades shorthand 57
4.4. Margen, relleno, bordes y box model 58
4.5. Fondos 61
Capítulo 5. Posicionamiento y visualización 69
5.1. Tipos de elementos 69
5.2. Posicionamiento 70
5.3. Posicionamiento normal 71
5.4. Posicionamiento float 74
5.5. Posicionamiento absoluto 81
5.6. Visualización 85
5.6.1. Display y visibility 85
5.6.2. Overflow 87
5.6.3. Z-index 89
Capítulo 6. Texto 91
6.1. Tipografía 91
6.2. Texto 97
Capítulo 7. Enlaces 111
7.1. Estilos básicos 111
7.1.1. Tamaño, color y decoración 111
7.1.2. Pseudo-clases 111
7.2. Estilos avanzados 113
7.2.1. Decoración personalizada 113
7.2.2. Imágenes según el tipo de enlace 114
7.2.3. Mostrar los enlaces como si fueran botones 115
Capítulo 8. Imágenes 116
8.1. Estilos básicos 116
8.1.1. Eliminar el borde de las imágenes con enlaces 116
8.2. Estilos avanzados 116
8.2.1. Sombra (drop shadow) 116
Capítulo 9. Listas 119
9.1. Estilos básicos 119
9.1.1. Viñetas personalizadas 119

9.1.2. Menú vertical sencillo ... 123
9.1.3. Menú vertical avanzado ... 126
9.2. Estilos avanzados ... 126
9.2.1. Menú horizontal básico ... 126
9.2.2. Menú horizontal con solapas ... 130
9.2.3. Menú horizontal avanzado ... 130
Capítulo 10. Tablas ... 132
10.1. Estilos básicos ... 132
10.2. Estilos avanzados ... 135
Capítulo 11. Formularios ... 140
11.1. Estilos básicos ... 140
11.1.1. Mostrar un botón como un enlace ... 140
11.1.2. Mejoras en los campos de texto ... 140
11.1.3. Labels alineadas y formateadas ... 141
11.2. Estilos avanzados ... 145
11.2.1. Formulario en varias columnas ... 145
11.2.2. Resaltar el campo seleccionado ... 146
Capítulo 12. Layout ... 148
12.1. Estilos básicos ... 148
12.1.1. Centrar una página completa ... 148
12.2. Estructura o layout ... 151
12.2.1. Diseño a 2 columnas con cabecera y pie de página ... 151
12.2.2. Diseño a 3 columnas con cabecera y pie de página ... 154
12.3. Alturas/anchuras máximas y mínimas ... 156
12.4. Estilos avanzados ... 113
Capítulo 13. Otros ... 160
13.1. Propiedades shorthand ... 160
13.2. Versión para imprimir ... 161
13.3. Personalizar el cursor ... 163
13.4. Hacks y filtros ... 167
13.5. Prioridad de las declaraciones CSS ... 168
13.6. Validador ... 170
13.7. Recomendaciones generales sobre CSS ... 170
13.7.1. Atributos ID y class ... 170
13.7.2. CLASSitis y DIVitis ... 171
13.7.3. Estructuración del código CSS ... 171
13.7.4. División de los estilos en varios archivos CSS ... 173
Capítulo 14. Recursos útiles ... 175
14.1. Extensiones de Firefox ... 175
14.1.1. Firebug ... 175
14.1.2. Web Developer ... 176
14.1.3. HTML Validator ... 176

14.1.4. Otras extensiones 177
14.2. Enlaces de interés 179
14.2.1. Recomendación oficial 179
14.2.2. Recursos 179
14.2.3. Foros 179
14.2.4. Galerías de páginas 179
Capítulo 15. Ejercicios 180
Capítulo 16. Ejercicios resueltos 199

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:

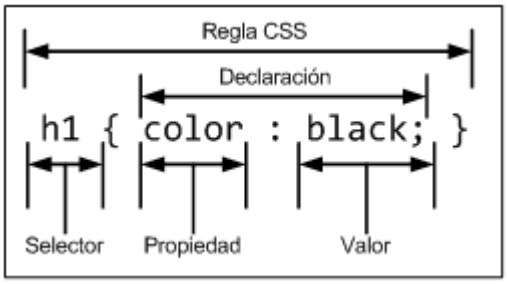


Figura 1.1. Componentes de un estilo CSS básico

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:

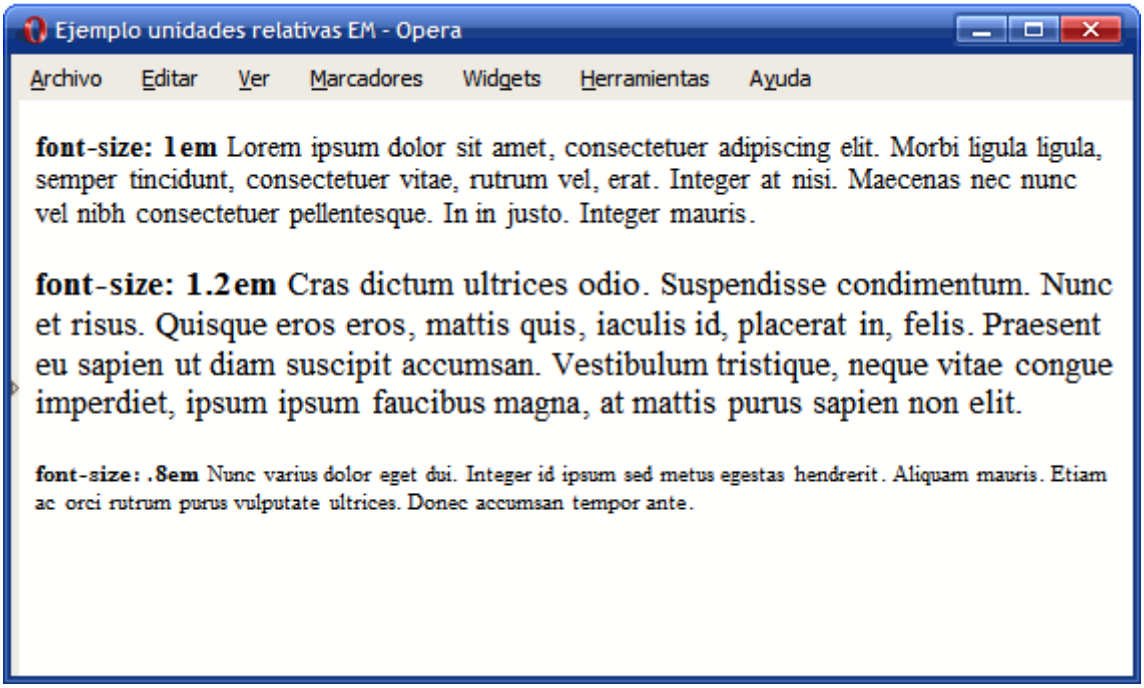


Figura 3.1. Ejemplo de tamaño de letra definido con la unidad relativa em

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:

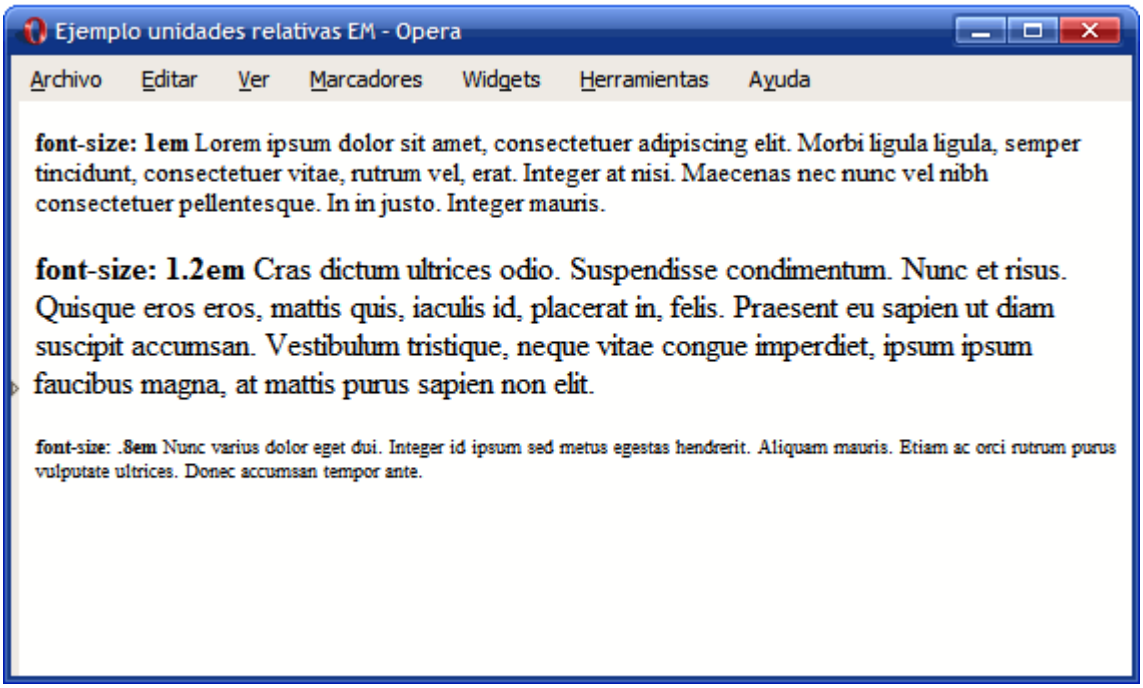


Figura 3.2. Ejemplo de tamaño de letra definido con la unidad relativa em

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

HTML 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 HTML

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:

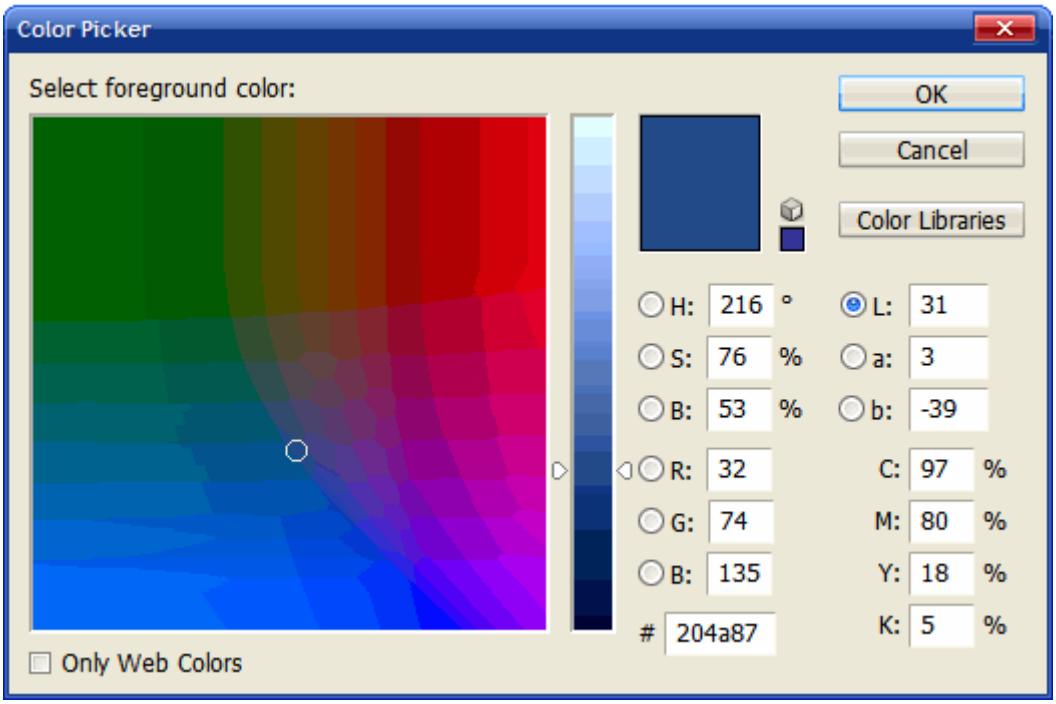


Figura 3.5. Herramienta de color de Photoshop para definir los colores según los modelos RGB, CMYK, Lab, HSB y RGB hexadecimal

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:

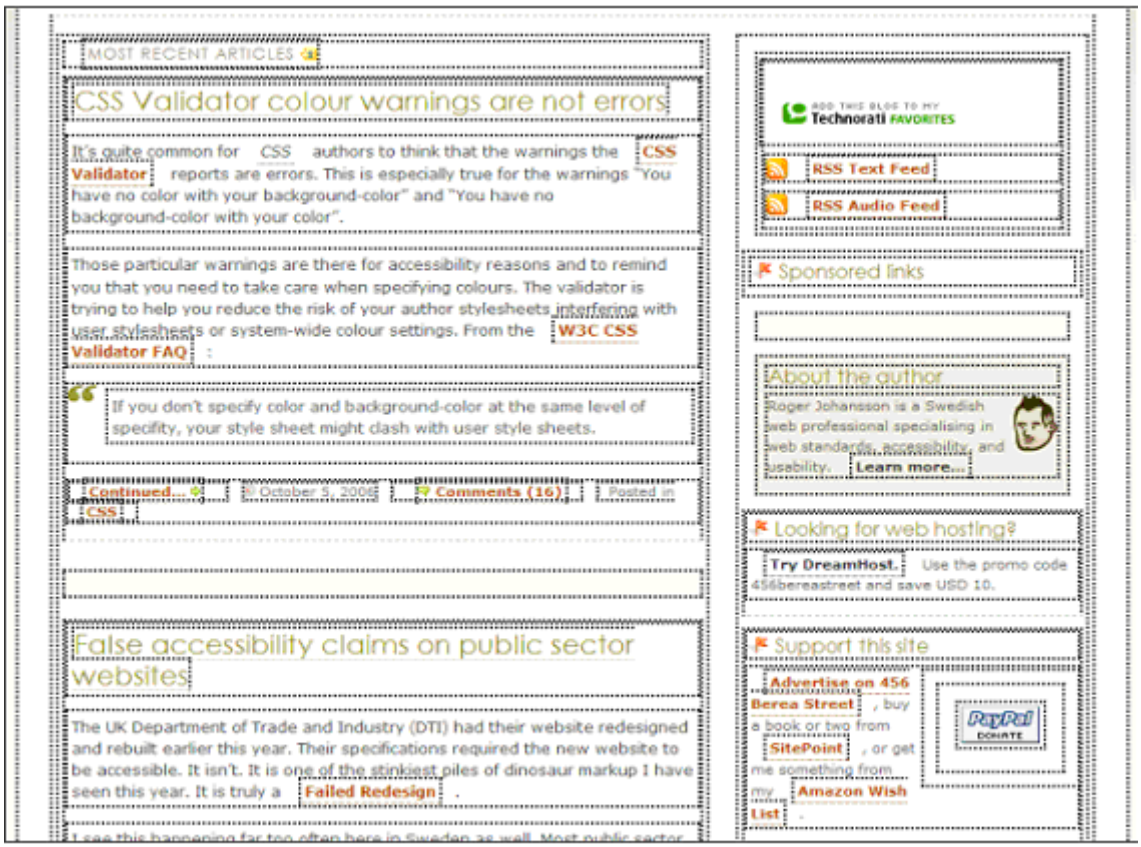


Figura 4.1. Cajas que forman el box model de la página 456bereastreet.com

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:

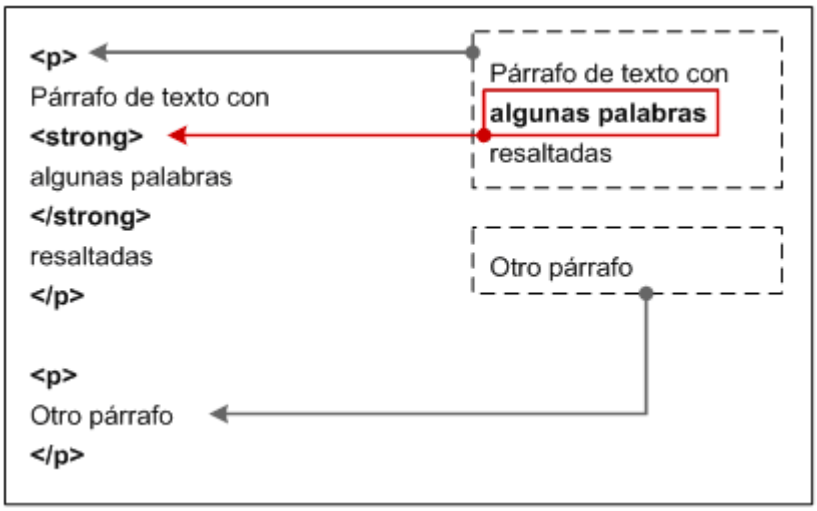


Figura 4.2. Las cajas se crean automáticamente al definir cada elemento HTML

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:

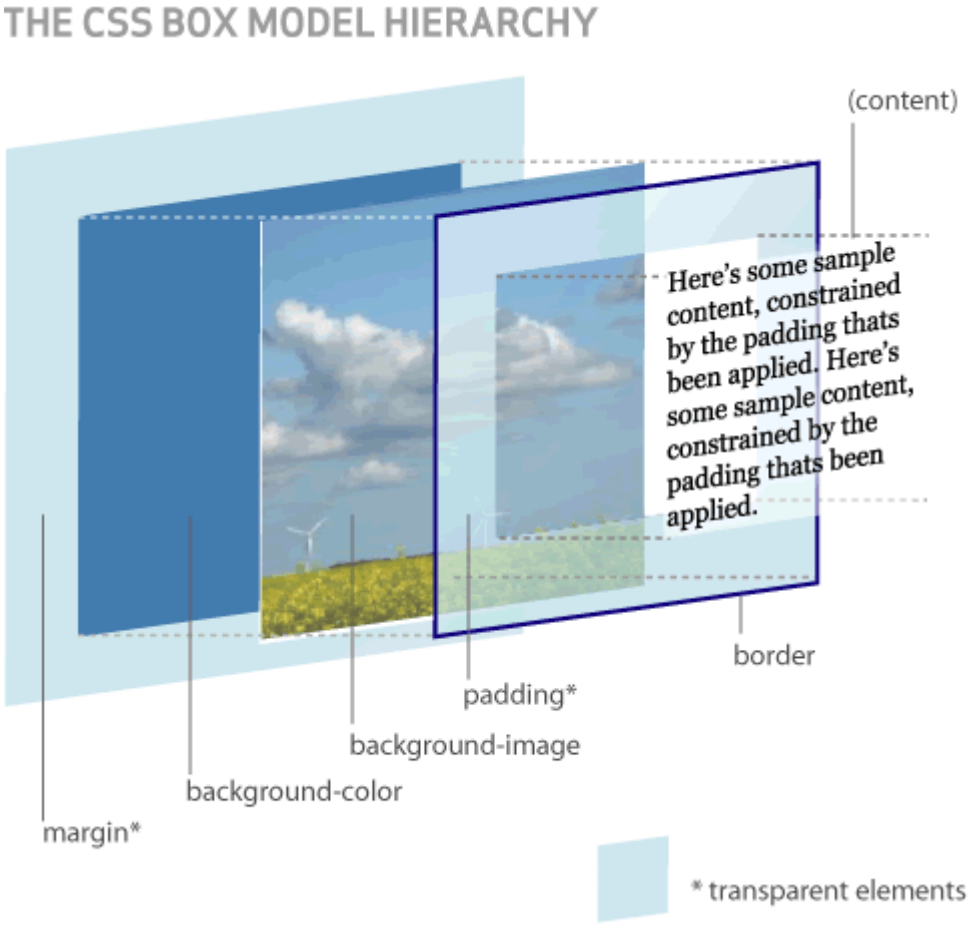


Figura 4.3. Representación tridimensional del box model de CSS

(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:

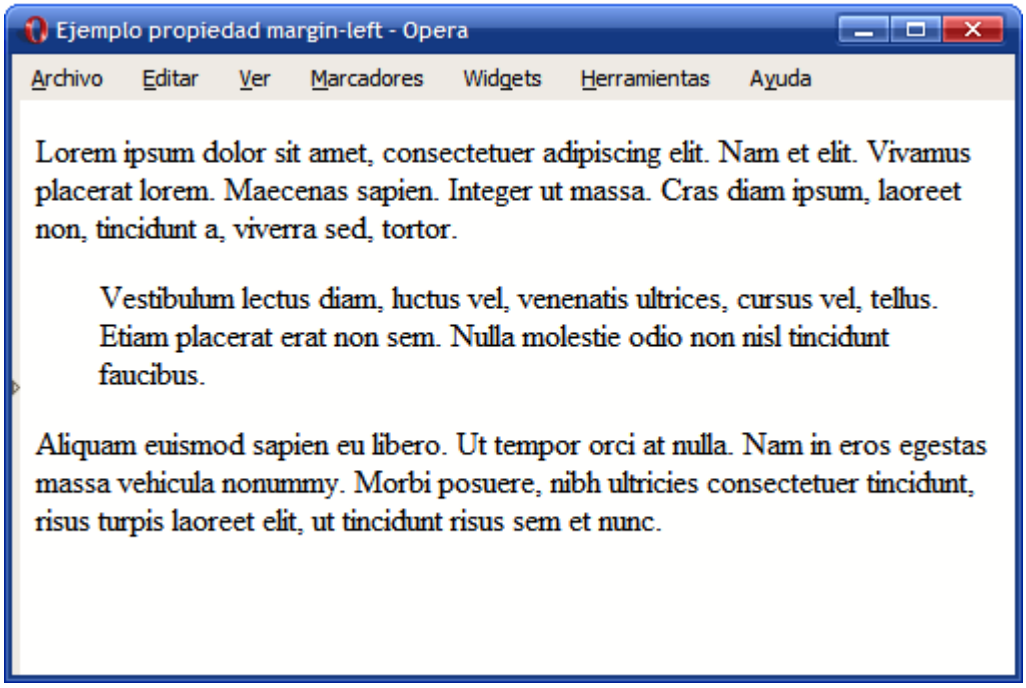


Figura 4.4. Ejemplo de propiedad margin-left

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

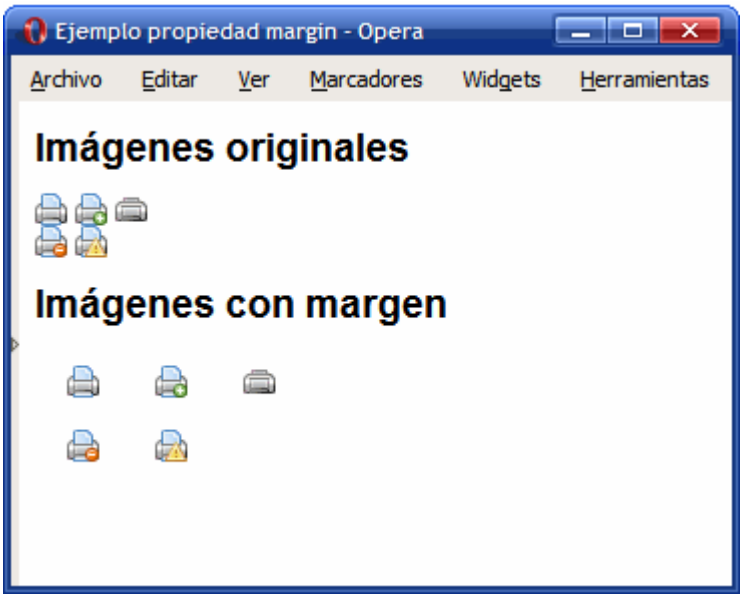


Figura 4.5. Ejemplo de propiedad margin

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:

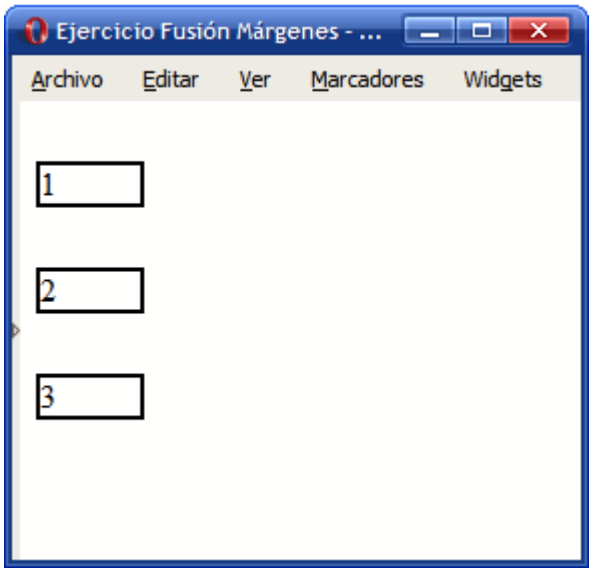


Figura 4.6. CSS fusiona de forma automática los márgenes verticales

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.

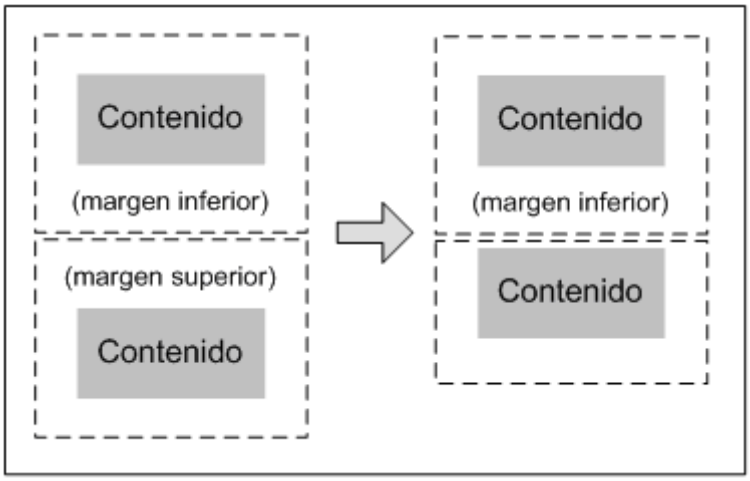


Figura 4.7. Fusión automática de los márgenes verticales

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:

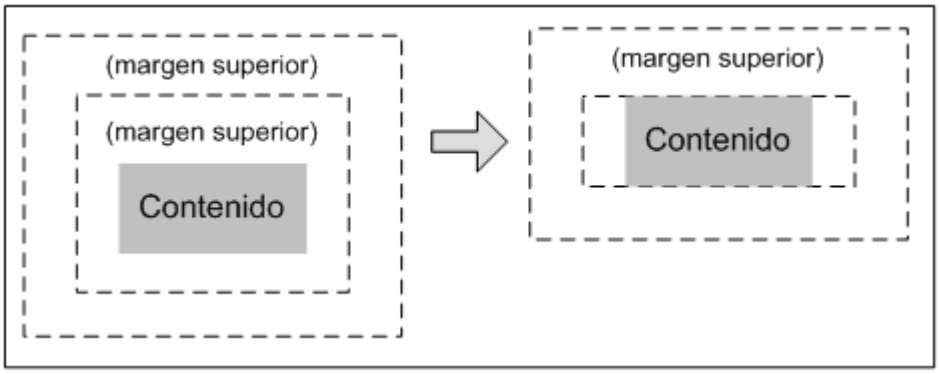


Figura 4.8. Fusión de los márgenes de los elementos interiores

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.

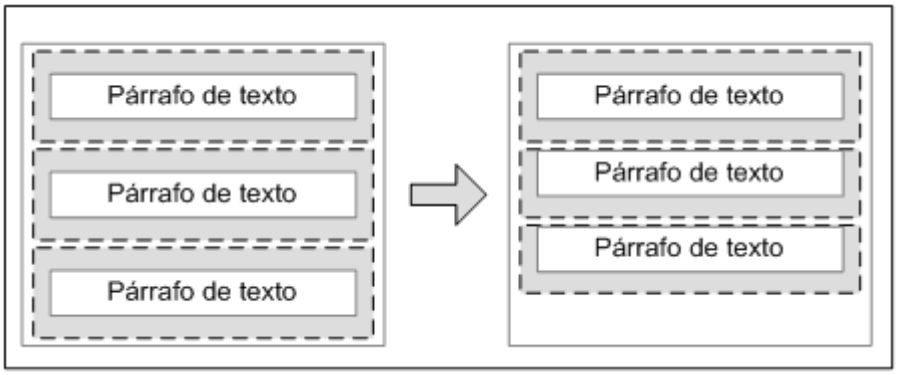


Figura 4.9. Motivo por el que se fusionan automáticamente los márgenes verticales

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:

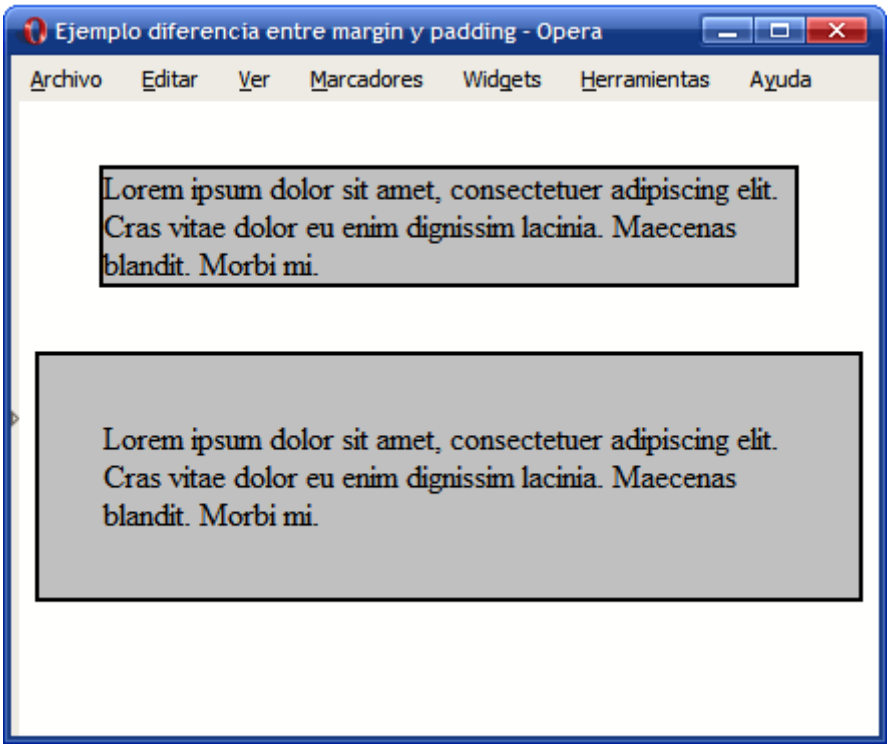


Figura 4.10. Diferencia entre relleno (padding) y margen (margin)

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:

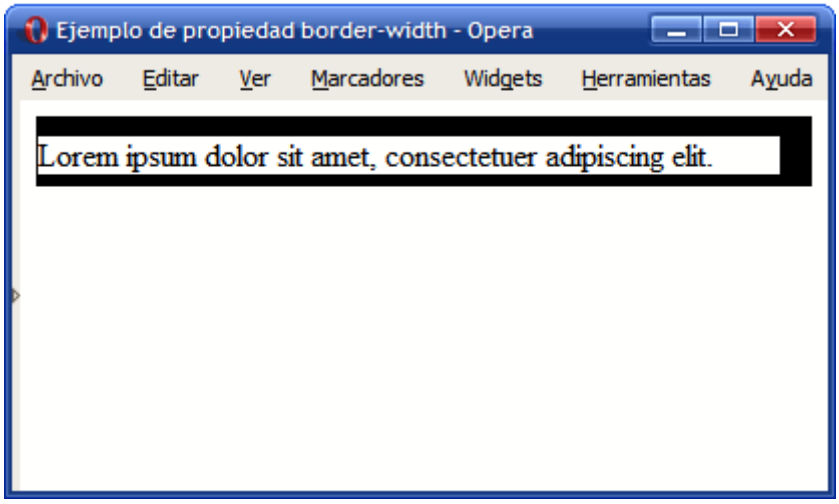


Figura 4.11. Ejemplo de propiedad border-width

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:

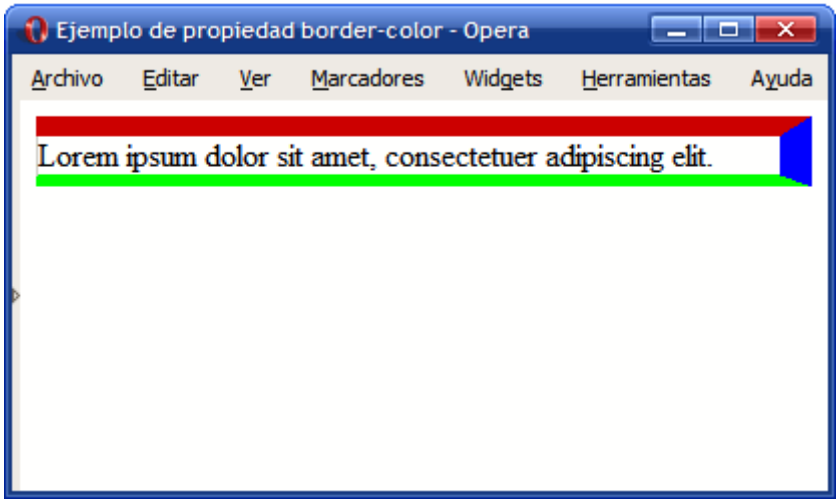


Figura 4.12. Ejemplo de propiedad border-color

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:

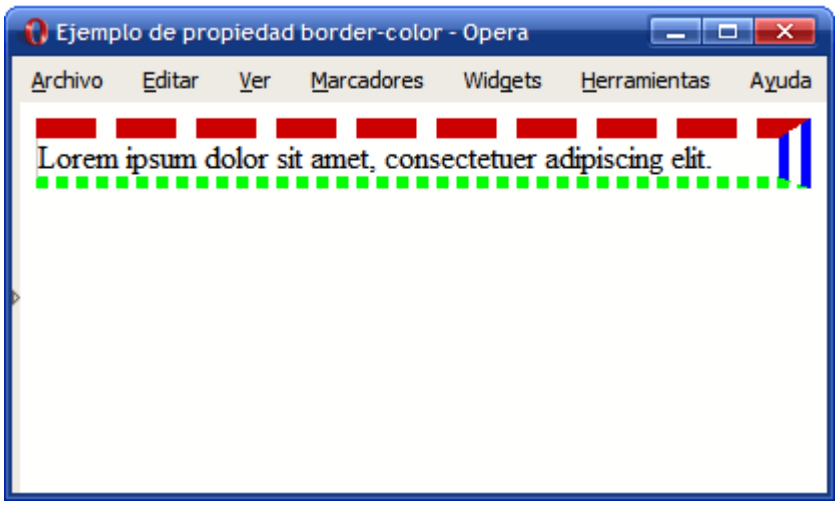


Figura 4.13. Ejemplo de propiedad border-style

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:

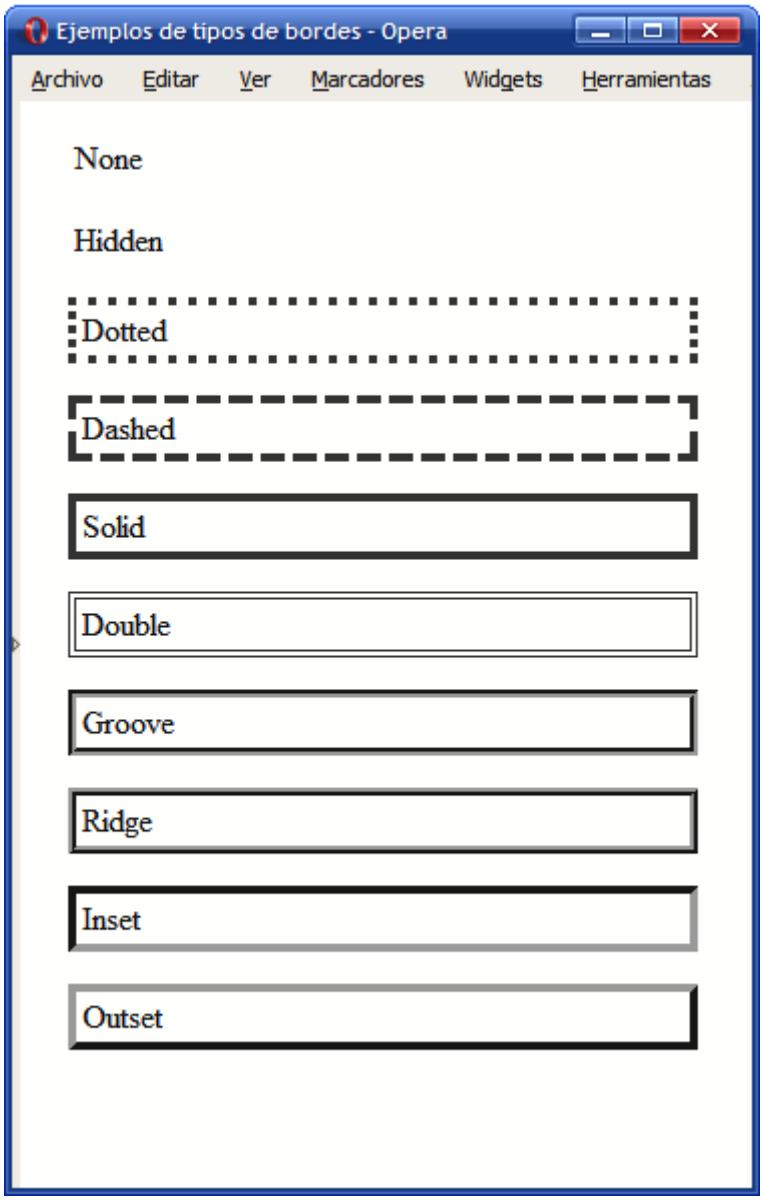


Figura 4.14. Tipos de bordes definidos por CSS

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:

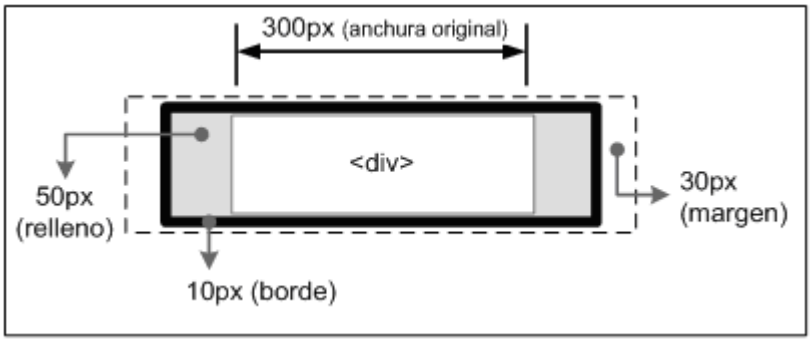


Figura 4.15. La anchura total de un elemento tiene en cuenta los márgenes, rellenos y bordes

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:

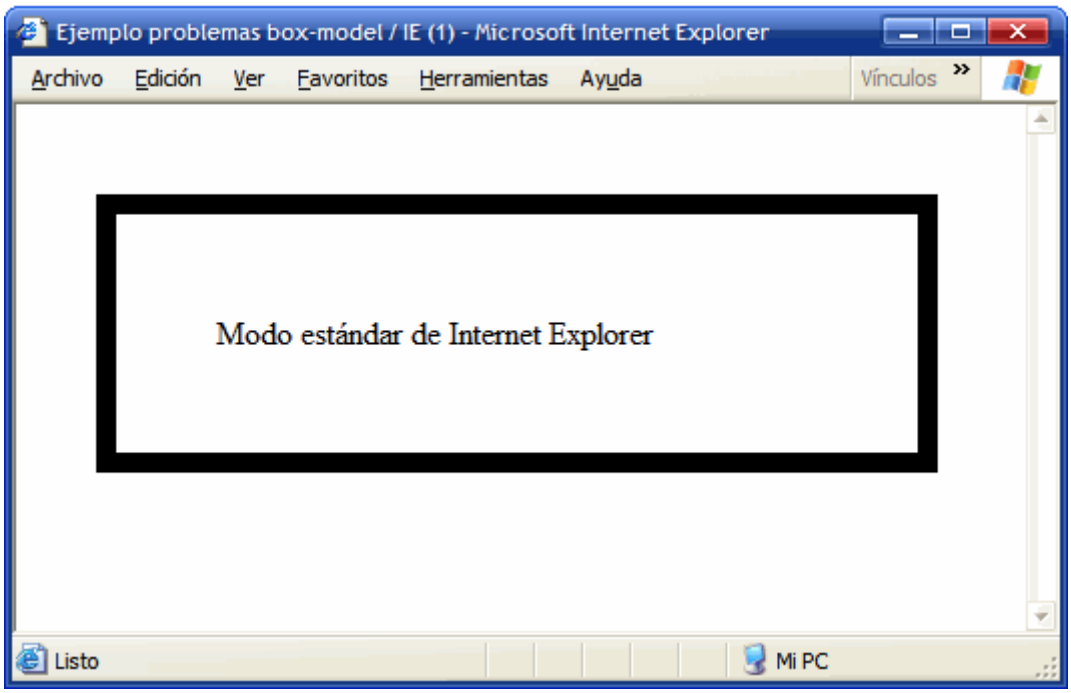


Figura 4.16. Internet Explorer 6 en modo estándar

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:

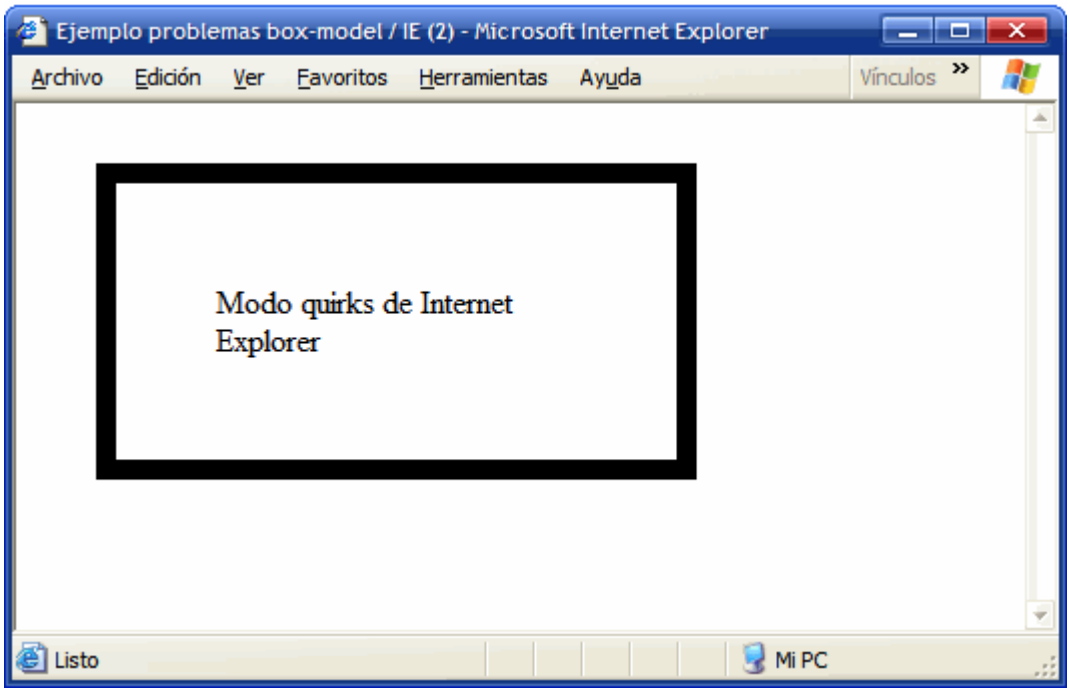


Figura 4.17. Internet Explorer 6 en modo quirks

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

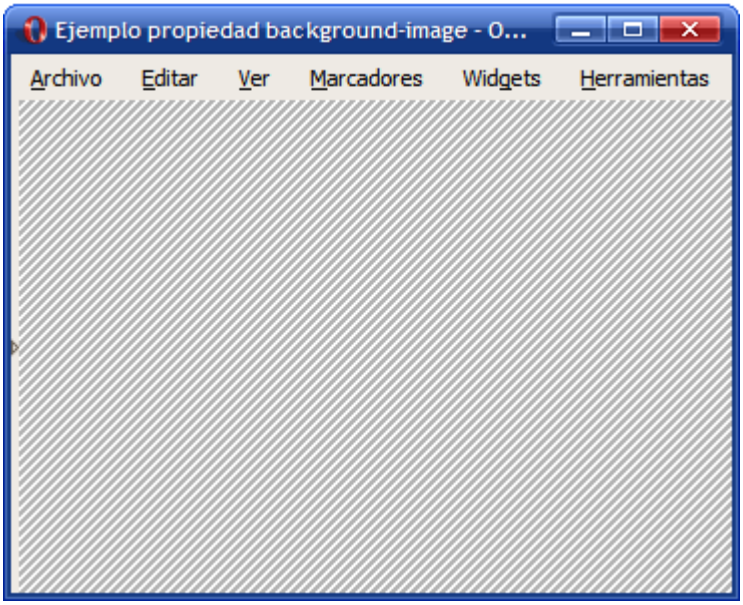


Figura 4.19. Página con una imagen de fondo

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:

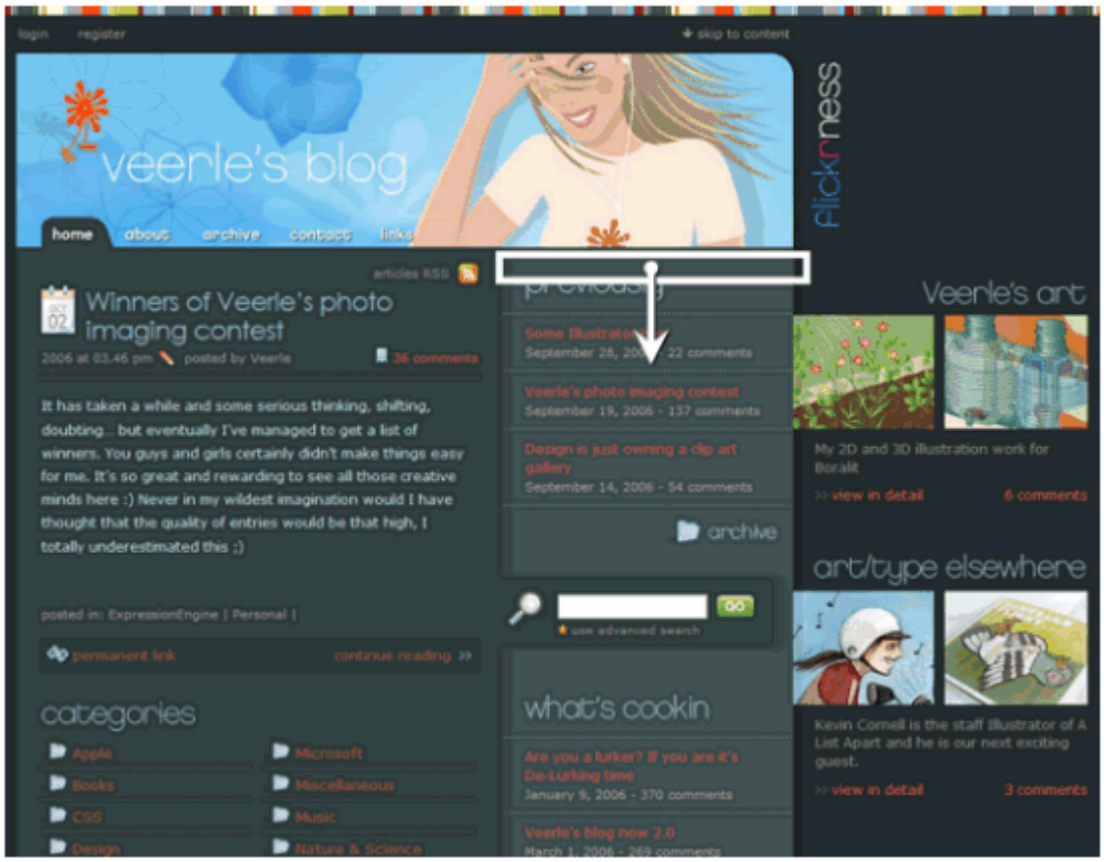


Figura 4.21. Uso de repeat-y en la página de Veerle.duoh.com

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:

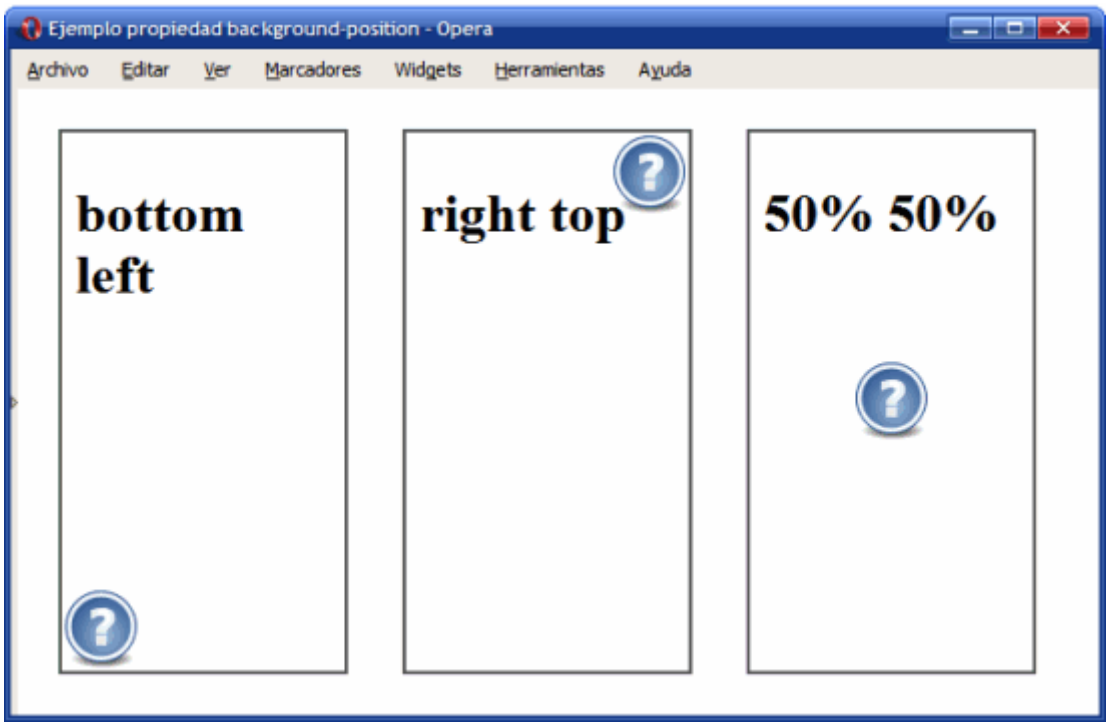


Figura 4.22. Ejemplo de propiedad background-position

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

Capítulo 5. Posicionamiento y visualización

5.1. Tipos de elementos

Los elementos HTML se clasifican en dos grandes grupos: elementos en línea y elementos de bloque.

Los elementos de bloque siempre empiezan en una nueva línea y ocupan todo el espacio disponible hasta el final de la línea. Los elementos en línea sin embargo no empiezan en nueva línea y sólo ocupan el espacio ocupado por sus contenidos.

La siguiente imagen muestra el espacio ocupado por los elementos en línea y los elementos de bloque en un documento HTML:

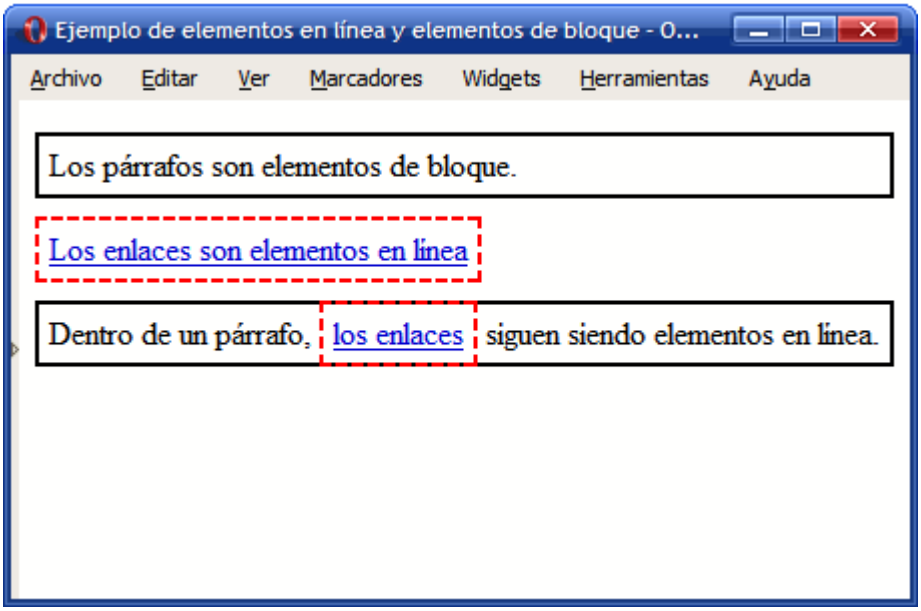


Figura 5.1. Cajas creadas por los elementos de línea y los elementos de bloque

Un elemento de bloque no puede aparecer dentro de un elemento en línea. En cambio, un elemento en línea puede aparecer dentro de un elemento de bloque y dentro de otro elemento en línea.

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

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

Los siguientes elementos también se considera que son de bloque: `dd`, `dt`, `frameset`, `li`, `tbody`, `td`, `tfoot`, `th`, `thead`, `tr`.

Los siguientes elementos pueden ser en línea y de bloque según las circunstancias: `button`, `del`, `iframe`, `ins`, `map`, `object`, `script`.

5.2. Posicionamiento

Cuando los navegadores cargan una página web, muestran sus elementos de la mejor manera posible teniendo en cuenta las limitaciones de espacio y el tipo de cada elemento (de bloque o en línea). Además de esta forma automática, CSS permite controlar de forma precisa el lugar en el que se coloca cada elemento, es decir, el posicionamiento de cada elemento.

CSS define tres mecanismos para posicionar las cajas de cada elemento HTML:

- Normal: posicionamiento por defecto de los elementos en línea y los de bloque. Además, incluye el posicionamiento relativo.
- Float: posicionamiento que consiste en posicionar el elemento según el esquema normal y una vez colocado, desplazarlo todo lo posible hacia la izquierda o hacia la derecha.
- Absoluto: posicionamiento que consiste en extraer por completo el elemento de su posicionamiento normal y colocarlo en la posición indicada respecto de su elemento padre.

La propiedad de CSS que define el posicionamiento del elemento se denomina `position`.

Tabla 5.1. Propiedad position

`position`	**Posicionamiento**
Valores	`static` | `relative` | `absolute` | `fixed` | `inherit`
Se aplica a	Todos los elementos
Valor inicial	static
Descripción	Selecciona el posicionamiento con el que se mostrará el elemento

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

- `static`: es el posicionamiento que se utiliza por defecto y todos los elementos inicialmente se muestran de esta forma. No se tienen en cuenta los valores de las propiedades `top`, `right`, `bottom` y `left`.
- `relative`: la nueva posición del elemento se calcula a partir de la posición que tendría si no se utilizara la propiedad `position`. Las posiciones de los siguientes elementos no se ven afectadas por el desplazamiento de este elemento.
- `absolute`: la nueva posición del elemento se determina mediante las propiedades `top`, `right`, `bottom` y `left`. Los valores de esas propiedades indican la posición del elemento respecto de la posición de su elemento padre. La posición del siguiente elemento se calcula como si no existiera el elemento que se desplaza, ya que este último se desentiende por completo del posicionamiento normal.

- `fixed`: la nueva posición del elemento se calcula de forma idéntica al posicionamiento absoluto. La diferencia reside en que en el caso de `fixed`, el elemento no se mueve cuando se desplaza la ventana del navegador. En los medios visuales (como la pantalla) el elemento se muestra en una posición fija independiente del movimiento de la ventana del navegador. En los medios impresos, el elemento se muestra en todas las páginas.

Las cuatro propiedades relacionadas con la propiedad `position` son las que determinan el desplazamiento de los elementos respecto de sus posiciones originales. CSS define para ello las propiedades `top`, `right`, `bottom` y `left`.

Tabla 5.2. Propiedad top, right, bottom, left

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

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

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

5.3. Posicionamiento normal

El **posicionamiento normal** es la estrategia que utilizan por defecto los navegadores para mostrar los elementos de las páginas.

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

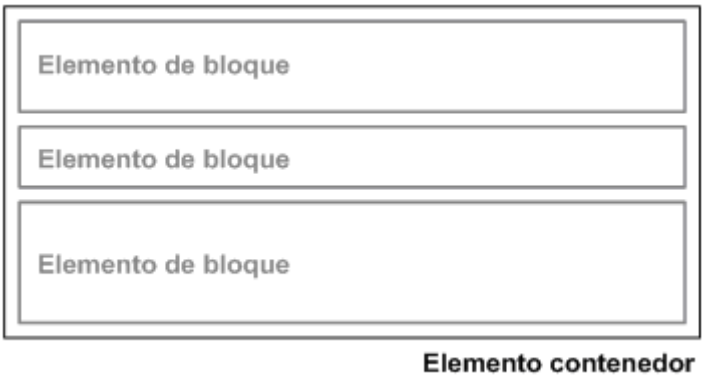


Figura 5.2. Posicionamiento normal de los elementos de bloque

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

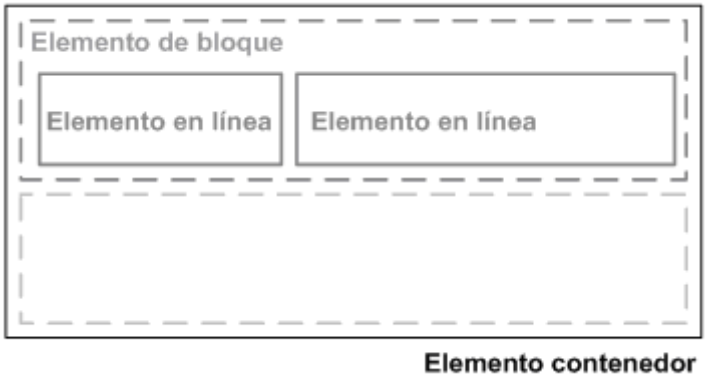


Figura 5.3. Posicionamiento normal de los elementos en línea

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

Por otra parte, el **posicionamiento relativo** también se considera parte del posicionamiento normal. El posicionamiento relativo consiste en desplazar un elemento respecto de su posición original. En este caso, el desplazamiento de una caja no afecta al resto, que se muestran en la misma posición que si la caja desplazada no se hubiera movido de su sitio original.

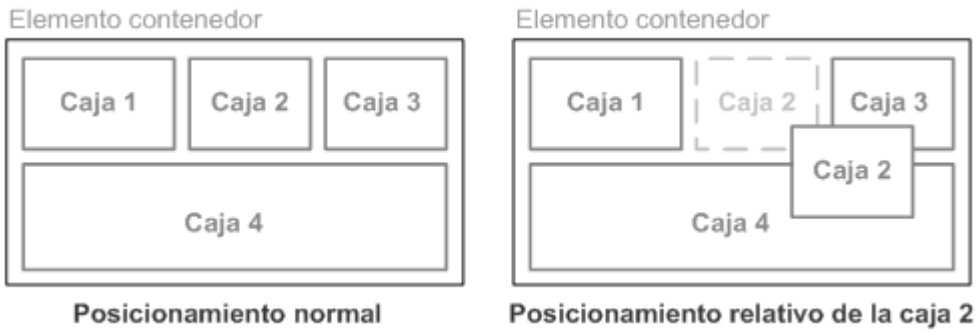


Figura 5.4. Diferencias visuales entre el posicionamiento normal y el posicionamiento relativo

El siguiente ejemplo muestra tres imágenes con el posicionamiento normal definido por CSS:

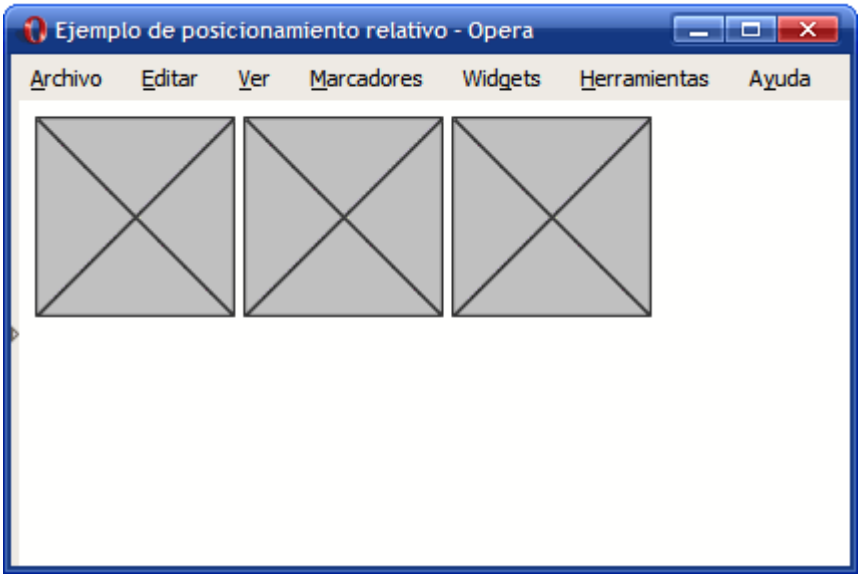


Figura 5.5. Elementos posicionados de forma normal

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

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

<img class="desplazada" src="imagenes/imagen.png" alt="Imagen genérica" />
<img src="imagenes/imagen.png" alt="Imagen genérica" />
<img src="imagenes/imagen.png" alt="Imagen genérica" />
```

El aspecto que muestran ahora las imágenes es el siguiente:

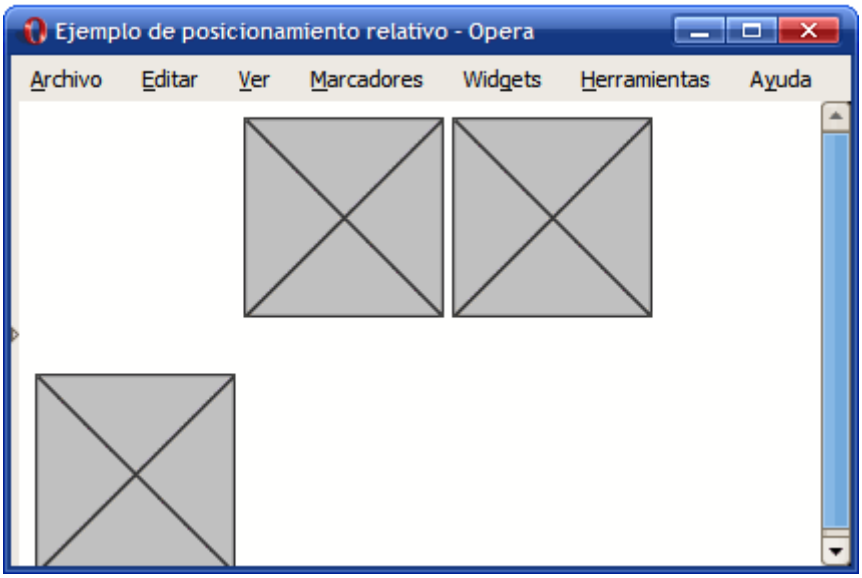


Figura 5.6. Elemento posicionado de forma relativa

El resto de imágenes no varían su posición y no ocupan el hueco dejado por la primera imagen, porque el posicionamiento relativo no influye en el resto de elementos de la página. La única incidencia que tiene el posicionamiento relativo sobre el resto de elementos, es que el elemento desplazado puede solaparse con otros elementos de la página.

5.4. Posicionamiento float

El **posicionamiento float** es uno de los más utilizados, sobre todo al definir la estructura de las páginas, como se verá más adelante. Una caja posicionada mediante la propiedad `float`, se desplaza hasta la zona más a la izquierda o más a la derecha de la línea en la que se debería mostrar si no se desplazara.

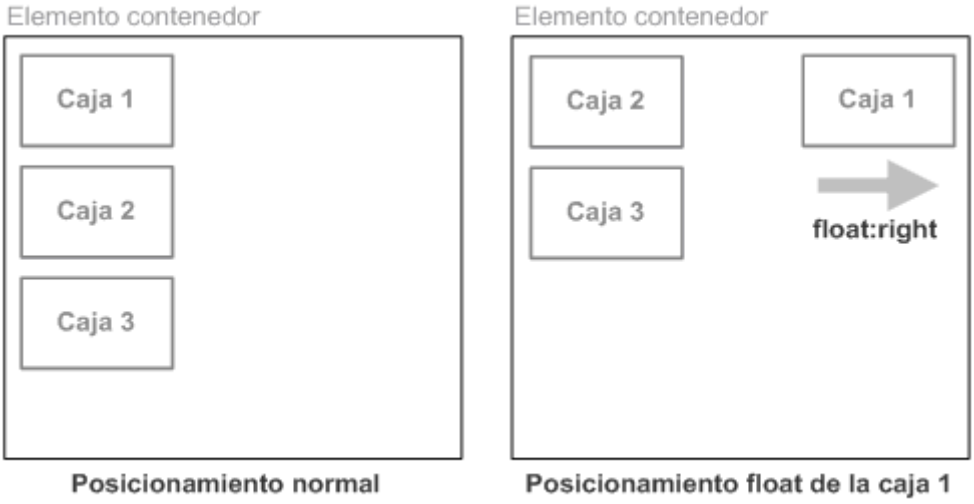


Figura 5.7. Ejemplo de posicionamiento float de una caja

Una caja desplazada mediante `float` no pertenece al posicionamiento normal de un documento, por lo que los elementos de bloque anteriores y posteriores se visualizan como si la caja desplazada no existiera.

Si en el anterior ejemplo la "Caja 1" se posiciona mediante un `float: left` el resultado sería el que se muestra en la imagen:

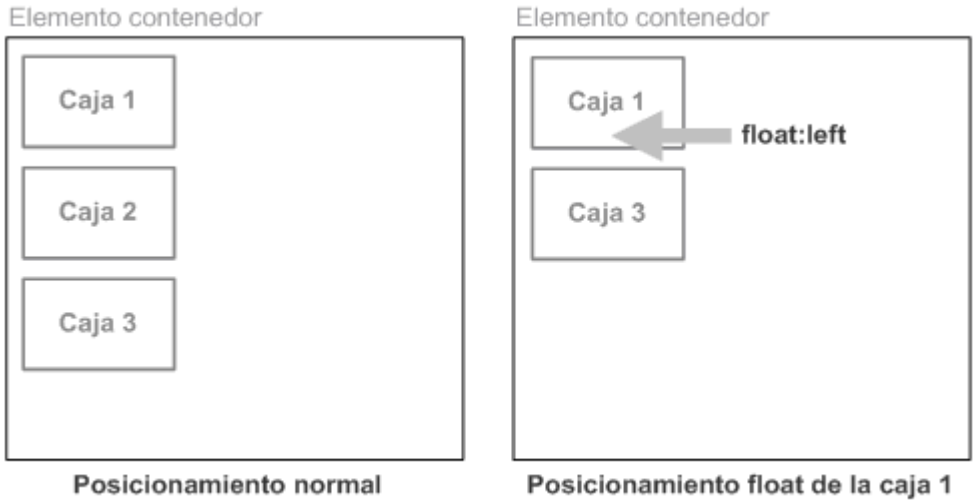


Figura 5.8. Ejemplo de posicionamiento float de una caja

En las imágenes anteriores, el resto de cajas ocupan la posición libre dejada por la "Caja 1" original. Como la "Caja 1" está desplazada a la izquierda, se coloca por encima de la nueva posición que ocupa la "Caja 2", a la que cubre por completo.

Si existen otras cajas desplazadas hacia la izquierda o derecha, la nueva caja desplazada se coloca al lado de las demás cajas. El siguiente ejemplo desplaza las tres cajas:

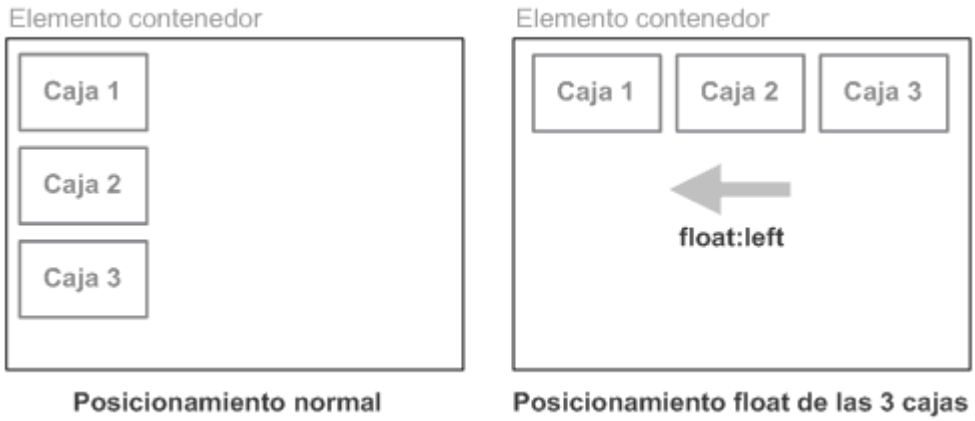


Figura 5.9. Ejemplo de posicionamiento float de varias cajas

Si no existe sitio en la línea actual, la caja pasa a la siguiente línea hasta que encuentra el sitio necesario para mostrarse a la izquierda o derecha de la línea.

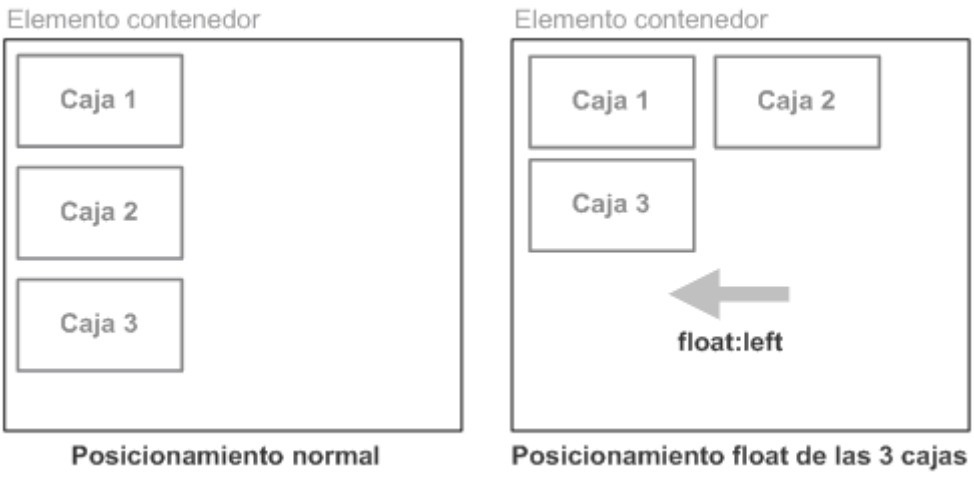


Figura 5.10. Ejemplo de posicionamiento float cuando no existe sitio suficiente

Una caja posicionada mediante `float` influye en la disposición de todas las demás cajas. CSS permite definir si el resto de los elementos fluyen alrededor de la caja desplazada o no lo hacen.

Los elementos en línea que se encuentran al lado de las cajas desplazadas mediante `float` adaptan su anchura al espacio libre dejado por la caja desplazada. Si en la línea donde se encuentra la caja desplazada no existe sitio necesario para los contenidos de los elementos en línea, estos se visualizan en la línea inmediatamente inferior.

Tabla 5.3. Propiedad float

`float`	**Posicionamiento float**
Valores	`left` | `right` | `none` | `inherit`
Se aplica a	Todos los elementos
Valor inicial	none
Descripción	Establece el tipo de posicionamiento float del elemento

Los posibles valores de la propiedad determinan el posicionamiento del elemento y el comportamiento de los elementos adyacentes. Si se indica un valor `left`, el elemento se desplaza hasta el punto más a la izquierda posible en esa misma línea (si no existe sitio en esa línea, el elemento baja una línea y se muestra lo más a la izquierda posible en esa nueva línea). El resto de elementos adyacentes se adaptan y fluyen alrededor del elemento desplazado.

El valor `right` tiene un funcionamiento idéntico, salvo que en este caso el elemento se desplaza hacia la derecha. El valor `none` permite eliminar el posicionamiento y que el elemento se muestre en su posición original.

Ejercicio 6 Ver enunciado en la página 186

Los elementos que se encuentran alrededor de un elemento que ha sido posicionado mediante `float`, adaptan sus contenidos para que fluyan alrededor del elemento posicionado:

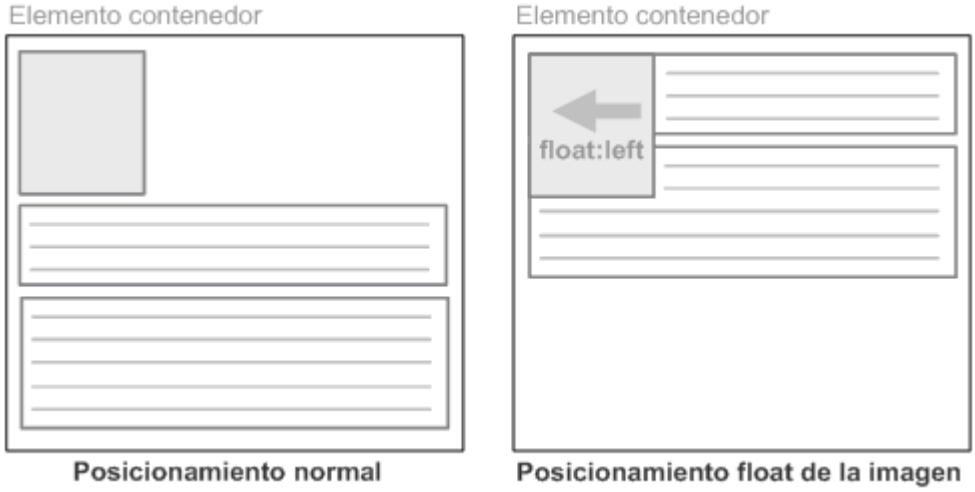


Figura 5.11. Elementos que fluyen alrededor de un elemento posicionado mediante float

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

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

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

La flexibilidad de CSS permite controlar la forma en la que los contenidos fluyen alrededor de los contenidos posicionados mediante `float`. De hecho, en muchas ocasiones es admisible que el texto del primer párrafo fluya alrededor de una imagen, pero el resto de los párrafos deberían mostrarse en su totalidad y no fluyendo alrededor de la imagen:

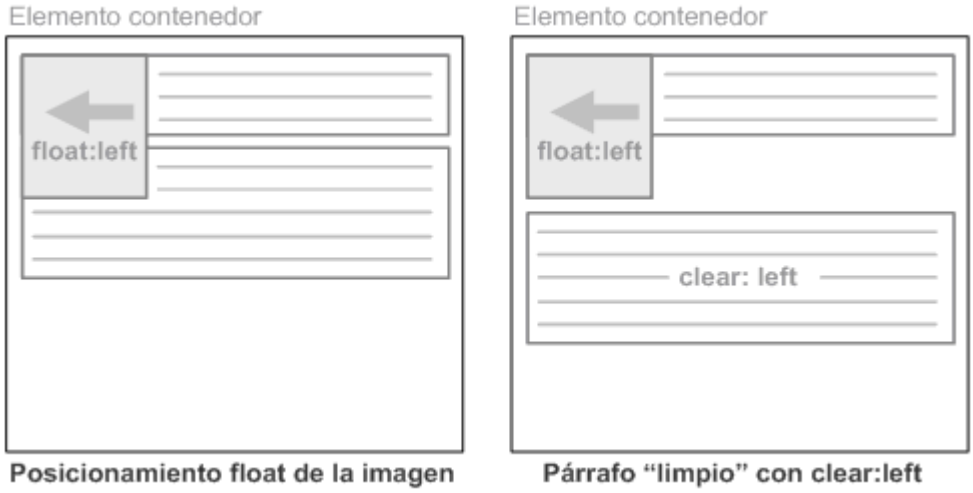


Figura 5.12. Forzando a que un elemento no fluya alrededor de otro elemento posicionado mediante float

La propiedad `clear` permite contrarrestar el comportamiento por defecto de los `float` y permite forzar a un elemento a mostrarse debajo de cualquier elemento posicionado con `float`.

La regla CSS que se aplica al segundo párrafo del ejemplo anterior sería la siguiente:

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

Tabla 5.4. Propiedad clear

`clear`	**Despejar los elementos adyacentes**
Valores	`none` | `left` | `right` | `both` | `inherit`
Se aplica a	Todos los elementos de bloque
Valor inicial	none
Descripción	Permite limpiar el lado derecho, izquierdo o ambos de un elemento, para que no se muestren elementos adyacentes

La propiedad `clear` indica el lado del elemento HTML que no debe ser adyacente a otros elementos desplazados mediante `float`.

Si se indica el valor `left`, el elemento se desplaza de forma descendente hasta que pueda colocarse en una línea en la que no haya ningún elemento en el lado izquierdo. La especificación oficial de CSS explica este comportamiento como *"un desplazamiento descendente hasta que el borde superior del elemento esté por debajo del borde inferior de cualquier elemento desplazado hacia la izquierda"*.

Si se indica el valor `right`, el comportamiento es el mismo, salvo que en este caso se tienen en cuenta los elementos desplazados hacia la derecha.

El valor `both` despeja los lados izquierdo y derecho del elemento, ya que desplaza el elemento de forma descendente hasta que el borde superior se encuentre por debajo de cualquier borde inferior de los elementos desplazados hacia la izquierda y hacia la derecha.

Como se verá más adelante, la propiedad `clear` es imprescindible para la creación de estructuras y layouts de páginas complejas. En el ejercicio anterior, se utilizaba `float` para desplazar los dos elementos:

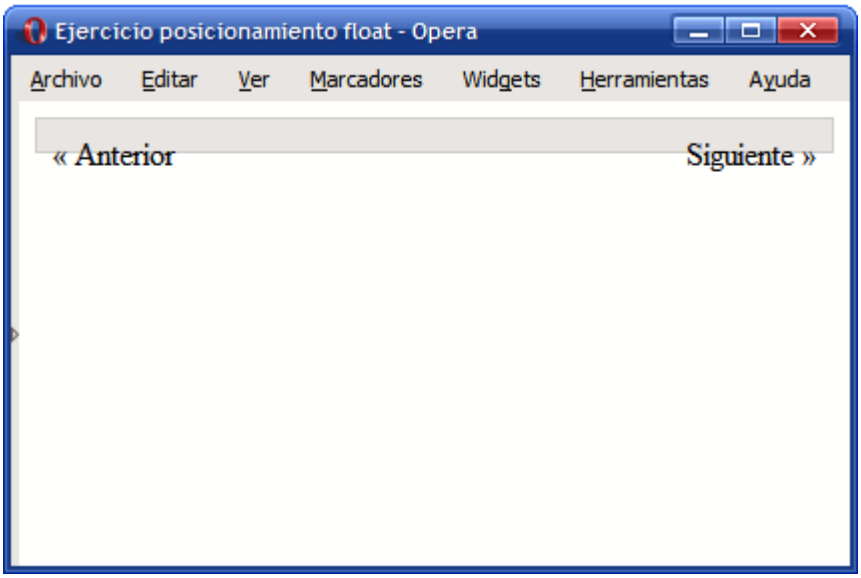


Figura 5.13. Visualización incorrecta de dos elementos posicionados mediante float

Como los dos elementos `<span>` creados dentro del elemento `<div>` se han posicionado mediante `float`, los dos han salido del flujo normal del documento. Así, el elemento `<div>` no tiene contenidos y por eso no llega a cubrir el texto de los dos elementos `<span>`:

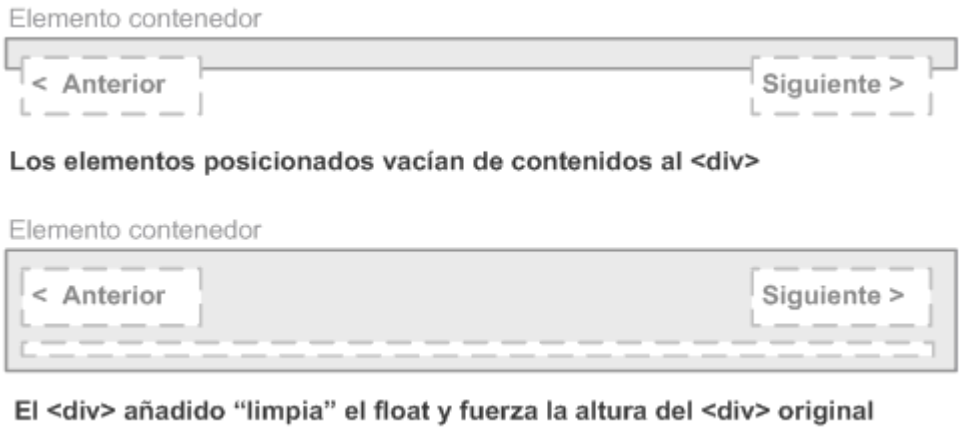


Figura 5.14. Esquema del problema y solución de la visualización incorrecta de 2 elementos posicionados mediante float

La solución consiste en añadir un elemento adicional invisible que *limpie* el `float` forzando a que el `<div>` original cubra completamente los dos elementos `<span>`. El código HTML y CSS final se muestra a continuación:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
<title>Ejercicio posicionamiento float</title>
<style type="text/css">
```

```
div#paginacion {
  border: 1px solid #CCC;
  background-color: #E0E0E0;
  padding: .5em;
}

.derecha {
  float: right;
}

.izquierda {
  float: left;
}

div.clear {
  clear: both;
}
</style>
</head>

<body>
<div id="paginacion">
  <span class="izquierda">&laquo; Anterior</span>
  <span class="derecha">Siguiente &raquo;</span>
  <div class="clear"></div>
</div>
</body>
</html>
```

Al añadir un `<div>` con la propiedad `clear: both`, se tiene la seguridad de que el `<div>` añadido se va a mostrar debajo de cualquier elemento posicionado con `float` y por tanto, se asegura que el `<div>` original tenga la altura necesaria como para encerrar a todos sus contenidos posicionados con `float`.

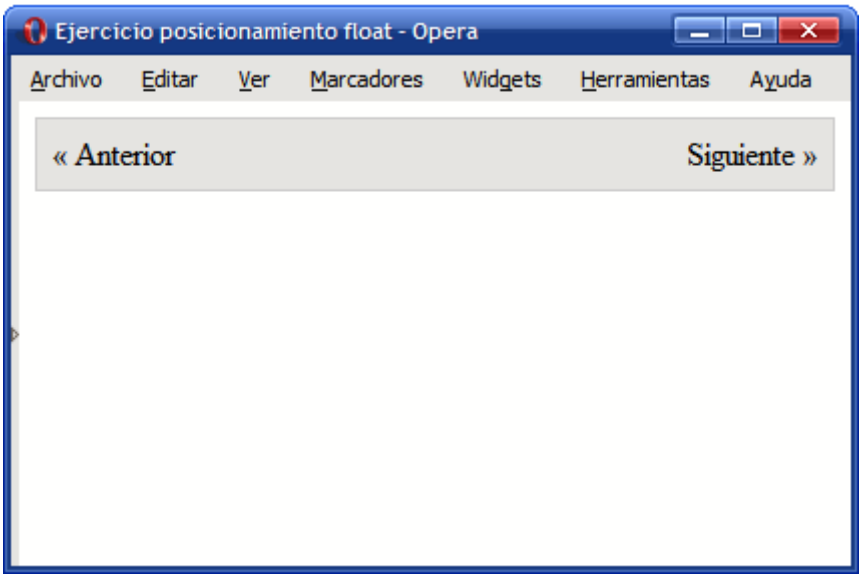


Figura 5.15. Visualización correcta de 2 elementos posicionados mediante float

Además de un elemento `<div>` invisible, también se puede utilizar un `<p>` invisible o un `<hr/>` invisible.

5.5. Posicionamiento absoluto

El **posicionamiento absoluto** implica que el elemento desplazado sale por completo del flujo normal del documento y por tanto, la posición del resto de elementos se determina como si no existiera el elemento desplazado.

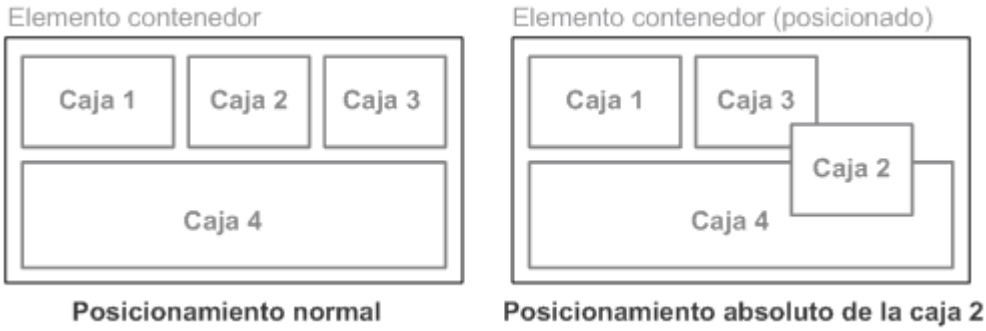


Figura 5.16. Diferencias visuales entre el posicionamiento normal y el posicionamiento absoluto

La referencia que toma el posicionamiento absoluto para determinar la nueva posición de un elemento es el primer elemento padre que esté posicionado. Si ningún elemento padre del elemento posicionado tiene establecido un posicionamiento, la referencia se toma respecto del documento HTML completo.

El siguiente ejemplo muestra las diferencias entre un elemento padre posicionado y otro que no lo está:

Situación original:

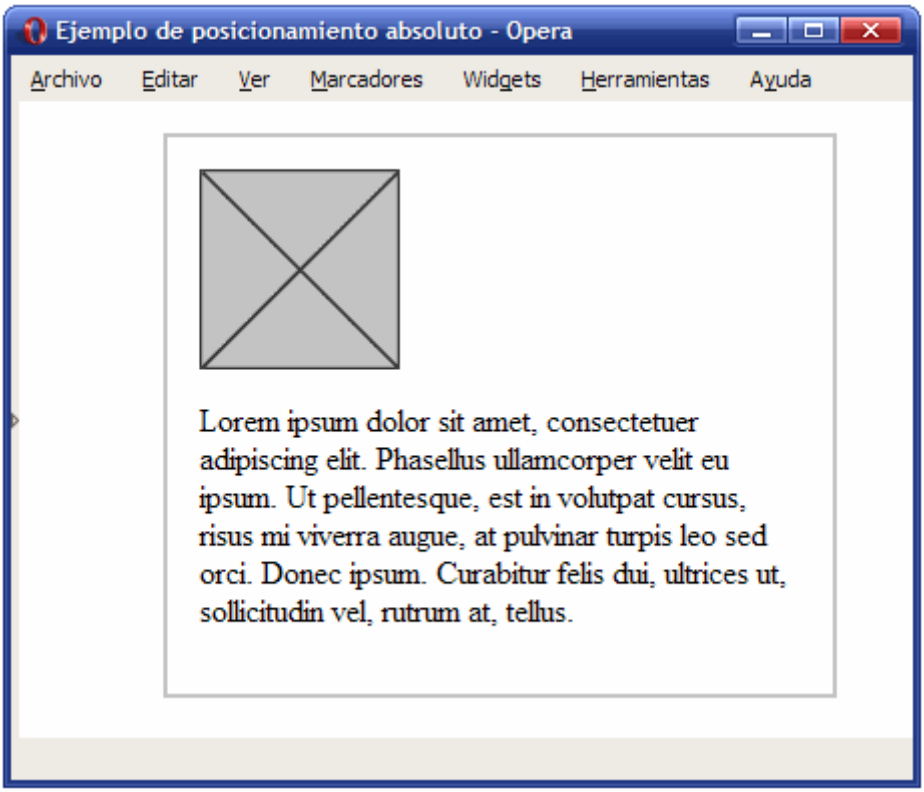


Figura 5.17. Situación original antes de modificar el posicionamiento

Código HTML original:

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

<div>
  <img src="imagenes/imagen.png" alt="Imagen genérica" />
  <p>Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Phasellus
ullamcorper velit eu ipsum. Ut pellentesque, est in volutpat cursus, risus
mi viverra augue, at pulvinar turpis leo sed orci. Donec ipsum. Curabitur
felis dui, ultrices ut, sollicitudin vel, rutrum at, tellus.</p>
</div>
```

A continuación, se posiciona la imagen de forma absoluta y con unos valores `top` y `left` iguales a `3em`:

```
div img {
  position: absolute;
  top: 3em;
  left: 3em;
}
```

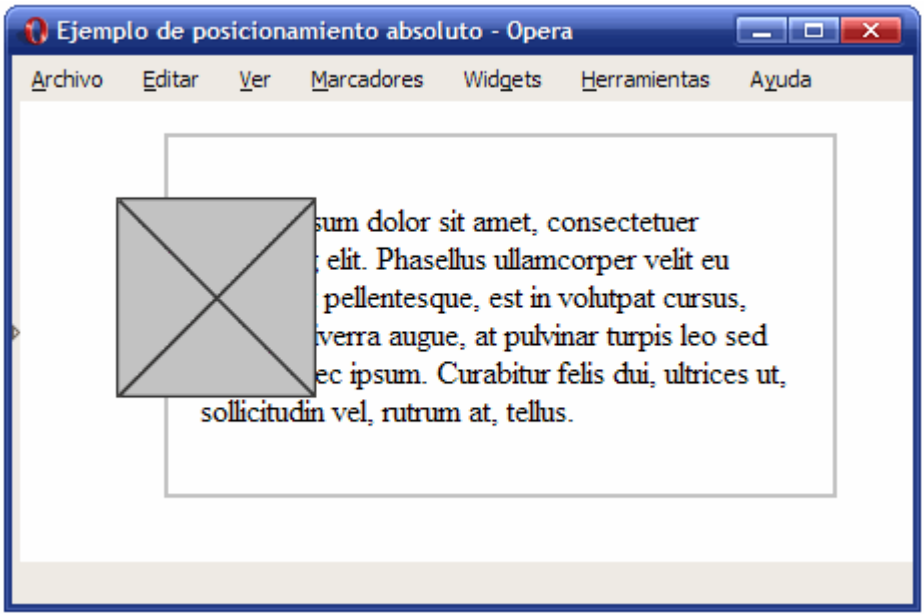


Figura 5.18. Imagen posicionada de forma absoluta

La imagen posicionada absolutamente no toma como origen la esquina superior izquierda del `<div>` en el que se encuentra, sino que su referencia es la esquina superior izquierda de la página:

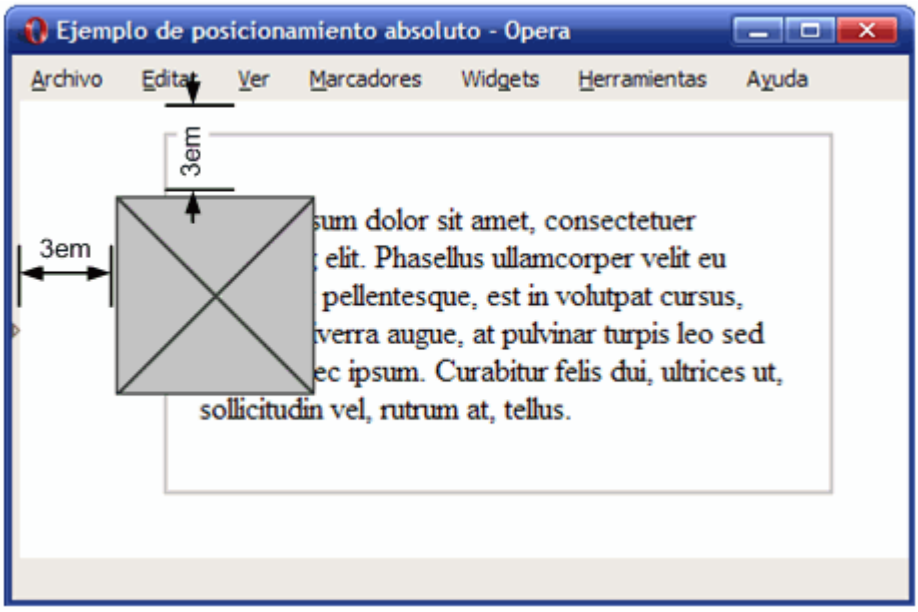


Figura 5.19. La referencia del posicionamiento absoluto es la página entera

Sin embargo, si el elemento contenedor de la imagen (es decir, el elemento `<div>`) se posiciona de forma relativa, la imagen posicionada de forma absoluta varía su posición:

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

```
  position: relative;
}

div img {
  position: absolute;
  top: 3em;
  left: 3em;
}
```

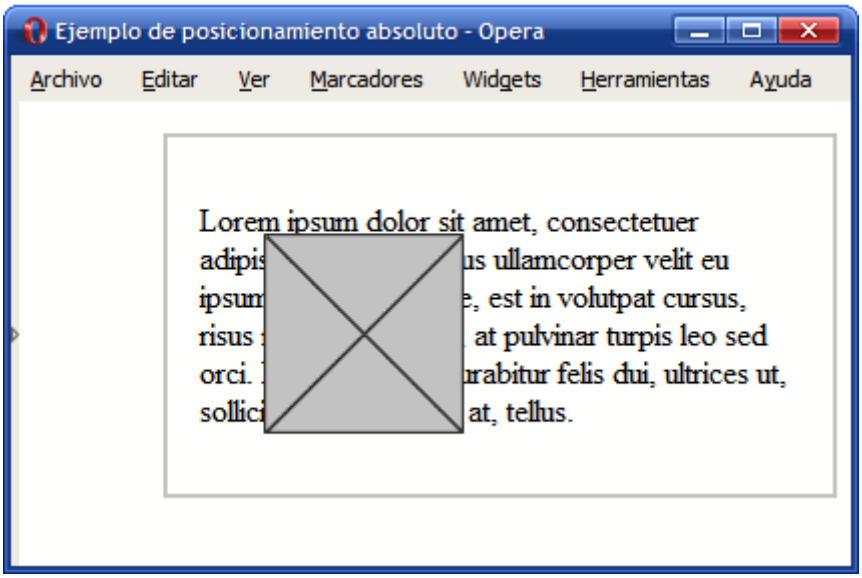


Figura 5.20. Imagen posicionada de forma absoluta

En este caso, la imagen sí que toma correctamente la referencia de su posición desde la esquina superior izquierda el elemento `<div>`:

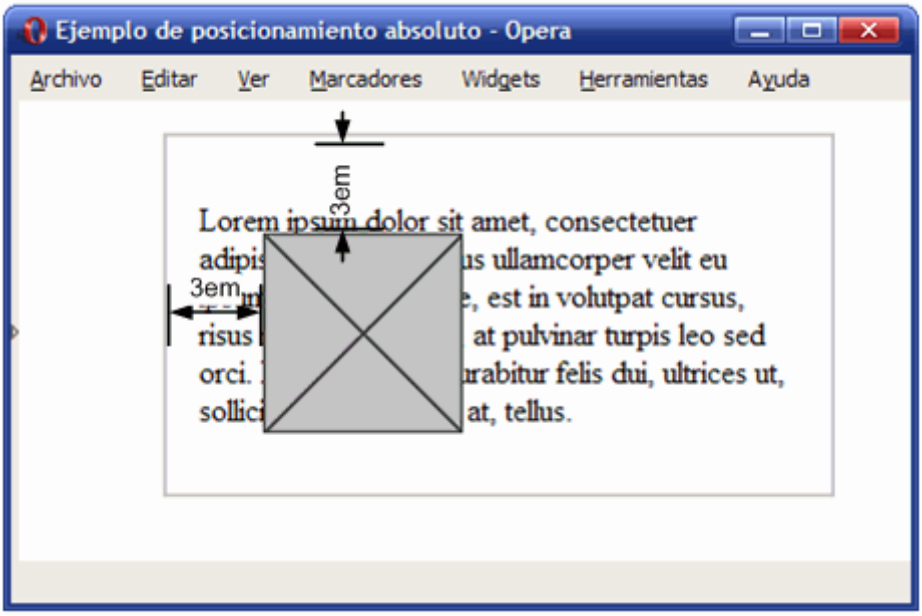


Figura 5.21. La referencia del posicionamiento absoluto es el elemento contenedor de la imagen

Por otra parte, el **posicionamiento fijo** es un caso especial de posicionamiento absoluto. Los elementos que utilizan un posicionamiento fijo se muestran de forma estática en la ventana del navegador y no varían su posición ni aunque se desplace la ventana del navegador o se haga *scroll* sobre ella.

Desafortunadamente, Internet Explorer 6 y las versiones anteriores no soportan el valor `fixed` para la propiedad `position` y se debe simular su comportamiento mediante JavaScript o mediante *hacks* específicos de ese navegador, como se verá más adelante.

5.6. Visualización

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

5.6.1. Display y visibility

Las propiedades `display` y `visibility` controlan la visualización de los elementos. Las dos propiedades permiten hacer invisible cualquier elemento de la página. Habitualmente se utilizan junto con JavaScript para crear efectos y aplicaciones dinámicas (mostrar y ocultar determinados textos, hacer aparecer imágenes, etc.)

La siguiente imagen muestra las diferencias entre ocultar una caja (la número 5) con la propiedad `display` y con la propiedad `visibility`.

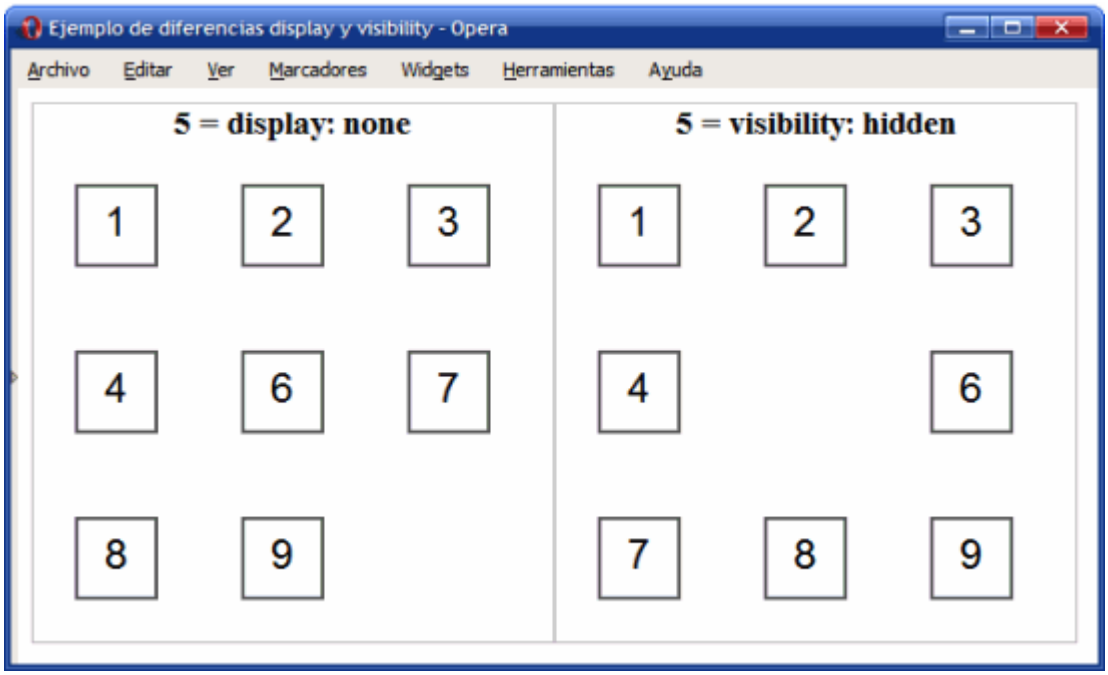


Figura 5.22. Diferencias visuales entre las propiedades display y visibility

La propiedad `display` se utiliza para no visualizar el elemento y para que el resto de elementos se muestren como si el elemento oculto no existiera. La propiedad `visibility` permite mantener la estructura de la página pero sin mostrar el elemento. En este caso, la posición del resto de elementos sí que se mantiene igual que si el elemento no

estuviera oculto. Por este motivo, la propiedad `display` se utiliza mucho más que la propiedad `visibility`.

Tabla 5.5. Propiedad display

`display`	**Visualización de un elemento**
Valores	`inline` | `block` | `none` | `list-item` | `run-in` | `inline-block` | `table` | `inline-table` | `table-row-group` | `table-header-group` | `table-footer-group` | `table-row` | `table-column-group` | `table-column` | `table-cell` | `table-caption` | `inherit`
Se aplica a	Todos los elementos
Valor inicial	inline
Descripción	Permite controlar la forma de visualizar un elemento e incluso ocultarlo

Los valores más utilizados son `inline`, `block` y `none`. El valor `block` permite mostrar un elemento como si fuera un elemento de bloque, independientemente del tipo de elemento que se trate. El valor `inline` permite visualizar un elemento en forma de elemento en línea, independientemente del tipo de elemento que se trate.

El valor `none` permite ocultar un elemento y no mostrarlo en el documento. El resto de elementos se visualizan como si no existiera el elemento oculto, es decir, pueden ocupar el espacio en el que se debería visualizar el elemento.

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

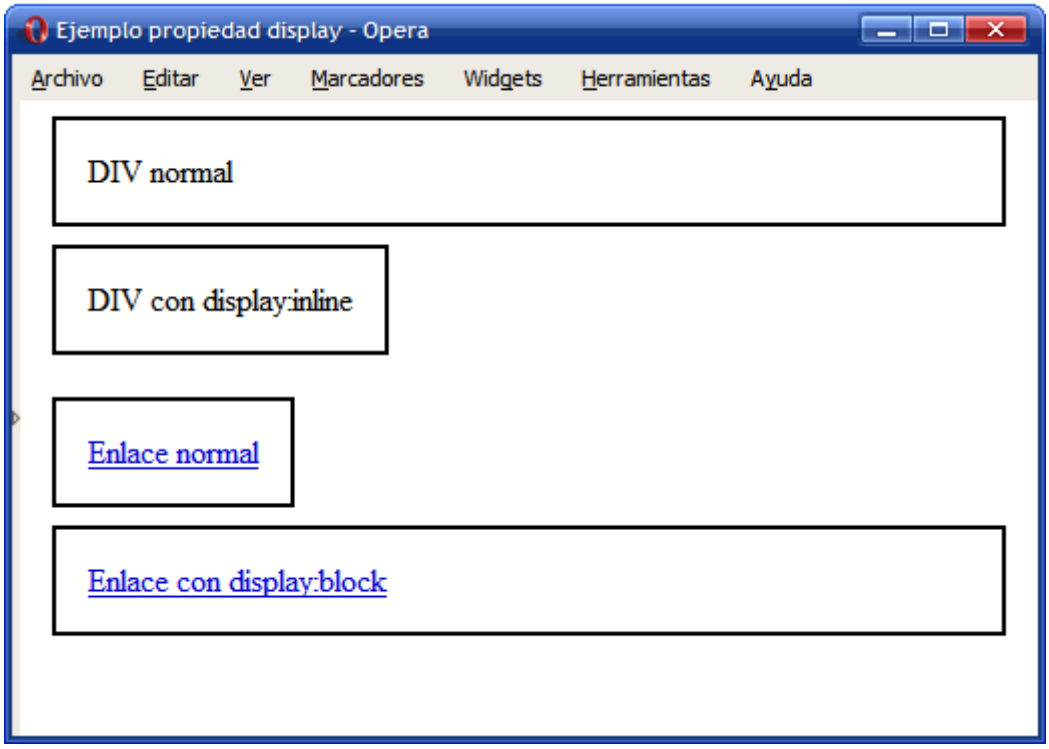


Figura 5.23. Ejemplo de propiedad display

Las reglas CSS del ejemplo anterior son las siguientes:

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

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

La propiedad `display:inline` se utiliza habitualmente con los elementos `<div>` que se quieren posicionar mediante `float`. También se utiliza para las listas (`<ul>`, `<ol>`) que se quieren mostrar horizontalmente. La propiedad `display:block` se emplea frecuentemente para los enlaces que forman el menú de navegación.

Tabla 5.6. Propiedad visibility

`visibility`	**Visibilidad de un elemento**
Valores	`visible` | `hidden` | `collapse` | `inherit`
Se aplica a	Todos los elementos
Valor inicial	visible
Descripción	Permite hacer visibles e invisibles a los elementos

Por defecto, todos los elementos se consideran como visibles (`visibility: visible`). Sin embargo, es posible hacer invisible un elemento asignándole el valor `hidden`. Aunque el elemento no sea visible en pantalla, el resto de elementos se muestran como si el elemento sí que fuera visible, es decir, que en el lugar donde originalmente se mostraba el elemento, ahora se muestra un hueco vacío.

5.6.2. Overflow

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

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

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

Tabla 5.7. Propiedad overflow

`overflow`	**Parte sobrante de un elemento**
Valores	`visible` | `hidden` | `scroll` | `auto` | `inherit`

`overflow`	**Parte sobrante de un elemento**
Se aplica a	Elementos de bloque y celdas de tablas
Valor inicial	visible
Descripción	Permite controlar los contenidos sobrantes de un elemento

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

- `visible`: el contenido no se corta y se muestra sobresaliendo la zona reservada para visualizar el elemento.
- `hidden`: el contenido sobrante se oculta y sólo se visualiza la parte del contenido que cabe dentro de la zona reservada para el elemento.
- `scroll`: solamente se visualiza el contenido que cabe dentro de la zona reservada para el elemento, pero también se muestran barras de *scroll* que permiten visualizar el resto del contenido.
- `auto`: el comportamiento depende del navegador, aunque normalmente es el mismo que la propiedad `scroll`.

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

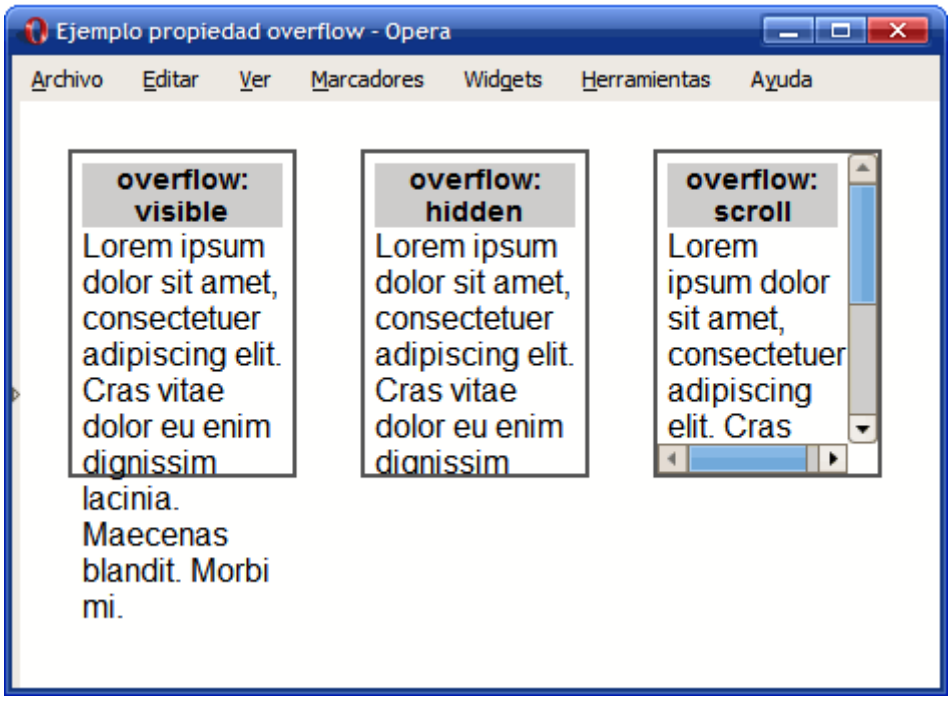


Figura 5.24. Ejemplo de propiedad overflow

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

```
div {
  display: inline;
  float: left;
```

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

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

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

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

El valor `visible` es el valor por defecto que se aplica a los elementos HTML. El valor `hidden` oculta todos los elementos que no caben en la zona reservada para el elemento. Por último, el valor `scroll` solamente muestra los contenidos que caben en la zona reservada pero también muestra barras de *scroll* que permiten visualizar el resto del contenido.

5.6.3. Z-index

Además de la posición horizontal y vertical, CSS permite asignar una posición tridimensional a cada caja del documento. La posición de las cajas se controla mediante un eje adicional llamado Z, por lo que la propiedad se denomina `z-index`.

La propiedad `z-index` permite crear páginas web complejas con varios niveles o capas. El orden de las capas determina los elementos que se mostrarán encima o debajo de otros elementos. Por este motivo, si se utiliza la propiedad `z-index` se pueden producir solapamientos entre los diferentes elementos de la página.

Tabla 5.8. Propiedad z-index

`z-index`	**Orden tridimensional**
Valores	`auto` | `<numero>` | `inherit`
Se aplica a	Elementos que han sido posicionado explícitamente
Valor inicial	auto
Descripción	Establece el nivel tridimensional en el que se muestra el elemento

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

Cuanto más alto sea el valor, más cerca del usuario se mostrará ese nivel. Un elemento con `z-index: 10` se mostrará por delante de otros elementos con `z-index: 8` o `z-index: 9`, pero por detrás de elementos con `z-index: 20` o `z-index: 50`.

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

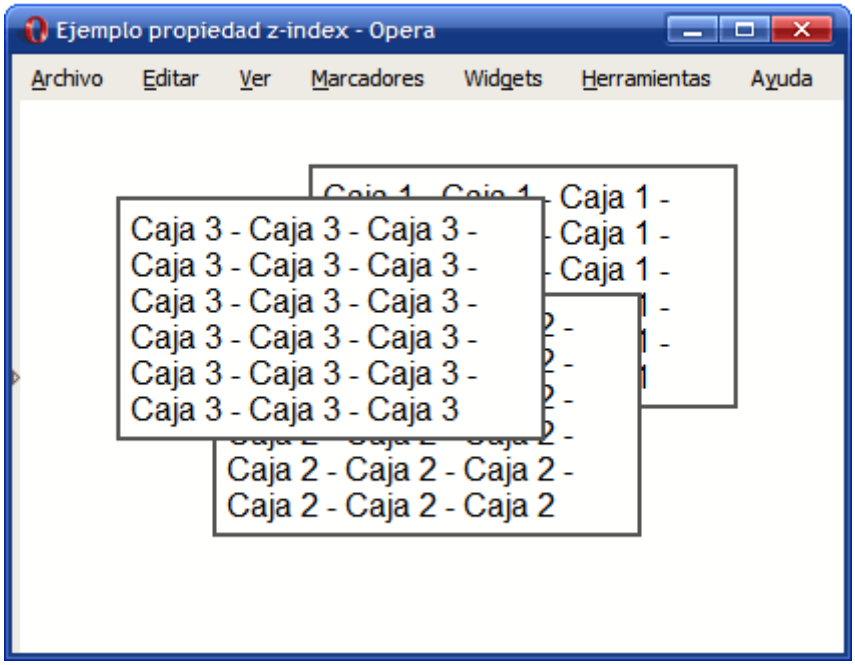


Figura 5.25. Ejemplo de propiedad z-index

El código CSS utilizado se muestra a continuación:

```
div { position: absolute; }
#caja1 { z-index: 5;  top: 1em; left: 8em;}
#caja2 { z-index: 15;  top: 5em; left: 5em;}
#caja3 { z-index: 25;  top: 2em; left: 2em;}

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

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

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

Capítulo 6. Texto

6.1. Tipografía

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

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

Tabla 6.1. Propiedad font-family

font-family	Tipo de letra
Valores	((<nombre_familia> | <familia_generica>) (,<nombre_familia> | <familia_generica>)*) | inherit
Se aplica a	Todos los elementos
Valor inicial	Depende del navegador
Descripción	Establece el tipo de letra utilizado para el texto

El tipo de letra del texto se puede indicar de dos formas diferentes:

- Mediante el nombre de una *familia* tipográfica: en otras palabras, mediante el nombre del tipo de letra, como por ejemplo "Arial", "Verdana", "Garamond", etc.
- Mediante el nombre genérico de una *familia* tipográfica: los nombres genéricos no se refieren a ninguna fuente en concreto, sino que hacen referencia al estilo del tipo de letra. Las familias genéricas definidas son `serif` (tipo de letra similar a *Times New Roman*), `sans-serif` (tipo *Arial*), `cursive` (tipo *Comic Sans*), `fantasy` (tipo *Impact*) y `monospace` (tipo *Courier New*).

Los navegadores muestran el texto de las páginas web utilizando los tipos de letra instalados en el ordenador del propio usuario. De esta forma, si el diseñador indica en la propiedad `font-family` que el texto debe mostrarse con un tipo de letra especialmente raro o rebuscado, casi ningún usuario dispondrá de ese tipo de letra.

Para evitar el problema común de que el usuario no tenga instalada la fuente que quiere utilizar el diseñador, CSS permite indicar en la propiedad `font-family` más de un tipo de letra. El navegador probará en primer lugar con el primer tipo de letra indicado. Si el usuario la tiene instalada, el texto se muestra con ese tipo de letra.

Si el usuario no dispone del primer tipo de letra indicado, el navegador irá probando con el resto de tipos de letra hasta que encuentre alguna fuente que esté instalada en el ordenador del usuario. Evidentemente, el diseñador no puede indicar para cada propiedad `font-family` tantos tipos de letra como posibles fuentes parecidas existan.

Para solucionar este problema se utilizan las familias tipográficas genéricas. Cuando la propiedad `font-family` toma un valor igual a `serif`, el diseñador no indica al navegador que debe utilizar la fuente Arial, sino que debe utilizar *"la fuente que más se parezca a Arial de todas las que tiene instaladas el usuario"*.

Por todo ello, el valor de `font-family` suele definirse como una lista de tipos de letra alternativos separados por comas. El último valor de la lista es el nombre de la familia tipográfica genérica que más se parece al tipo de letra que se quiere utilizar.

Las listas de tipos de letra más utilizadas son las siguientes:

```
font-family: Arial, Helvetica, sans-serif;
font-family: "Times New Roman", Times, serif;
font-family: "Courier New", Courier, monospace;
font-family: Georgia, "Times New Roman", Times, serif;
font-family: Verdana, Arial, Helvetica, sans-serif;
```

Ya que las fuentes que se utilizan en la página deben estar instaladas en el ordenador del usuario, cuando se quiere disponer de un diseño complejo con fuentes muy especiales, se debe recurrir a soluciones alternativas.

La solución más sencilla consiste en crear imágenes en las que se muestra el texto con la fuente deseada. Esta técnica solamente es viable para textos cortos (por ejemplo los titulares de una página) y puede ser manual (creando las imágenes una por una) o automática, utilizando JavaScript, PHP y/o CSS.

Otra alternativa es la de la sustitución automática de texto basada en Flash. La técnica más conocida es la de sIFR, de la que se puede encontrar más información en `http://wiki.novemberborn.net/sifr`

Una vez seleccionado el tipo de letra, se puede modificar su tamaño mediante la propiedad `font-size`.

Tabla 6.2. Propiedad font-size

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

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

- `tamaño_absoluto`: indica el tamaño de letra de forma absoluta mediante alguna de las siguientes palabras clave: `xx-small`, `x-small`, `small`, `medium`, `large`, `x-large`, `xx-large`.

- `tamaño_relativo`: indica de forma relativa el tamaño de letra del texto mediante dos palabras clave (`larger`, `smaller`) que toman como referencia el tamaño de letra del elemento padre.

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

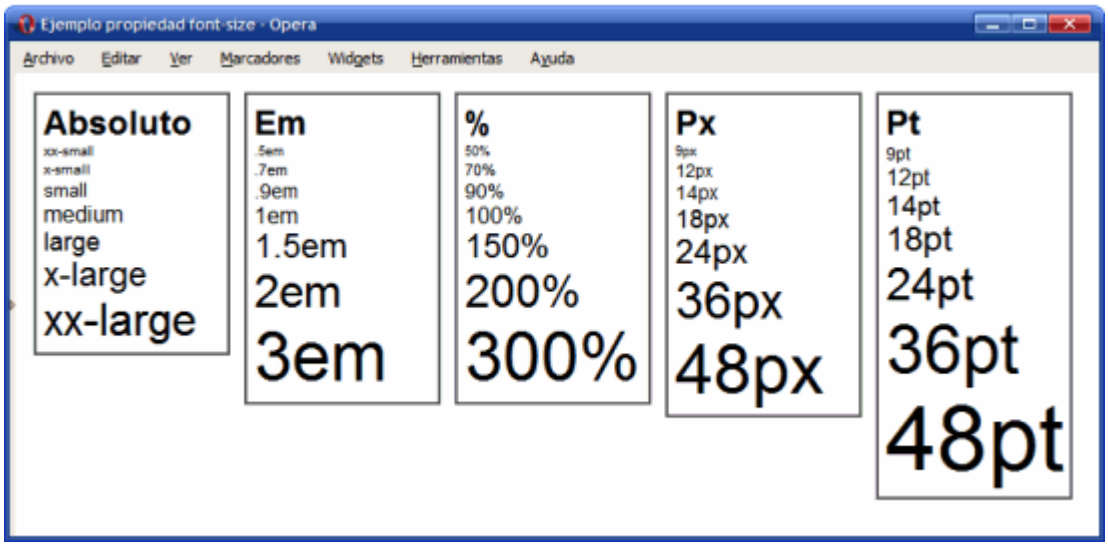


Figura 6.1. Comparación visual de las distintas unidades para indicar el tamaño del texto

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

Por defecto los navegadores asignan los siguientes tamaños a los títulos de sección: `<h1>` = `xx-large`, `<h2>` = `x-large`, `<h3>` = `large`, `<h4>` = `medium`, `<h5>` = `small`, `<h6>` = `xx-small`.

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

Tabla 6.3. Propiedad font-weight

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

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

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

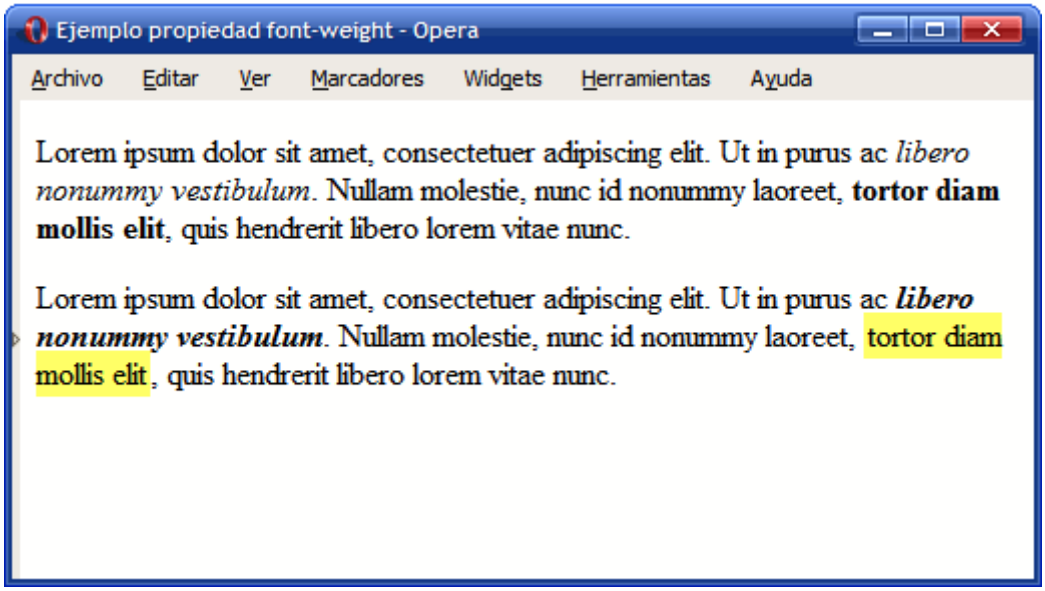


Figura 6.2. Ejemplo de propiedad font-weight

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

Las reglas CSS del ejemplo anterior se muestran a continuación:

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

<p>Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut in
purus ac <em>libero nonummy vestibulum</em>. Nullam molestie, nunc id
nonummy laoreet, <strong>tortor diam mollis elit</strong>, quis hendrerit
libero lorem vitae nunc.</p>

<p id="especial">Lorem ipsum dolor sit amet, consectetuer adipiscing elit.
Ut in purus ac <em>libero nonummy vestibulum</em>. Nullam molestie, nunc id
nonummy laoreet, <strong>tortor diam mollis elit</strong>, quis hendrerit
libero lorem vitae nunc.</p>
```

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

Tabla 6.4. Propiedad font-style

font-style	Estilo de la letra
Valores	normal | italic | oblique | inherit
Se aplica a	Todos los elementos
Valor inicial	normal
Descripción	Establece el estilo de la letra utilizada para el texto

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

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

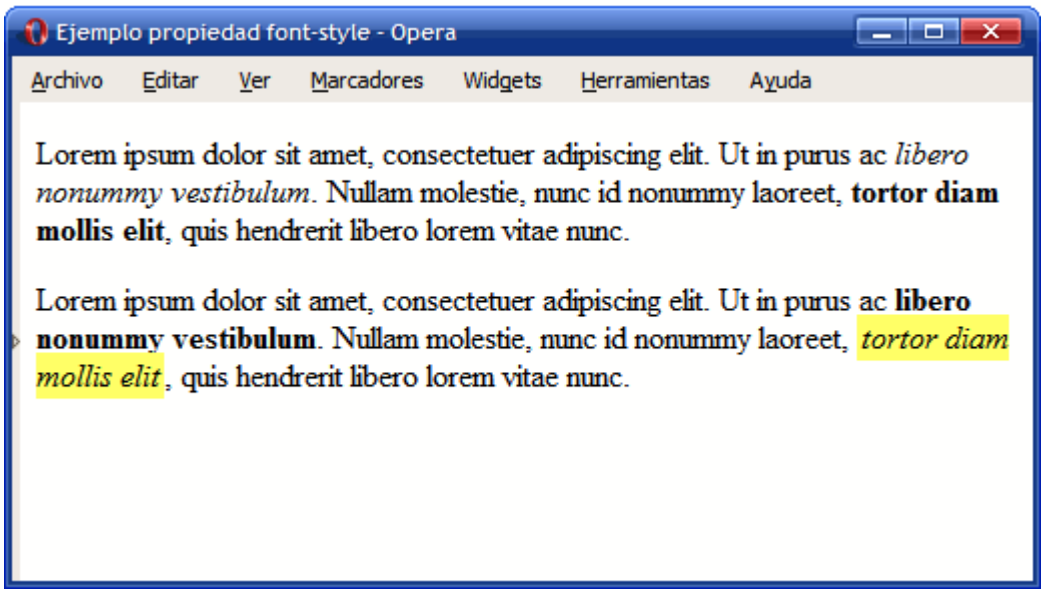


Figura 6.3. Ejemplo de propiedad font-style

Ahora, el texto del elemento se muestra como un texto en negrita y el texto del elemento se muestra como un texto en cursiva con un color de fondo muy destacado.

El único cambio necesario en las reglas CSS anteriores es el de añadir la propiedad font-style:

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

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

Tabla 6.5. Propiedad font-variant

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

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

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

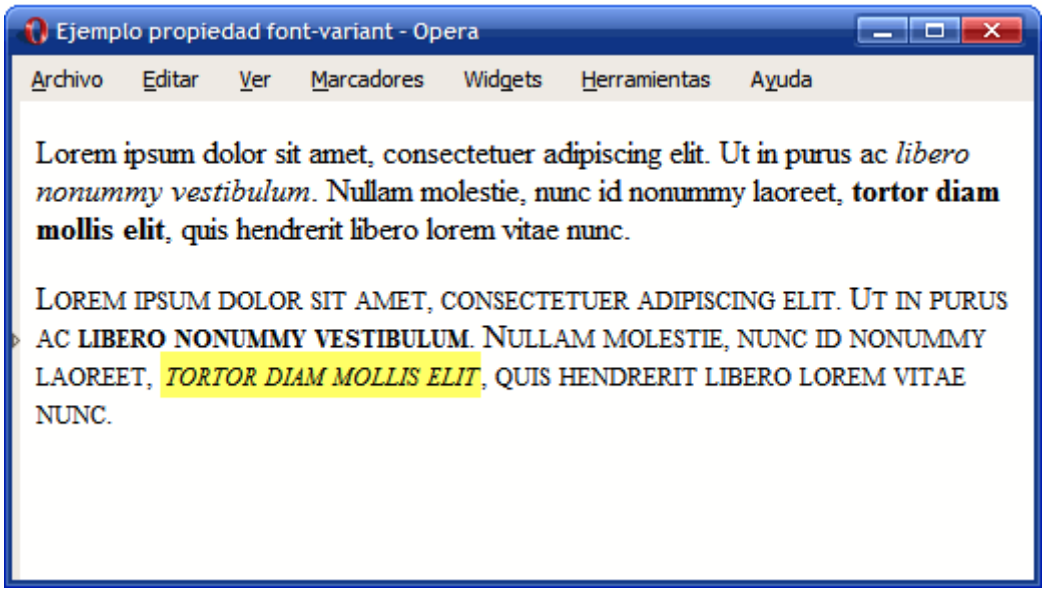


Figura 6.4. Ejemplo de propiedad font-variant

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

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

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

Tabla 6.6. Propiedad font

font	Tipografía
Valores	((<font-style> || <font-variant> || <font-weight>)? <font-size> (/ <line-height>)? <font-family>) | caption | icon | menu | message-box | small-caption | status-bar | inherit
Se aplica a	Todos los elementos
Valor inicial	-
Descripción	Permite indicar de forma directa todas las propiedades de la tipografía de un texto

El orden en el que se deben indicar las propiedades del texto es el siguiente:

- En primer lugar y de forma opcional se indican el `font-style`, `font-variant` y `font-weight` en cualquier orden.
- A continuación, se indica el valor de `font-size` seguido opcionalmente por el valor de `line-height`.
- Por último, se indica el tipo de letra a utilizar.

Ejemplos de uso de la propiedad `font`:

```
font: 76%/140% Verdana,Arial,Helvetica,sans-serif;
font: normal 24px/26px "Century Gothic","Trebuchet MS",Arial,Helvetica,sans-serif;
font: normal .94em "Trebuchet MS",Arial,Helvetica,sans-serif;
font: bold 1em "Trebuchet MS",Arial,Sans-Serif;
font: normal 0.9em "Lucida Grande", Verdana, Arial, Helvetica, sans-serif;
font: normal 1.2em/1em helvetica, arial, sans-serif;
font: 11px verdana, sans-serif;
font: normal 1.4em/1.6em "helvetica", arial, sans-serif;
font: bold 14px georgia, times, serif;
```

Aunque su uso no es muy común, la propiedad `font` permite indicar el tipo de letra a utilizar mediante una serie de palabras clave: `caption`, `icon`, `menu`, `message-box`, `small-caption`, `status-bar`.

Si por ejemplo se utiliza la palabra `status-bar`, el navegador mostrará el texto con el mismo tipo de letra que la que utiliza el sistema operativo para mostrar los textos de la barra de estado de las ventanas. La palabra `icon` se utilizaría para mostrar el texto con el mismo tipo de letra que utiliza el sistema operativo para mostrar el nombre de los iconos y así sucesivamente.

Ejercicio 7 Ver enunciado en la página 187

6.2. Texto

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

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

Tabla 6.7. Propiedad text-align

`text-align`	**Alineación del texto**
Valores	`left` | `right` | `center` | `justify` | `inherit`
Se aplica a	Elementos de bloque y celdas de tabla
Valor inicial	left
Descripción	Establece la alineación del contenido del elemento

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

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

Figura 6.5. Ejemplo de propiedad text-align

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

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

Tabla 6.8. Propiedad line-height

`line-height`	**Interlineado**
Valores	`normal` | `<numero>` | `<medida>` | `<porcentaje>` | `inherit`
Se aplica a	Todos los elementos
Valor inicial	normal
Descripción	Permite establecer la altura de línea de los elementos

Además de todas las unidades de medida y el uso de porcentajes, la propiedad `line-height` permite indicar un número sin unidades que se interpreta como el múltiplo del tamaño normal de la letra. Por tanto, estas tres reglas CSS son equivalentes:

```
p { line-height: 1.2; font-size: 1em }
p { line-height: 1.2em; font-size: 1em }
p { line-height: 120%; font-size: 1em }
```

Siempre que se utilice de forma moderada, el interlineado mejora notablemente la legibilidad de un texto, como se puede observar en la siguiente imagen:

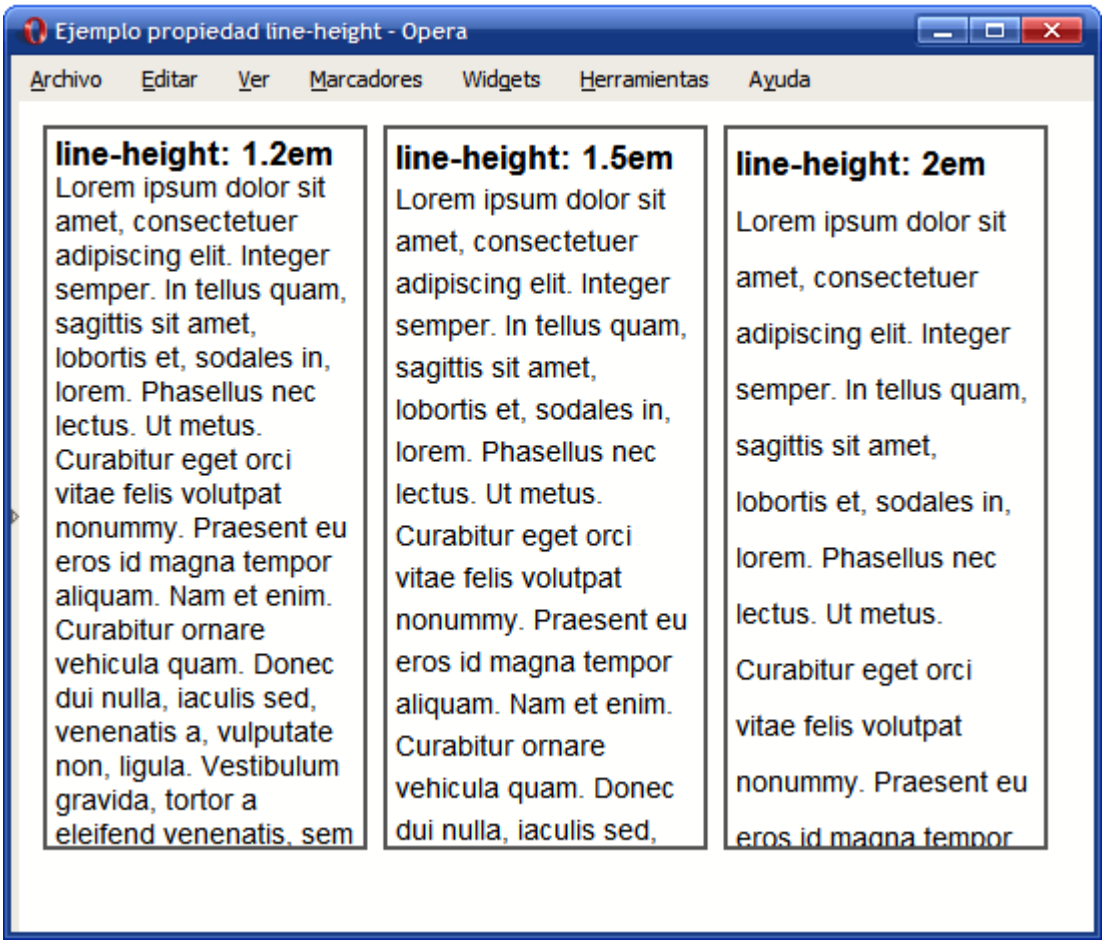


Figura 6.6. Ejemplo de propiedad line-height

Además de la decoración que se puede aplicar a la tipografía que utilizan los textos, CSS define otros estilos y decoraciones para el texto en su conjunto. La propiedad que decora el texto se denomina `text-decoration`.

Tabla 6.9. Propiedad text-decoration

`text-decoration`	**Decoración del texto**
Valores	`none | ( underline || overline || line-through || blink ) | inherit`
Se aplica a	Todos los elementos

`text-decoration`	**Decoración del texto**
Valor inicial	none
Descripción	Establece la decoración del texto (subrayado, tachado, parpadeante, etc.)

El valor `underline` subraya el texto, por lo que puede confundir a los usuarios haciéndoles creer que se trata de un enlace. El valor `overline` añade una línea en la parte superior del texto, un aspecto que raramente es deseable. El valor `line-through` muestra el texto tachado con una línea continua, por lo que su uso tampoco es muy habitual. Por último, el valor `blink` muestra el texto parpadeante y se recomienda evitar su uso por las molestias que genera a la mayoría de usuarios.

Una de las propiedades de CSS más desconocidas y que puede ser de gran utilidad en algunas circunstancias es la propiedad `text-transform`, que puede variar de forma sustancial el aspecto del texto.

Tabla 6.10. Propiedad text-transform

`text-transform`	**Transformación del texto**
Valores	`capitalize` | `uppercase` | `lowercase` | `none` | `inherit`
Se aplica a	Todos los elementos
Valor inicial	none
Descripción	Transforma el texto original (lo transforma a mayúsculas, a minúsculas, etc.)

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

La siguiente imagen muestra cada uno de los posibles valores:

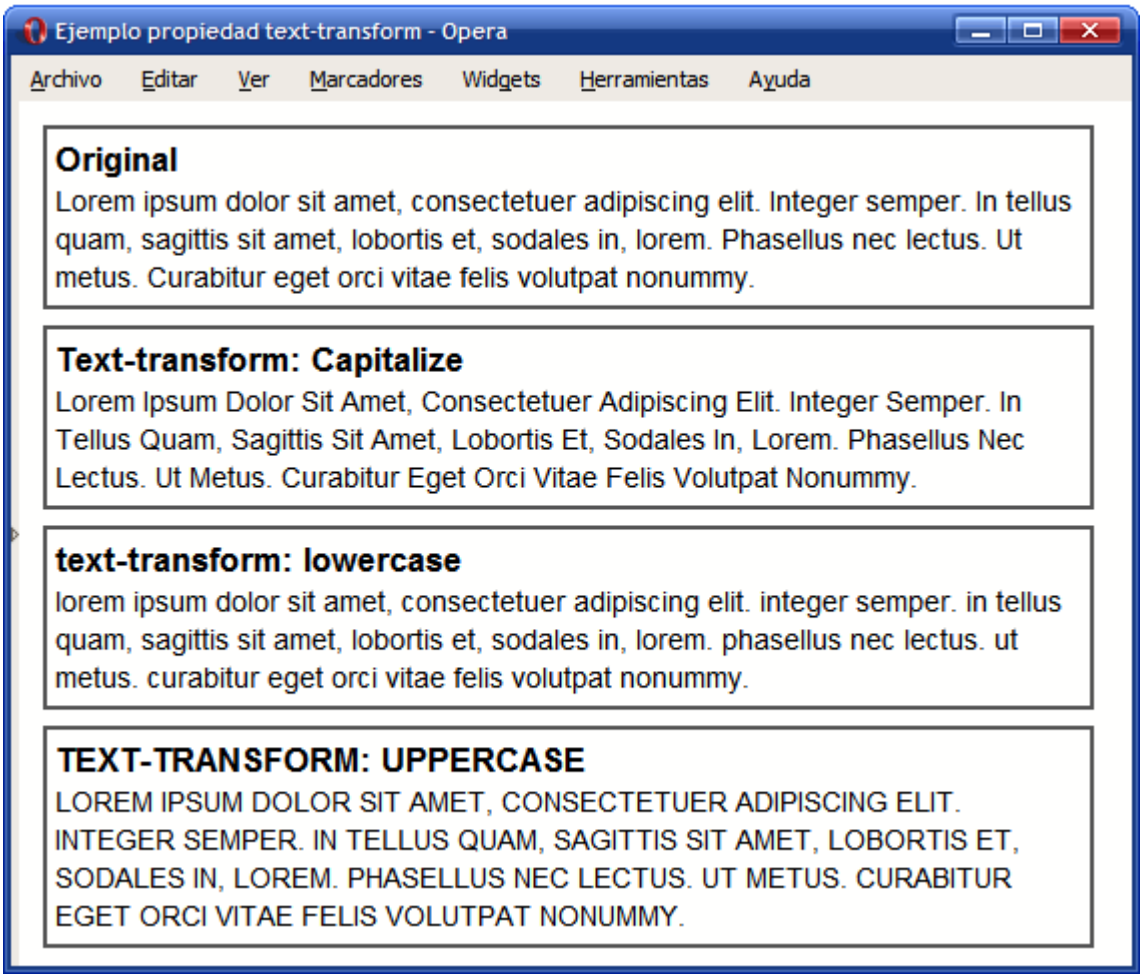


Figura 6.7. Ejemplo de propiedad text-transform

Las reglas CSS del ejemplo anterior se muestran a continuación:

```
<div style="text-transform: none"><h1>Original</h1>Lorem ipsum dolor
sit amet, consectetuer adipiscing elit. Integer semper. In tellus quam,
sagittis sit amet, lobortis et, sodales in, lorem. Phasellus nec lectus.
Ut metus. Curabitur eget orci vitae felis volutpat nonummy.</div>

<div style="text-transform: capitalize"><h1>text-transform: capitalize</h1>
Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Integer semper.
In tellus quam, sagittis sit amet, lobortis et, sodales in, lorem. Phasellus
nec lectus. Ut metus. Curabitur eget orci vitae felis volutpat nonummy.</div>

<div style="text-transform: lowercase"><h1>text-transform: lowercase</h1>
Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Integer semper.
In tellus quam, sagittis sit amet, lobortis et, sodales in, lorem. Phasellus
nec lectus. Ut metus. Curabitur eget orci vitae felis volutpat nonummy.</div>

<div style="text-transform: uppercase"><h1>text-transform: uppercase</h1>
Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Integer semper.
In tellus quam, sagittis sit amet, lobortis et, sodales in, lorem. Phasellus
nec lectus. Ut metus. Curabitur eget orci vitae felis volutpat nonummy.</div>
```

Uno de los principales problemas del diseño de documentos y páginas mediante CSS consiste en la alineación vertical en una misma línea de varios elementos diferentes como imágenes y texto. Para controlar esta alineación, CSS define la propiedad `vertical-align`.

Tabla 6.11. Propiedad vertical-align

`vertical-align`	**Alineación vertical**
Valores	`baseline` | `sub` | `super` | `top` | `text-top` | `middle` | `bottom` | `text-bottom` | `<porcentaje>` | `<medida>` | `inherit`
Se aplica a	Elementos en línea y celdas de tabla
Valor inicial	baseline
Descripción	Determina la alineación vertical de los contenidos de un elemento

A continuación se muestra una imagen con el aspecto que muestran los navegadores para cada uno de los posibles valores de la propiedad `vertical-align`:

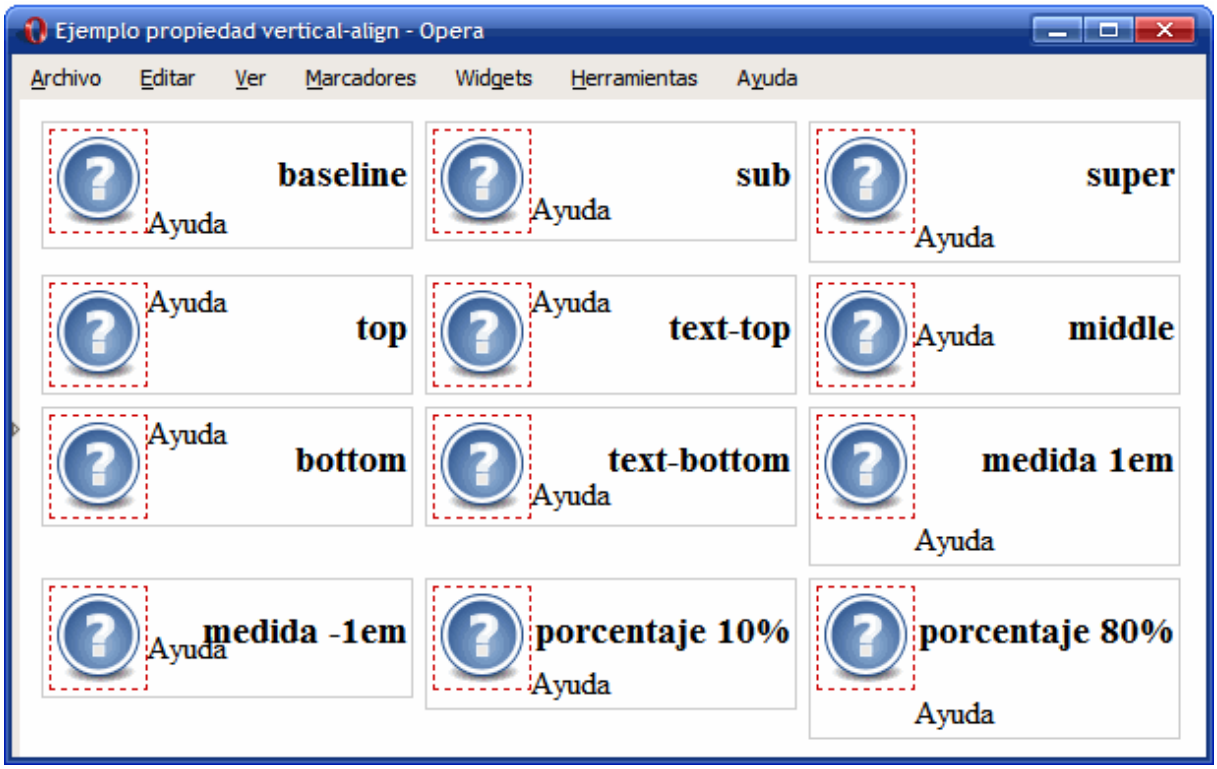


Figura 6.8. Ejemplo de propiedad vertical-align

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

```
img {border:1px dashed #CC0000; padding: 1px;}
div#baseline img { vertical-align:baseline; }
div#sub img { vertical-align:sub; }
div#super img { vertical-align:super; }
div#top img { vertical-align:top; }
div#text-top img { vertical-align:text-top; }
```

```
div#middle img { vertical-align:middle; }
div#bottom img { vertical-align:bottom; }
div#text-bottom img { vertical-align:text-bottom; }
div#medida_1em img { vertical-align:1em; }
div#medida_-1em img { vertical-align:-1em; }
div#porcentaje_10 img { vertical-align:10%; }
div#porcentaje_80 img { vertical-align:80%; }

<div id="baseline">
  <h1>baseline</h1>
  <img src="images/help.png" alt="Ayuda" />
  <span>Ayuda</span>
</div>

<div id="sub">
  <h1>sub</h1>
  <img src="images/help.png" alt="Ayuda" />
  <span>Ayuda</span>
</div>

<div id="super">
  <h1>super</h1>
  <img src="images/help.png" alt="Ayuda" />
  <span>Ayuda</span>
</div>

<div id="top">
  <h1>top</h1>
  <img src="images/help.png" alt="Ayuda" />
  <span>Ayuda</span>
</div>

<div id="text-top">
  <h1>text-top</h1>
  <img src="images/help.png" alt="Ayuda" />
  <span>Ayuda</span>
</div>

<div id="middle">
  <h1>middle</h1>
  <img src="images/help.png" alt="Ayuda" />
  <span>Ayuda</span>
</div>

<div id="bottom">
  <h1>bottom</h1>
  <img src="images/help.png" alt="Ayuda" />
  <span>Ayuda</span>
</div>

<div id="text-bottom">
  <h1>text-bottom</h1>
  <img src="images/help.png" alt="Ayuda" />
  <span>Ayuda</span>
</div>
```

```
<div id="medida_1em">
  <h1>medida 1em</h1>
  <img src="images/help.png" alt="Ayuda" />
  <span>Ayuda</span>
</div>

<div id="medida_-1em">
  <h1>medida -1em</h1>
  <img src="images/help.png" alt="Ayuda" />
  <span>Ayuda</span>
</div>

<div id="porcentaje_10">
  <h1>porcentaje 10%</h1>
  <img src="images/help.png" alt="Ayuda" />
  <span>Ayuda</span>
</div>

<div id="porcentaje_80">
  <h1>porcentaje 80%</h1>
  <img src="images/help.png" alt="Ayuda" />
  <span>Ayuda</span>
</div>
```

El valor por defecto es `baseline` y el valor más utilizado cuando se establece la propiedad `vertical-align` es `middle`.

En muchas publicaciones impresas suele ser habitual tabular la primera línea de cada párrafo para facilitar su lectura. CSS permite controlar esta tabulación mediante la propiedad `text-indent`.

Tabla 6.12. Propiedad text-indent

`text-indent`	**Tabulación del texto**
Valores	`<medida> | <porcentaje> | inherit`
Se aplica a	Todos los elementos
Valor inicial	0
Descripción	Tabula desde la izquierda la primera línea del texto original

La siguiente imagen muestra la comparación entre un texto largo formado por varios párrafos sin tabular y el mismo texto con la primera línea de cada párrafo tabulada:

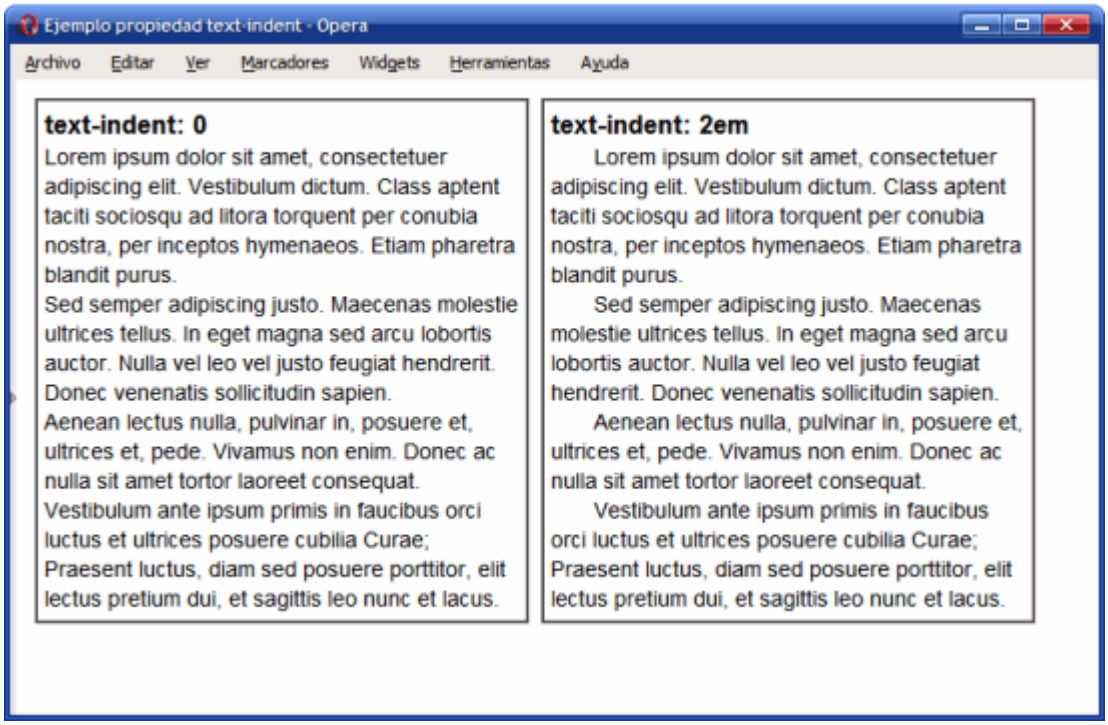


Figura 6.9. Ejemplo de propiedad text-indent

CSS también permite controlar la separación entre las letras que forman las palabras y la separación entre las palabras que forman los textos. La propiedad que controla la separación entre letras se llama `letter-spacing` y la separación entre palabras se controla mediante `word-spacing`.

Tabla 6.13. Propiedad letter-spacing

`letter-spacing`	**Espaciado entre letras**
Valores	`normal` | `<medida>` | `inherit`
Se aplica a	Todos los elementos
Valor inicial	normal
Descripción	Permite establecer el espacio entre las letras que forman las palabras del texto

Tabla 6.14. Propiedad word-spacing

`word-spacing`	**Espaciado entre palabras**
Valores	`normal` | `<medida>` | `inherit`
Se aplica a	Todos los elementos
Valor inicial	normal
Descripción	Permite establecer el espacio entre las palabras que forman el texto

La siguiente imagen muestra la comparación entre un texto normal y otro con las propiedades `letter-spacing` y `word-spacing` aplicadas:

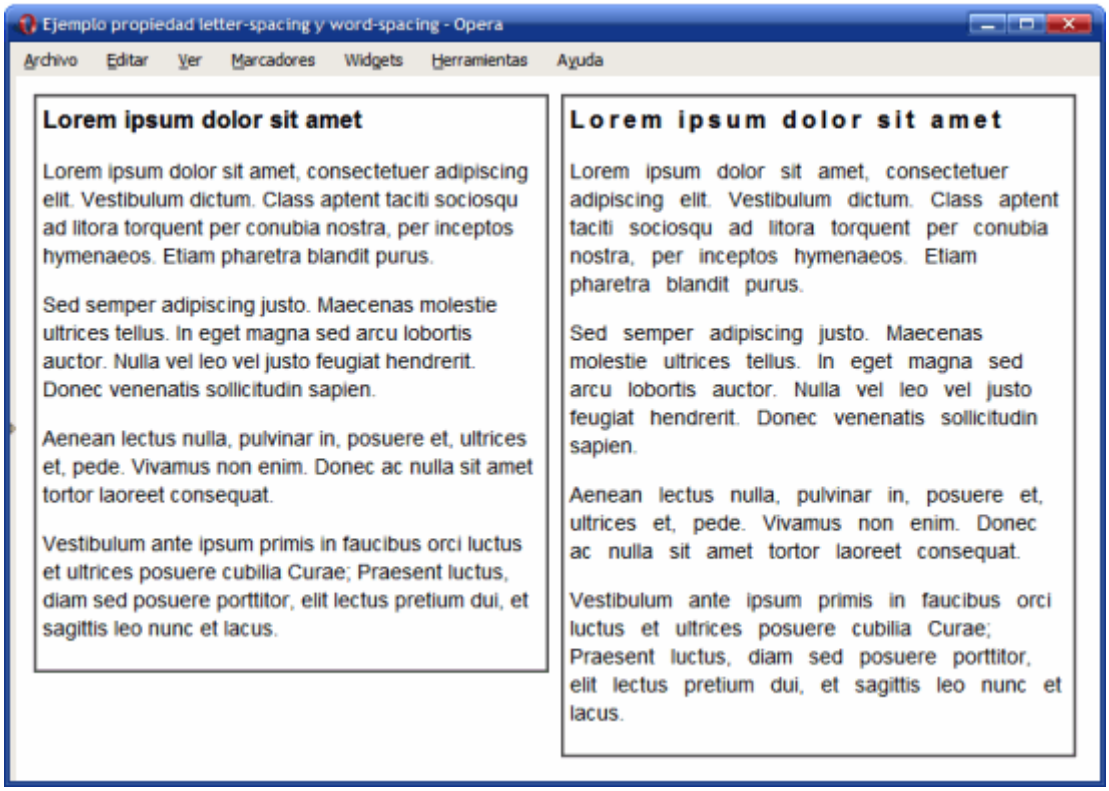


Figura 6.10. Ejemplo de propiedades letter-spacing y word-spacing

Las reglas CSS del ejemplo anterior se muestran a continuación:

```
.especial h1 { letter-spacing: .2em; }
.especial p { word-spacing: .5em; }

<div><h1>Lorem ipsum dolor sit amet</h1>
<p>Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Vestibulum
dictum. Class aptent taciti sociosqu ad litora torquent per conubia nostra,
per inceptos hymenaeos. Etiam pharetra blandit purus.</p>
...
</div>

<div class="especial"><h1>Lorem ipsum dolor sit amet</h1>
<p>Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Vestibulum
dictum. Class aptent taciti sociosqu ad litora torquent per conubia nostra,
per inceptos hymenaeos. Etiam pharetra blandit purus.</p>
...
</div>
```

Como ya se sabe. el tratamiento que hace HTML de los espacios en blanco es bastante problemático cuando se empiezan a crear las primeras páginas. Básicamente, HTML elimina todos los espacios en blanco sobrantes (es decir, todos salvo el espacio en blanco entre cada palabra) y todas las nuevas líneas.

Para forzar los espacios en blanco adicionales se debe utilizar la entidad HTML ` ` y para forzar nuevas líneas, se utiliza el elemento `<br/>`. Además, HTML proporciona el

elemento `<pre>` que muestra el contenido tal y como se escribe, respetando todos los espacios en blanco y todas las nuevas líneas.

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

Tabla 6.15. Propiedad white-space

`white-space`	**Tratamiento de los espacios en blanco**
Valores	`normal` | `pre` | `nowrap` | `pre-wrap` | `pre-line` | `inherit`
Se aplica a	Todos los elementos
Valor inicial	normal
Descripción	Establece el tratamiento de los espacios en blanco del texto

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

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

La siguiente imagen muestra las diferencias entre los valores permitidos para `white-space`. El párrafo original contiene espacios en blanco y nuevas líneas y se ha limitado el espacio en el que se debería mostrar los contenidos:

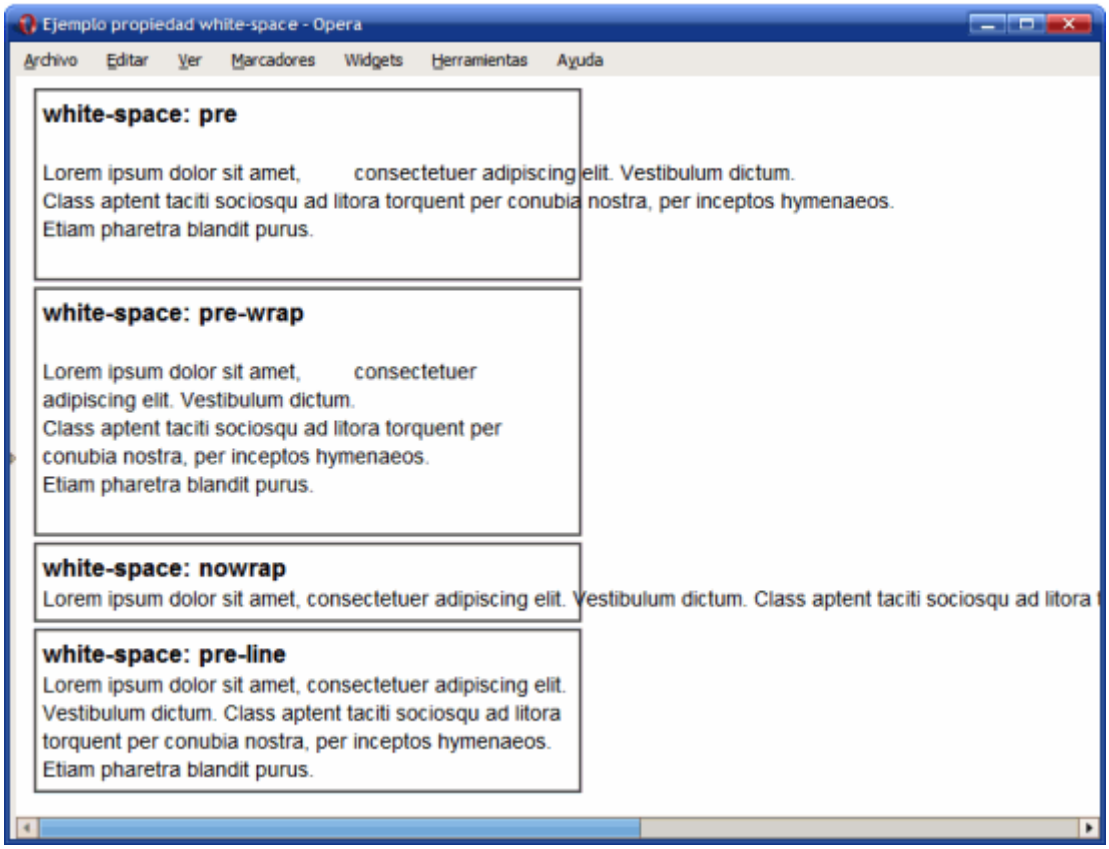


Figura 6.11. Ejemplo de propiedad white-space

Por último, CSS define unos elementos especiales llamados *"pseudo-elementos"* que permiten aplicar estilos a ciertas partes de un texto. En concreto, CSS permite definir estilos especiales a la primera frase de un texto y a la primera letra de un texto.

El pseudo-elemento `:first-line` permite aplicar estilos a la primera línea de un texto. Las palabras que forman la primera línea de un texto dependen del espacio reservado para mostrar el texto o del tamaño de la ventana del navegador, por lo que CSS calcula de forma automática las palabras que forman la primera línea de texto en cada momento.

La siguiente imagen muestra cómo aplica CSS los estilos indicados a la primera línea calculando para cada anchura las palabras que forman la primera línea:

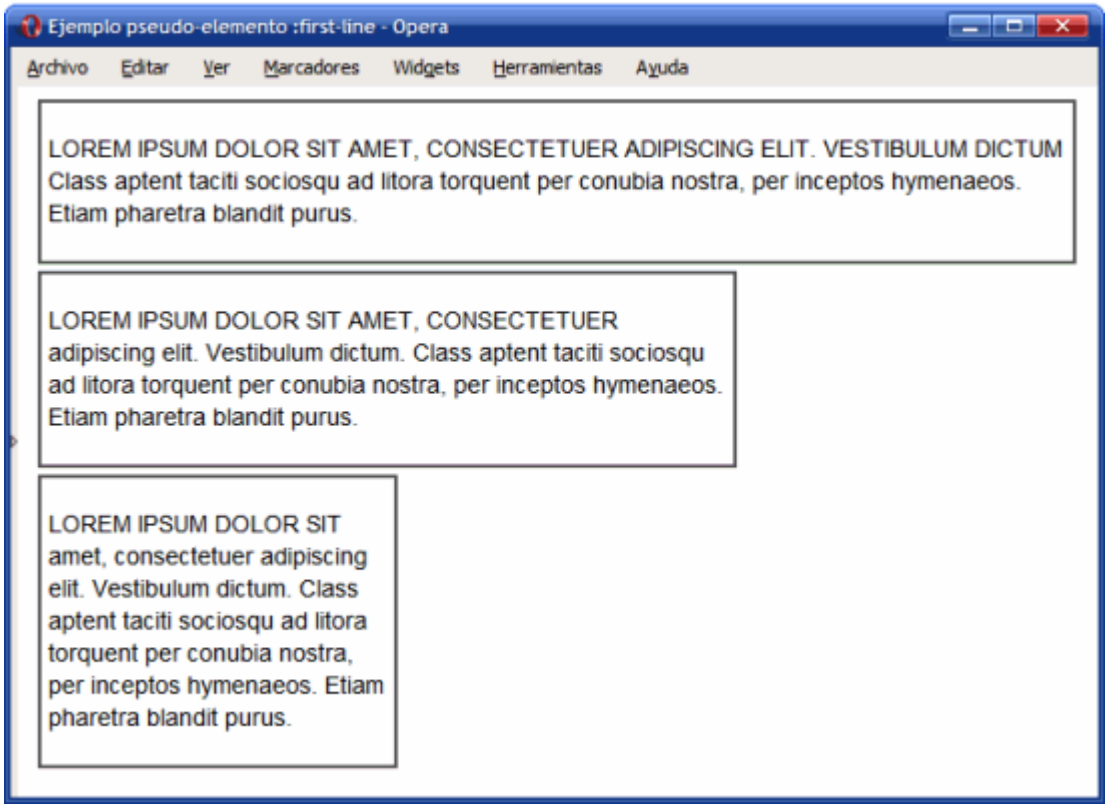


Figura 6.12. Ejemplo de pseudo-elemento first-line

La regla CSS utilizada para los párrafos del ejemplo se muestra a continuación:

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

De la misma forma, CSS permite aplicar estilos a la primera letra del texto mediante el pseudo-elemento `:first-letter`. La siguiente imagen muestra el uso del pseudo-elemento `:first-letter` para crear una letra capital:

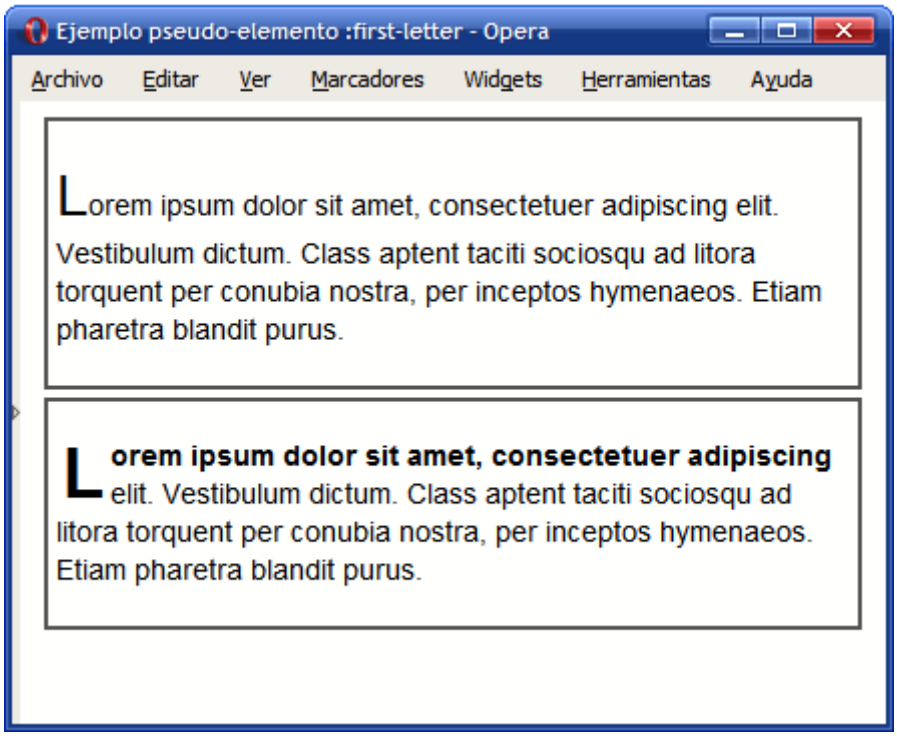


Figura 6.13. Ejemplo de pseudo-elemento first-letter

El código HTML y CSS se muestra a continuación:

```
#normal p:first-letter {
  font-size: 2em;
}
#avanzado p:first-letter {
  font-size: 2.5em;
  font-weight: bold;
  line-height: .9em;
  float: left;
  margin: .1em;
}
#avanzado p:first-line {
  font-weight: bold;
}

<div id="normal">
<p>Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Vestibulum
dictum. Class aptent taciti sociosqu ad litora torquent per conubia nostra,
per inceptos hymenaeos. Etiam pharetra blandit purus.</p>
</div>
<div id="avanzado">
<p>Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Vestibulum
dictum. Class aptent taciti sociosqu ad litora torquent per conubia nostra,
per inceptos hymenaeos. Etiam pharetra blandit purus.</p>
</div>
```

Capítulo 7. Enlaces

7.1. Estilos básicos

7.1.1. Tamaño, color y decoración

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

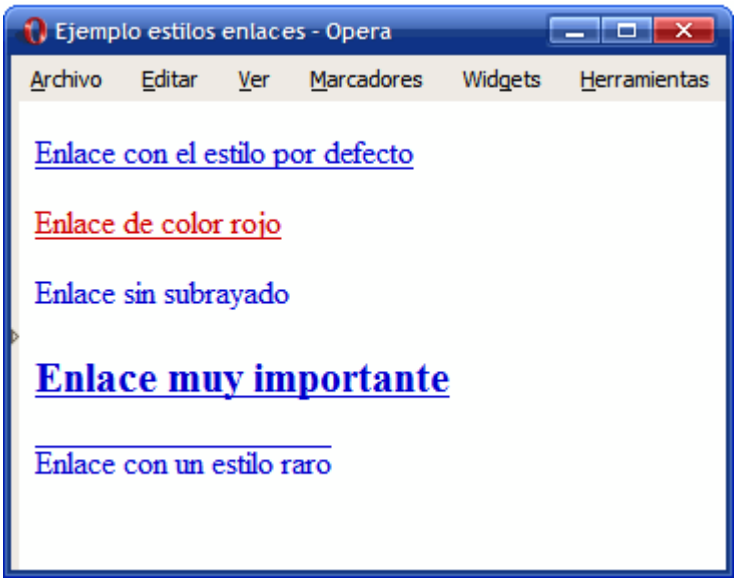


Figura 7.1. Ejemplo de enlaces con estilos diferentes

A continuación se muestran las reglas CSS del ejemplo anterior:

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

.alternativo {color: #CC0000;}
.simple {text-decoration: none;}
.importante {font-weight: bold; font-size: 1.3em;}
.raro {text-decoration:overline;}

<a href="#">Enlace con el estilo por defecto</a>
<a class="alternativo" href="#">Enlace de color rojo</a>
<a class="simple" href="#">Enlace sin subrayado</a>
<a class="importante" href="#">Enlace muy importante</a>
<a class="raro" href="#">Enlace con un estilo raro</a>
```

7.1.2. Pseudo-clases

CSS define cuatro *pseudo-clases* que permiten aplicar estilos avanzados para los enlaces de los documentos. Las *pseudo-clases* permiten aplicar diferentes estilos a un mismo

enlace en función de su estado: enlace no visitado, enlace visitado, enlace en el que se pasa el puntero del ratón por encima y enlace activo en ese momento.

Cada una de las pseudo-clases definidas se muestra a continuación:

- `:link`, permite aplicar estilos para los enlaces que aún no han sido pinchados.
- `:visited`, aplica estilos a los enlaces que han sido pinchados anteriormente (el navegador del usuario elimina automáticamente el historial de enlaces visitados cada cierto tiempo).
- `:hover`, estilos que muestra el enlace cuando el usuario posiciona el puntero del ratón sobre el enlace.
- `:active`, estilos que se aplican al enlace cuando el usuario está pinchando sobre el enlace (el tiempo durante el que se aplica este estilo es muy breve).

El orden recomendado para la definición de las pseudo-clases de los enlaces es `:link`, `:visited`, `:hover`, `:active`.

Las pseudo-clases `:link` y `:visited` solamente están definidas para los enlaces, pero las pseudo-clases `:hover` y `:active` se definen para todos los elementos HTML. Desafortunadamente, Internet Explorer 6 y sus versiones anteriores solamente las soportan para los enlaces.

Las pseudo-clases de los enlaces permiten variar el comportamiento por defecto que los navegadores aplican a los enlaces. El siguiente ejemplo oculta el subrayado a los enlaces cuando el usuario pasa el ratón por encima de un enlace:

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

El código CSS necesario se muestra a continuación:

```
a { }

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

Ejercicio 8 Ver enunciado en la página 188

7.2. Estilos avanzados

7.2.1. Decoración personalizada

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

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

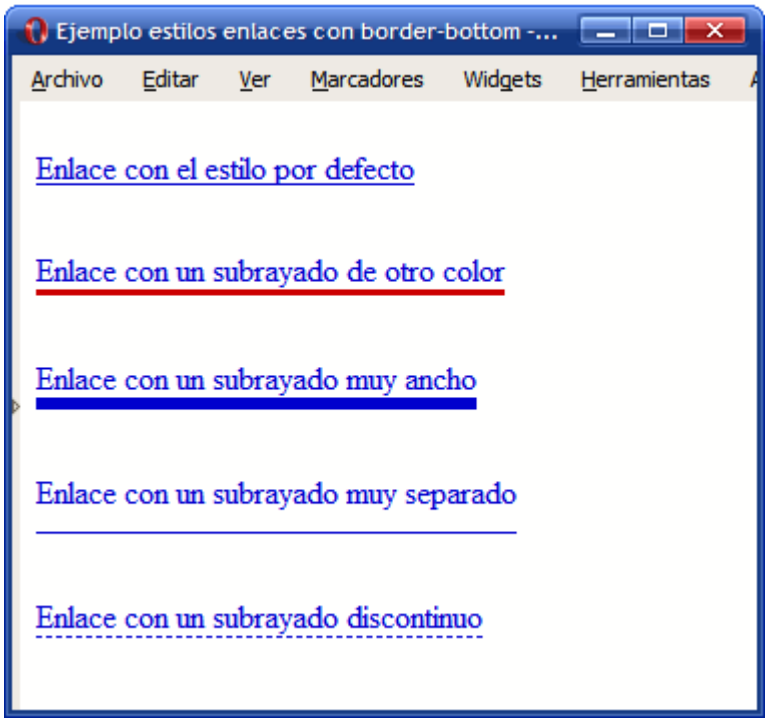


Figura 7.3. Enlaces con subrayado personalizado mediante la propiedad border

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

```
a { margin: 1em 0; float: left; clear: left; text-decoration: none;}
.simple {text-decoration: underline;}
.color { border-bottom: medium solid #CC0000;}
.ancho {border-bottom: thick solid;}
.separado {border-bottom: 1px solid; padding-bottom: .6em;}
.discontinuo {border-bottom: thin dashed;}

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

<a class="color" href="#">Enlace un subrayado de otro color</a>

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

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

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

7.2.2. Imágenes según el tipo de enlace

En ocasiones, puede resultar útil incluir un pequeño icono al lado de un enlace para indicar el tipo de contenido que enlaza, como por ejemplo un archivo comprimido o un documento con formato PDF.

Este tipo de imágenes son puramente decorativas en vez de imágenes de contenido, por lo que se deberían añadir con CSS y no con elementos de tipo `<img>`. Utilizando la propiedad `background` (y `background-image`) se puede personalizar el aspecto de los enlaces para que incluyan un pequeño icono a su lado.

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

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

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

Las reglas CSS necesarias se muestran a continuación:

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

.rss {
  color: #E37529;
  padding: 0 0 0 18px;
  background: #FFF url(imagenes/rss.gif) no-repeat left center;
}

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

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

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

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

7.2.3. Mostrar los enlaces como si fueran botones

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

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

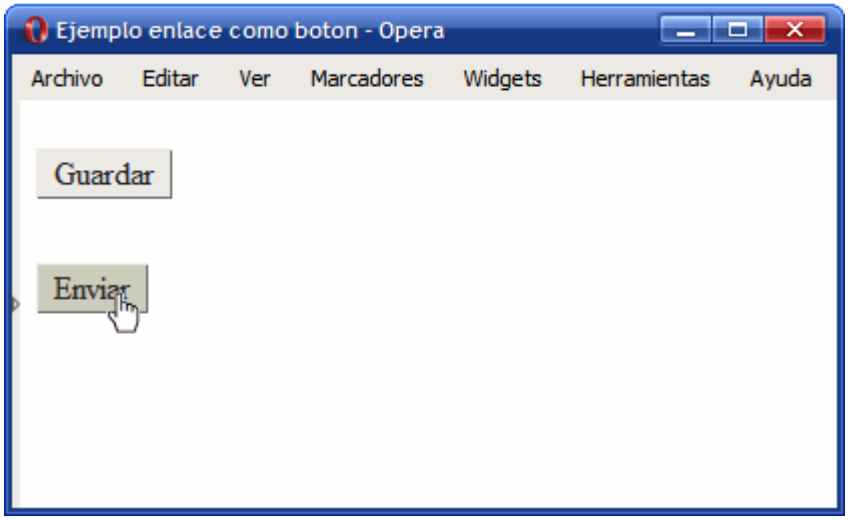


Figura 7.5. Mostrando un enlace como si fuera un botón

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

```
a { margin: 1em 0; float: left; clear: left; }
a.boton {
  text-decoration: none;
  background: #EEE;
  color: #222;
  border: 1px outset #CCC;
  padding: .1em .5em;
}
a.boton:hover {
  background: #CCB;
}
a.boton:active {
  border: 1px solid #000;
}

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

Capítulo 8. Imágenes

8.1. Estilos básicos

8.1.1. Eliminar el borde de las imágenes con enlaces

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

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

Ejercicio 9 Ver enunciado en la página 189

8.2. Estilos avanzados

8.2.1. Sombra (drop shadow)

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

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

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

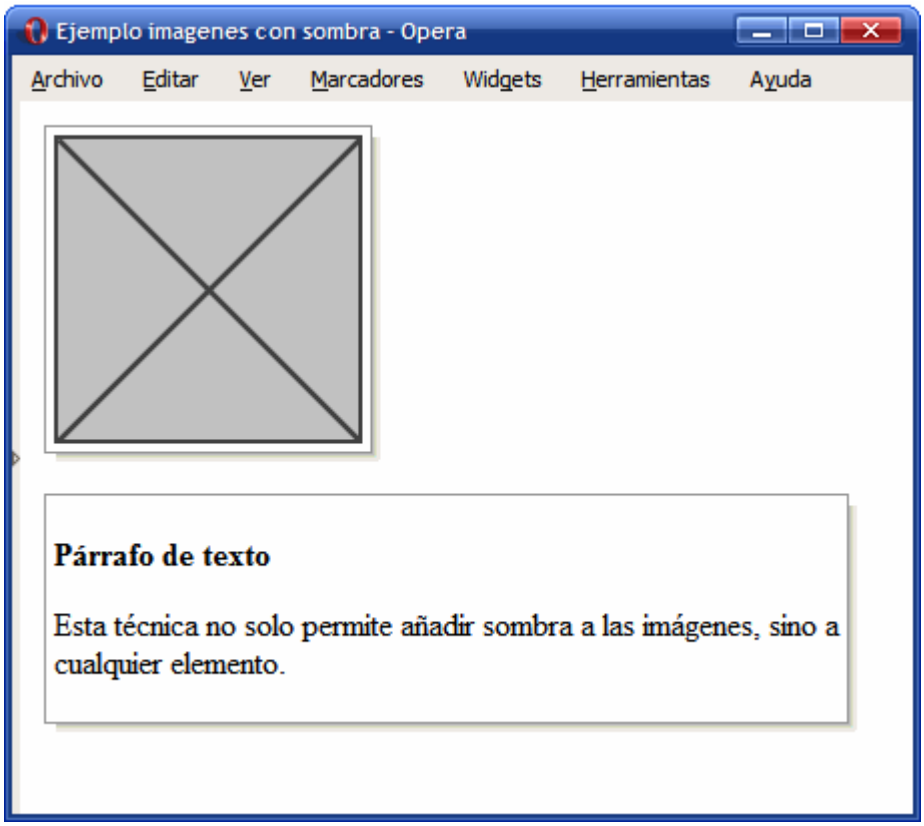


Figura 8.1. Sombra aplicada a las imágenes y otros elementos mediante CSS

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

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

El mayor inconveniente de este método es que la sombra siempre se muestra en la misma dirección (derecha inferior) y que en Internet Explorer 6 y versiones anteriores sólo muestran la sombra correctamente cuando el color de fondo de la página es blanco (ya que Internet Explorer 6 y versiones anteriores no soportan imágenes en formato PNG con transparencias).

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

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

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

<div class="dropshadow">
<div class="innerbox">
  <img src="imagenes/imagen.png" alt="Imagen genérica" />
</div>
</div>

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

Capítulo 9. Listas

9.1. Estilos básicos

9.1.1. Viñetas personalizadas

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

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

Tabla 9.1. Propiedad list-style-type

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

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

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

- Los valores gráficos son `disc`, `circle` y `square` y muestran como viñeta un círculo relleno, un círculo vacío y un cuadrado relleno respectivamente.
- Los valores numéricos están formados por `decimal`, `decimal-leading-zero`, `lower-roman`, `upper-roman`, `armenian` y `georgian`.
- Por último, los valores alfanuméricos se controlan mediante `lower-latin`, `lower-alpha`, `upper-latin`, `upper-alpha` y `lower-greek`.

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

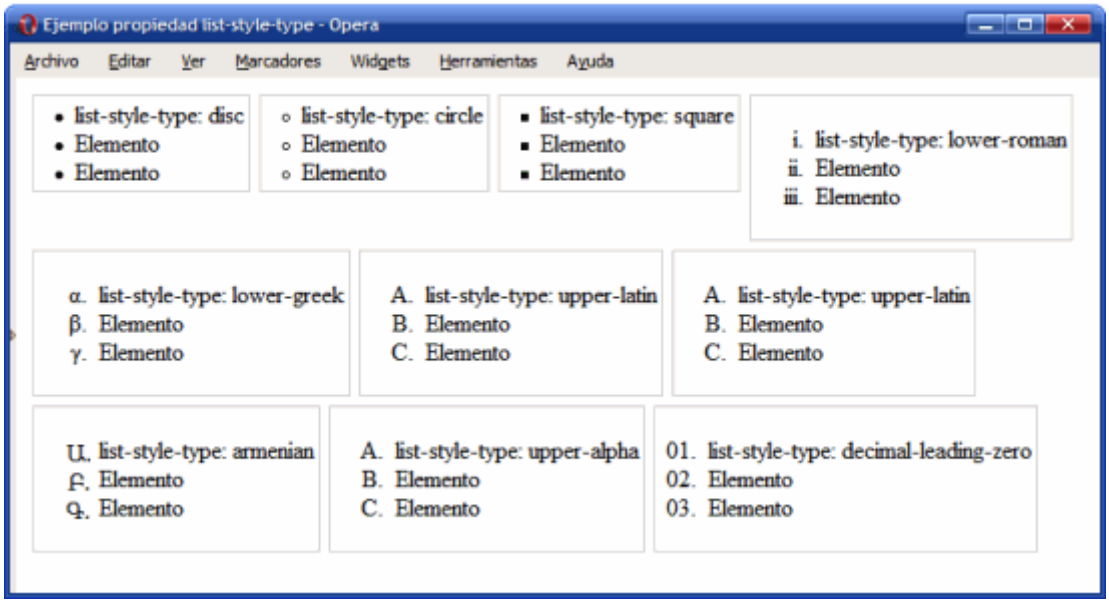


Figura 9.1. Ejemplo de propiedad list-style-type

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

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

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

Tabla 9.2. Propiedad list-style-position

`list-style-position`	**Posición de la viñeta**
Valores	`inside` | `outside` | `inherit`
Se aplica a	Elementos de una lista
Valor inicial	outside
Descripción	Permite establecer la posición de la viñeta de cada elemento de una lista

La diferencia entre los valores `outside` y `inside` se hace evidente cuando los elementos contienen mucho texto, como en la siguiente imagen:

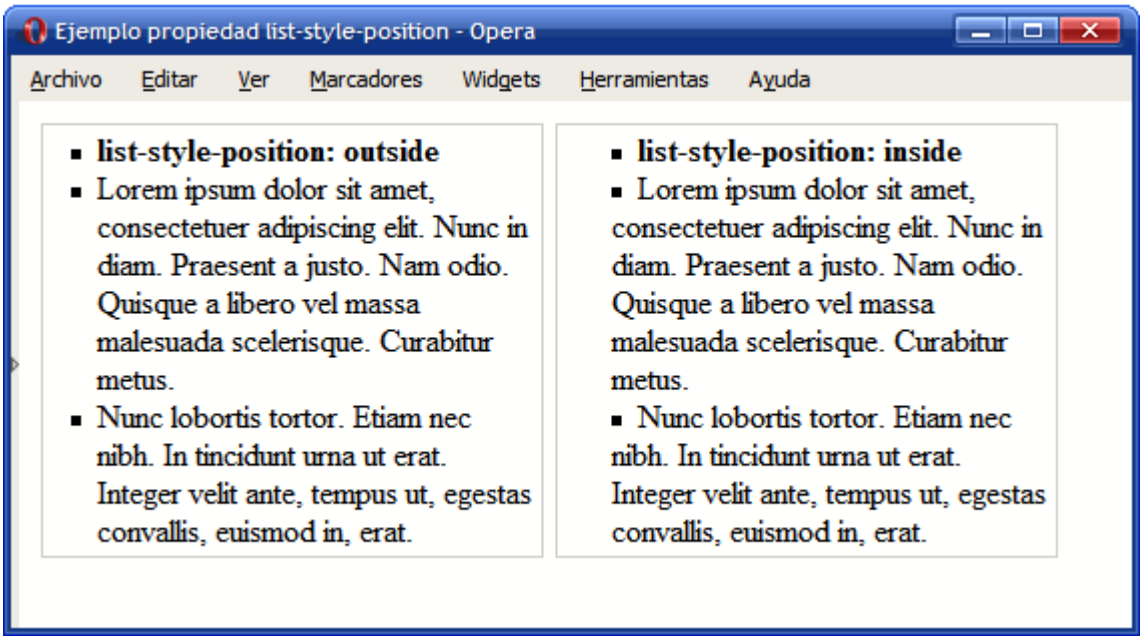


Figura 9.2. Ejemplo de propiedad list-style-position

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

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

Tabla 9.3. Propiedad list-style-image

`list-style-image`	**Imagen de la viñeta**
Valores	`<url>` | `none` | `inherit`
Se aplica a	Elementos de una lista
Valor inicial	none
Descripción	Permite reemplazar las viñetas automáticas por una imagen personalizada

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

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

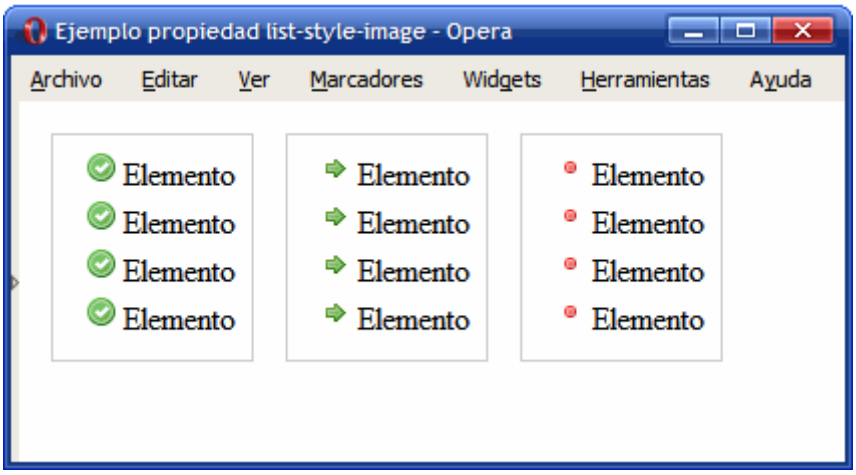


Figura 9.3. Ejemplo de propiedad list-style-image

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

```
ul {
  margin:0;
  padding-left: 1.5em;
  line-height: 1.5em;
}
ul li { padding-left: .2em; }
ul.ok { list-style-image: url(imagenes/ok.png); }
ul.go { list-style-image: url(imagenes/bullet_go.png); }
ul.redondo { list-style-image: url(imagenes/bullet_red.png); }

<div>
<ul class="ok">
  <li>Elemento</li>
  <li>Elemento</li>
  <li>Elemento</li>
  <li>Elemento</li>
</ul>
</div>
<div>
<ul class="go">
  <li>Elemento</li>
  <li>Elemento</li>
  <li>Elemento</li>
  <li>Elemento</li>
</ul>
</div>
<div>
<ul class="redondo">
  <li>Elemento</li>
  <li>Elemento</li>
  <li>Elemento</li>
  <li>Elemento</li>
</ul>
</div>
```

Como es habitual, CSS define una propiedad de tipo *"shorthand"* que permite establecer todas las propiedades de una lista de forma directa. La propiedad se denomina `list-style`.

Tabla 9.4. Propiedad list-style

`list-style`	**Estilo de una lista**
Valores	`( <list-style-type> || <list-style-position> || <list-style-image> ) | inherit`
Se aplica a	Elementos de una lista
Valor inicial	-
Descripción	Propiedad que permite establecer de forma simultánea todas las opciones de una lista

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

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

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

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

9.1.2. Menú vertical sencillo

Las listas HTML se suelen emplear, además de para su función natural, para la creación de menús de navegación verticales y horizontales.

A continuación se muestra la transformación de una lista sencilla de enlaces en un menú vertical de navegación.

Lista de enlaces original:

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

Aspecto final del menú vertical:

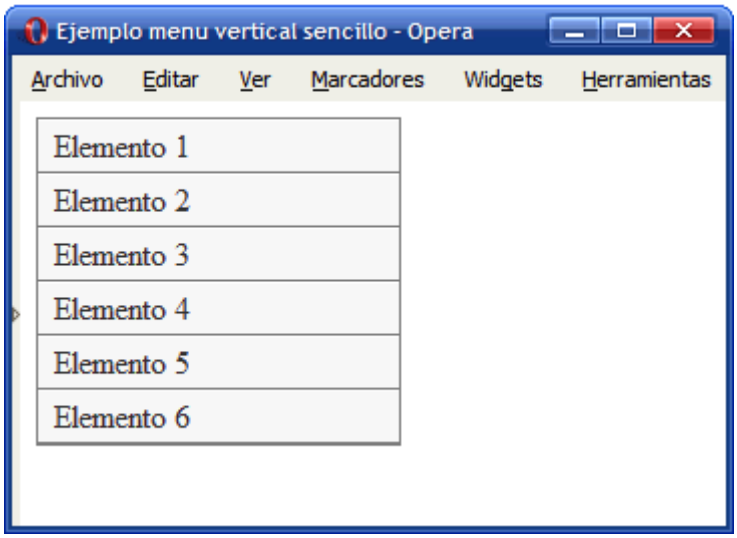


Figura 9.4. Menú vertical sencillo creado con CSS

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

1) Definir la anchura del menú:

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

Figura 9.5. Menú vertical: definiendo su anchura

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

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

Figura 9.6. Menú vertical: eliminar viñetas por defecto

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

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

Figura 9.7. Menú vertical: añadiendo bordes

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

```
ul.menu li a:link, ul.menu li a:visited {
  padding: .2em 0 .2em .5em;
  display: block;
  text-decoration: none;
  color: #333;
}
```

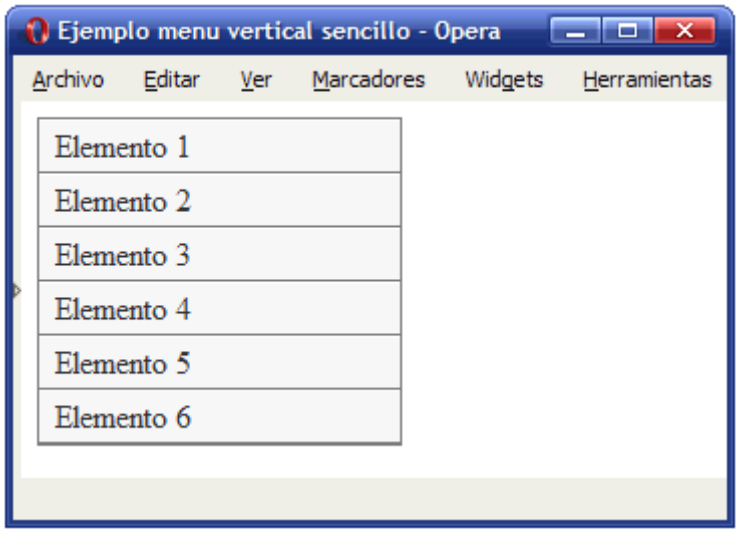


Figura 9.8. Aspecto final del menú vertical sencillo creado con CSS

Los tipos de menús verticales que se pueden definir mediante las propiedades CSS son innumerables, como se puede observar en la página `http://www.exploding-boy.com/images/EBmenus/menus.html`

9.1.3. Menú vertical avanzado

Ejercicio 10 Ver enunciado en la página 190

9.2. Estilos avanzados

9.2.1. Menú horizontal básico

A partir de un menú vertical sencillo, es posible crear un menú horizontal sencillo aplicando las propiedades CSS conocidas hasta el momento.

A continuación se muestra la transformación del anterior menú vertical sencillo en un menú horizontal sencillo. En este ejemplo, las propiedades para establecer el aspecto de los elementos del menú se definen en los elementos `<a>` en lugar de definirlas para los elementos `<li>` como en el ejemplo anterior. En cualquier caso, es indiferente el elemento en el que se aplican los estilos que definen el aspecto de cada opción del menú.

Código HTML del menú horizontal:

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

Código CSS del menú vertical anterior:

```
ul.menu {
  width: 180px;
  list-style: none;
  margin: 0;
  padding: 0;
  border: 1px solid #7C7C7C;
}
ul.menu li {
  border-bottom:1px solid #7C7C7C;
  border-top: 1px solid #FFF;
  background: #F4F4F4;
}
ul.menu li a:link, ul.menu li a:visited {
  padding: .2em 0 .2em .5em;
  display:block;
  text-decoration: none;
  color: #333;
}
```

Aspecto final del menú horizontal:

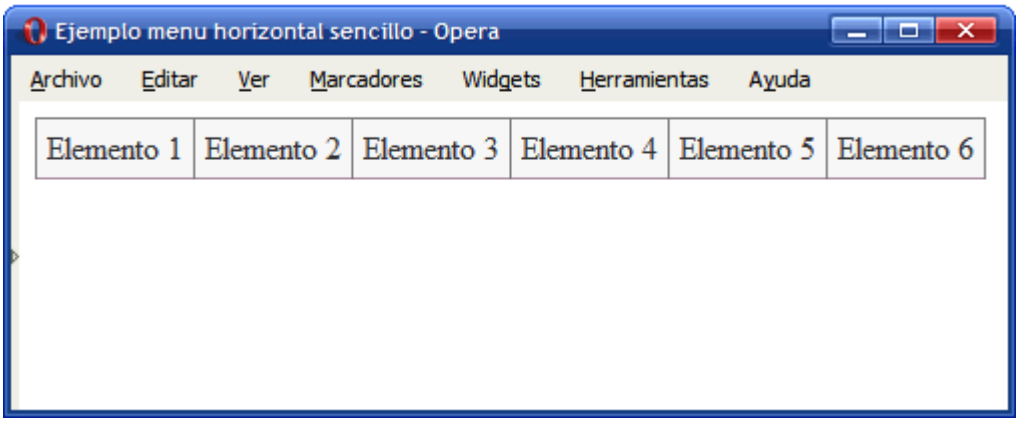


Figura 9.9. Menú horizontal sencillo creado con CSS

El proceso de transformación del menú vertical en un menú horizontal consta de los siguientes pasos:

1) Eliminar la anchura y el borde del elemento `<ul>` y aplicar las propiedades `float` y `clear`:

```
ul.menu {
  clear: both;
  float: left;
```

```
  width: 100%;
  list-style: none;
  margin: 0;
  padding: 0;
}
```

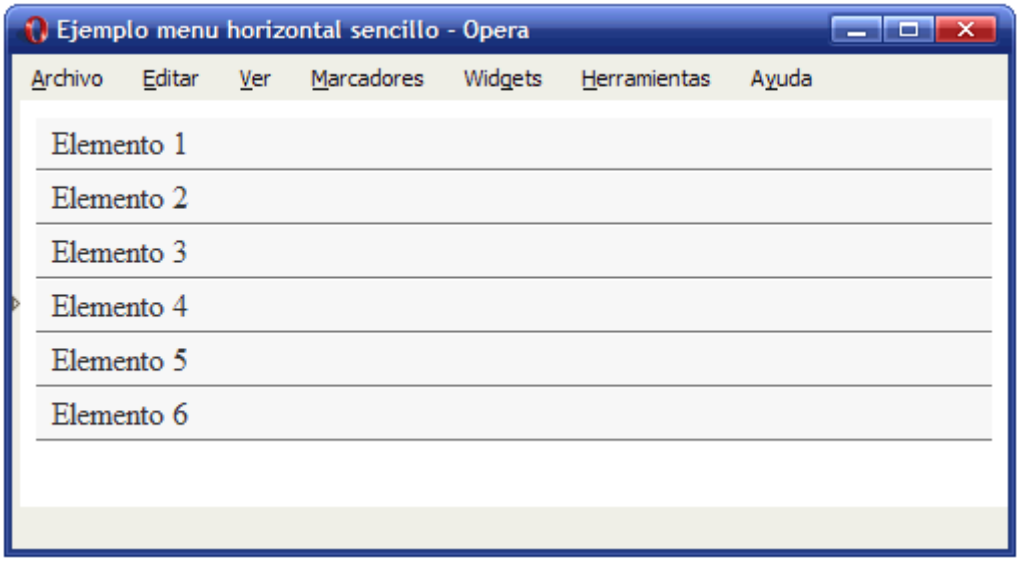


Figura 9.10. Menú horizontal: definiendo anchura y bordes

2) La clave de la transformación reside en modificar las propiedades `float` y `display` de los elementos `<li>` del menú:

```
ul.menu li {
  display: inline;
  float: left;
}
```

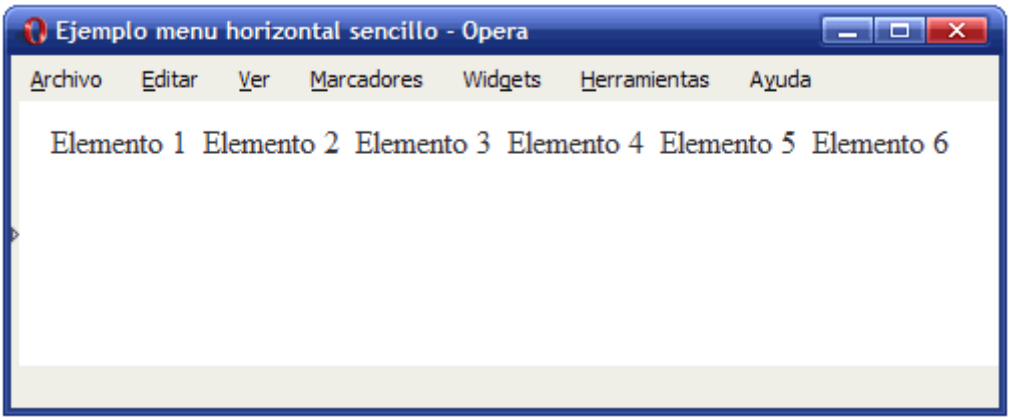


Figura 9.11. Menú horizontal: propiedades float y display

3) Modificar el `padding` y los bordes de los enlaces que forman el menú:

```
ul.menu li a:link, ul.menu li a:visited {
  padding: .3em;
  display: block;
  text-decoration: none;
  color: #333;
```

```
  background: #F4F4F4;
  border-top: 1px solid #7C7C7C;
  border-right: 1px solid #7C7C7C;
  border-bottom: 1px solid #9C9C9C;
}
```

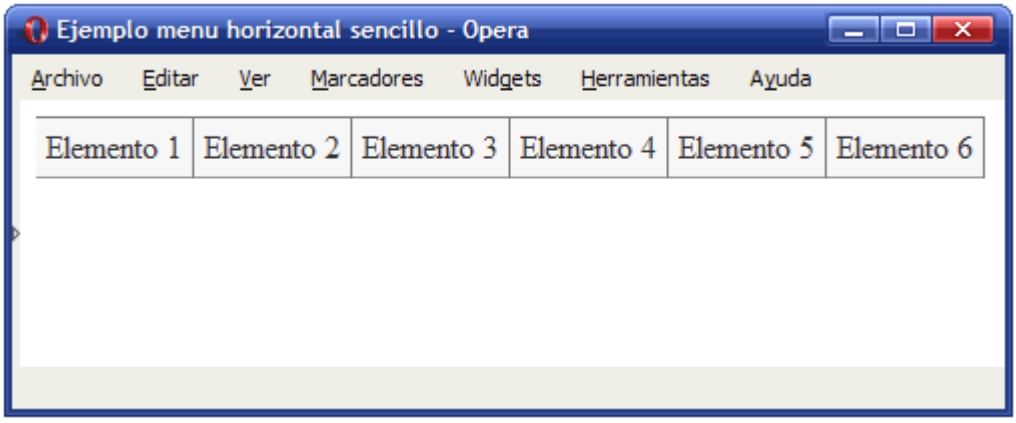


Figura 9.12. Menú horizontal: relleno y borde de los elementos

4) Por último, se añade un borde izquierdo en el elemento `<ul>` para homogeneizar el aspecto de los elementos del menú:

```
ul.menu {
  clear: both;
  float: left;
  width: 100%;
  list-style: none;
  margin: 0;
  padding: 0;
  border-left: 1px solid #7C7C7C;
}
```

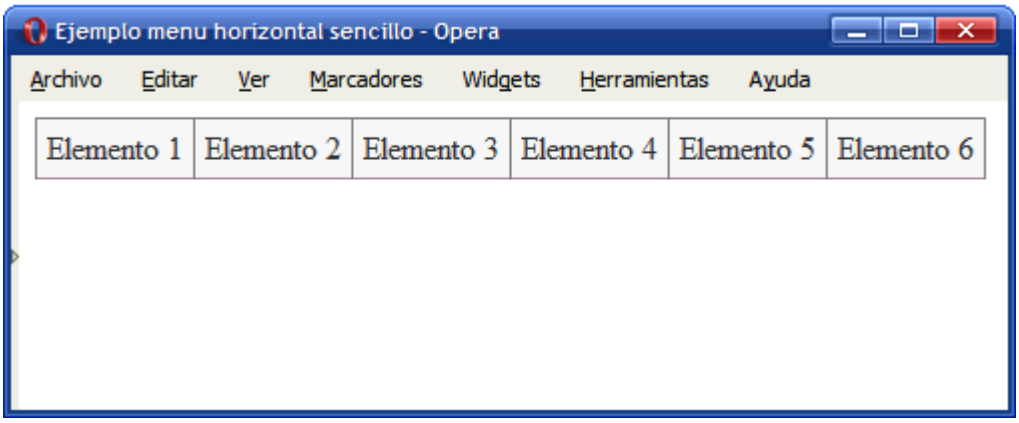


Figura 9.13. Aspecto final del menú horizontal sencillo creado con CSS

El código CSS final se muestra a continuación:

```
ul.menu {
  clear: both;
  float: left;
  width: 100%;
```

```
  list-style: none;
  margin: 0;
  padding: 0;
  border-left:1px solid #7C7C7C;
}
ul.menu li {
  display: inline;
  float: left;
}
ul.menu li a:link, ul.menu li a:visited {
  padding: .3em;
  display: block;
  text-decoration: none;
  color: #333;
  background: #F4F4F4;
  border-top:1px solid #7C7C7C;
  border-right:1px solid #7C7C7C;
  border-bottom:1px solid #9C9C9C;
}
```

9.2.2. Menú horizontal con solapas

Modificando los estilos de cada elemento del menú y utilizando imágenes de fondo y las pseudo-clases `:hover` y `:active`, se pueden crear menús horizontales complejos, incluso con el aspecto de un menú de solapas o pestañas:

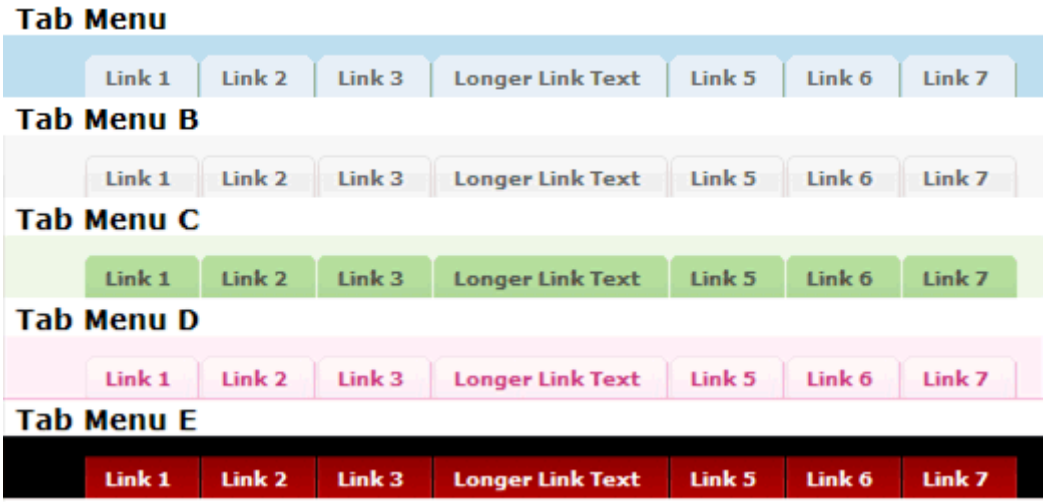


Figura 9.14. Ejemplos de menús horizontales con pestañas creados con CSS

El código fuente de los menús de la imagen anterior y muchos otros se puede encontrar en `http://exploding-boy.com/images/cssmenus/menus.html`

9.2.3. Menú horizontal avanzado

Además de los menús horizontales de solapas, existen muchos otros tipos diferentes de menús horizontales (y verticales) que se pueden realizar empleando exclusivamente CSS:

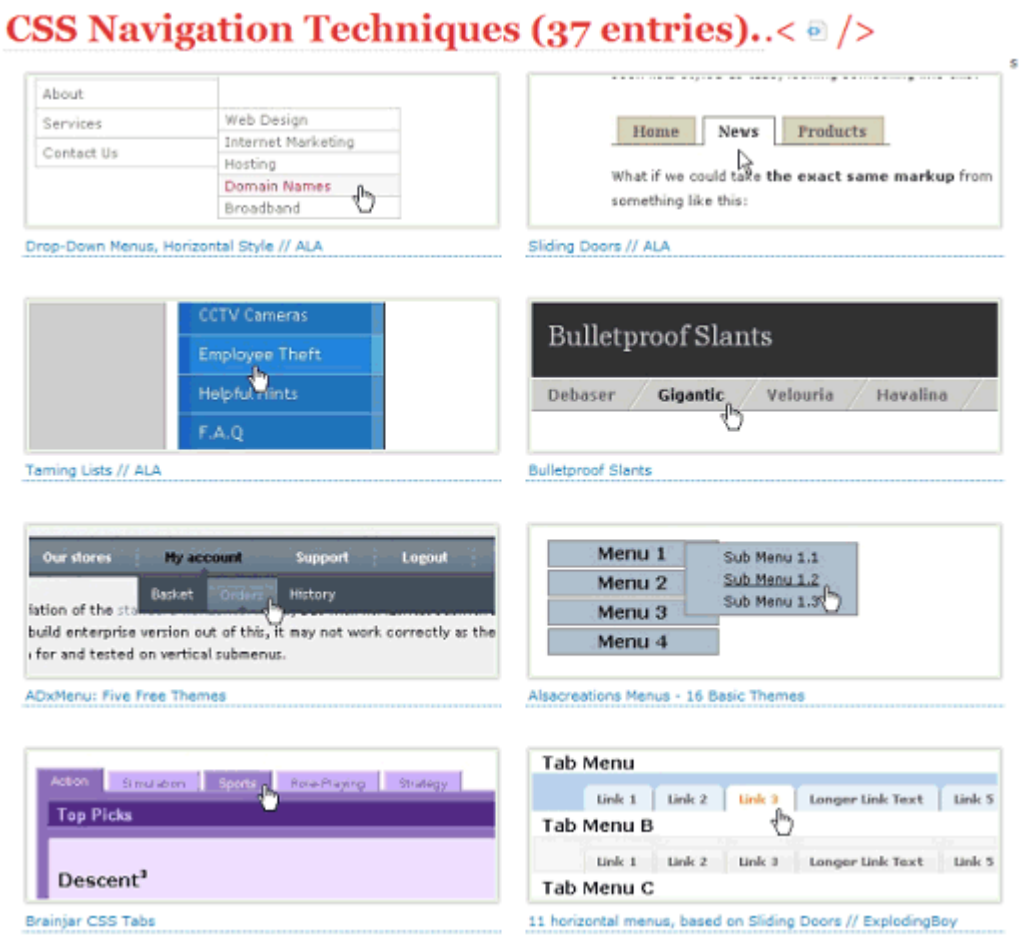


Figura 9.15. Ejemplos de menús horizontales y verticales complejos creados con CSS

El código CSS de todos los ejemplos de la imagen anterior y muchos otros se pueden encontrar en: `http://alvit.de/css-showcase/css-navigation-techniques-showcase.php`

Capítulo 10. Tablas

10.1. Estilos básicos

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

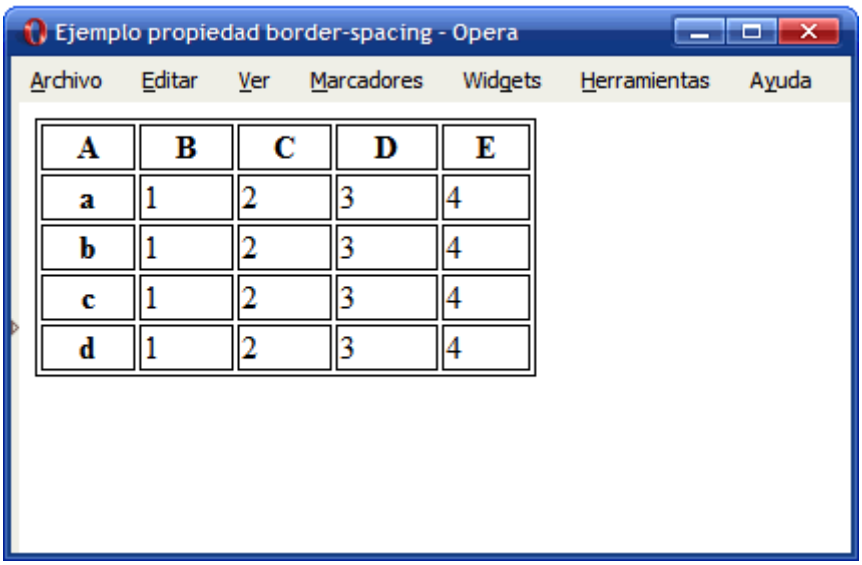


Figura 10.1. Aspecto por defecto de los bordes de una tabla

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

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

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

Normalmente, la separación entre los bordes de las diferentes celdas no es deseable, ya que es preferible mostrar las celdas sin separación. CSS define la propiedad `border-spacing` para controlar la separación entre las celdas de una tabla.

Tabla 10.1. Propiedad border-spacing

border-spacing	**Espaciado entre bordes**
Valores	`<medida> <medida>? | inherit`
Se aplica a	Todas las tablas
Valor inicial	0
Descripción	Establece la separación entre los bordes de las celdas adyacentes de una tabla

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

Por tanto, el ejemplo anterior se puede modificar para no mostrar ningún espaciado entre las celdas adyacentes:

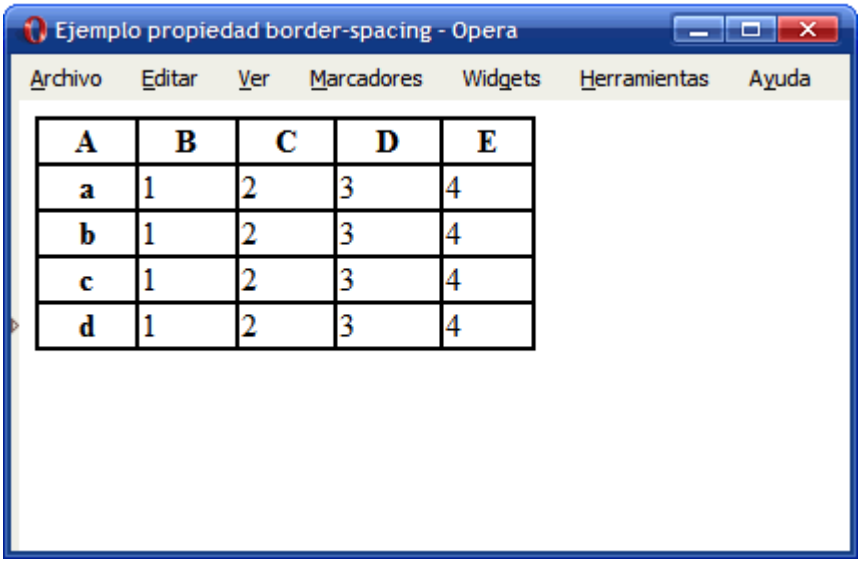


Figura 10.2. Ejemplo de propiedad border-spacing

La única regla añadida al código CSS es la siguiente:

```
.normal {
  width: 250px;
  border: 1px solid #000;
  border-spacing: 0;
}
```

Aunque la propiedad `border-spacing` soluciona el problema de la separación de las celdas adyacentes, origina otro problema: los bordes de las celdas adyacentes se juntan y por tanto, se muestran como si fueran un borde de anchura doble.

La propiedad `border-collapse` permite definir el mecanismo a seguir cuando se juntan dos bordes de celdas adyacentes.

Tabla 10.2. Propiedad border-collapse

border-collapse	**Fusión de bordes**
Valores	`collapse` | `separate` | `inherit`
Se aplica a	Todas las tablas
Valor inicial	separate
Descripción	Define el mecanismo de fusión de los bordes de las celdas adyacentes de una tabla

El valor `collapse` indica al navegador que los bordes de las celdas adyacentes se deben fusionar para que no se muestren bordes de anchura doble. Por lo tanto, el ejemplo anterior se puede rehacer para mostrar la tabla con bordes sencillos y sin separación entre celdas:

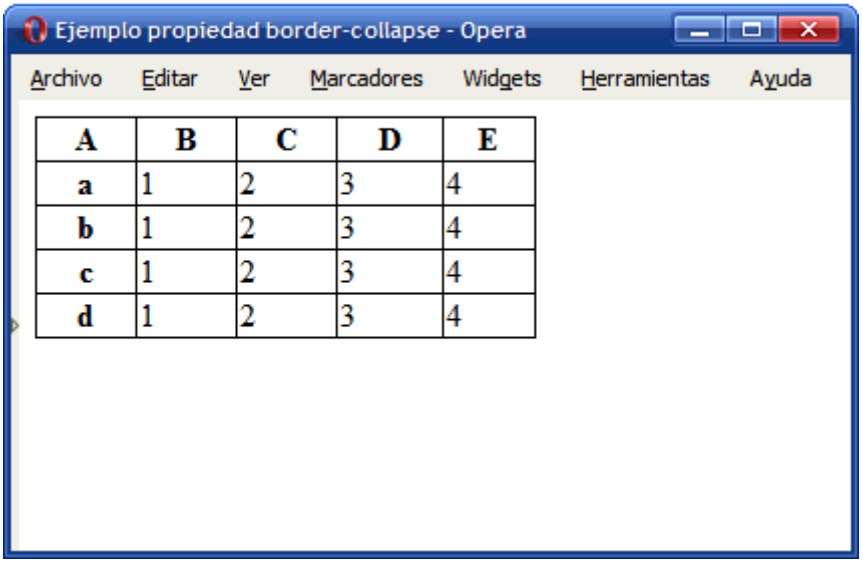


Figura 10.3. Ejemplo de propiedad border-collapse

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

```
.normal {
  width: 250px;
  border: 1px solid #000;
  border-spacing: 0;
  border-collapse: collapse;
}
.normal th, .normal td {
  border: 1px solid #000;
}

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

```
  </tr>
  ...
</table>
```

El mecanismo que sigue el valor `collapse` es muy complejo, ya que para fusionar diferentes bordes tiene en cuenta la anchura de cada borde, su estilo, el tipo de celda que contiene el borde (columna, fila, grupo de filas, grupo de columnas), etc.

10.2. Estilos avanzados

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

Tabla 10.3. Propiedad empty-cells

`empty-cells`	**Tratamiento de las celdas vacías**
Valores	`show` | `hide` | `inherit`
Se aplica a	Celdas de una tabla
Valor inicial	show
Descripción	Define el mecanismo utilizado para el tratamiento de las celdas vacías de una tabla

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

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

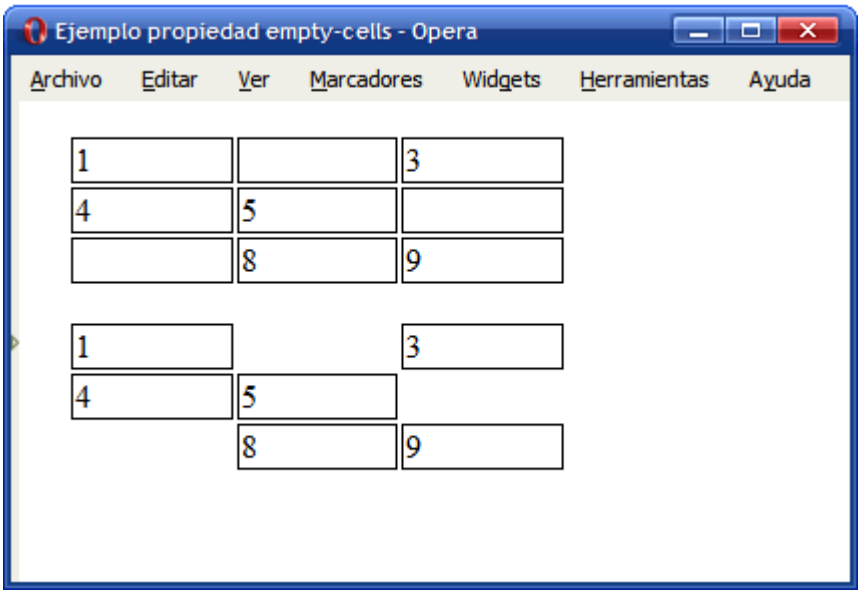


Figura 10.4. Ejemplo de propiedad empty-cells

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

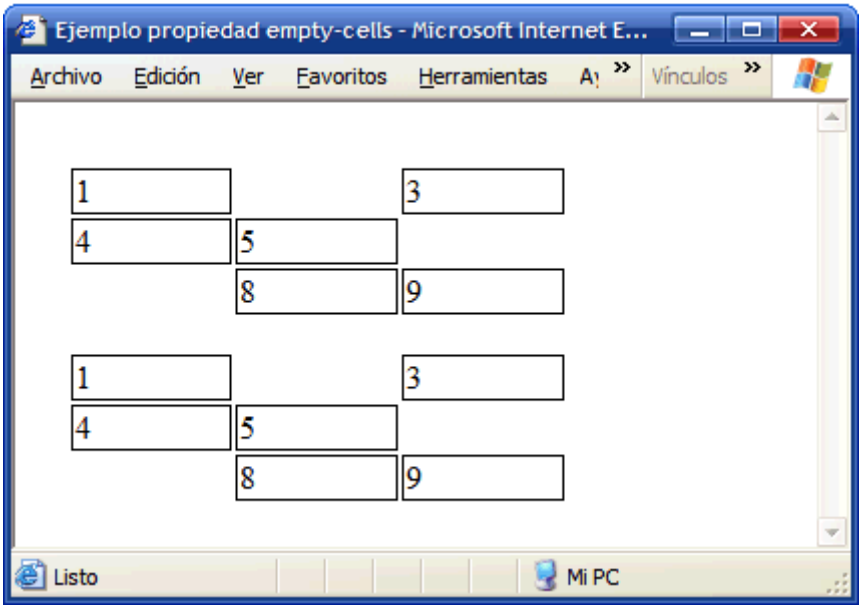


Figura 10.5. Internet Explorer no soporta la propiedad empty-cells

Por otra parte, el título de las tablas se establece mediante el elemento `<caption>`, que por defecto se muestra encima de los contenidos de la tabla. La propiedad `caption-side` permite controlar la posición del título de la tabla.

Tabla 10.4. Propiedad caption-side

`caption-side`	**Posición del título de la tabla**
Valores	`top` | `bottom` | `inherit`
Se aplica a	Los elementos caption
Valor inicial	top
Descripción	Establece la posición del título de la tabla

El valor `bottom` indica que el título de la tabla se debe mostrar después de los contenidos de la tabla. La alineación horizontal se controla mediante la propiedad `text-align`.

A continuación se muestra el código HTML y CSS de un ejemplo sencillo de uso de la propiedad `caption-side`:

```
.especial {
  caption-side: bottom;
}

<table class="especial" summary="Tabla genérica">
  <caption>Tabla 2.- Título especial</caption>
  <tr>
    <td>1</td>
    <td>2</td>
```

```
    <td>3</td>
  </tr>
  ...
</table>
```

Desafortunadamente, el navegador Opera 9 y versiones anteriores y el navegador Internet Explorer 6 y versiones anteriores no soportan esta propiedad y muestran el título de la tabla siempre encima de sus contenidos:

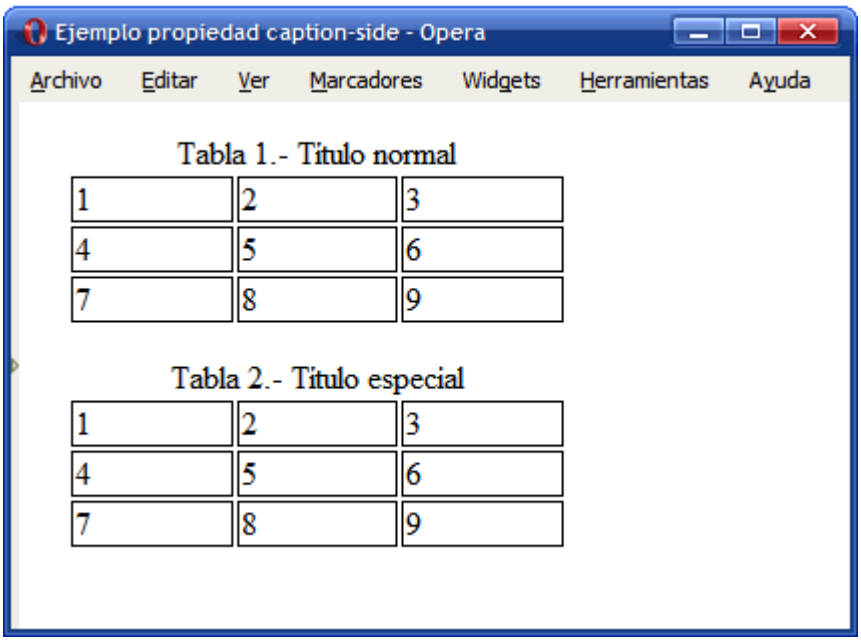


Figura 10.6. Opera e Internet Explorer no soportan la propiedad caption-side

El navegador Firefox sí que soporta esta propiedad y muestra el título de la segunda tabla debajo de sus contenidos, tal y como se ha indicado en el ejemplo:

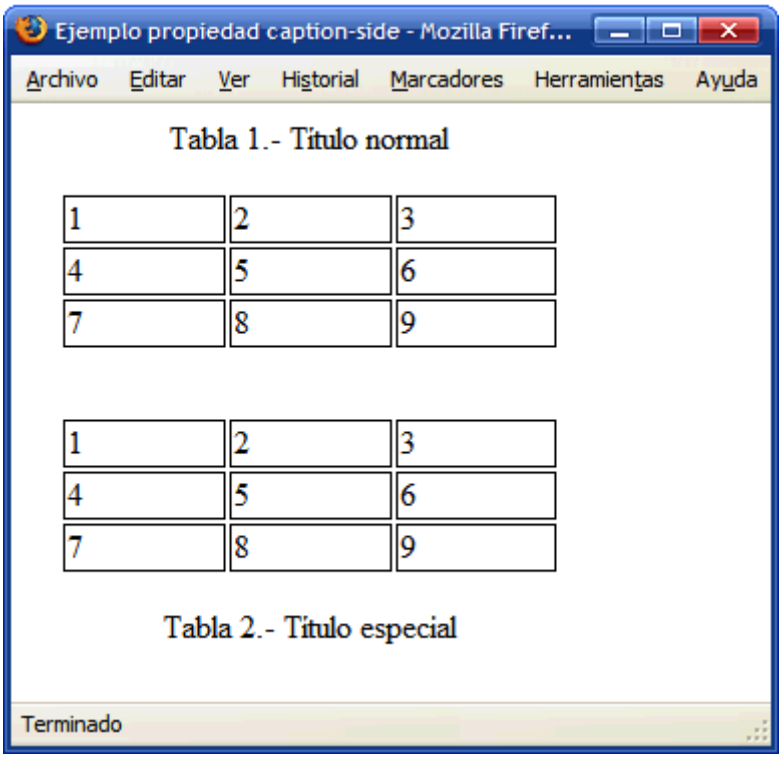


Figura 10.7. Ejemplo de propiedad caption-side

Ejercicio 11 Ver enunciado en la página 192

El resultado final del ejercicio anterior se podría completar añadiendo una pequeña mejora: que el color de las filas varíe cuando el usuario pasa el ratón por encima de cada fila. La *pseudo-clase* `:hover` permite añadir fácilmente esta característica:

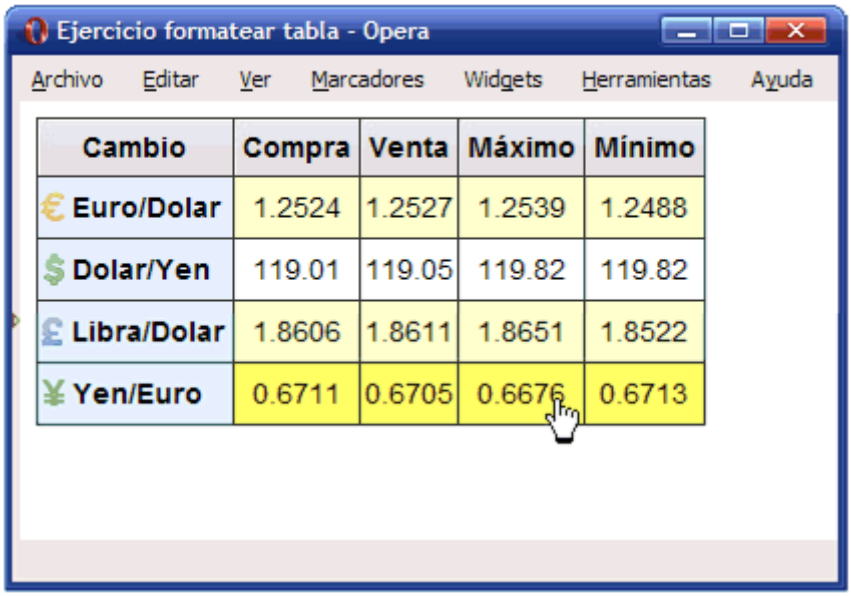


Figura 10.8. Pseudo-clase :hover en las filas de una tabla

La regla CSS necesaria se muestra a continuación:

```
table tr:hover {
  background: #FFFF66;
}
```

Desafortunadamente, Internet Explorer 6 y las versiones anteriores no soportan la *pseudo-clase* `:hover` en elementos diferentes a los enlaces, por lo que se debe recurrir a soluciones con JavaScript para mostrar de otro color la fila activa.

Capítulo 11. Formularios

11.1. Estilos básicos

11.1.1. Mostrar un botón como un enlace

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

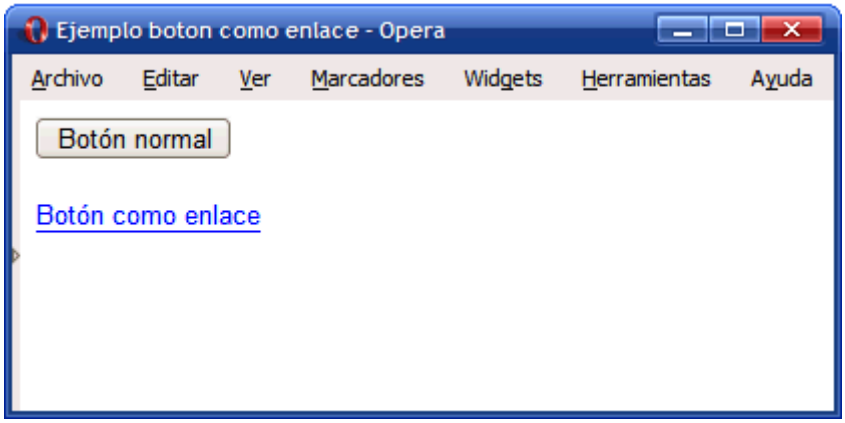


Figura 11.1. Mostrando un botón como si fuera un enlace

Las reglas CSS del ejemplo anterior son las siguientes:

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

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

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

11.1.2. Mejoras en los campos de texto

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

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

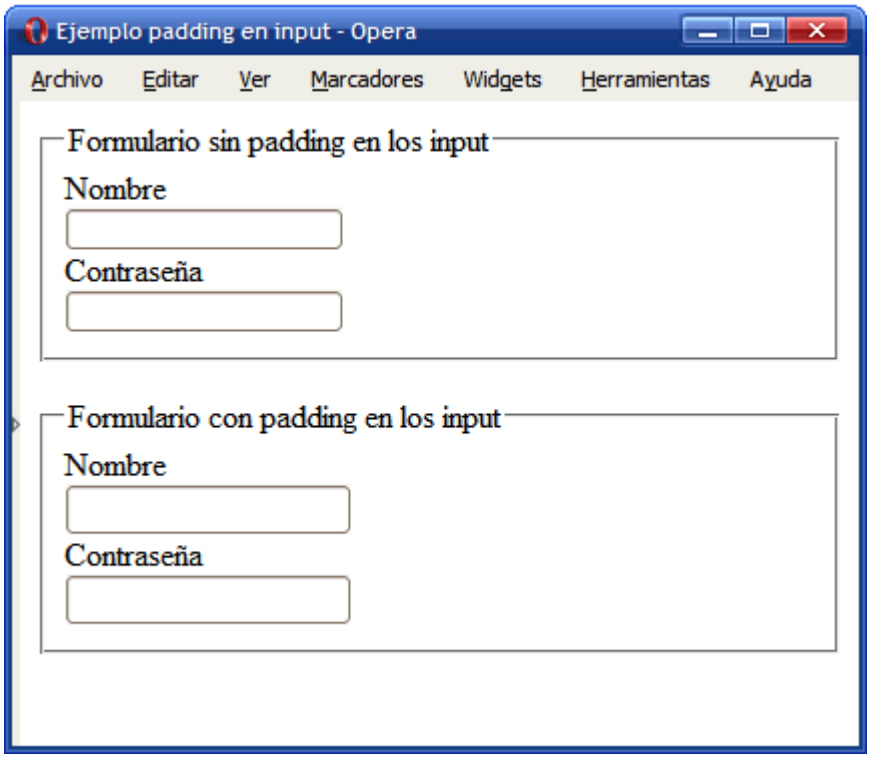


Figura 11.2. Mejorando el aspecto de los formularios gracias al padding

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

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

11.1.3. Labels alineadas y formateadas

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

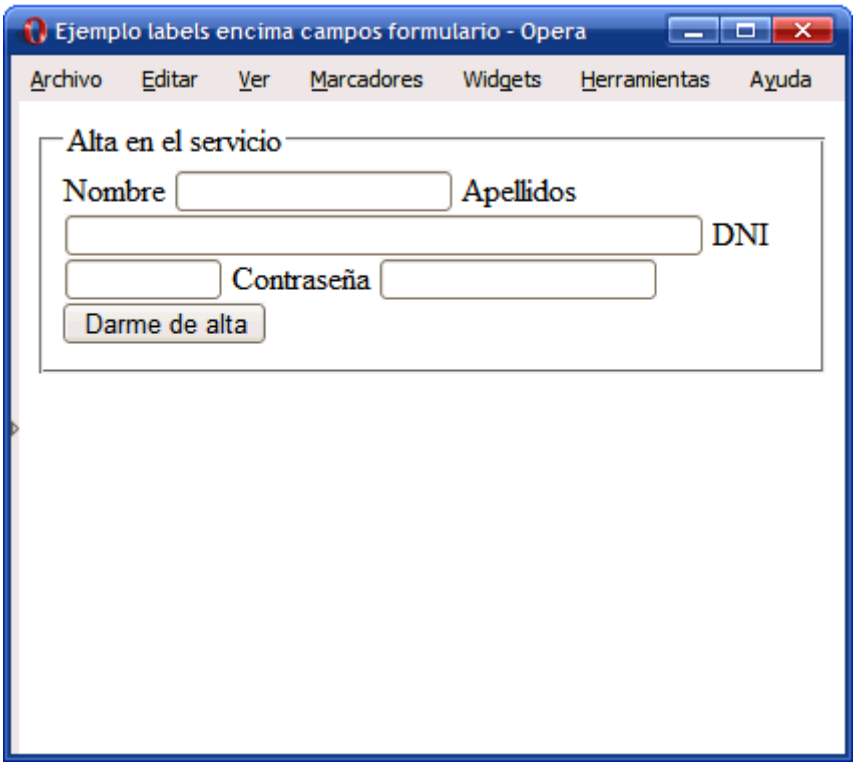


Figura 11.3. Aspecto por defecto que muestran los formularios

El código HTML del ejemplo anterior es el siguiente:

```
<form>
<fieldset>
  <legend>Alta en el servicio</legend>

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

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

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

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

  <input class="btn" type="submit" value="Darme de alta" />
</fieldset>
</form>
```

Aprovechando los elementos `<label>`, se pueden aplicar unos estilos CSS sencillos que permitan mostrar el formulario con el aspecto de la siguiente imagen:

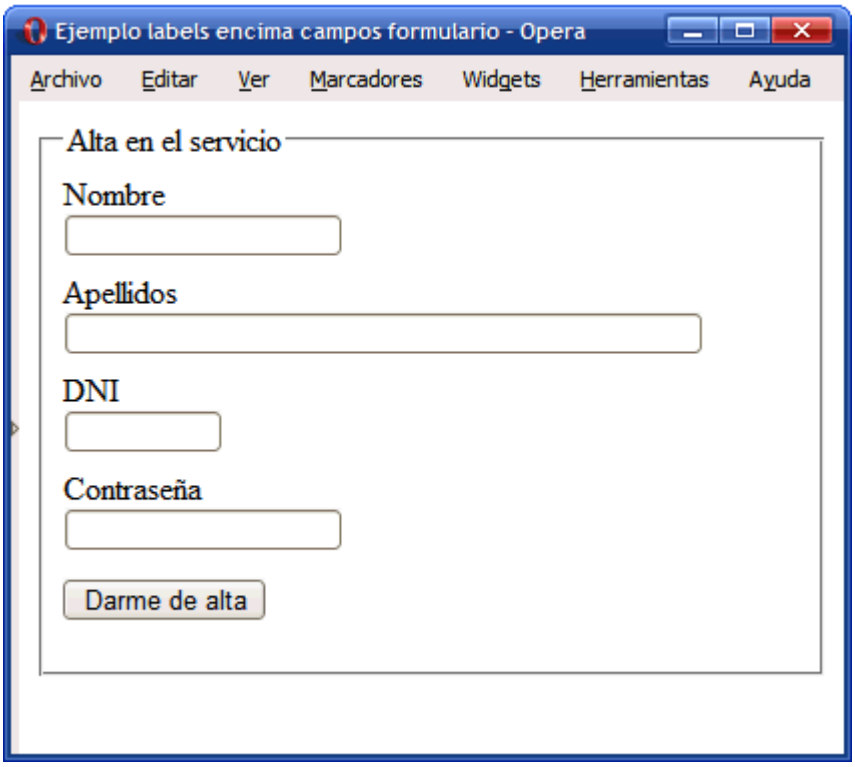


Figura 11.4. Mostrando las etiquetas label encima de los campos del formulario

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

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

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

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

Ejercicio 12 Ver enunciado en la página 195

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

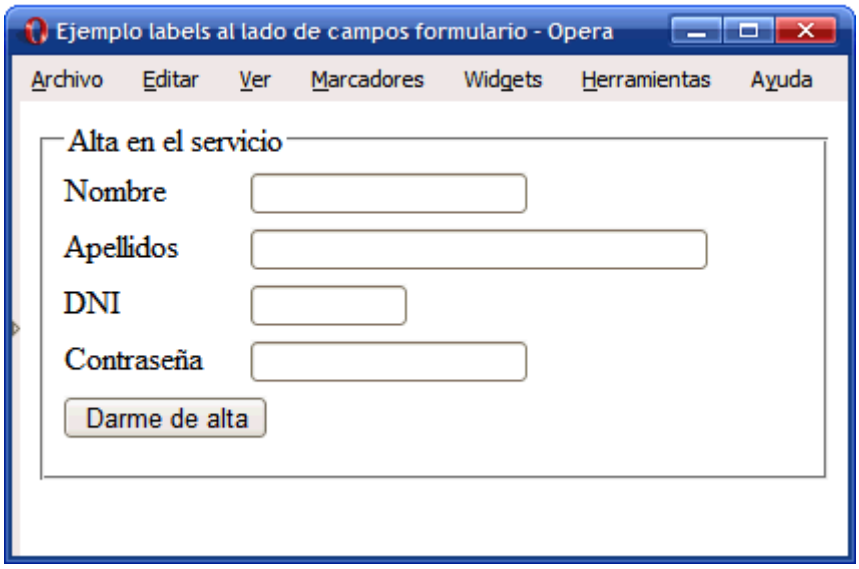


Figura 11.5. Mostrando las etiquetas label alineadas con los campos del formulario

Para mostrar un formulario tal y como aparece en la imagen anterior no es necesario crear una tabla y controlar la anchura de sus columnas para conseguir una alineación perfecta. Sin embargo, sí que es necesario añadir un nuevo elemento (`<div>`) que encierre a cada uno de los campos del formulario (`<label>` y `<input>`). El esquema de la solución propuesta es el siguiente:

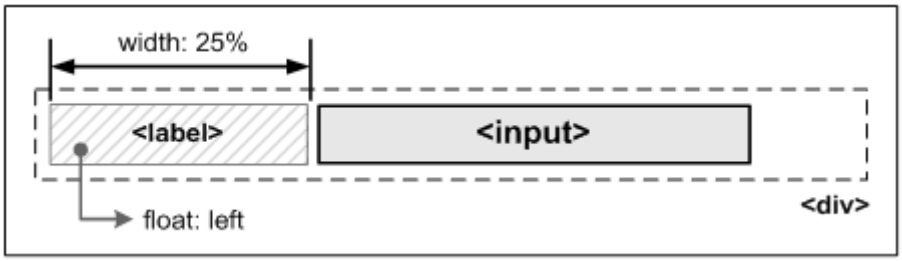


Figura 11.6. Esquema de la técnica de alineación de etiquetas label y campos de formulario

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

```
<form>
  <fieldset>
  <legend>Alta en el servicio</legend>

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

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

```
</fieldset>
</form>
```

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

```
div {
  margin: .4em 0;
}
div label {
  width: 25%;
  float: left;
}
```

11.2. Estilos avanzados

11.2.1. Formulario en varias columnas

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

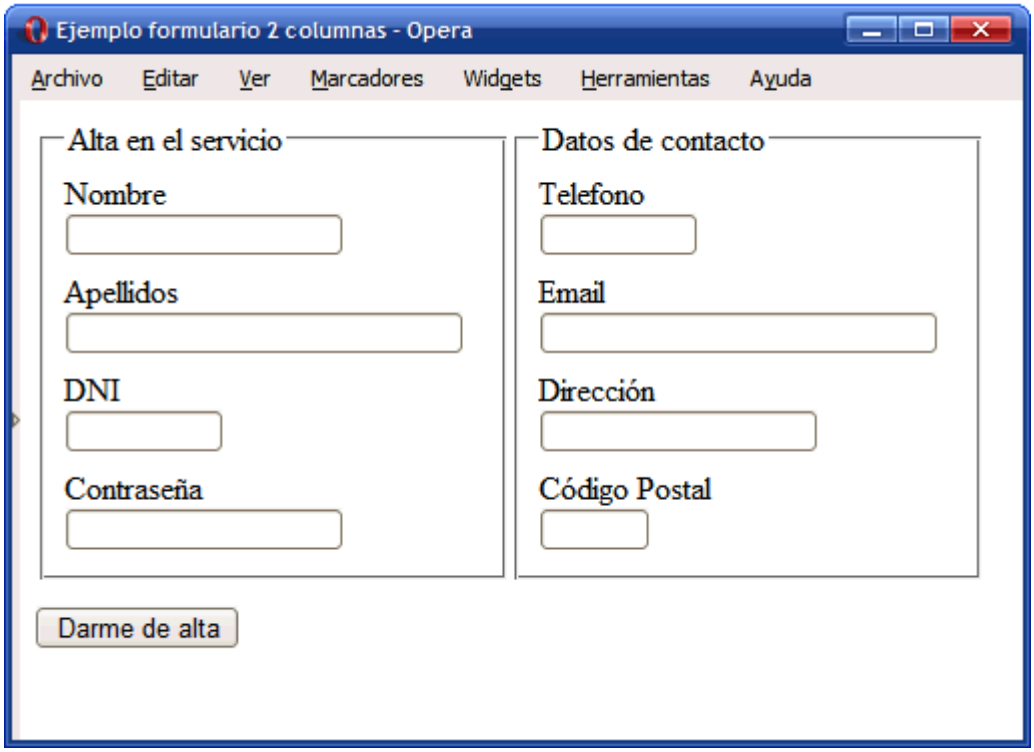


Figura 11.7. Ejemplo de formulario a dos columnas

La solución consiste en añadir un elemento `<div>` que agrupe a cada elemento `<fieldset>` y aplicar las siguientes reglas CSS:

```
form div {
  display: inline;
  float: left;
```

```
  width: 49%;
}

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

  ...
</form>
```

11.2.2. Resaltar el campo seleccionado

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

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

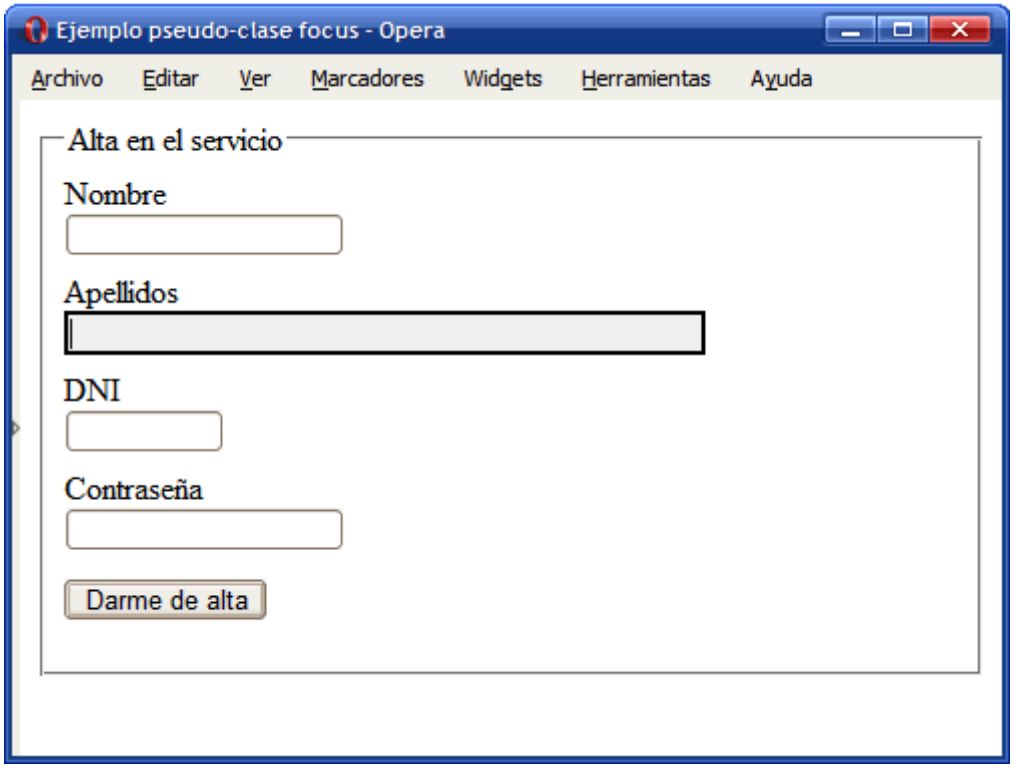


Figura 11.8. Ejemplo de pseudo-clase :focus

La pseudo-clase `:focus` permite definir el estilo del campo seleccionado de forma sencilla:

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

```
  background: #F3F3F3;
}
```

Desafortunadamente, Internet Explorer 6 y las versiones anteriores no soportan la pseudo-clase `:focus` y es preciso recurrir a soluciones con JavaScript.

Capítulo 12. Layout

12.1. Estilos básicos

12.1.1. Centrar una página completa

A medida que aumenta el tamaño y la resolución de las pantallas de ordenador, se hace más difícil diseñar páginas que se adapten completamente al tamaño de la ventana. Con resoluciones superiores a 1024 x 768 píxel, las líneas en las que se muestra el texto son demasiado largas como para leerlas con comodidad. Por ese motivo se suele optar por diseños con una anchura fija limitada a un valor aceptable para mantener la legibilidad del texto.

Por otra parte, los navegadores alinean por defecto las páginas a la izquierda de la ventana, lo que provoca que algunas páginas parezcan muy estrechas y den una sensación de vacío en los monitores con más resolución.

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

Figura 12.1. Página de anchura fija centrada mediante CSS

Figura 12.2. Página de anchura fija centrada mediante CSS

Figura 12.3. Página de anchura fija centrada mediante CSS

La solución se basa en agrupar todos los contenidos de la página en un `<div>` llamado `contenedor` (en inglés se denomina `wrapper`) y asignarle a ese `<div>` unos márgenes laterales automáticos:

```
#contenedor {
  width: 300px;
  margin: 0 auto;
}

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

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

elemento <body>), por lo que se consigue centrar el elemento respecto de la ventana del navegador.

Modificando ligeramente el código CSS anterior se puede conseguir un diseño dinámico (también llamado *líquido*) que se adapta a la anchura de la ventana del navegador y permanece siempre centrado:

```
#contenedor {
  width: 70%;
  margin: 0 auto;
}
```

Indicando la anchura del elemento contenedor como un porcentaje, la anchura de la página se adapta de forma continua a la anchura de la ventana del navegador. Si el usuario minimiza la ventana del navegador, la página se verá estrecha. Si el usuario maximiza la ventana de su navegador, la página se verá muy ancha.

Las siguientes imágenes muestran cómo se adapta el diseño dinámico a la anchura de la ventana del navegador, mostrando cada vez más contenidos a medida que se agranda la ventana.

Figura 12.4. Página de anchura variable centrada mediante CSS

Figura 12.5. Página de anchura variable centrada mediante CSS

Figura 12.6. Página de anchura variable centrada mediante CSS

12.2. Estructura o layout

El diseño tradicional de páginas web se basaba en el uso de tablas para definir las estructuras de las páginas y posicionar cada uno de los bloques que forman las páginas: cabecera, menús, contenidos, columnas laterales, pies de página, etc.

Sin embargo, la estructura basada en tablas es compleja (complica en exceso el código HTML), poco flexible (no permite cambios sencillos en la estructura) y es poco *semántica* (no es sencillo dotar de significado a la estructura).

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

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

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

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

Figura 12.7. Esquema del diseño a 2 columnas con cabecera y pie de página

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

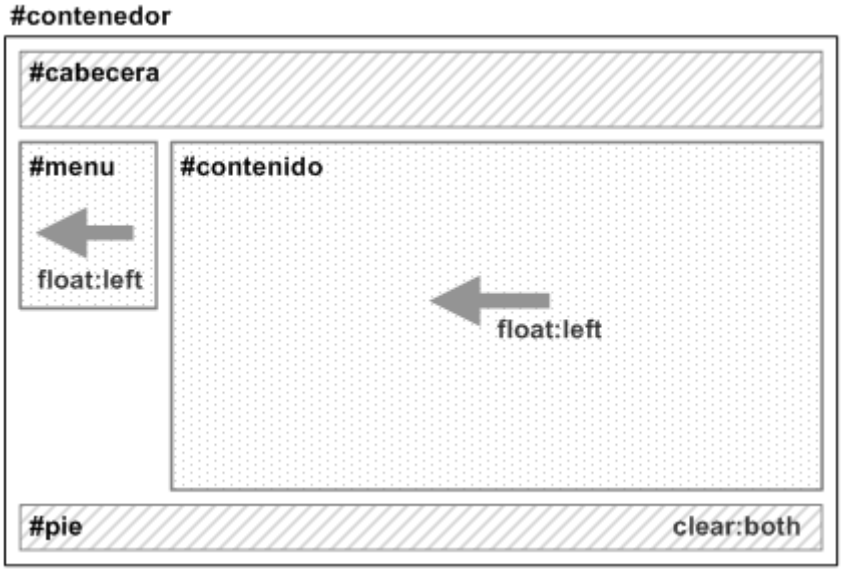


Figura 12.8. Propiedades CSS necesarias en el diseño a dos columnas con cabecera y pie de página

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

```
#contenedor {
  width: 700px;
}
#cabecera {
}
#menu {
```

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

<body>
<div id="contenedor">
  <div id="cabecera">
  </div>

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

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

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

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

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

```
#contenedor {
}
#cabecera {
}
#menu {
  float: left;
  width: 15%;
}
#contenido {
  float: left;
  width: 85%;
}
#pie {
  clear: both;
}
```

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

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

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

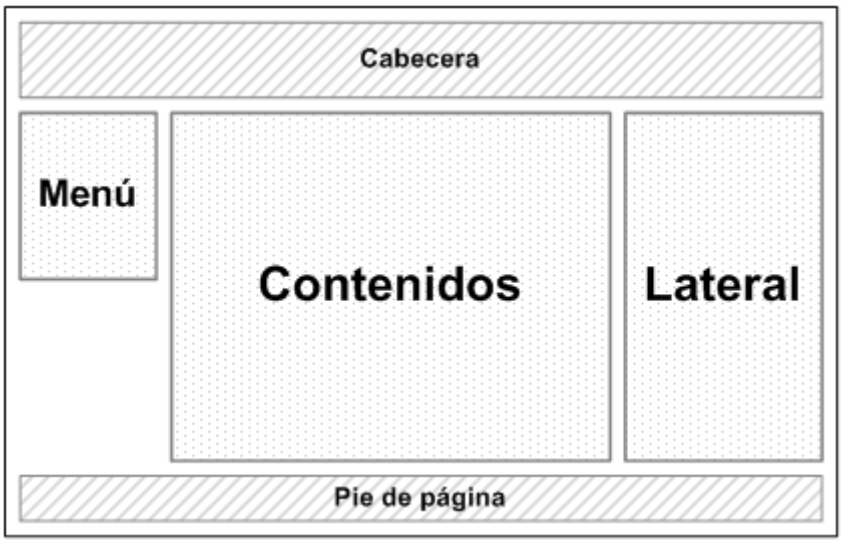


Figura 12.9. Esquema del diseño a tres columnas con cabecera y pie de página

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

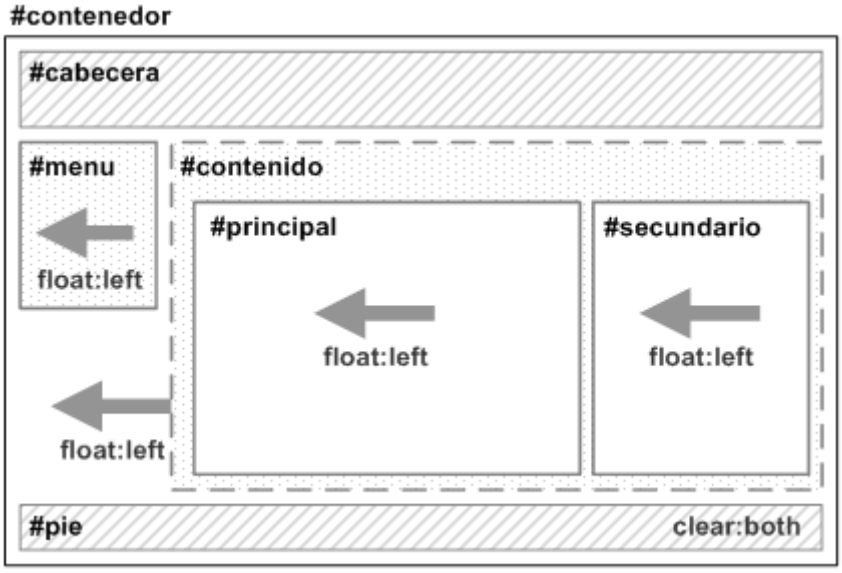


Figura 12.10. Propiedades CSS necesarias en el diseño a 3 columnas con cabecera y pie de página

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

```
#contenedor {
}
#cabecera {
}
#menu {
  float: left;
  width: 15%;
}
#contenido {
  float: left;
  width: 85%;
}
#contenido #principal {
  float: left;
  width: 80%;
}
#contenido #secundario {
  float: left;
  width: 20%;
}

#pie {
  clear: both;
}

<body>
<div id="contenedor">
  <div id="cabecera">
  </div>

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

  <div id="contenido">
    <div id="principal">
    </div>

    <div id="secundario">
    </div>
  </div>

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

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

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

principal de contenidos y `float: right` para el contenedor de los contenidos y la zona secundaria de contenidos.

Ejercicio 13 Ver enunciado en la página 196

12.3. Alturas/anchuras máximas y mínimas

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

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

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

Tabla 12.1. Propiedad max-width

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

Tabla 12.2. Propiedad min-width

`min-width`	**Anchura mínima**
Valores	`<medida> | <porcentaje> | inherit`
Se aplica a	Todos los elementos salvo filas y grupos de filas de tablas
Valor inicial	0
Descripción	Permite definir la anchura mínima de un elemento

Tabla 12.3. Propiedad max-height

`max-height`	**Altura máxima**
Valores	`<medida> | <porcentaje> | none | inherit`
Se aplica a	Todos los elementos salvo columnas y grupos de columnas de tablas
Valor inicial	none

max-height	Altura máxima
Descripción	Permite definir la altura máxima de un elemento

Tabla 12.4. Propiedad min-height

min-height	Altura mínima
Valores	`<medida> | <porcentaje> | inherit`
Se aplica a	Todos los elementos salvo columnas y grupos de columnas de tablas
Valor inicial	0
Descripción	Permite definir la altura mínima de un elemento

De esta forma, para conseguir un diseño de anchura variable pero controlada, se podrían utilizar reglas CSS como la siguiente:

```
#contenedor {
  min-width: 500px;
  max-width: 900px;
}
```

Las propiedades que definen la altura y anchura máxima y mínima se pueden aplicar a cualquier elemento, aunque solamente suelen utilizarse para estructurar la página. En general, las propiedades más utilizadas son `max-width` y `min-width`, ya que no es muy habitual definir alturas máximas y mínimas.

Desafortunadamente, Internet Explorer 6 y las versiones anteriores no soportan ninguna de las cuatro propiedades sobre ningún elemento. Hasta que se incorpore en las nuevas versiones del navegador, es preciso recurrir a trucos que simulen el comportamiento de las propiedades:

`max-width` equivalente para Internet Explorer:

```
div {
  max-width: 800px;
  width: expression(document.body.clientWidth > 801? "800px": "auto");
}
```

`min-width` equivalente para Internet Explorer:

```
div {
  min-width:800px;
  width: expression(document.body.clientWidth < 801? "800px": "auto" );
}
```

`max-height` equivalente para Internet Explorer:

```
div {
  max-height: 300px;
  overflow: hidden;
  height: expression(this.scrollHeight > 301? "300px" : "auto" );
}
```

`min-height` equivalente para Internet Explorer:

```
div {
  min-height:300px;
  overflow: hidden;
  height: expression(this.scrollHeight < 301? "300px" : "auto" );
}
```

Los equivalentes para Internet Explorer han sido extraídos de: `http://www.svendtofte.com/code/max_width_in_ie/`

12.4. Estilos avanzados

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

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

Desde la aparición de este problema se han presentado numerosas soluciones. La más conocida es la técnica *faux columns* (*"columnas falsas"*) y que simula el color/imagen de fondo de las columnas laterales mediante la imagen de fondo de la columna central de contenidos.

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

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

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

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

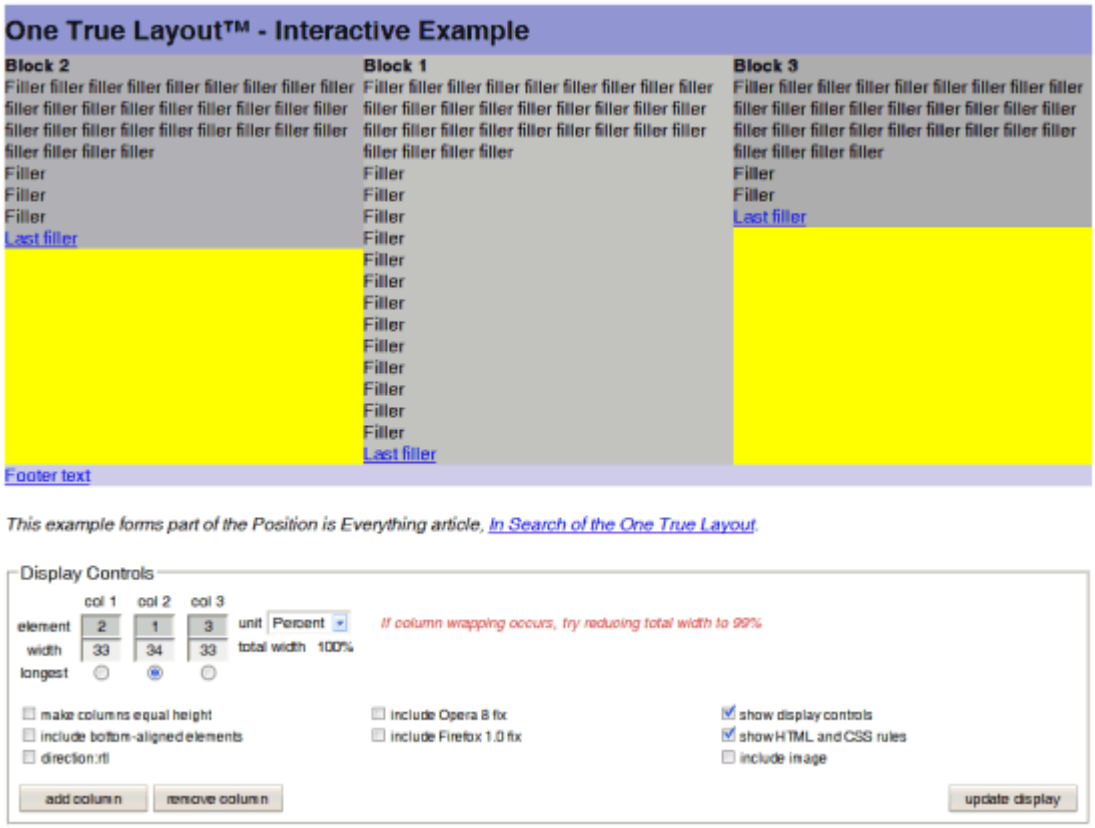


Figura 12.11. Herramienta online para diseñar layouts de varias columnas con CSS

La herramienta interactiva se puede encontrar en: `http://www.fu2k.org/alex/css/onetruelayout/example/interactive`

Capítulo 13. Otros

13.1. Propiedades shorthand

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

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

Tabla 13.1. Propiedad font

font	Tipografía
Valores	((<font-style> || <font-variant> || <font-weight>)? <font-size> (/ <line-height>)? <font-family>) | caption | icon | menu | message-box | small-caption | status-bar | inherit
Se aplica a	Todos los elementos
Valor inicial	-
Descripción	Permite indicar de forma directa todas las propiedades de la tipografía de un texto

Tabla 13.2. Propiedad margin

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

Tabla 13.3. 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

Tabla 13.4. 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

Tabla 13.5. 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

Tabla 13.6. Propiedad list-style

`list-style`	**Estilo de una lista**
Valores	`( <list-style-type> || <list-style-position> || <list-style-image> ) | inherit`
Se aplica a	Elementos de una lista
Valor inicial	-
Descripción	Propiedad que permite establecer de forma simultanea todas las opciones de una lista

13.2. Versión para imprimir

Muchos de los elementos que incluyen las páginas web carecen de sentido en otros medios como el medio impreso. Uno de los elementos básicos que pierde su significado al cambiar de medio es el enlace.

La solución más sencilla sería la de eliminar todos los enlaces cuando se imprime el texto de una página. Otra solución alternativa sería la de crear al final de la impresión una tabla con las direcciones de los enlaces del texto.

Utilizando los medios definidos por CSS, es posible adaptar fácilmente las páginas web para que no se impriman los elementos propios del medio online.

El sitio web *A List Apart* (`www.alistapart.com`) es un buen ejemplo de cómo crear una versión específica para impresora de las páginas web originales. Los artículos de este sitio web muestran el logotipo del sitio, el menú principal de navegación, una barra lateral de utilidades que incluye un buscador y enlaces a los comentarios del artículo:

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

El mismo artículo cuando se imprime muestra el siguiente aspecto:

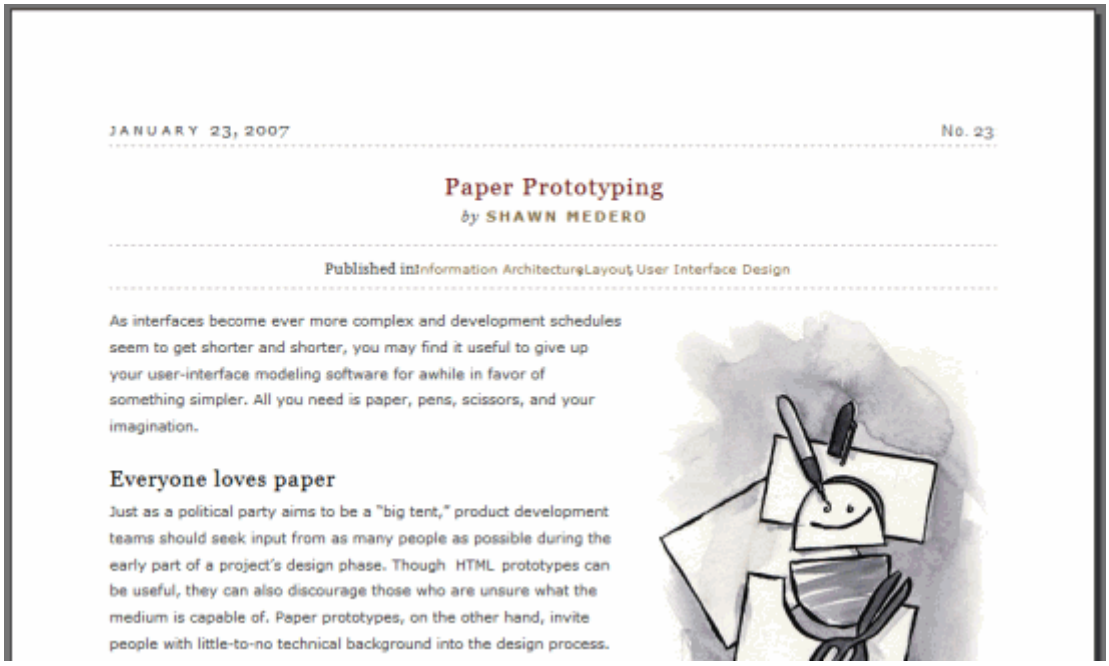


Figura 13.2. Aspecto de un artículo de A List Apart como se ve cuando se imprime

En un artículo impreso es absurdo incluir el formulario de búsqueda, un menú de navegación o el enlace a los comentarios del artículo, por lo que este sitio web define una hoja

de estilos específica para la impresora que permite imprimir los artículos de forma más adecuada.

Para crear la *"versión para imprimir"* de una página web, se utiliza el medio `print` definido por CSS. Por lo tanto, en primer lugar se crea una nueva hoja de estilos y se enlaza específicamente para la impresora:

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

En general, las hojas de estilos para impresoras son muy sencillas comparadas con las hojas de estilos para la web y siempre suelen construirse siguiendo estos pasos:

1) Ocultar los elementos que no se van a imprimir:

```
#cabecera, #menu, #lateral, #comentarios {
  display: none !important;
}
```

Los bloques (normalmente encerrados en elementos de tipo `<div>`) que no se van a imprimir se ocultan con la propiedad `display` y su valor `none`. La palabra clave `!important` aumenta la prioridad de esta regla CSS y más adelante se explica su significado.

2) Corregir la estructura de la página:

```
body, #contenido, #principal, #pie {
  float: none !important;
  width: auto !important;
  margin: 0 !important;
  padding: 0 !important;
}
```

Normalmente, las páginas web complejas están formadas por bloques posicionados mediante `float`. Si se elimina el bloque que contiene el menú de navegación, quizás es conveniente no posicionar mediante `float` el bloque de contenidos y aumentar su anchura.

3) Modificar los colores y tipos de letra:

```
body { color: #000; font: 100%/150% Georgia, "Times New Roman", Times, serif; }
```

Aunque el uso de impresoras en color es mayoritario, suele ser conveniente imprimir todo el texto de las páginas de color negro, para ahorrar costes y para aumentar el contraste cuando se imprime sobre hojas de color blanco.

13.3. Personalizar el cursor

CSS no permite modificar los elementos propios del navegador o de la interfaz de usuario del sistema operativo. Sin embargo, el puntero del ratón es una excepción muy importante, ya que se puede modificar mediante la propiedad `cursor`.

Tabla 13.7. Propiedad cursor

`cursor`	**Puntero del ratón**
Valores	`( (<url> ,)* ( auto | crosshair | default | pointer | move | e-resize | ne-resize | nw-resize | n-resize | se-resize |`

`cursor`	**Puntero del ratón**
	`sw-resize | s-resize | w-resize | text | wait | help | progress ) ) | inherit`
Se aplica a	Todos los elementos
Valor inicial	auto
Descripción	Permite personalizar el puntero del ratón

La propiedad `cursor` no sólo permite seleccionar un puntero entre los disponibles en el sistema operativo (flecha, mano, reloj de arena, redimensionar, etc.) sino que incluso permite indicar la URL de una imagen que se quiere mostrar como puntero personalizado.

Se pueden indicar varias URL para que CSS intente cargar varias imágenes: si la primera imagen del puntero no se carga o no la soporta el navegador, se pasa a la siguiente imagen y así sucesivamente hasta que se pueda cargar alguna imagen.

El siguiente ejemplo muestra el caso de un puntero definido con varias URL y un valor estándar:

```
:link, :visited { cursor: url(puntero.svg), url(puntero.cur), pointer }
```

Si el navegador soporta las imágenes en formato SVG, el puntero del ratón cambia su aspecto por la imagen `puntero.svg`. Si el navegador no soporta el formato SVG, intenta cargar la siguiente URL que define un puntero en formato `.cur`. Si no se puede cargar correctamente, se mostraría el valor preestablecido `pointer`.

Los valores preestablecidos para el puntero se muestran a continuación:

Puntero	**Navegadores que lo soportan**
Figura 13.3. cursor: default	Todos
Figura 13.4. cursor: crosshair	Todos
Figura 13.5. cursor: hand	Solo Internet Explorer
Figura 13.6. cursor: pointer	Todos salvo Internet Explorer

Figura 13.7. cursor:pointer; cursor: hand	Todos
Figura 13.8. cursor: move	Todos
Figura 13.9. cursor: text	Todos
Figura 13.10. cursor: wait	Todos
Figura 13.11. cursor: help	Todos
Figura 13.12. cursor: n-resize	Todos
Figura 13.13. cursor: ne-resize	Todos
Figura 13.14. cursor: e-resize	Todos
Figura 13.15. cursor: se-resize	Todos
Figura 13.16. cursor: s-resize	Todos
Figura 13.17. cursor: sw-resize	Todos
	Todos

Figura 13.18. cursor: w-resize	
Figura 13.19. cursor: nw-resize	Todos
Figura 13.20. cursor: progress	Solo Internet Explorer
Figura 13.21. cursor: not-allowed	Solo Internet Explorer
Figura 13.22. cursor: no-drop	Solo Internet Explorer
Figura 13.23. cursor: vertical-text	Solo Internet Explorer
Figura 13.24. cursor: all-scroll	Solo Internet Explorer
Figura 13.25. cursor: col-resize	Solo Internet Explorer
Figura 13.26. cursor: row-resize	Solo Internet Explorer
Figura 13.27. cursor: url(...)	Solo Internet Explorer

El puntero personalizado más utilizado es la opción `cursor: pointer` y `cursor: hand` que muestra en el puntero una mano que puede pinchar sobre el elemento. Otro puntero muy utilizado es `cursor: move` que permite indicar en las aplicaciones web dinámicas los elementos de la página que se pueden mover.

Se puede ver un ejemplo de cada uno de los punteros y la compatibilidad con los diferentes navegadores en la siguiente página: `http://www.echoecho.com/csscursors.htm`

13.4. Hacks y filtros

Los diferentes navegadores y las diferentes versiones de cada navegador incluyen defectos y carencias en su implementación del estándar CSS 2.1. Algunos navegadores no soportan ciertas propiedades, otros las soportan a medias y otros ignoran el estándar e incorporan su propio comportamiento.

De esta forma, diseñar una página compleja que presente un aspecto homogéneo en varios navegadores y varias versiones diferentes de cada navegador es una tarea que requiere mucho esfuerzo. Para facilitar la creación de hojas de estilos homogéneas, se han introducido los filtros y los *hacks*.

A pesar de que utilizar filtros y *hacks* es una solución poco ortodoxa, en ocasiones es la única forma de conseguir que una página web muestre un aspecto idéntico en cualquier navegador.

En primer lugar, los filtros permiten definir u ocultar ciertas reglas CSS para algunos navegadores específicos. Los filtros se definen aprovechando los errores de algunos navegadores (sobre todo los antiguos) a la hora de procesar las hojas de estilos.

Un caso especial de filtro son los comentarios condicionales, que es un mecanismo propietario del navegador Internet Explorer. Los comentarios condicionales permiten incluir hojas de estilos o definir reglas CSS específicamente para una versión de Internet Explorer.

El siguiente ejemplo carga la hoja de estilos `basico_ie.css` solamente para los navegadores de tipo Internet Explorer:

```
<!--[if IE]>
  <style type="text/css">
    @import ("ie.css");
  </style>
<![endif]-->
```

Los navegadores que no son Internet Explorer ignoran las reglas CSS anteriores ya que interpretan el código anterior como un comentario de HTML (gracias a los caracteres `<!-` y `->`) mientras que los navegadores de la familia Internet Explorer lo interpretan como una instrucción propia y válida.

El filtro `[if IE]` indica que esos estilos CSS sólo deben tenerse en cuenta si el navegador es cualquier versión de Internet Explorer. Utilizando comentarios condicionales, también es posible incluir reglas CSS para versiones específicas de Internet Explorer:

```
<!--[if gte IE 6]>
  <style type="text/css">
    @import ("basico_ie6.css");
  </style>
<![endif]-->
```

El anterior ejemplo solamente carga la hoja de estilos `basico_ie6.css` si el navegador es la versión 6 o superior de Internet Explorer.

Una de las mejores listas actualizadas con todos los filtros disponibles para los navegadores de los diferentes sistemas operativos se puede encontrar en `http://centricle.com/ref/css/filters/`

Por otra parte, los *hacks* permiten forzar el comportamiento de un navegador para que se comporte tal y como se espera. Se trata de una forma poco elegante de crear las hojas de estilos y los *hacks* se pueden considerar pequeños *parches* y *chapuzas* que permiten que la hoja de estilos completa se muestre tal y como se espera.

Uno de los *hacks* más conocidos y utilizados es el llamado `* html`. Todas las propiedades CSS que se establezcan mediante el selector `* html` son interpretadas exclusivamente por el navegador Internet Explorer 6 y sus versiones anteriores:

```
div {
  border-bottom: 1px dotted #000;
}
* html div {
  border-bottom: 1px solid #000;
}
```

El ejemplo anterior utiliza el *hack* `* html` para mostrar un borde inferior punteado en los `<div>` en todos los navegadores salvo Internet Explorer 6. Como en este navegador no se muestran correctamente los bordes punteados de 1 píxel de anchura, se decide mostrar un borde formado por una línea continua.

El otro *hack* más conocido y utilizado por su sencillez es el *"underscore hack"*. Las propiedades cuyos nombres se indiquen con un guión bajo por delante, sólo son interpretadas por el navegador Internet Explorer 6 y sus versiones anteriores:

```
#menu {
  position: fixed;
  _position: static;
}
```

Los navegadores más modernos soportan el valor `fixed` para la propiedad `position`, pero Internet Explorer 6 no la soporta. Por este motivo, la regla CSS anterior establece el valor de la propiedad `position` y seguidamente define la propiedad `_position`.

Los navegadores que siguen los estándares rechazan la propiedad `_position`, ya que su nombre no se corresponde con ninguna propiedad válida de CSS. Internet Explorer 6 y las versiones anteriores, consideran correcta tanto `position` como `_position`, por lo que el valor utilizado será el que se haya definido en último lugar (`static` en este caso).

Una de las mejores listas actualizadas con los hacks más útiles para varios navegadores de diferentes sistemas operativos se puede encontrar en: `http://css-discuss.incutio.com/?page=CssHack`

13.5. Prioridad de las declaraciones CSS

Además de las hojas de estilos definidas por los diseñadores, los navegadores aplican a cada página otras dos hojas de estilos: la del navegador y la del usuario.

La hoja de estilos del navegador es la primera que se aplica y se utiliza para establecer el estilo inicial por defecto a todos los elementos HTML (tamaños de letra iniciales, decoración del texto, márgenes entre elementos, etc.)

La otra hoja de estilos externa que se aplica es la que puede definir el usuario mediante su navegador. Aunque la inmensa mayoría de usuarios no utiliza esta característica, en teoría es posible que los usuarios puedan establecer el tipo de letra, color y tamaño de los textos de las páginas que ven en sus navegadores.

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

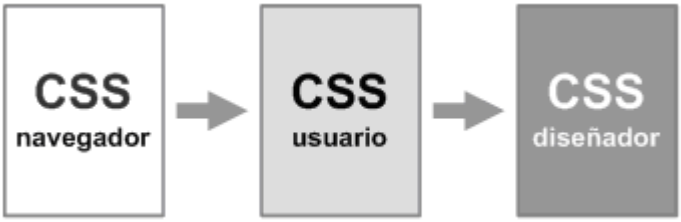


Figura 13.28. Orden en el que se aplican las diferentes hojas de estilos

Por tanto, las reglas que menos prioridad tienen son las del CSS por defecto de los navegadores, ya que son las primeras que se aplican. A continuación se aplican las reglas definidas por los usuarios y por último se aplican las reglas CSS definidas por el diseñador, que por tanto son las que más prioridad tienen.

Además de estas hojas de estilos, CSS define la palabra reservada `!important` para controlar la prioridad de las declaraciones de las diferentes hojas de estilos.

Si a una declaración CSS se le añade la palabra reservada `!important`, se aumenta su prioridad. El siguiente ejemplo muestra el uso de `!important`:

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

Si la primera declaración no tuviera añadido el valor `!important`, el color de los párrafos sería azul, ya que en el caso de declaraciones de la misma importancia, prevalece la indicada en último lugar.

Sin embargo, como la primera declaración se ha marcado como de alta prioridad (gracias al valor `!important`), el color de los párrafos será el rojo.

El valor `!important` no sólo afecta a las declaraciones simples, sino que varía la prioridad de las hojas de estilo. Cuando se indican declaraciones de alta prioridad, el orden en el que se aplican las hojas de estilo es el siguiente:

Figura 13.29. Orden en el que se aplican las diferentes hojas de estilos cuando se utiliza la palabra resevada important

Los estilos del usuario marcados como `!important` tienen más prioridad que los estilos marcados como `!important` en la hoja de estilos del diseñador. De esta forma, ninguna página web puede sobreescribir o redefinir ninguna propiedad de alta prioridad establecida por el usuario.

13.6. Validador

La validación del código CSS y de las reglas que lo forman es un concepto similar a la validación de documentos XHTML.

La validación es un mecanismo que permite comprobar que el código CSS creado cumple las reglas de la sintaxis del lenguaje CSS y que por tanto es una hoja de estilos válida para aplicarla a cualquier documento XHTML.

La validación suele ser útil cuando se producen errores en los estilos definidos o comportamientos no deseados al aplicar las reglas CSS. En muchas ocasiones, los errores se producen porque el navegador está ignorando algunas reglas que contienen propiedades mal definidas o errores de sintaxis.

El W3C (*World Wide Web Consortium*) dispone de un validador online que permite validar reglas CSS sueltas, páginas XHTML con CSS incluído y archivos CSS independientes. El validador se puede acceder en `http://jigsaw.w3.org/css-validator/`

13.7. Recomendaciones generales sobre CSS

13.7.1. Atributos ID y class

El atributo `id` identifica de forma única a cada elemento HTML, por lo que los identificadores deben ser únicos en una misma página. En otras palabras, dos elementos HTML diferentes de una misma página no pueden tener un mismo valor en el atributo `id`.

Por otra parte, el atributo `class` se emplea para indicar la clase o clases a las que pertenece el elemento. Una misma clase se puede aplicar a varios elementos diferentes y un único elemento puede tener asignadas varias clases (se indican separadas por espacios en blanco).

Aunque los dos atributos tienen muchos otros propósitos (sobre todo el atributo `id`), CSS los emplea principalmente como selectores para indicar los elementos sobre los que se aplicarán los diferentes estilos.

En el siguiente ejemplo, las dos listas están formadas por un mismo elemento HTML `<ul>`, pero sus atributos `id` las distinguen de forma adecuada:

```
<ul id="menu">
  ...
</ul>

<ul id="enlaces">
```

```
  ...
</ul>
```

Una de las recomendaciones más importantes relacionadas con el diseño de páginas XHTML y hojas de estilos CSS está relacionada con los valores asignados a los atributos `id` y `class`. Siempre que sea posible, estos atributos se deben utilizar para mejorar la semántica del documento, es decir, para añadir significado a cada elemento de la página.

Por este motivo, los valores asignados a `id` y `class` deberían indicad la función del elemento y no debería estar relacionado con su aspecto o su posición:

Valores incorrectos	Valores correctos
negrita	importante
arial15	titular
verdanaPequena	normal
menuIzquierdo	menuSecundario
letraRoja	error

13.7.2. CLASSitis y DIVitis

Un error común al comenzar a desarrollar páginas con estilos CSS es la utilización excesiva de etiquetas `<div>` y atributos `class`.

Ejemplo de *divitis* y *classitis*:

```
<div id="menu">
<ul class="menu">
  <li class="elemento_menu"><span class="texto_elemento_menu">...</span></li>
  <li class="elemento_menu"><span class="texto_elemento_menu">...</span></li>
  <li class="elemento_menu"><span class="texto_elemento_menu">...</span></li>
  <li class="elemento_menu"><span class="texto_elemento_menu">...</span></li>
</ul>
</div>
```

Los selectores de CSS permiten prescindir de la mayoría de etiquetas `<div>` y atributos `id` y `class`. Diseñar páginas con exceso de etiquetas `<div>` no mejora la semántica del documento y sólo consigue complicar el código HTML.

13.7.3. Estructuración del código CSS

La posibilidad de incluir todo el código CSS en archivos externos exclusivamente dedicados a contener las reglas CSS, permite ordenar de forma lógica las reglas, mejorando su legibilidad y facilitando su actualización.

Las reglas CSS de las hojas de estilos complejas se suelen agrupar según su funcionalidad y se suelen incluir en el siguiente orden:

- Estilos básicos (`<body>`, tipo de letra por defecto, márgenes de `<ul>`, `<ol>` y `<li>`, etc.)

- Estilos de la estructura o layout (anchura, altura y posición de la cabecera, pie de página, zonas de contenidos, menús de navegación, etc.)
- Enlaces (estilos normales, estilos `:hover`, etc.)
- Estilos de cada una de las zonas (elementos de la cabecera, titulares y texto de la zona de contenidos, enlaces, listas e imágenes de las zonas laterales, etc.)

Otra característica común de los mejores sitios web es el uso de comentarios CSS para mejorar la estructura de las hojas de estilos muy largas.

Ejemplo de código CSS estructurado de `http://veerle.duoh.com/`

```
/* Veerle's blog Main stylesheet
-------------------------------------------------------------------*/

/* Wide browser windows
-------------------------------------------------------------------*/
#wrap {
  width:995px;
}


/* Global
-------------------------------------------------------------------*/
html, body, form, h1, h2, h3, h4, h5, h6, p, pre, blockquote, ul, ol, dl {
  margin:0;
  padding:0;
}
ul,li {
  list-style-type:none;
}
...

/* Wide layout
-------------------------------------------------------------------*/
.wide #wrap-main {
  ...
}
...

/* Links
-------------------------------------------------------------------*/
a:link,
a:visited {
  text-decoration:none;
  color:#e45a49;
}

/* Header
-------------------------------------------------------------------*/
#header {
  ...
}
...
```

```
/* Main navigation
-----------------------------------------------------------------*/
ul#nav {
  ...
}
...
```

Ejemplo de código CSS estructurado de `http://www.uxmag.com/`

```
/*        ------------------------------------------------------------
        UX Magazine
        Design | Technology | Strategy | Common Sense
        - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
        Description:        Base setup styles
        Filename:                uxm.base.css
        Version:                1.9
        Date:                        Feb 9, 2006
        ------------------------------------------------------------        */

/*        ------------------------------------------------------------
        Base Body Styles
        - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -        */

/*        ------------------------------------------------------------
        Print Styles
        - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -        */

/*        ------------------------------------------------------------
        Top Bar Styles
        - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -        */

/*        Slogan
        - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -        */
/*        Search Form
        - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -        */
/*        Channels
        - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -        */
```

13.7.4. División de los estilos en varios archivos CSS

Normalmente, los estilos de una página compleja se dividen en varios archivos CSS diferentes para hacerlos más manejables. En primer lugar, se suele utilizar un archivo común que contiene todos los estilos básicos de las páginas HTML del sitio web.

Además, si existe alguna sección especial del sitio web que requiera nuevos estilos, se crea un archivo CSS con todos esos estilos. También es habitual preparar una hoja de estilos específica para impresora y otra preparada para los dispositivos móviles.

Una vez creados los archivos CSS, existen dos estrategias para enlazar varios archivos CSS en las páginas HTML:

Si se puede modificar fácilmente la cabecera del documento (por ejemplo porque las páginas se generan dinámicamente) lo habitual es incluir tantos elementos `<link>` como archivos CSS se enlazan:

```
<head>
  ...
  <link rel="stylesheet" type="text/css" href="/css/basico.css" media="screen" />
  <link rel="stylesheet" type="text/css" href="/css/seccion.css" media="screen" />
  <link rel="stylesheet" type="text/css" href="/css/impresora.css" media="print" />
  <link rel="stylesheet" type="text/css" href="/css/movil.css" media="handheld" />
  ...
</head>
```

Si no se puede modificar de forma sencilla la cabecera de los documentos para añadir, eliminar y modificar los archivos CSS que se enlazan, lo habitual es enlazar un único archivo CSS que se encarga de importar todos los demás:

```
<head>
  ...
  <link rel="stylesheet" type="text/css" href="/css/estilos.css" media="all" />
  ...
</head>
```

El contenido del archivo `estilos.css` debería ser el siguiente para ser equivalente al ejemplo anterior:

```
@import url("basico.css") screen;
@import url("seccion.css") screen;
@import url("impresora.css") print;
@import url("movil.css") handheld;
```

Capítulo 14. Recursos útiles

14.1. Extensiones de Firefox

Si te dedicas profesionalmente al diseño de páginas y aplicaciones web, seguramente tu navegador preferido para trabajar es Firefox. Si no lo conoces, puedes descargar gratuitamente Firefox desde su sitio web oficial: `http://www.mozilla.com/firefox/`.

Las principales ventajas de Firefox desde el punto de vista del diseñador y creador de páginas web es que respeta los estándares del W3C mucho más que los navegadores de la familia Internet Explorer. Además, permite instalar pequeños añadidos, llamados extensiones, que añaden funcionalidades al navegador.

Una extensión se puede considerar como un pequeño programa que se instala dentro del navegador y que añade alguna característica interesante que el navegador no incorpora de serie. Lo mejor de las extensiones de Firefox es que existen cientos de extensiones, prácticamente todas son gratuitas y casi todas son realmente útiles.

El sitio web de Firefox incluye una sección especial llamada "Complementos" en la que se puede encontrar el listado completo de extensiones disponibles para Firefox: `http://addons.mozilla.org/es-ES/firefox/`. A continuación se muestra una pequeña selección de algunas de las extensiones más interesantes para los diseñadores de páginas web.

14.1.1. Firebug

Firebug es probablemente la extensión más útil y completa de todas las que están relacionadas con el diseño web. No importa si tu especialidad es XHTML, CSS, JavaScript, DOM o Ajax, ya que Firebug proporciona toda la información posible sobre cada uno de estos temas. Como Firebug tiene tantas opciones, lo mejor es instalar la extensión y probarla en tus proyectos web.

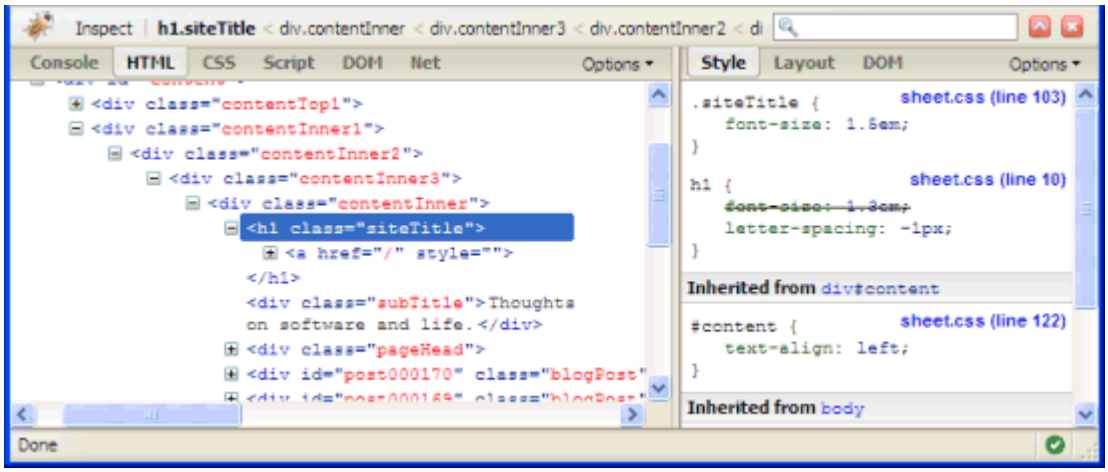


Figura 14.1. Extensión Firebug para Firefox

Probablemente, después de varios meses de uso de Firebug, seguirás descubriendo nuevas opciones muy útiles y seguirás agradeciendo a su creador las horas de trabajo que te ha ahorrado.

Descargar Firebug: `https://addons.mozilla.org/firefox/1843/`

14.1.2. Web Developer

Antes de que existiera Firebug, la extensión *Web Developer* era la más útil para los diseñadores web. Se trata de una barra que se instala junto con el resto de herramientas del navegador y que básicamente se puede utilizar para obtener información sobre la página:

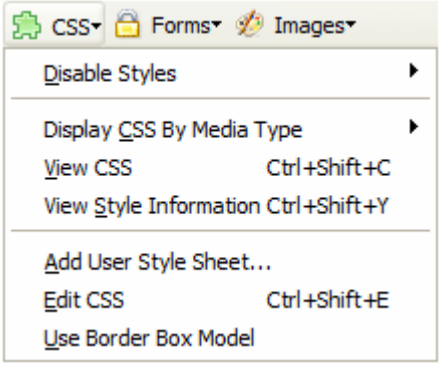


Figura 14.2. Extensión Web Developer para Firefox

Además de proporcionar información, la extensión Web Developer incluye utilidades que Firebug todavía no incorpora, como la redimensión de la ventana del navegador a las dimensiones estándar, recuadrar todos los elementos de un determinado tipo (celdas de tabla, divs, etc.), mostrar una lupa, una regla redimensionable, mostrar los elementos de tipo `hidden`, mostrar la ruta de cada imagen, etc.

Descargar Web Developer: `https://addons.mozilla.org/firefox/60/`

14.1.3. HTML Validator

Una buena práctica, y un requisito impuesto por muchos clientes, es que las páginas XHTML creadas sean válidas y por tanto, pasen el validador de HTML y de CSS disponible en el W3C. Para facilitar la validación de las páginas, esta extensión indica en todo momento los errores y las recomendaciones sobre el código HTML de la página que muestra el navegador.

Figura 14.3. Extensión HTML Validator para Firefox

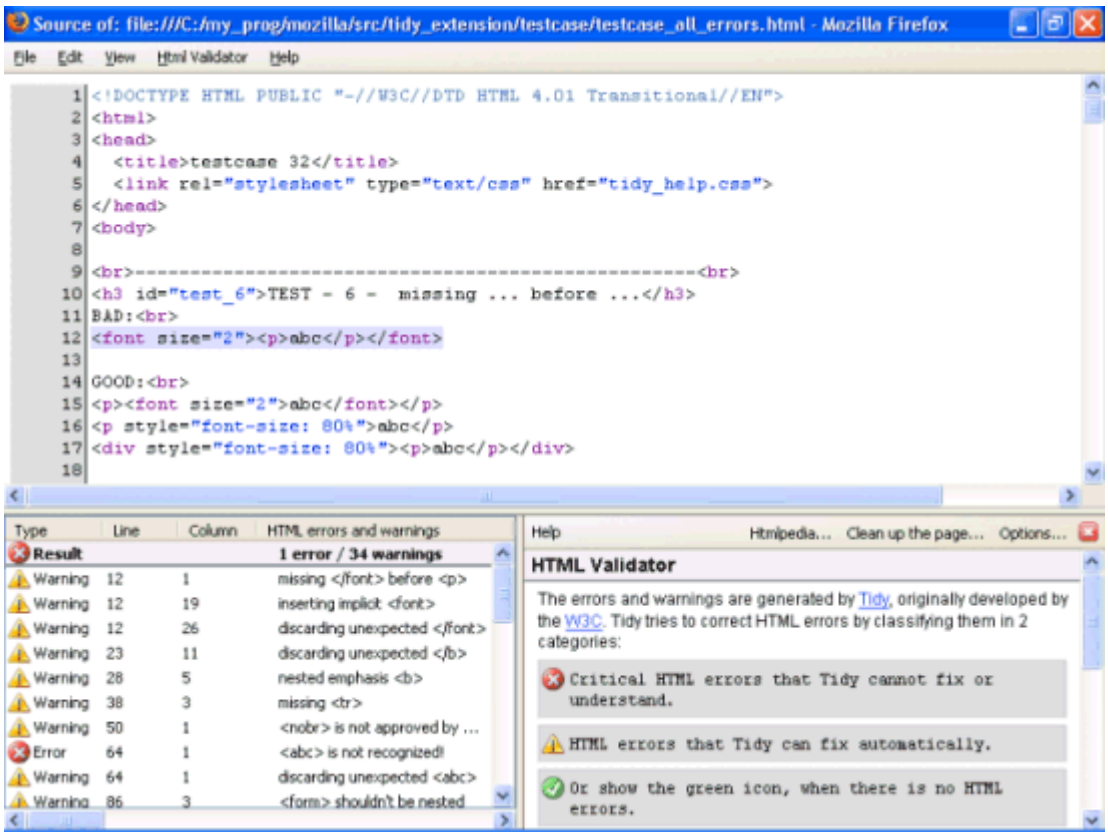


Figura 14.4. Extensión HTML Validator para Firefox

Descargar HTML Validator: `https://addons.mozilla.org/firefox/249/`

14.1.4. Otras extensiones

ColorZilla permite obtener los colores de cualquier elemento de la página mediante una herramienta similar a la de los programas de diseño gráfico:

Figura 14.5. Extensión ColorZilla para Firefox

Descargar ColorZilla: `https://addons.mozilla.org/firefox/271/`

MeasureIt permite medir la altura y anchura de cualquier elemento de la página:

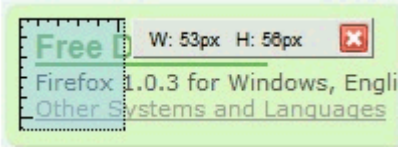


Figura 14.6. Extensión MeasureIt para Firefox

Descargar MeasureIt: `https://addons.mozilla.org/firefox/539/`

View Source Chart, muestra el código fuente de una página estructurado según el anidamiento de los diferentes elementos. Facilita el aprendizaje de HTML y mejora la búsqueda de posibles errores en el código:

Figura 14.7. Extensión View Source Chart para Firefox

Descargar View Source Chart: `https://addons.mozilla.org/firefox/655/`

14.2. Enlaces de interés

14.2.1. Recomendación oficial

Todas las recomendaciones oficiales de CSS, herramientas y otro tipo de información se puede encontrar en la página oficial del W3C:

`http://www.w3.org/Style/CSS/`

14.2.2. Recursos

Web Developers Handbook reúne miles de enlaces con todos los recursos útiles para diseñadores web. `http://www.alvit.de/handbook/`

Blue Vertigo también reúne varios centenares de enlaces con recursos útiles para diseñadores web. `http://www.bluevertigo.com.ar/bluevertigo.htm`

14.2.3. Foros

Los foros de **Foros del Web** disponen de foros específicos dedicados al diseño web: `http://www.forosdelweb.com/`

Ovillo es una de las mejores listas de distribución en castellano: `http://www.ovillo.org/`

En inglés, los foros de **SitePoint** son una buena fuente de información sobre diseño en general, diseño web, programación web, etc. `http://www.sitepoint.com/forums/`

14.2.4. Galerías de páginas

Muchas veces es útil disponer de buenos ejemplos de páginas diseñadas completamente con CSS para tomarlas como referencia.

Web Creme incluye diariamente varios ejemplos de las mejores páginas diseñadas con CSS. `http://www.webcreme.com/`

CSS Remix muestra centenares de páginas diseñadas exclusivamente con CSS y con la posibilidad de puntuar su diseño. `http://www.cssremix.com/`

css Zen Garden es una galería diferente a las tradicionales, pero se ha convertido en una referencia en cuanto a diseños complejos realizados mediante CSS. `http://www.csszengarden.com/`

Capítulo 15. Ejercicios

15.1. Ejercicio 1

A partir del código HTML y CSS que se muestra, añadir los selectores CSS que faltan para aplicar los estilos deseados. Cada regla CSS incluye un comentario en el que se explica los elementos a los que debe aplicarse:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/
xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
<title>Ejercicio de selectores</title>
<style type="text/css">
/* Todos los elementos de la pagina */
{ font: 1/1.3em Arial, Helvetica, sans-serif; }

/* Todos los parrafos de la pagina */
{ color: #555; }

/* Todos los párrafos contenidos en #primero */
{ color: #336699; }

/* Todos los enlaces la pagina */
{ color: #CC3300; }

/* Los elementos "em" contenidos en #primero */
{ background: #FFFFCC; padding: .1em; }

/* Todos los elementos "em" de clase "especial" en toda la pagina */
{ background: #FFCC99; border: 1px solid #FF9900; padding: .1em; }

/* Elementos "span" contenidos en .normal */
{ font-weight: bold; }

</style>
</head>

<body>

<div id="primero">
<p>Lorem ipsum dolor sit amet, <a href="#">consectetuer adipiscing elit</a>. Praesent
blandit nibh at felis. Sed nec diam in dolor vestibulum aliquet. Duis ullamcorper, nisi
non facilisis molestie, <em>lorem sem aliquam nulla</em>, id lacinia velit mi
vestibulum enim.</p>

</div>

<div class="normal">
<p>Phasellus eu velit sed lorem sodales egestas. Ut feugiat. <span><a href="#">Donec
porttitor</a>, magna eu varius luctus,</span> metus massa tristique massa, in imperdiet
est velit vel magna. Phasellus erat. Duis risus. <a href="#">Maecenas dictum</a>, nibh
```

```
vitae pellentesque auctor, tellus velit consectetuer tellus, tempor pretium felis
tellus at metus.</p>

<p>Cum sociis natoque <em class="especial">penatibus et magnis</em> dis parturient
montes, nascetur ridiculus mus. Proin aliquam convallis ante. Pellentesque habitant
morbi tristique senectus et netus et malesuada fames ac turpis egestas. Nunc aliquet.
Sed eu metus. Duis justo.</p>

<p>Donec facilisis blandit velit. Vestibulum nisi. Proin volutpat, <em
class="especial">enim id iaculis congue</em>, orci justo ultrices tortor, <a
href="#">quis lacinia eros libero in eros</a>. Sed malesuada dui vel quam. Integer at
eros.</p>
</div>

</body>
</html>
```

15.2. Ejercicio 2

A partir del código HTML y CSS que se muestra:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/
xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
<title>Ejercicio Fusión Márgenes</title>
<style type="text/css">
div {
  border: 2px solid black;
  width: 50px;
}
</style>
</head>

<body>
  <div>1</div>
  <div>2</div>
  <div>3</div>
</body>
</html>
```

1. Añadir un margen superior de 30 píxel a todos los `<div>`
2. Añadir un margen inferior de 30 píxel a todos los `<div>`

15.3. Ejercicio 3

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

Figura 15.1. Página original

1. El elemento `#cabecera` debe tener un relleno de `1em` en todos los lados.
2. El elemento `#menu` debe tener un relleno de `0.5em` en todos los lados y un margen inferior de `0.5em`.
3. El resto de elementos debe tener `0.5em` de relleno en todos sus lados, salvo el elemento `#pie`, que solo debe tener relleno en la zona superior e inferior.
4. Los elementos `.articulo` deben mostrar una separación entre ellos de `1em`.
5. Las imágenes de los artículos muestran un margen de `0.5em` en todos sus lados.
6. El elemento `#publicidad` está separado `1em` de su elemento superior.
7. El elemento `#pie` debe tener un margen superior de `1em`.

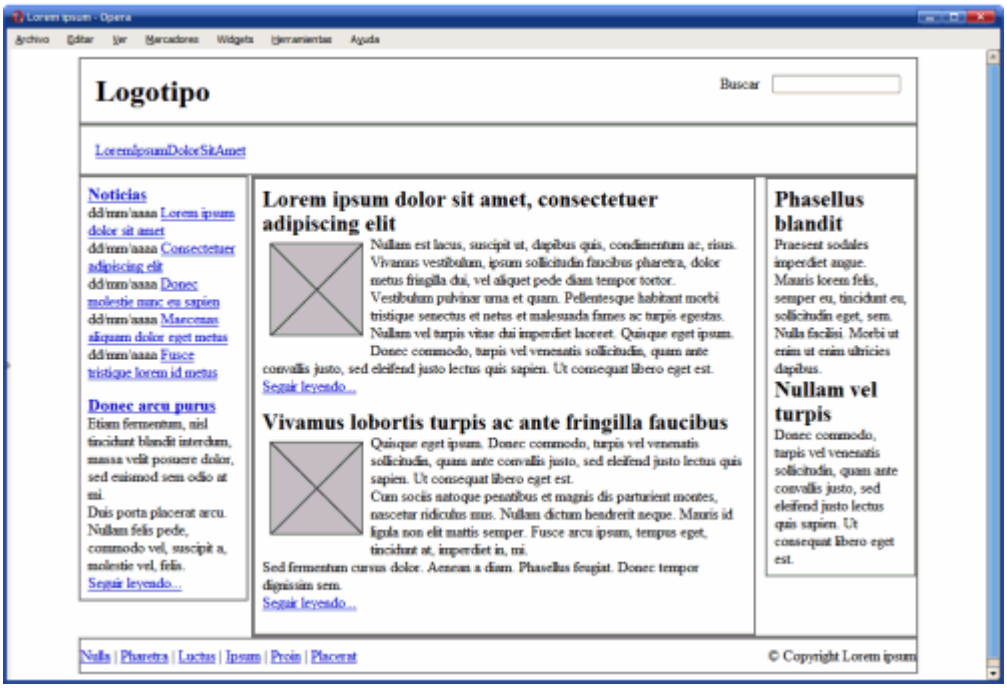


Figura 15.2. Página con márgenes y rellenos

15.4. Ejercicio 4

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

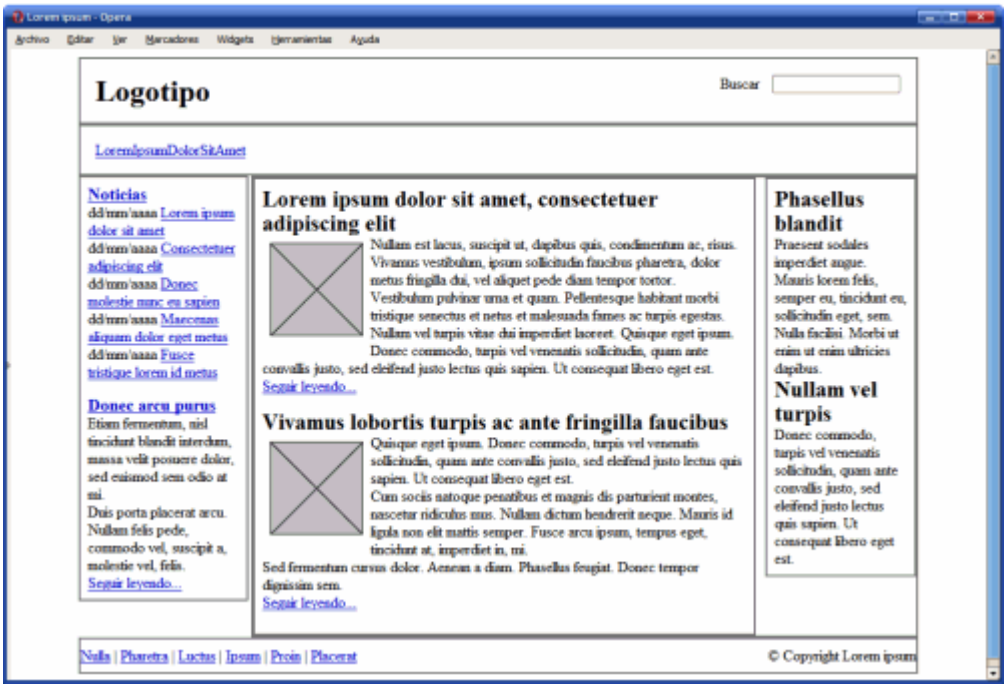


Figura 15.3. Página original

1. Eliminar el borde gris que muestran por defecto todos los elementos.
2. El elemento `#menu` debe tener un borde inferior de 1 píxel y azul (`#004C99`).

3. El elemento `#noticias` muestra un borde de 1 píxel y gris claro (`#C5C5C5`).
4. El elemento `#publicidad` debe mostrar un borde discontinuo de 1 píxel y de color `#CC6600`.
5. El lateral formado por el elemento `#secundario` muestra un borde de 1 píxel y de color `#CC6600`.
6. El elemento `#pie` debe mostrar un borde superior y otro inferior de 1 píxel y color gris claro `#C5C5C5`.

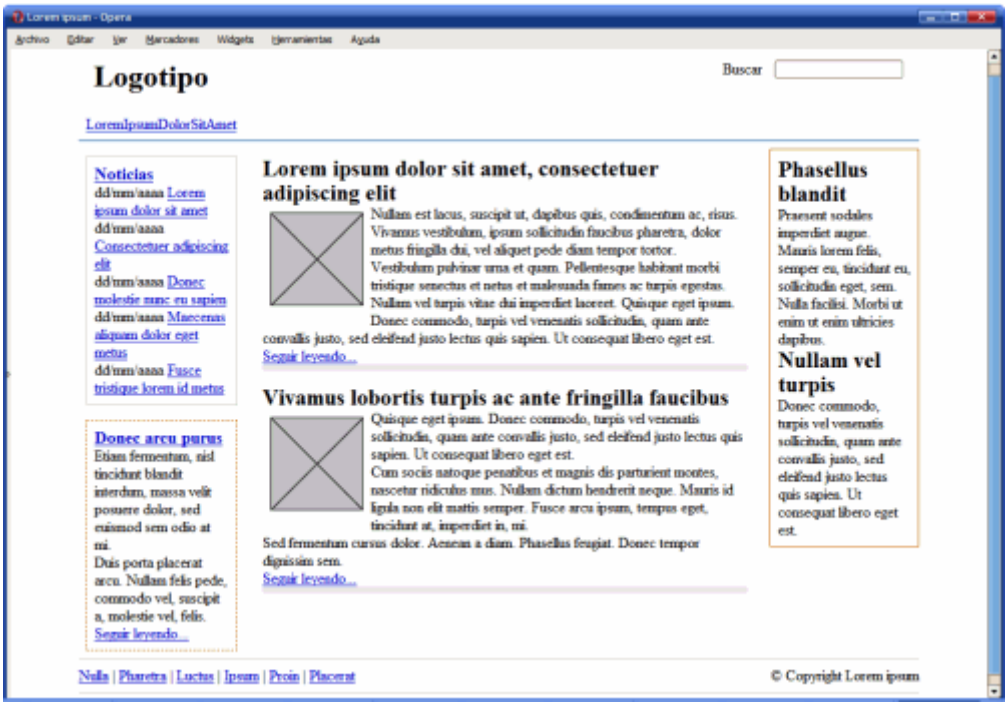


Figura 15.4. Página con bordes

15.5. Ejercicio 5

A partir del código HTML y CSS proporcionados, determinar las reglas CSS necesarias para añadir los siguientes colores e imágenes de fondo:

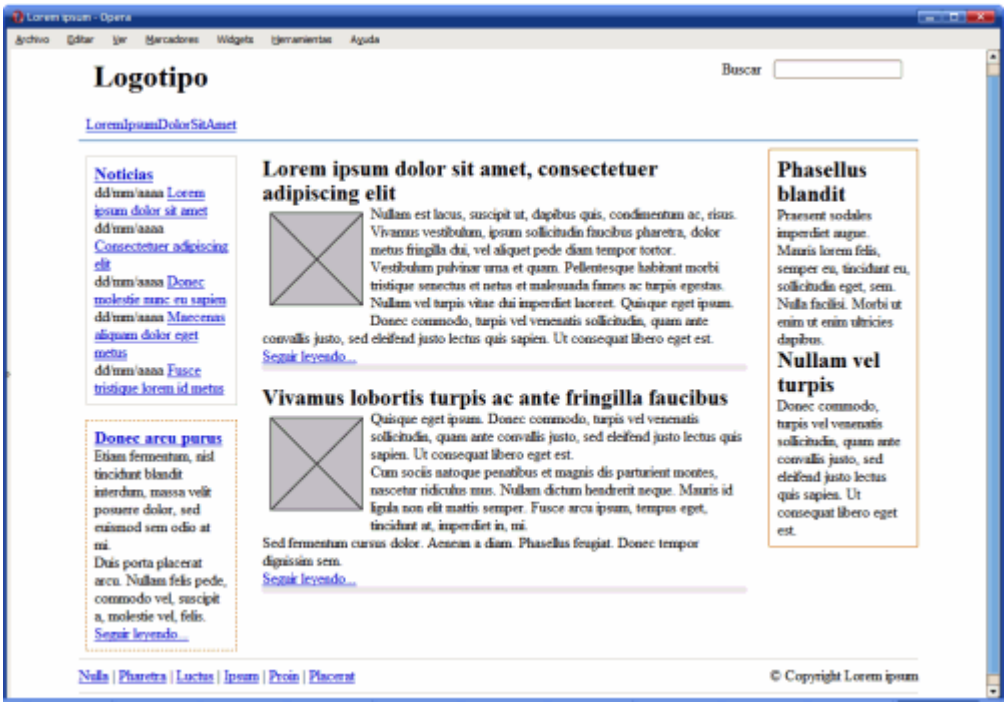


Figura 15.5. Página original

1. Los elementos `#noticias` y `#pie` tiene un color de fondo gris claro (`#F8F8F8`).
2. El elemento `#publicidad` muestra un color de fondo amarillo claro (`#FFF6CD`).
3. Los elementos `<h2>` del lateral `#secundario` muestran un color de fondo `#DB905C` y un pequeño padding de `.2em`.
4. El fondo del elemento `#menu` se construye mediante una pequeña imagen llamada `fondo_menu.gif`.
5. El logotipo del sitio se muestra mediante una imagen de fondo del elemento `<h1>` contenido en el elemento `#cabecera` (la imagen se llama `logo.gif`).

Figura 15.6. Página con colores e imágenes de fondo

15.6. Ejercicio 6

A partir del código HTML proporcionado:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/
xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
  <title>Ejercicio posicionamiento float</title>
  <style type="text/css">
  </style>
</head>

<body>
  <div>
    &laquo; Anterior   Siguiente &raquo;
  </div>
</body>
</html>
```

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

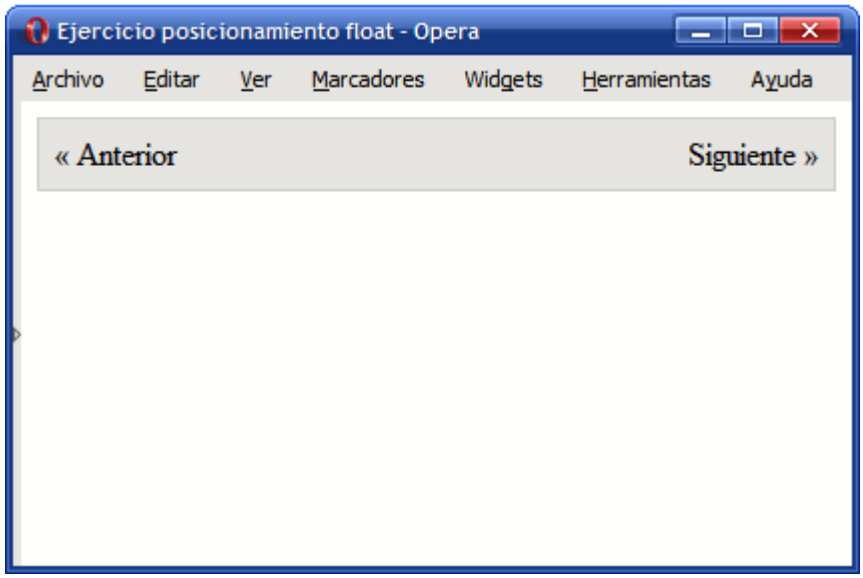


Figura 15.7. Elementos posicionados mediante float

15.7. Ejercicio 7

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

Figura 15.8. Página original

1. La fuente base de la página debe ser: color negro, tipo Arial, tamaño `0.9em`, interlineado `1.4em`.
2. Los elementos `<h2>` de `.articulo` se muestran en color `#CC6600`, con un tamaño de letra de `1.6em`, un interlineado de `1.2em` y un margen inferior de `0.3em`.

3. Los elementos del `#menu` deben mostrar un margen a su derecha de `1em` y los enlaces deben ser de color blanco y tamaño de letra `1.3em`.
4. El tamaño del texto de todos los contenidos de `#lateral` debe ser de `0.9em`. La fecha de cada noticia debe ocupar el espacio de toda su línea y mostrarse en color gris claro `#999`. El elemento `<h3>` de `#noticias` debe mostrarse de color `#003366`.
5. El texto del elemento `#publicidad` es de color gris oscuro `#555` y todos los enlaces de color `#CC6600`.
6. Los enlaces contenidos dentro de `.articulo` son de color `#CC6600` y todos los párrafos muestran un margen superior e inferior de `0.3em`.
7. Añadir las reglas necesarias para que el contenido de `#secundario` se vea como en la imagen que se muestra.
8. Añadir las reglas necesarias para que el contenido de `#pie` se vea como en la imagen que se muestra.

Figura 15.9. Página con propiedades tipográficas

15.8. Ejercicio 8

Definir las reglas CSS que permiten mostrar los enlaces con los siguientes estilos:

1. En su estado normal, los enlaces se muestran de color rojo `#CC0000`.
2. Cuando el usuario pasa su ratón sobre el enlace, se muestra con un color de fondo rojo `#CC0000` y la letra de color blanco `#FFF`.

3. Los enlaces visitados se muestran en color gris claro #CCC.

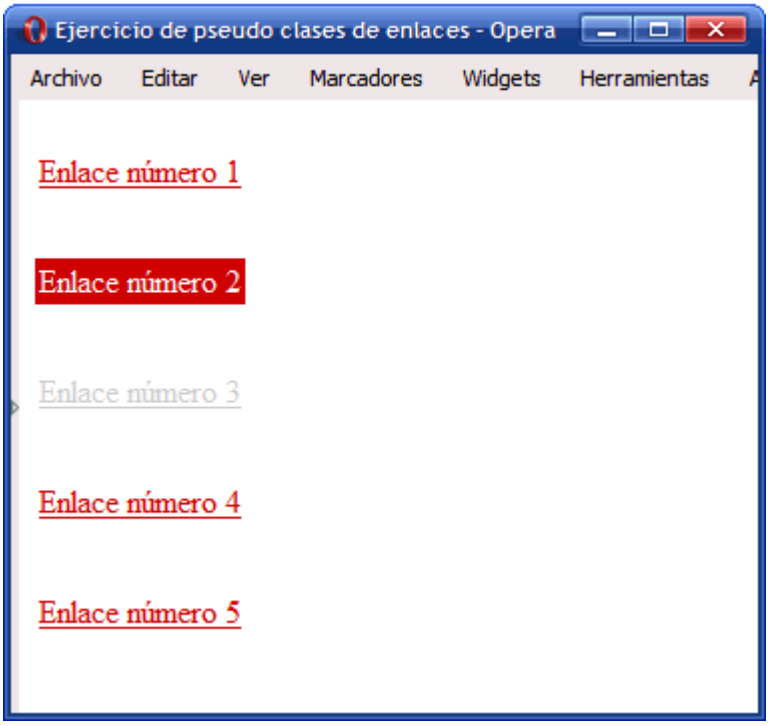


Figura 15.10. Enlaces con estilos aplicados mediante CSS

15.9. Ejercicio 9

Definir las reglas CSS que permiten mostrar una galería de imágenes similar a la que se muestra en la siguiente imagen:

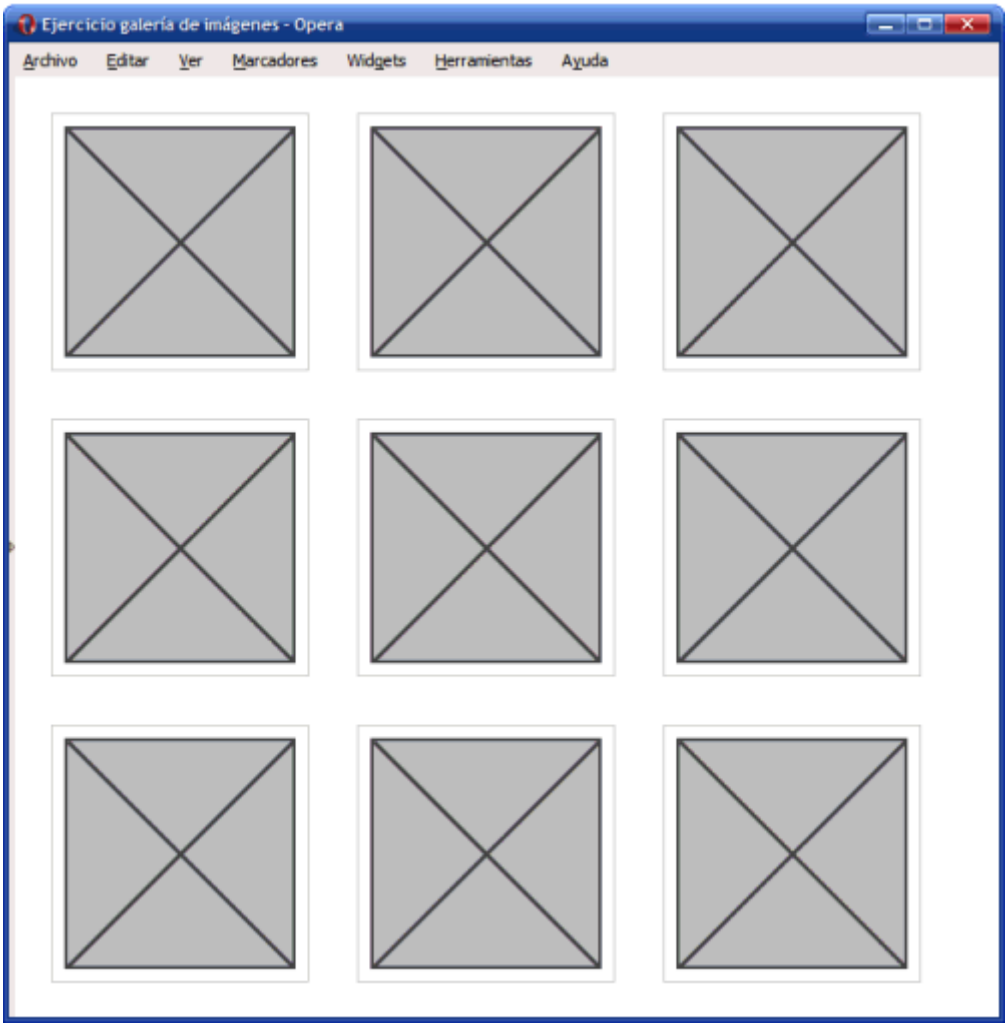


Figura 15.11. Galería de imágenes construida con CSS

15.10. Ejercicio 10

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

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

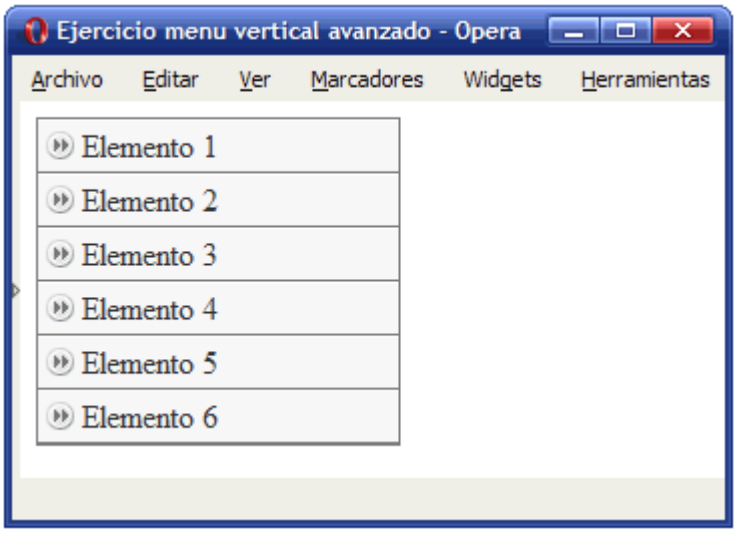


Figura 15.12. Menú vertical con imagen de fondo

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

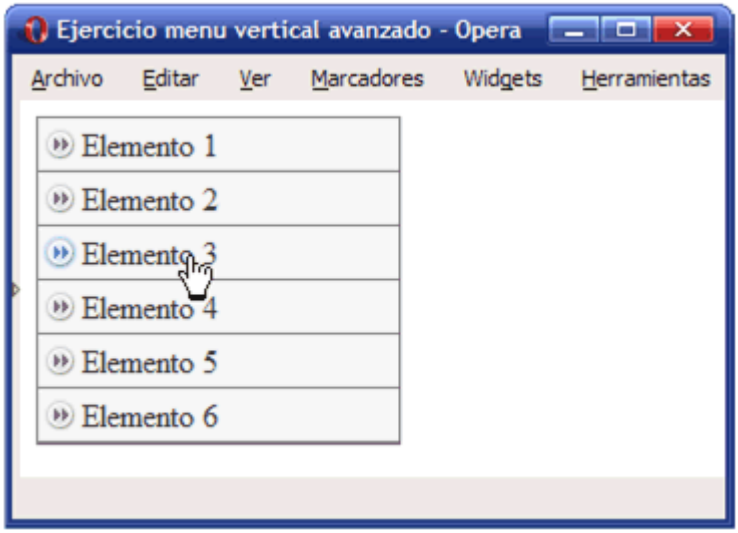


Figura 15.13. Menú vertical con imagen de fondo alternativa

1. El color de fondo del elemento también debe variar ligeramente y mostrar un color gris más oscuro (`#E4E4E4`) cuando se pasa el ratón por encima:

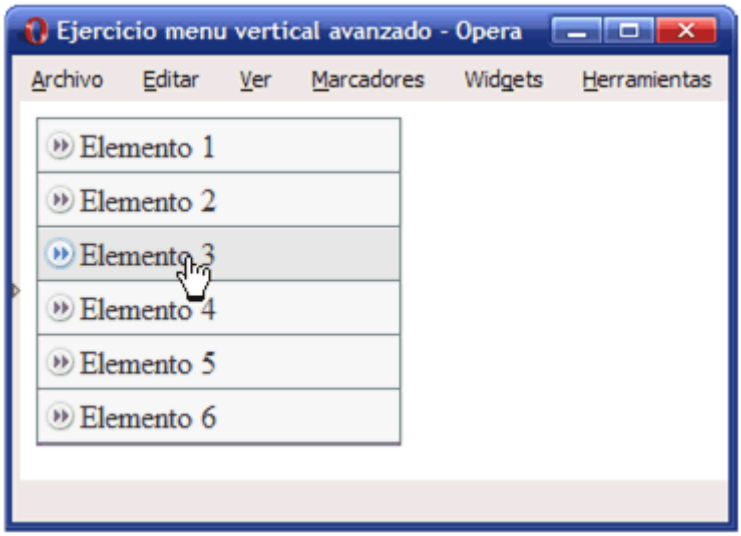


Figura 15.14. Menú vertical con imagen de fondo y color alternativos

1. El comportamiento anterior se debe producir cuando el usuario pasa el ratón por encima de cualquier zona del elemento del menú, no solo cuando se pasa el ratón por encima del texto del elemento (este problema solo sucede con Internet Explorer):

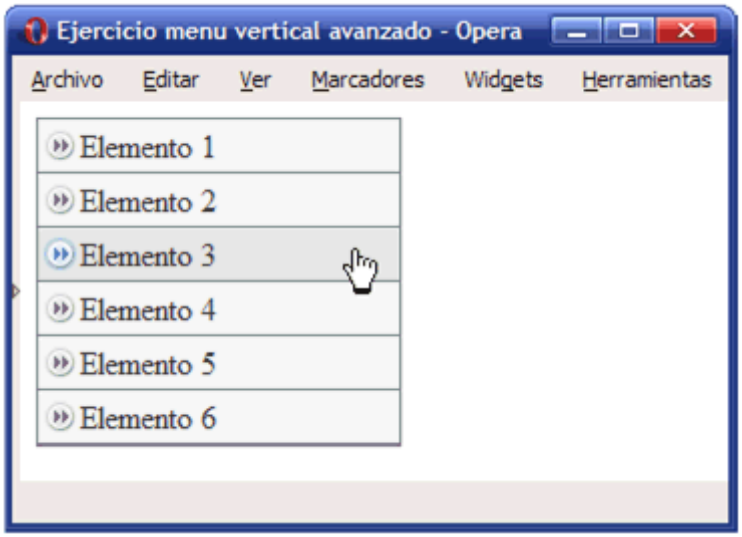
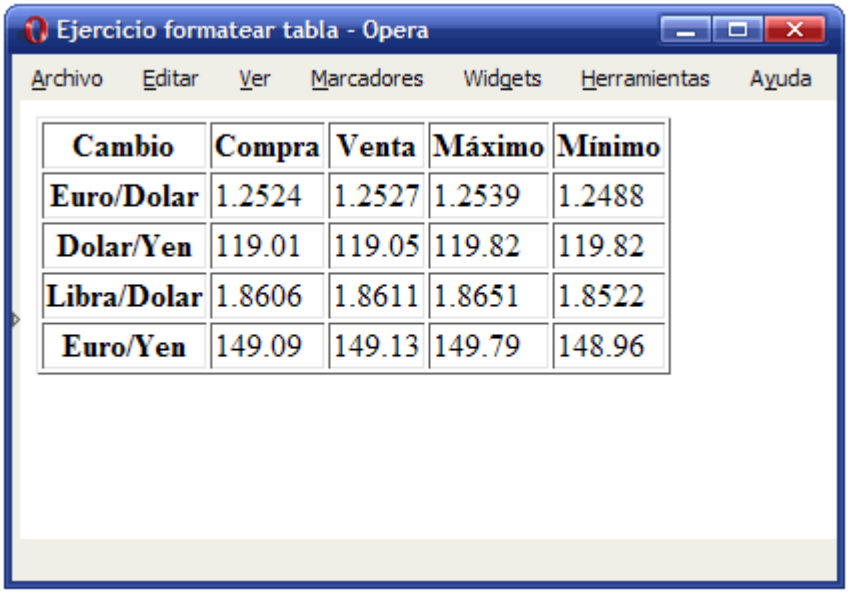


Figura 15.15. Aspecto final del menú vertical avanzado creado con CSS

15.11. Ejercicio 11

Determinar las reglas CSS necesarias para mostrar la siguiente tabla con el aspecto final mostrado en la imagen (modificar el código HTML que se considere necesario añadiendo los atributos `class` oportunos).

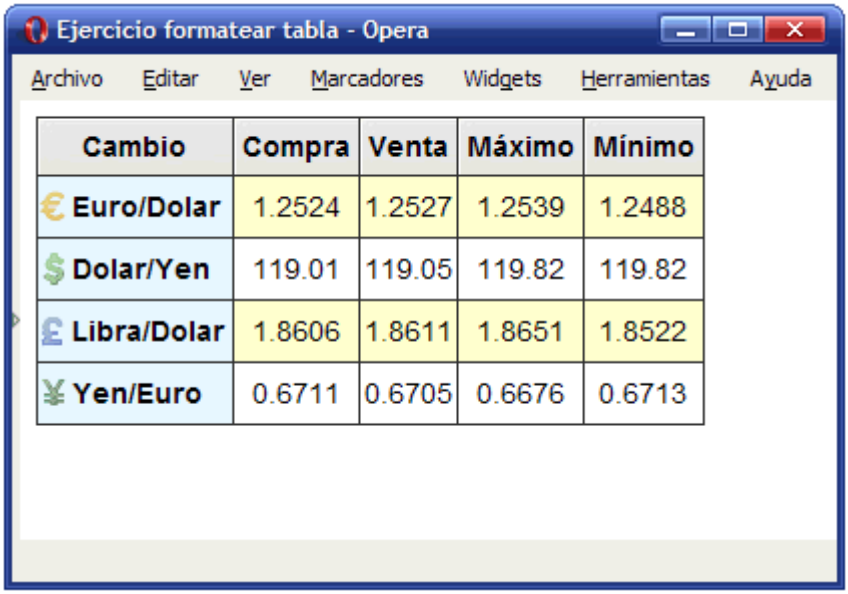
Tabla original:



Cambio	Compra	Venta	Máximo	Mínimo
Euro/Dolar	1.2524	1.2527	1.2539	1.2488
Dolar/Yen	119.01	119.05	119.82	119.82
Libra/Dolar	1.8606	1.8611	1.8651	1.8522
Euro/Yen	149.09	149.13	149.79	148.96

Figura 15.16. Aspecto original de la tabla

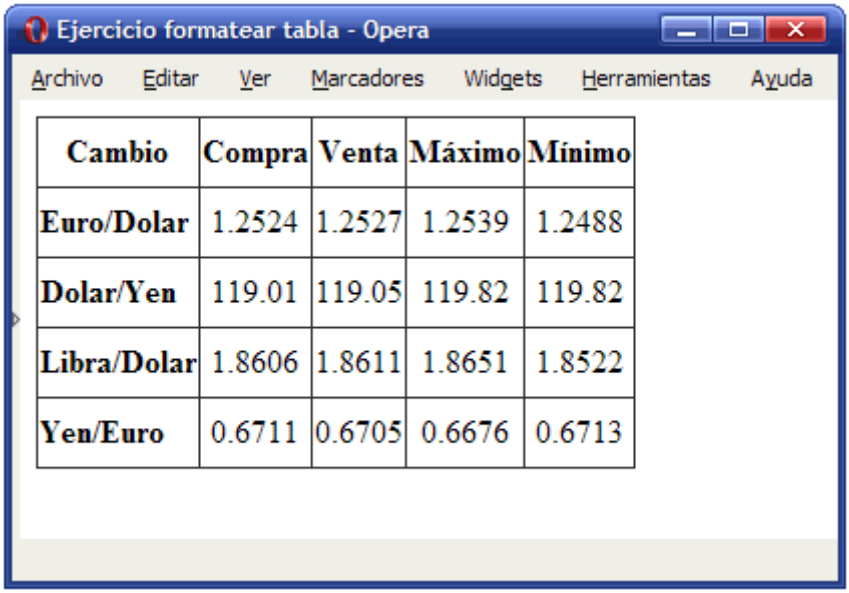
Tabla final:



Cambio	Compra	Venta	Máximo	Mínimo
€ Euro/Dolar	1.2524	1.2527	1.2539	1.2488
$ Dolar/Yen	119.01	119.05	119.82	119.82
£ Libra/Dolar	1.8606	1.8611	1.8651	1.8522
¥ Yen/Euro	0.6711	0.6705	0.6676	0.6713

Figura 15.17. Aspecto definitivo de la tabla después de aplicar estilos CSS

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



Cambio	Compra	Venta	Máximo	Mínimo
Euro/Dolar	1.2524	1.2527	1.2539	1.2488
Dolar/Yen	119.01	119.05	119.82	119.82
Libra/Dolar	1.8606	1.8611	1.8651	1.8522
Yen/Euro	0.6711	0.6705	0.6676	0.6713

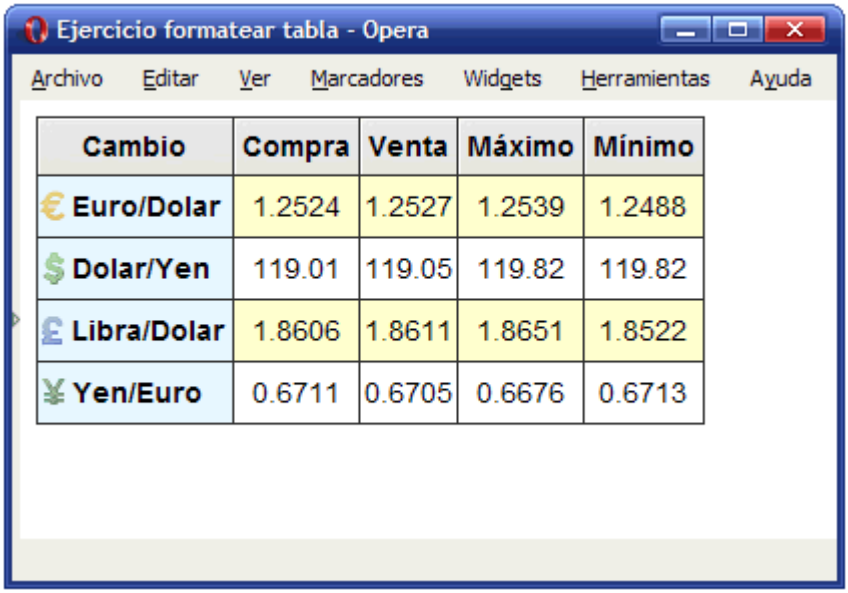
Figura 15.18. Tabla con texto alineado y bordes

1. Formatear las cabeceras de fila y columna con la imagen de fondo correspondiente en cada caso (`fondo_gris.gif`, `euro.png`, `dolar.png`, `yen.png`, `libra.png`). Modificar el tipo de letra de la tabla y utilizar `Arial`. El color azul claro es `#E6F3FF`.

Cambio	Compra	Venta	Máximo	Mínimo
€ Euro/Dolar	1.2524	1.2527	1.2539	1.2488
$ Dolar/Yen	119.01	119.05	119.82	119.82
£ Libra/Dolar	1.8606	1.8611	1.8651	1.8522
¥ Yen/Euro	0.6711	0.6705	0.6676	0.6713

Figura 15.19. Tabla con colores e imágenes de fondo

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



Cambio	Compra	Venta	Máximo	Mínimo
€ Euro/Dolar	1.2524	1.2527	1.2539	1.2488
$ Dolar/Yen	119.01	119.05	119.82	119.82
£ Libra/Dolar	1.8606	1.8611	1.8651	1.8522
¥ Yen/Euro	0.6711	0.6705	0.6676	0.6713

Figura 15.20. Tabla con colores de fila alternos

, , , ,

15.12. Ejercicio 12

A partir del código HTML que se indica, aplicar las reglas CSS necesarias para mostrar los mensajes de ayuda del formulario con el aspecto de la siguiente imagen:

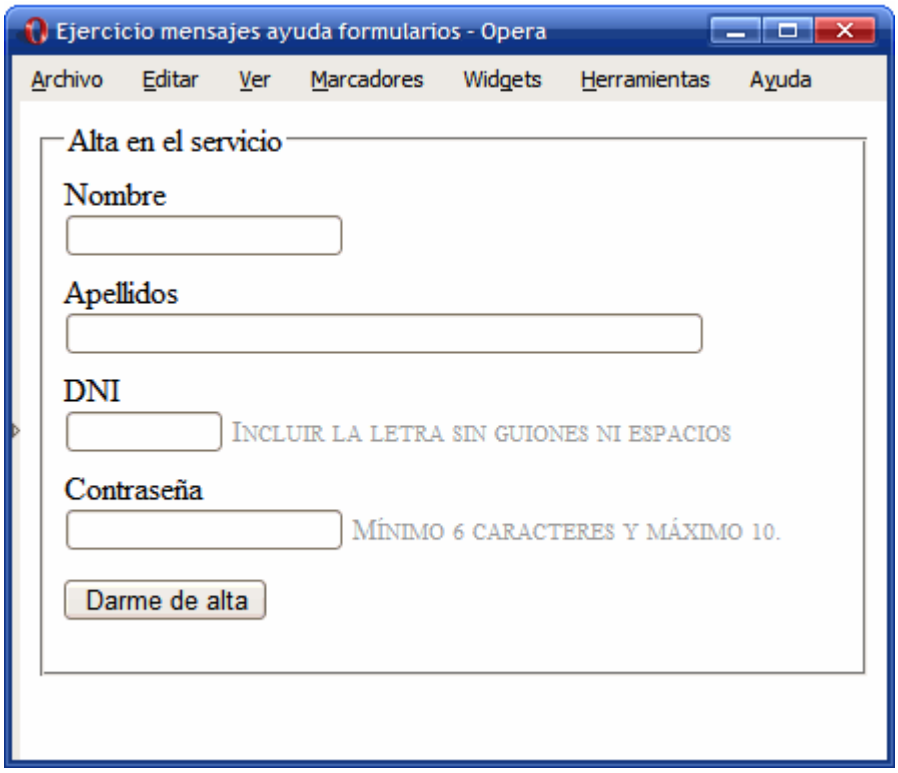


Figura 15.21. Aspecto final del formulario

Código HTML:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/
xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
<title>Ejercicio mensajes ayuda formularios</title>
</head>

<body>
<form>
<fieldset>
<legend>Alta en el servicio</legend>
<label for="nombre">Nombre</label>
<input type="text" id="nombre" />

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

<label for="dni">DNI</label>
<input type="text" id="dni" size="10" maxlength="9" />
<p class="ayuda">Incluir la letra sin guiones ni espacios</p>

<label for="contrasena">Contraseña</label>
<input type="password" id="contrasena" />
<p class="ayuda">Mínimo 6 caracteres y máximo 10.</p>

<input class="btn" type="submit" value="Darme de alta" />
</fieldset>
</form>

</body>
</html>
```

15.13. Ejercicio 13

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

Figura 15.22. Aspecto final que debe mostrar la página HTML proporcionada

A continuación se indica una propuesta de los pasos que se pueden seguir para obtener el aspecto final deseado:

- Añadir los estilos básicos de la página (tipo de letra Verdana, color de letra `#192666`, imagen de fondo llamada `fondo.gif`).
- Definir la estructura básica de la página: anchura fija de `770` píxel, centrada en la ventana del navegador, cabecera y pie, columna central de contenidos de anchura `530` píxel y columna secundaria de contenidos de `200` píxel de anchura.
- La cabecera tiene una altura de `100` píxel y una imagen de fondo llamada `cabecera.jpg`.
- Los elementos del menú de navegación tienen un color de fondo `#253575`, un color de letra `#B5C4E3`. Cuando el ratón pasa por encima de cada elemento, su color de fondo cambia a `#31479B`. Los elementos seleccionados se muestran con un color de fondo blanco y un color de letra `#FF9000`:

Figura 15.23. Imagen detallada del aspecto que muestran los elementos del menú de navegación

- Con la ayuda de las imágenes que se proporcionan, mostrar cada uno de los artículos de contenido con el estilo que se muestra en la siguiente imagen:

Figura 15.24. Aspecto de un artículo de la sección principal de contenidos

- Añadir los estilos adecuados para mostrar los elementos de la columna secundaria de contenidos con el siguiente aspecto.

Figura 15.25. Aspecto de las secciones de la columna secundaria de contenidos

Capítulo 16. Ejercicios resueltos

16.1. Ejercicio 1

```
/* Todos los elementos de la pagina */
* { font: 1/1.3em Arial, Helvetica, sans-serif; }

/* Todos los párrafos de la pagina */
p { color: #555; }

/* Solo los párrafos contenidos en #primero */
#primero p { color: #336699; }

/* Todos los enlaces la pagina */
a { color: #CC3300; }

/* Los elementos em contenidos en #primero */
#primero em { background: #FFFFCC; padding: .1em; }

/* Todos los elementos em de tipo especial en toda la pagina */
em.especial { background: #FFCC99; border: 1px solid #FF9900; padding: .1em; }

/* Elementos span contenidos en .normal */
.normal span { font-weight: bold; }
```

16.2. Ejercicio 2

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/
xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
<title>Ejercicio Fusión Márgenes</title>
<style type="text/css">
div {
  border: 2px solid black;
  width: 50px;
  margin-top: 30px;
  margin-bottom: 30px;
}
</style>
</head>

<body>
  <div>1</div>
  <div>2</div>
  <div>3</div>
</body>
</html>
```

16.3. Ejercicio 3

```
/*-----------------------------------------------------------
General
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - -        */
body { }

ul { margin: 0; padding: 0; list-style: none; }
ul li { margin: 0; padding: 0; list-style: none; }

h1, h2, h3, p, form { margin: 0; padding: 0; }

.clear { clear: both; }

img { border: none; }

/*-----------------------------------------------------------
Layout
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - -        */
#contenedor {
        width: 90%;
        max-width: 900px;
        width: expression(document.body.clientWidth > 901? "900px": "auto");
        margin: 0 auto;
}

#cabecera, #menu, #lateral, #contenido, #contenido #principal, #contenido #secundario,
#pie {
        border: 2px solid #777;
        padding: .5em;
}

#cabecera {
        clear: both;
        padding: 1em;
}

#menu {
        clear: both;
        padding: 1em;
}

#lateral {
        float: left;
        width: 18%;
}

#contenido {
        float: right;
        width: 79%;
        padding: 0;
}

#contenido #principal {
        float: left;
```

```
        width: 73%;
}

#contenido #secundario {
        float: right;
        width: 20%;
}

#pie {
        clear: both;
        padding: .5em 0;
        margin-top: 1em;
}

/*-----------------------------------------------------------
Enlaces
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - -        */

/*-----------------------------------------------------------
Secciones
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - -        */

/*Cabecera
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - -        */
#cabecera #logo { float: left; }
#cabecera #buscador { float: right; }

/*Menu
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - -        */
#menu ul#menu_principal { }
#menu ul#menu_principal li { display: inline; float: left; }

/*Contenidos
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - -        */

/*Principal
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - -        */
#contenido #principal .articulo img { width: 100px; float: left; margin: .5em; }
#contenido #principal .articulo { margin-bottom: 1em; }


/*Secundario
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - -        */

/*Lateral
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - -        */
#lateral #publicidad { margin-top: 1em; }

/*Pie
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - -        */
#pie .enlaces { float: left; }
#pie .copyright { float: right; }
```

16.4. Ejercicio 4

```
/*-----------------------------------------------------------
General
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - -        */
body { }

ul { margin: 0; padding: 0; list-style: none; }
ul li { margin: 0; padding: 0; list-style: none; }

h1, h2, h3, p, form { margin: 0; padding: 0; }

.clear { clear: both; }

img { border: none; }

/*-----------------------------------------------------------
Layout
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - -        */
#contenedor {
        width: 90%;
        max-width: 900px;
        width: expression(document.body.clientWidth > 901? "900px": "auto");
        margin: 0 auto;
}

#cabecera, #menu, #lateral, #contenido, #contenido #principal, #contenido #secundario,
#pie {
        padding: .5em;
}

#cabecera {
        clear: both;
        padding: 1em;
}

#menu {
        clear: both;
        padding: .5em;
        margin-bottom: .5em;
        border-bottom: 1px solid #004C99;
}

#lateral {
        float: left;
        width: 18%;
}

#contenido {
        float: right;
        width: 79%;
        padding: 0;
}

#contenido #principal {
```

```
        float: left;
        width: 73%;
}

#contenido #secundario {
        float: right;
        width: 20%;
}

#pie {
        clear: both;
        padding: .5em 0;
        margin-top: 1em;
        border-top: 1px solid #C5C5C5;
        border-bottom: 1px solid #C5C5C5;
}

/*-----------------------------------------------------------
Enlaces
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - -        */

/*-----------------------------------------------------------
Secciones
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - -        */

/*Cabecera
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - -        */
#cabecera #logo { float: left; }
#cabecera #buscador { float: right; }

/*Menu
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - -        */
#menu ul#menu_principal { }
#menu ul#menu_principal li { display: inline; float: left; }

/*Contenidos
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - -        */

/*Principal
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - -        */
#contenido #principal .articulo img { width: 100px; float: left; margin: .5em; }
#contenido #principal .articulo { margin-bottom: 1em; border-bottom: thick solid
#E5E5E5; }


/*Secundario
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - -        */
#contenido #secundario { border: 1px solid #CC6600; }

/*Lateral
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - -        */
#lateral #publicidad { margin-top: 1em; padding: .5em; border: 1px dashed #CC6600; }
#lateral #noticias { padding: .5em; border: 1px solid #C5C5C5; }

/*Pie
```

```
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - -        */
#pie .enlaces { float: left; }
#pie .copyright { float: right; }
```

16.5. Ejercicio 5

```
/*-----------------------------------------------------------
General
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - -        */
body { }

ul { margin: 0; padding: 0; list-style: none; }
ul li { margin: 0; padding: 0; list-style: none; }

h1, h2, h3, p, form { margin: 0; padding: 0; }

.clear { clear: both; }

img { border: none; }

/*-----------------------------------------------------------
Layout
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - -        */
#contenedor {
        width: 90%;
        max-width: 900px;
        width: expression(document.body.clientWidth > 901? "900px": "auto");
        margin: 0 auto;
}

#cabecera, #menu, #lateral, #contenido, #contenido #principal, #contenido #secundario,
#pie {
        padding: .5em;
}

#cabecera {
        clear: both;
        padding: 1em;
}

#menu {
        clear: both;
        padding: .5em;
        margin-bottom: .5em;
        border-bottom: 1px solid #004C99;
        background: #003366 url(imagenes/fondo_menu.gif);
}

#lateral {
        float: left;
        width: 18%;
}

#contenido {
        float: right;
```

```
        width: 79%;
        padding: 0;
}

#contenido #principal {
        float: left;
        width: 73%;
}

#contenido #secundario {
        float: right;
        width: 20%;
}

#pie {
        clear: both;
        padding: .5em 0;
        margin-top: 1em;
        border-top: 1px solid #C5C5C5;
        border-bottom: 1px solid #C5C5C5;
        background: #F8F8F8;
}

/*-----------------------------------------------------------
Enlaces
- - - - - - - - - - - - - - - - - - - - - - - - - - - - -        */

/*-----------------------------------------------------------
Secciones
- - - - - - - - - - - - - - - - - - - - - - - - - - - - -        */

/*Cabecera
- - - - - - - - - - - - - - - - - - - - - - - - - - - - -        */
#cabecera #logo { float: left; }
#cabecera #buscador { float: right; }
#cabecera #logo { background: transparent url(imagenes/logo.gif) no-repeat top left;
width: 200px; height: 55px; }
#cabecera #logo h1 span {visibility:hidden;}

/*Menu
- - - - - - - - - - - - - - - - - - - - - - - - - - - - -        */
#menu ul#menu_principal { }
#menu ul#menu_principal li { display: inline; float: left; }

/*Contenidos
- - - - - - - - - - - - - - - - - - - - - - - - - - - - -        */

/*Principal
- - - - - - - - - - - - - - - - - - - - - - - - - - - - -        */
#contenido #principal .articulo img { width: 100px; float: left; margin: .5em; }
#contenido #principal .articulo { margin-bottom: 1em; border-bottom: thick solid
#E5E5E5; }


/*Secundario
```

```
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - -        */
#contenido #secundario { border: 1px solid #CC6600; }
#contenido #secundario h2 { background: #DB905C; padding: .2em; }

/*Lateral
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - -        */
#lateral #publicidad { margin-top: 1em; padding: .5em; border: 1px dashed #CC6600;
background: #FFF6CD;}
#lateral #noticias { padding: .5em; border: 1px solid #C5C5C5; background: #F8F8F8; }

/*Pie
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - -        */
#pie .enlaces { float: left; }
#pie .copyright { float: right; }
```

16.6. Ejercicio 6

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/
xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
<title>Ejercicio posicionamiento float</title>
<style type="text/css">
div#paginacion {
  border: 1px solid #CCC;
  background-color: #E0E0E0;
  padding: .5em;
}
.derecha {
  float: right;
}
.izquierda {
  float: left;
}
div.clear {
  clear: both;
}
</style>
</head>

<body>
<div id="paginacion">
<span class="izquierda">&laquo; Anterior</span> <span class="derecha">Siguiente
&raquo;</span>
<div class="clear"></div>
</div>
</body>
</html>
```

16.7. Ejercicio 7

```
/*-------------------------------------------------------------
General
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - -        */
```

```
body { color: #000; font: normal .9em/1.4em Arial, Helvetica, sans-serif; }

ul { margin: 0; padding: 0; list-style: none; }
ul li { margin: 0; padding: 0; list-style: none; }

h1, h2, h3, p, form { margin: 0; padding: 0; }

.clear { clear: both; }

img { border: none; }

/*-----------------------------------------------------------
Layout
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - -        */
#contenedor {
        width: 90%;
        max-width: 900px;
        width: expression(document.body.clientWidth > 901? "900px": "auto");
        margin: 0 auto;
}

#cabecera, #menu, #lateral, #contenido, #contenido #principal, #contenido #secundario,
#pie {
        padding: .5em;
}

#cabecera {
        clear: both;
        padding: 1em;
}

#menu {
        clear: both;
        padding: .5em;
        margin-bottom: .5em;
        border-bottom: 1px solid #004C99;
        background: #003366 url(imagenes/fondo_menu.gif);
}

#lateral {
        float: left;
        width: 18%;
}

#contenido {
        float: right;
        width: 79%;
        padding: 0;
}

#contenido #principal {
        float: left;
        width: 73%;
}
```

```
#contenido #secundario {
        float: right;
        width: 20%;
}

#pie {
        clear: both;
        padding: .5em 0;
        margin-top: 1em;
        border-top: 1px solid #C5C5C5;
        border-bottom: 1px solid #C5C5C5;
        background: #F8F8F8;
}

/*-----------------------------------------------------------
Enlaces
- - - - - - - - - - - - - - - - - - - - - - - - - - - - -       */

/*-----------------------------------------------------------
Secciones
- - - - - - - - - - - - - - - - - - - - - - - - - - - - -       */

/*Cabecera
- - - - - - - - - - - - - - - - - - - - - - - - - - - - -       */
#cabecera #logo { float: left; }
#cabecera #buscador { float: right; }
#cabecera #logo { background: transparent url(imagenes/logo.gif) no-repeat top left;
width: 200px; height: 55px; }
#cabecera #logo h1 span {visibility:hidden;}

/*Menu
- - - - - - - - - - - - - - - - - - - - - - - - - - - - -       */
#menu ul#menu_principal { }
#menu ul#menu_principal li { display: inline; float: left; margin-left: 1em; }
#menu ul#menu_principal li a { color: white; font-size: 1.2em; }

/*Contenidos
- - - - - - - - - - - - - - - - - - - - - - - - - - - - -       */

/*Principal
- - - - - - - - - - - - - - - - - - - - - - - - - - - - -       */
#contenido #principal .articulo img { width: 100px; float: left; margin: .5em; }
#contenido #principal .articulo { margin-bottom: 1em; border-bottom: thick solid
#E5E5E5; }
#contenido #principal .articulo h2 { color: #CC6600; font-size: 1.6em;
line-height:1.2em; margin-bottom: .3em; }
#contenido #principal .articulo a { color: #CC6600; }
#contenido #principal .articulo p { margin: .3em 0; }

/*Secundario
- - - - - - - - - - - - - - - - - - - - - - - - - - - - -       */
#contenido #secundario { border: 1px solid #CC6600; font-size: .8em; color: #555; }
#contenido #secundario h2 { background: #DB905C; padding: .2em; font-size: 1em; color:
#fff; }
#contenido #secundario p { margin-bottom: 1em; }
```

```
/*Lateral
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - -        */
#lateral { font-size: .9em; }
#lateral #publicidad { margin-top: 1em; padding: .5em; border: 1px dashed #CC6600;
background: #FFF6CD; color: #555; }
#lateral #publicidad a { color: #CC6600;}

#lateral #noticias { padding: .5em; border: 1px solid #C5C5C5; background: #F8F8F8; }
#lateral #noticias h3 a { color: #003366; }
#lateral #noticias .fecha { display: block; color: #999; }

/*Pie
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - -        */
#pie {font-size:.7em; color: #666; }
#pie .enlaces { float: left; }
#pie .enlaces a { color: #666; }
#pie .copyright { float: right; }
```

16.8. Ejercicio 8

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/
xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
<title>Ejercicio de pseudo clases de enlaces</title>
<style type="text/css">
a {
  margin: 1em 0;
  color: #CC0000;
  float: left;
  clear: left;
  padding: 2px;
}
a:hover {
  text-decoration: none;
  background-color: #CC0000;
  color: #FFF;
}
a:visited {
  color: #CCC;
}
</style>
</head>

<body>
<a href="#">Enlace número 1</a>

<a href="#">Enlace número 2</a>

<a href="#">Enlace número 3</a>

<a href="#">Enlace número 4</a>
```

```
<a href="#">Enlace número 5</a>
</body>
</html>
```

16.9. Ejercicio 9

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/
xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
<title>Ejercicio galería de imágenes</title>
<style type="text/css">
#galeria {
  /* En el navegador Internet Explorer versión 6 y anteriores no funciona la propiedad
"max-width" */
  max-width: 650px;
}
#galeria img {
  float: left;
  margin: 1em;
  padding: .5em;
  border: 1px solid #CCC;
}
</style>
</head>

<body>
<div id="galeria">
<img src="../comun/imagenes/imagen.png" alt="Imagen generica" />
<img src="../comun/imagenes/imagen.png" alt="Imagen generica" />
<img src="../comun/imagenes/imagen.png" alt="Imagen generica" />
<img src="../comun/imagenes/imagen.png" alt="Imagen generica" />
<img src="../comun/imagenes/imagen.png" alt="Imagen generica" />
<img src="../comun/imagenes/imagen.png" alt="Imagen generica" />
<img src="../comun/imagenes/imagen.png" alt="Imagen generica" />
<img src="../comun/imagenes/imagen.png" alt="Imagen generica" />
<img src="../comun/imagenes/imagen.png" alt="Imagen generica" />
</div>
</body>
</html>
```

16.10. Ejercicio 10

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/
xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
<title>Ejercicio menu vertical avanzado</title>
<style type="text/css">
ul.menu {
  width: 180px;
  list-style: none;
  margin: 0;
```

```
  padding: 0;
  border:1px solid #7C7C7C;
}
ul.menu li {
  border-bottom:1px solid #7C7C7C;
  border-top: 1px solid #FFF;
  background: #F4F4F4;
}
ul.menu li a:link, ul.menu li a:visited {
  width: 158px;
  padding: .2em 0 .2em 1.3em;
  display:block;
  text-decoration: none;
  color: #333;
  background: #F4F4F4 url(imagenes/flecha_inactiva.png) no-repeat .2em center;
}
ul.menu li a:hover, ul.menu li a:active {
  background: #E4E4E4 url(imagenes/flecha_activa.png) no-repeat .2em center;
}
</style>
</head>

<body>
<ul class="menu">
  <li><a href="#" title="Enlace genérico">Elemento 1</a></li>
  <li><a href="#" title="Enlace genérico">Elemento 2</a></li>
  <li><a href="#" title="Enlace genérico">Elemento 3</a></li>
  <li><a href="#" title="Enlace genérico">Elemento 4</a></li>
  <li><a href="#" title="Enlace genérico">Elemento 5</a></li>
  <li><a href="#" title="Enlace genérico">Elemento 6</a></li>
</ul>
</body>
</html>
```

16.11. Ejercicio 11

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/
xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
<title>Ejercicio formatear tabla</title>
<style type="text/css">
table {
        font: .9em Arial, Helvetica, sans-serif;
        border: 1px solid #333;
        border-collapse: collapse;
        text-align: center;
}
table th {
        background: #F5F5F5 url(imagenes/fondo_gris.gif) repeat-x;
        padding: 0 .3em;
}
table th.euro {
        background: #E6F3FF url(imagenes/euro.png) no-repeat left center;
```

```
        padding: 0 .3em 0 1.2em;
}
table th.dolar {
        background: #E6F3FF url(imagenes/dolar.png) no-repeat left center;
        padding: 0 .3em 0 1.2em;
}
table th.libra {
        background: #E6F3FF url(imagenes/libra.png) no-repeat left center;
        padding: 0 .3em 0 1.2em;
}
table th.yen {
        background: #E6F3FF url(imagenes/yen.png) no-repeat left center;
        padding: 0 .3em 0 1.2em;
}
table th, table, td {
        border: 1px solid #333;
        line-height: 2em;
}
.fila {
        text-align: left;
}
.par {
        background-color:#FFFFCC;
}
table tr:hover {
        background: #FFFF66 !important;
}
</style>
</head>

<body>
<table summary="Tipos de cambio">
        <tr>
                <th scope="col">Cambio</th>
                <th scope="col">Compra</th>
                <th scope="col">Venta</th>
                <th scope="col">M&aacute;ximo</th>
                <th scope="col">M&iacute;nimo</th>
        </tr>
        <tr class="par">
                <th scope="row" class="fila euro">Euro/Dolar</th>
                <td>1.2524</td>
                <td>1.2527</td>
                <td>1.2539</td>
                <td>1.2488</td>
        </tr>
        <tr>
                <th scope="row" class="fila dolar">Dolar/Yen</th>
                <td>119.01</td>
                <td>119.05</td>
                <td>119.82</td>
                <td>119.82</td>
        </tr>
        <tr class="par">
                <th scope="row" class="fila libra">Libra/Dolar</th>
```

```
                <td>1.8606</td>
                <td>1.8611</td>
                <td>1.8651</td>
                <td>1.8522</td>
        </tr>
        <tr>
                <th scope="row" class="fila yen">Yen/Euro</th>
                <td>0.6711</td>
                <td>0.6705</td>
                <td>0.6676</td>
                <td>0.6713</td>
        </tr>
</table>

</body>
</html>
```

16.12. Ejercicio 12

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/
xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
<title>Ejercicio mensajes ayuda formularios</title>
<style type="text/css">
label {
  display: block;
  margin: .5em 0 0 0;
}
.btn {
  display: block;
  margin: 1em 0;
}
p.ayuda {
  display: inline;
  color: #999;
  font-variant:small-caps;
  font-size: .9em;
}
</style>
</head>

<body>
<form>
<fieldset>
<legend>Alta en el servicio</legend>
<label for="nombre">Nombre</label>
<input type="text" id="nombre" />

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

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

```
<p class="ayuda">Incluir la letra sin guiones ni espacios</p>

<label for="contrasena">Contraseña</label>
<input type="password" id="contrasena" />
<p class="ayuda">Mínimo 6 caracteres y máximo 10.</p>

<input class="btn" type="submit" value="Darme de alta" />
</fieldset>
</form>

</body>
</html>
```

16.13. Ejercicio 13

```
/* ---- Estilos básicos ---------------------------------------------------- */

body {
  margin: 0;
  padding: 0;
  background: #F2F5FE url("../imagenes/fondo.gif") 0 0 repeat-x;
  font: 70%/160% "verdana", sans-serif;
  color: #192666;
}

a {
  color:#192666;
}
a:hover {
  color:#4F6AD7;
}

p {
  margin: 1em 0;
  padding: 0;
}

.clear {
  clear: both;
}

h1, h2, h3, h4, h5 {
  margin: 1em 0;
  padding:0;
}

h1 {
  font-size:260%;
  font-family: "georgia", serif;
  font-weight:normal;
}

h2 {
  font-size:180%;
  font-family: "georgia", serif;
```

```
  font-weight:normal;
}

h3 {
  font-size:120%;
  font-weight:bold;
}

ul, ol {
  margin: 1em 0 1em 2em;
  padding:0;
}

/* ---- Layout ------------------------------------------------------------ */

#contenedor {
  width:770px;
  margin: 50px auto 0 auto;
}

#cabecera {
  width: 770px;
  position: relative;
  height: 100px;
  margin: 0;
  padding: 0;
  background: #233C9B url("../imagenes/cabecera.jpg") no-repeat;
  color: #FFF;
}

#contenido {
  width: 760px;
  margin: 0 5px;
  background: #FFF;
}

#contenido #principal {
  float: left;
  width: 530px;
  margin-top: 15px;
  padding:0 0 0 20px;
  background: #FFF;
}

#contenido #secundario {
  float: left;
  width: 200px;
  margin: 15px 0 0 0;
  padding: 0;
  background: #CEDBF9 url("../imagenes/fondo_columna.gif") no-repeat;
}

#pie {
  clear: both;
  height: 60px;
```

```
  margin-bottom: 50px;
  background: url("../imagenes/fondo_pie.jpg") no-repeat;
  color: #6685CC;
  position: relative;
}

/* ---- Cabecera ------------------------------------------------------------ */
#cabecera #logo {
  position: absolute;
  top: 35px;
  left: 35px;
  margin: 0;
}

#cabecera #logo a {
  color: #FFF;
}

#cabecera #logo a:hover {
  color: #B5C4E3;
  text-decoration: underline;
}

#cabecera #buscador {
  position: absolute;
  top: 35px;
  right: 20px;
}

#cabecera #buscador legend {
  display: none;
}

#cabecera #buscador fieldset {
  border: none;
}

/* ---- Menú ---------------------------------------------------------------- */
#menu {
  background: #192666;
  margin: 0 5px;
  padding: 10px 0 0 0;
}

#menu ul {
  margin: 0 10px;
  padding: 0;
  list-style:none;
}

#menu ul li {
  margin: 0 5px 0 0;
  padding: 0;
  float:left;
}
```

```
#menu ul li a {
  display: block;
  position: relative;
  padding: 5px 15px;
  border: 0;
  background: #253575;
  color: #B5C4E3;
  font-weight: bold;
  text-decoration: none;
  cursor: pointer;
}

#menu ul li a:hover {
  background: #31479B;
}

#menu ul li.seleccionado a {
  background: #FFF;
  color: #FF9000;
}

/* ---- Contenidos ------------------------------------------------------- */
#contenido .articulo {
  clear: both;
  margin: 0;
  padding: 20px;
  background: url("../imagenes/fondo_articulo.jpg") no-repeat;
}

#contenido .articulo h2 {
  margin: 0 -20px;
  padding: 10px;
  background: #DEE5FD;
  color: #192666;
}


#contenido .info {
  margin: 10px 0;
  padding-bottom: 8px;
  border-bottom: 1px solid #DEE5FD;
  color:#6685CC;
}

#contenido .info a { color:#6685CC; }
#contenido .info a:hover { color:#FF9000; }

#contenido .info span.fecha, #contenido .info span.categoria, #contenido .info
span.autor, #contenido .info span.comentarios {
  padding-left:15px;
}

#contenido .info span.fecha { background: url("../imagenes/icono_fecha.gif") 0 50%
no-repeat; }
```

```
#contenido .info span.categoria { background: url("../imagenes/icono_categoria.gif") 0
50% no-repeat; margin-left: 8px; }
#contenido .info span.autor { background: url("../imagenes/icono_autor.gif") 0 50%
no-repeat; margin-left: 8px; }
#contenido .info span.comentarios { background: url("../imagenes/
icono_comentarios.gif") 0 50% no-repeat; margin-left: 8px; }

#contenido #secundario h3 {
  padding: 10px 0 10px 10px;
  margin-top: 20px;
  background: #A0B9F3;
  color: #192666;
}

#contenido #secundario #sobremi img {
  float: left;
  margin: 4px 5px 0 0;
}

#contenido #secundario #sobremi p {
}

#contenido #secundario div {
  padding: 0 10px;
}

#contenido #secundario ul {
  margin: 15px 0;
  padding: 0;
  list-style:none;
}

#contenido #secundario li {
  margin: 0;
  padding: 0;
  border-bottom: 1px solid #E0E8FA;
}

#contenido #secundario li a {
  display: block;
  padding: 3px 0 3px 22px;
  background: url("../imagenes/icono_elemento.gif") 8px no-repeat;
  text-decoration: none;
}

#contenido #secundario li a:hover {
  background-color: #E0E8FA;
  color: #192666;
}

#contenido #secundario li.seleccionado a {
  background: #E0E8FA url("../imagenes/icono_elemento_seleccionado.gif") 8px no-repeat;
  font-weight: bold;
}
```

```
/* ---- Pie ------------------------------------------------------------- */
#pie p {
  margin: 0;
  padding: 0;
  position: absolute;
  top: 10px;
  left: 40px;
}
```