MÓDULO FORMATIVO 1: Programación Web en el entorno cliente: 60 HORAS

UNIDAD FORMATIVA 1: Elaboración de documentos mediante lenguajes de marcas

(continuación…)

Hojas de estilo Web CSS (http://librosweb.es/css/)

- Tipos de hojas de estilo: estáticas y dinámicas.
- Elementos y estructura de una hoja de estilo.
  - Creación de hojas de estilo.
  - Aplicación de estilos.
  - Herencia de estilos y aplicación en cascada.
  - Formateado de páginas mediante estilos.
  - Estructura de páginas mediante estilos.
- Tipos de hojas de estilo: estáticas y dinámicas.
- Elementos y estructura de una hoja de estilo.
  - Creación de hojas de estilo.
  - Aplicación de estilos.
  - Herencia de estilos y aplicación en cascada.
  - Formateado de páginas mediante estilos.
  - Estructura de páginas mediante estilos.
- Unidades de medida y colores (http://librosweb.es/css/).

INICIO

- **Soporte de CSS en los navegadores**

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

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

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

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

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

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

- **1.4. Especificación oficial**

  La especificación o norma oficial que se utiliza actualmente para diseñar páginas web con CSS es la versión CSS 2.1, actualizada por última vez el 7 de junio de 2011 y que se puede consultar libremente en w3.org/TR/CSS21

  Desde hace varios años, el organismo W3C trabaja en la elaboración de la próxima versión de CSS, conocida como CSS 3. Esta nueva versión incluye multitud de cambios importantes en todos los niveles y es mucho más avanzada y compleja que CSS 2. Puedes consultar el estado actual de cada componente de CSS 3 en w3.org/Style/CSS/current-work. También existe un blog oficial en el que se publican todas las novedades relacionadas con el estándar CSS.

- **1.6. Cómo incluir CSS en un documento XHTML**

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

  - **1.6.1. Incluir CSS en el mismo documento HTML**

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

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

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

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

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

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

- **1.6.2. Definir CSS en un archivo externo**

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

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

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

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

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

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

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
<title>Ejemplo de estilos CSS en un archivo externo</title>
<link rel="stylesheet" type="text/css" href="/css/estilos.css" media="screen" />
</head>

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

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

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

rel: indica el tipo de relación que existe entre el recurso enlazado (en este caso, el archivo CSS) y la página HTML. Para los archivos CSS, siempre se utiliza el valor stylesheet

type: indica el tipo de recurso enlazado. Sus valores están estandarizados y para los archivos CSS su valor siempre es text/css

href: indica la URL del archivo CSS que contiene los estilos. La URL indicada puede ser relativa o absoluta y puede apuntar a un recurso interno o externo al sitio web.

media: indica el medio en el que se van a aplicar los estilos del archivo CSS. Más adelante se explican en detalle los medios CSS y su funcionamiento.

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

Con este método, el mantenimiento del sitio web se simplifica al máximo, ya que un solo cambio en un solo archivo CSS permite variar de forma instantánea los estilos de todas las páginas HTML que enlazan ese archivo. Aunque generalmente se emplea la etiqueta <link> para enlazar los archivos CSS externos, también se puede utilizar la etiqueta <style>. La forma alternativa de incluir un archivo CSS externo se muestra a continuación:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
<title>Ejemplo de estilos CSS en un archivo externo</title>
<style type="text/css" media="screen">
 @import '/css/estilos.css';
</style>
</head>

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

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

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

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

- **1.6.3. Incluir CSS en los elementos HTML**

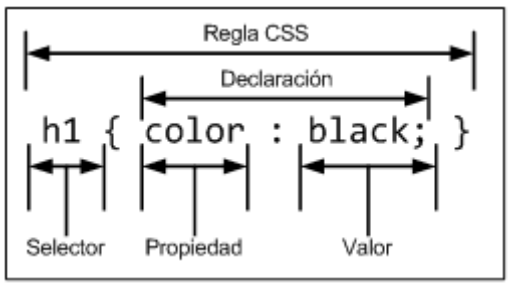
  El último método para incluir estilos CSS en documentos HTML es el peor y el menos utilizado, ya que tiene los mismos problemas que la utilización de las etiquetas <font>.

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

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

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

- **1.7. Glosario básico**

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



Componentes de un estilo CSS básico

- Regla: cada uno de los estilos que componen una hoja de estilos CSS. Cada regla está compuesta de una parte de *"selectores"*, un símbolo de *"llave de apertura"* ({), otra parte denominada *"declaración"* y por último, un símbolo de *"llave de cierre"* (}).
- Selector: indica el elemento o elementos HTML a los que se aplica la regla CSS.
- Declaración: especifica los estilos que se aplican a los elementos. Está compuesta por una o más propiedades CSS.
- Propiedad: característica que se modifica en el elemento seleccionado, como por ejemplo su tamaño de letra, su color de fondo, etc.
- Valor: establece el nuevo valor de la característica modificada en el elemento.

Un archivo CSS puede contener un número ilimitado de reglas CSS, cada regla se puede aplicar a varios selectores diferentes y cada declaración puede incluir tantos pares propiedad/valor como se desee.

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

- **1.8. Medios CSS**

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

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

Medio	Descripción
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).

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

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

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

- **1.8.1. Medios definidos con las reglas de tipo @media**

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

  A continuación se muestra un ejemplo sencillo:

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

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

- **1.8.2. Medios definidos con las reglas de tipo @import**

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

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

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

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

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

- **1.8.3. Medios definidos con la etiqueta** <link>

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

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

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

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

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

- **1.8.4. Medios definidos mezclando varios métodos**

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

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

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

@media print {

        /* Estilos específicos para impresora */

}
```

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

- **1.9. Comentarios**

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

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

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

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

```
/* Este es un

        comentario CSS de varias

        lineas */
```

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

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

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

<!-- Este es un

comentario HTML de varias

lineas -->

- **Capítulo 2. Selectores**
  Para crear diseños web profesionales, es imprescindible conocer y dominar los selectores de CSS. Como se vio en el capítulo anterior, una regla de CSS está formada por una parte llamada "selector" y otra parte llamada "declaración".

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

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

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

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

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

```
* {

    margin: 0;

    padding: 0;

}
```

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

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

- **2.1.2. Selector de tipo o etiqueta**

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

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

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

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

```
h1 {
    color: red;
}

h2 {
    color: blue;
}

p {
    color: black;
}
```

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

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

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

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

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

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

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

- **2.1.3. Selector descendente**

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

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

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

Si el código HTML de la página es el siguiente:

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

El selector p span selecciona tanto texto1 como texto2. El motivo es que en el selector descendente, un elemento no tiene que ser descendiente directo del otro. La única condición es que un elemento debe estar dentro de otro elemento, sin importar el nivel de profundidad en el que se encuentre.

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

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

```
p span { color: red;}

h1 span { color: blue; }
```

Con las reglas CSS anteriores:

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

La sintaxis formal del selector descendente se muestra a continuación:

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

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

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

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

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

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

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

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

Se puede restringir el alcance del selector descendente combinándolo con el selector universal. El siguiente ejemplo, muestra los dos enlaces de color rojo:

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

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

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

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

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

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

- **2.1.4. Selector de clase**

  Si se considera el siguiente código HTML de ejemplo:

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

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

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

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

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

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

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

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

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

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

A continuación se muestra otro ejemplo de selectores de clase:

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

.error {
    color: #930;
    font-weight: bold;
}
<span class="error">...</span>

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

```
.error { color: red; }

.error.destacado { color: blue; }

.destacado { font-size: 15px; }

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

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

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

- **2.1.5. Selectores de ID**

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

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

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

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

```
<p>Primer párrafo</p>

<p id="destacado">Segundo párrafo</p>

<p>Tercer párrafo</p>
```

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Si se modifica el anterior selector:

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

Ahora, el selector solamente selecciona aquellos elementos de tipo con un atributo class="especial" que estén dentro de cualquier elemento de tipo 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.

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

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

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

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

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

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

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

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

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

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

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

- **2.2.2. Selector adyacente**

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

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

Si se considera el siguiente código HTML:

```
<body>
<h1>Titulo1</h1>

<h2>Subtítulo</h2>
...

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

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

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

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

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

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

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

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

En muchos libros, suele ser habitual que la primera línea de todos los párrafos esté indentada, salvo la primera línea del primer párrafo. Con el selector p + p, se seleccionan todos los párrafos de la página que estén precedidos por otro párrafo, por lo que no se aplica al primer párrafo de la página.

- **2.2.3. Selector de atributos**

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

  Los cuatro tipos de selectores de atributos son:

  1. [nombre_atributo], selecciona los elementos que tienen establecido el atributo llamado nombre_atributo, independientemente de su valor.
  2. [nombre_atributo=valor], selecciona los elementos que tienen establecido un atributo llamado nombre_atributo con un valor igual a valor.
  3. [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.
  4. [nombre_atributo^=valor], selecciona los elementos que tienen establecido un atributo llamado nombre_atributo y cuyo valor comienza con valor. Este tipo de selector sólo es útil para los atributos de tipo lang que indican el idioma del contenido del elemento.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

- **2.4. Herencia**

  Una de las características principales de CSS es la herencia de los estilos definidos para los elementos. Cuando se establece el valor de una propiedad CSS en un elemento, sus elementos descendientes heredan de forma automática el valor de esa propiedad. Si se considera el siguiente ejemplo:

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

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

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

Aunque la herencia de estilos se aplica automáticamente, se puede anular su efecto estableciendo de forma explícita otro valor para la propiedad que se hereda, como se muestra en el siguiente ejemplo:

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

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

En el ejemplo anterior, se establece en primer lugar el color y tipo de letra del elemento <body>, por lo que todos los elementos de la página se mostrarían con ese mismo color y tipo de letra. No obstante, las otras reglas CSS modifican alguno de los estilos heredados.

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

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

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

- **2.5. Colisiones de estilos**

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

```
p { color: red; }

p { color: blue; }
```

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

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

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

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

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

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

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

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

```
p { color: red; }

p#especial { color: green; }

* { color: blue; }


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

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

- **3.1. Unidades de medida**

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

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

  Si el valor es 0, la unidad de medida es opcional. Si el valor es distinto a 0 y no se indica ninguna unidad, la medida se ignora completamente, lo que suele ser uno de los errores más habituales de los diseñadores que empiezan con CSS. Algunas propiedades permiten indicar medidas negativas, aunque habitualmente sus valores son positivos. Si el valor decimal de una medida es inferior a 1, se puede omitir el 0 de la izquierda (0.5em es equivalente a .5em).

- **3.1.1. Unidades absolutas**

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

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

A continuación se muestran ejemplos de utilización de unidades absolutas:

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

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

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

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

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

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

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

- **3.1.2. Unidades relativas**

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

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

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

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

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

Si se considera el siguiente ejemplo:

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

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

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

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

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

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

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

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

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

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

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

El funcionamiento de la unidad ex es idéntico a em, salvo que en este caso, la referencia es la altura de la letra x minúscula, por lo que su valor es aproximadamente la mitad que el de la unidad em.

Por último, las medidas indicadas en píxel también se consideran relativas, ya que el aspecto de los elementos dependerá de la resolución del dispositivo en el que se visualiza la página HTML. Si un elemento tiene una anchura de 400px, ocupará la mitad de una pantalla con una resolución de 800x600, pero ocupará menos de la tercera parte en una pantalla con resolución de 1440x900.

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

```
body { font-size: 10px; }

h1 { font-size: 2.5em; }
```

En primer lugar, se establece un tamaño de letra base de 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, el valor de la mayoría de propiedades CSS se hereda de padres a hijos. Así por ejemplo, si se establece el tamaño de letra al elemento <body>, todos los elementos de la página tendrán el mismo tamaño de letra, salvo que indiquen otro valor.

Sin embargo, el valor de las medidas relativas no se hereda directamente, sino que se hereda su valor real una vez calculado. El siguiente ejemplo muestra este comportamiento:

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

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

Si se heredara el valor 3em, al multiplicarlo por el valor de font-size del elemento <h1> (que vale 15px) el resultado sería 3em 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.

- **3.1.3. Porcentajes**

  El porcentaje también es una unidad de medida relativa, aunque por su importancia CSS la trata de forma separada a em, ex y px. Un porcentaje está formado por un valor numérico seguido del símbolo % y siempre está referenciado a otra medida. Cada una de las propiedades de CSS que permiten indicar como valor un porcentaje, define el valor al que hace referencia ese porcentaje.

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

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

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

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

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

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

En el ejemplo anterior, la referencia del valor 80% es la anchura de su elemento padre. Por tanto, el elemento 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, elaborado por el organismo W3C, recomienda el uso de la unidad em para indicar el tamaño del texto y para todas las medidas que sean posibles.

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

- **3.2. Colores**

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

- **3.2.1. Palabras clave**

  CSS define 140 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

Colores definidos mediante las palabras clave de CSS. Especificación oficial de CSS.

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

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

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

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

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

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

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

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

- **3.2.3. RGB porcentual**

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

  El mismo color del ejemplo anterior se puede representar de forma porcentual:

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

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

- **3.2.4. RGB hexadecimal**

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

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

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

  Como sólo conocemos 10 símbolos numéricos, el sistema hexadecimal utiliza también seis letras (de la A a la F) para representar los números. De esta forma, en el sistema hexadecimal, después del 9 no va el 10, sino la A. La letra B equivale al número 11, la C al 12, la D al 13, la E al 14 y la F al número 15.

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

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

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

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

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

#AAA = #AAAAAA

#FFF = #FFFFFF

#A0F = #AA00FF

#369 = #336699

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

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

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

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

- **3.2.5. Colores del sistema**

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

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

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

- **3.2.6. Colores web safe**

  Como cada componente RGB de los colores puede tomar un valor entre 0 y 255, el número total de colores que se pueden representar con este formato es de 256 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.

FIN

- Texto (http://librosweb.es/css/).
- Enlaces (http://librosweb.es/css/).
- Imágenes (http://librosweb.es/css/).
- Listas (http://librosweb.es/css/).
- Tablas (http://librosweb.es/css/).
- Formularios (http://librosweb.es/css/).

Capítulo 6. Texto

- **6.1. Tipografía**

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

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

Propiedad 	**color**
Valores	color | inherit
Se aplica a	Todos los elementos
Valor inicial	Depende del navegador
Descripción	Establece el color de letra utilizado para el texto

  Aunque el color por defecto del texto depende del navegador, todos los navegadores principales utilizan el color negro. Para establecer el color de letra de un texto, se puede utilizar cualquiera de las cinco formas que incluye CSS para definir un color.

A continuación se muestran varias reglas CSS que establecen el color del texto de diferentes formas:

```
h1 { color: #369; }

p  { color: black; }

a, span { color: #B1251E; }

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

Como el valor de la propiedad color se hereda, normalmente se establece la propiedad color en el elemento body para establecer el color de letra de todos los elementos de la página:

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

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

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

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

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

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

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

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

Los navegadores muestran el texto de las páginas web utilizando los tipos de letra instalados en el ordenador o dispositivo del propio usuario. De esta forma, si el diseñador indica en la propiedadfont-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 propiedadfont-family toma un valor igual a sans-serif, el diseñador no indica al navegador que debe utilizar la fuente Arial, sino que debe utilizar *"la fuente que más se parezca a Arial de todas las que tiene instaladas el usuario"*.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Las reglas CSS del ejemplo anterior se muestran a continuación:

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

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

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

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

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

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

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

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.

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

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

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

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

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

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

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

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

Ejemplos de uso de la propiedad font:

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

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

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

- **6.2. Texto**

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

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

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

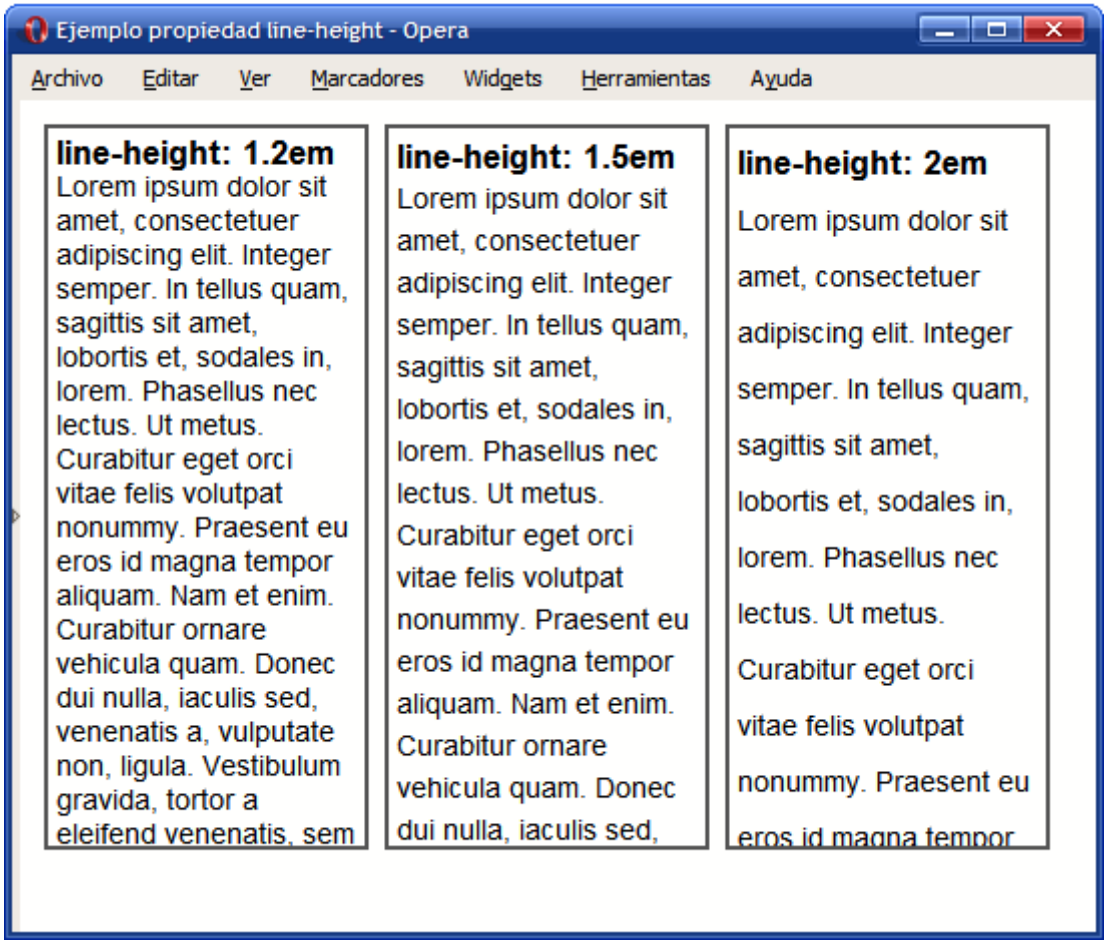
El interlineado de un texto se controla mediante la propiedad line-height, que permite controlar la altura ocupada por cada línea de texto:

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

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

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

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

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

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



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

Propiedad	**text-decoration**
Valores	none | (underline || overline || line-through || blink) | inherit
Se aplica a	Todos los elementos
Valor inicial	none
Descripción	Establece la decoración del texto (subrayado, tachado, parpadeante, etc.)

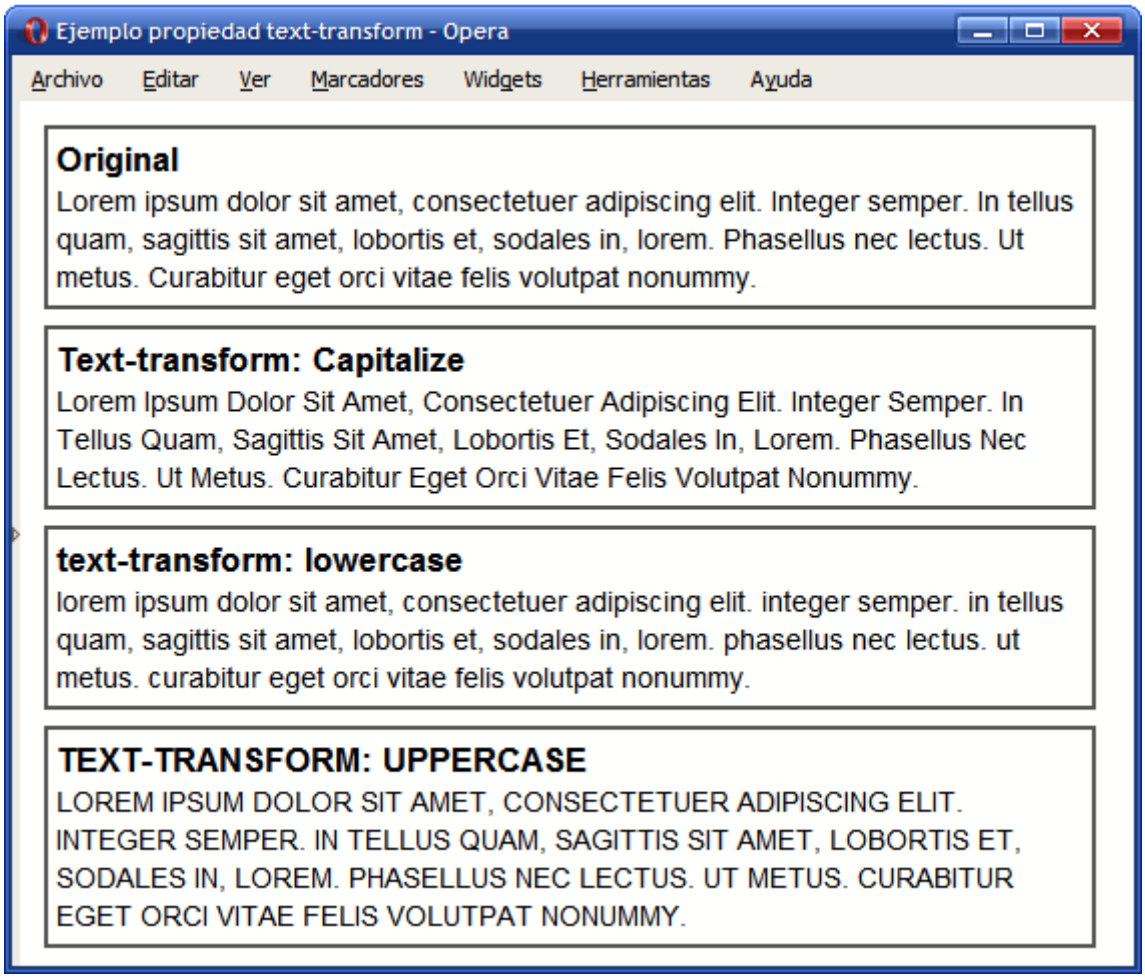
El valor underline subraya el texto, por lo que puede confundir a los usuarios haciéndoles creer que se trata de un enlace. El valor overline añade una línea en la parte superior del texto, un aspecto que raramente es deseable. El valor line-through muestra el texto tachado con una línea continua, por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

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

Propiedad	**text-transform**
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:



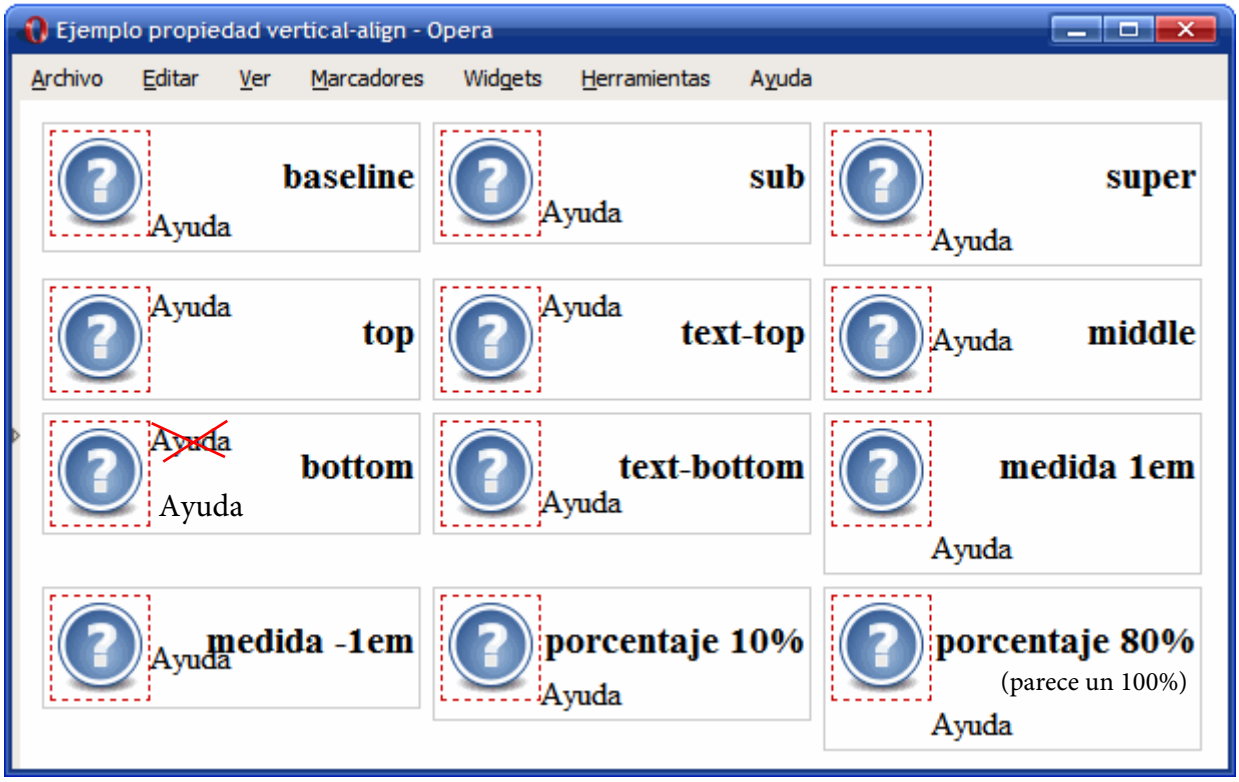
Las reglas CSS del ejemplo anterior se muestran a continuación:

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

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

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

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

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

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

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



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

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

Propiedad	**text-indent**
Valores	unidad de medida | porcentaje | inherit
Se aplica a	Los elementos de bloque y las celdas de tabla
Valor inicial	0
Descripción	Tabula desde la izquierda la primera línea del texto original

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

Propiedad	**letter-spacing**
Valores	normal | unidad de medida | inherit
Se aplica a	Todos los elementos
Valor inicial	normal
Descripción	Permite establecer el espacio entre las letras que forman las palabras del texto
Propiedad	**word-spacing**
Valores	normal | unidad de medida | inherit
Se aplica a	Todos los elementos
Valor inicial	normal
Descripción	Permite establecer el espacio entre las palabras que forman el texto

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

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

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

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

Propiedad	**white-space**
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

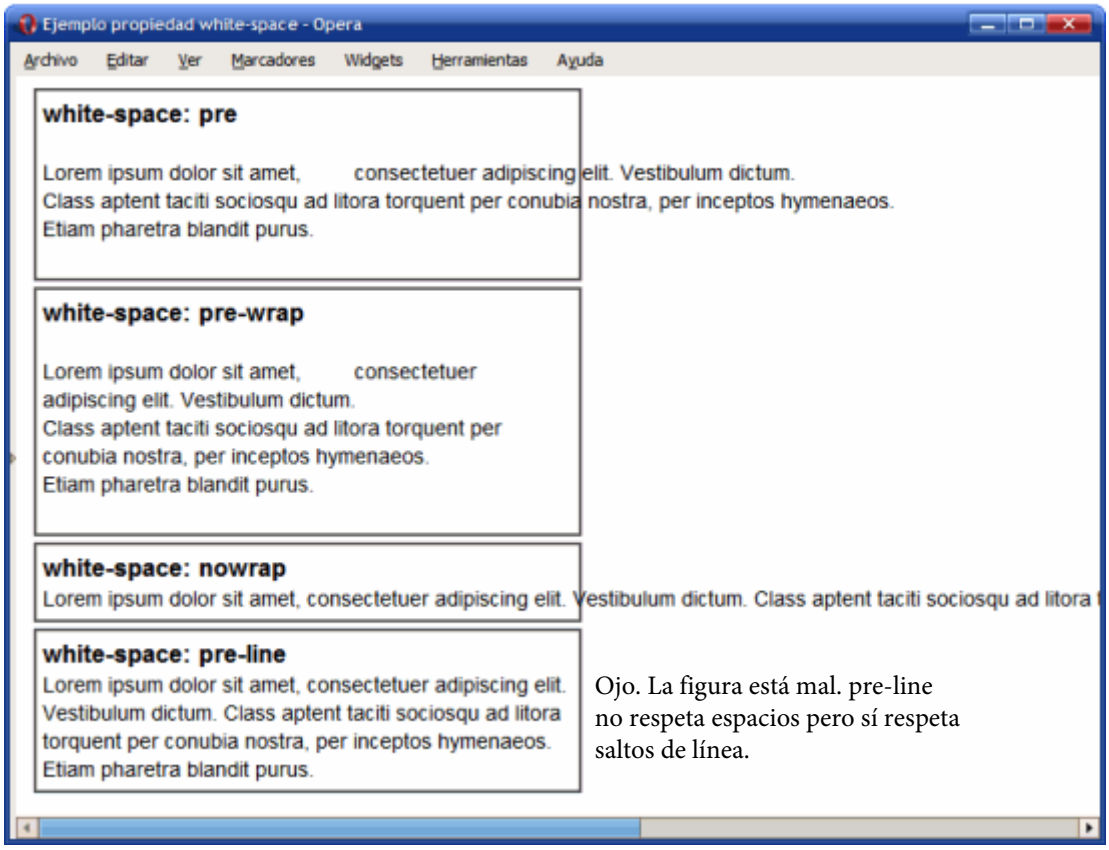
El significado de cada uno de los valores es el siguiente:

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

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

Valor	Respeta espacios en blanco	Respeta saltos de línea	Ajusta las líneas
normal	no	no	si
pre	si	si	no
nowrap	no	no	no
pre-wrap	si	si	si
pre-line	no	si	si

La siguiente imagen muestra las diferencias entre los valores permitidos para white-space. El párrafo original contiene espacios en blanco y nuevas líneas y se ha limitado la anchura de su elemento contenedor:

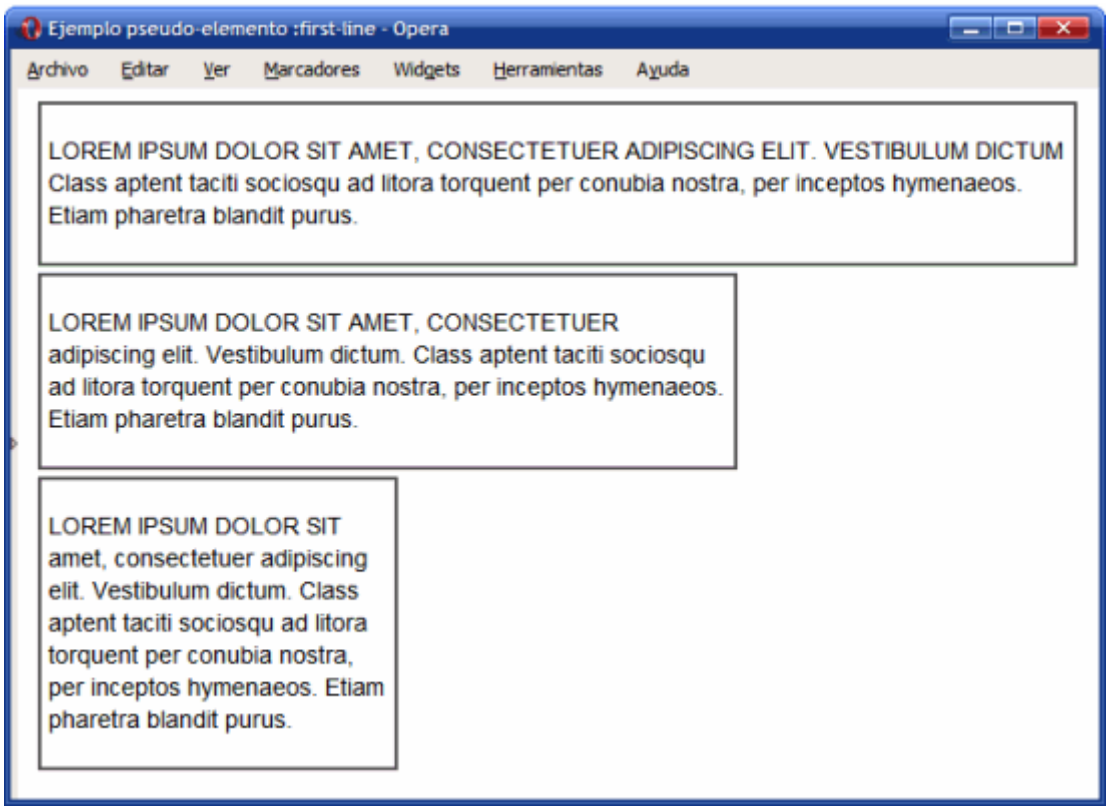


Ojo. La figura está mal. pre-line no respeta espacios pero sí respeta saltos de línea.

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.

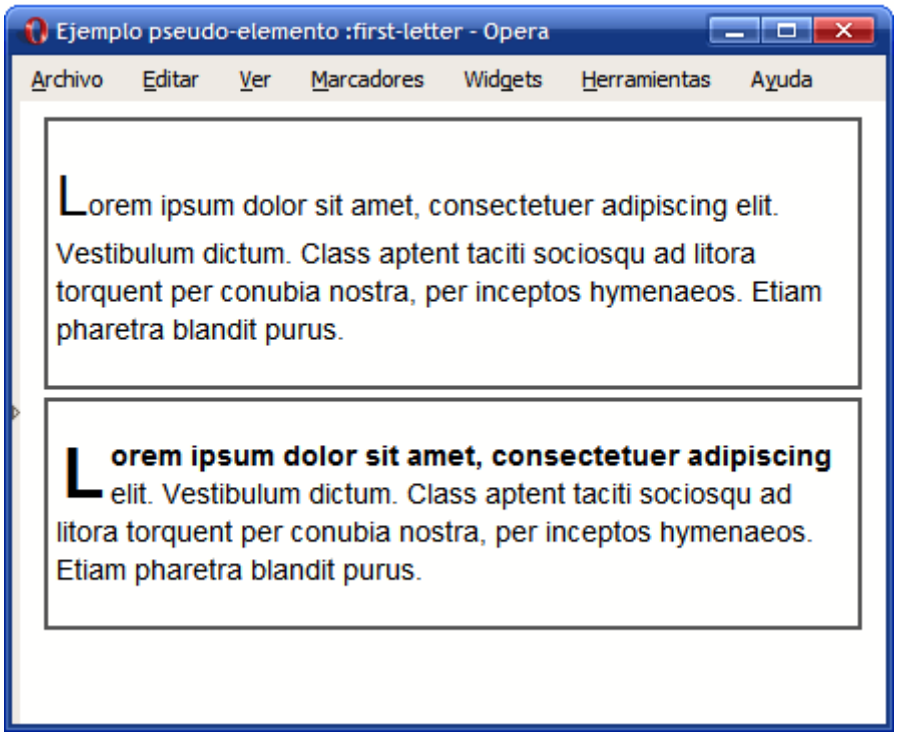
La siguiente imagen muestra cómo aplica CSS los estilos indicados a la primera línea calculando para cada anchura las palabras que forman la primera línea:



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-letterpara crear una letra capital:



El código HTML y CSS se muestra a continuación:

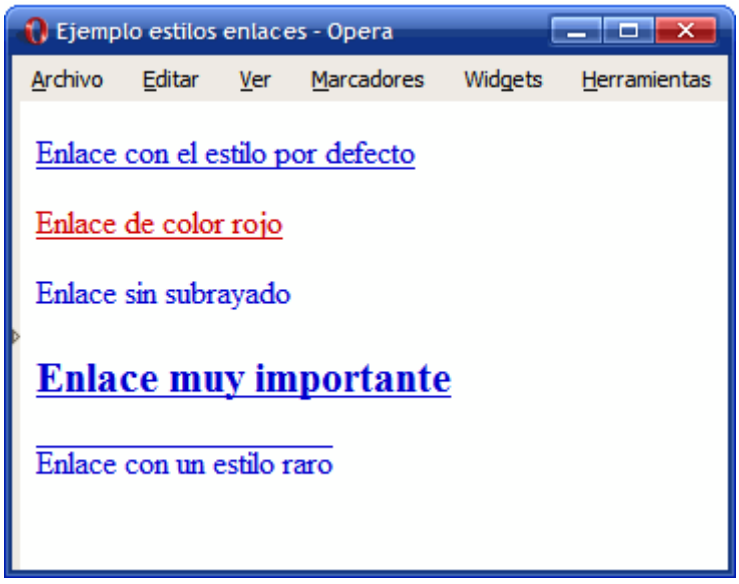
```
#normal p:first-letter {
        font-size: 2em;
}
#avanzado p:first-letter {
        font-size: 2.5em;
        font-weight: bold;
        line-height: .9em;
        float: left;
        margin: .1em;
}
#avanzado p:first-line {
        font-weight: bold;
}
```

```
<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 yfont-weight se pueden conseguir estilos como los que se muestran en la siguiente imagen:



A continuación se muestran las reglas CSS del ejemplo anterior:

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

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

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

- **7.1.2. Pseudo-clases**

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

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

- :link, aplica estilos a los enlaces que apuntan a páginas o recursos que aún no han sido visitados por el usuario.
- :visited, aplica estilos a los enlaces que apuntan a recursos que han sido visitados anteriormente por el usuario. El historial de enlaces visitados se borra automáticamente cada cierto tiempo y el usuario también puede borrarlo manualmente.
- :hover, aplica estilos al enlace sobre el que el usuario ha posicionado el puntero del ratón.

- :active, aplica estilos al enlace que está pinchando el usuario. Los estilos sólo se aplican desde que el usuario pincha el botón del ratón hasta que lo suelta, por lo que suelen ser unas pocas décimas de segundo.

Como se sabe, por defecto los navegadores muestran los enlaces no visitados de color azul y subrayados y los enlaces visitados de color morado. Las pseudo-clases anteriores permiten modificar completamente ese aspecto por defecto y por eso casi todas las páginas las utilizan.

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

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

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

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

```
/* Incorrecto: el nombre de la pseudo-clase no se puede separar de los dos puntos iniciales */

a: visited { ... }


/* Incorrecto: la pseudo-clase siempre se añade a continuación del elemento
               al que modifica */

a :visited { ... }


/* Correcto: la pseudo-clase modifica el comportamiento del elemento <a> */

a:visited { ... }
```

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

```
a.importante:visited { ... }

a#principal:hover { ... }

div#menu ul li a.primero:active { ... }
```

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

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

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

a:hover, a:active {
        ...
}
```

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

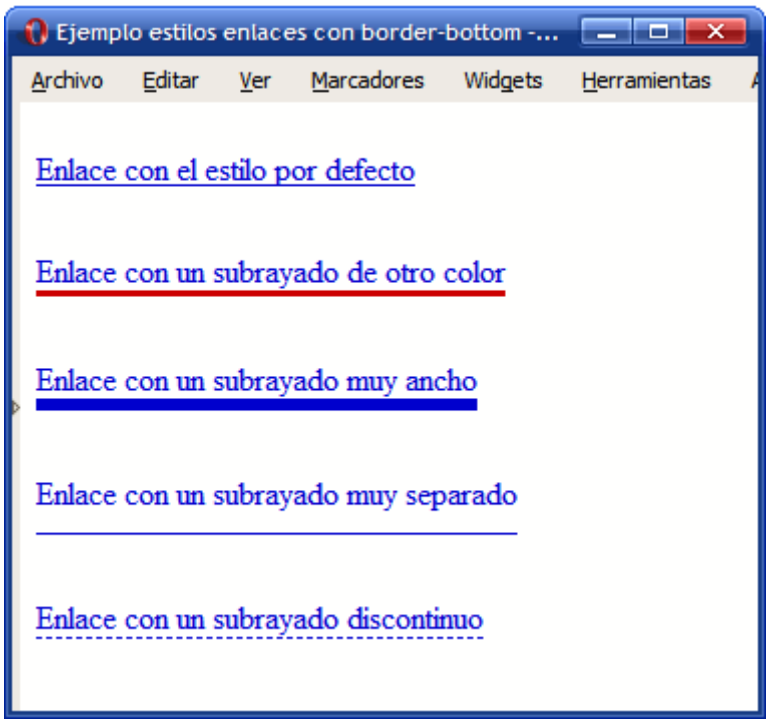
También es posible combinar en un mismo elemento las pseudo-clases que son compatibles entre sí:

```
/* Los estilos se aplican cuando el usuario pasa el ratón por encima de un
        enlace que todavía no ha visitado */
a:link:hover { ... }

/* Los estilos se aplican cuando el usuario pasa el ratón por encima de un
        enlace que ha visitado previamente */
a:visited:hover { ... }
```

- **7.2. Estilos avanzados de enlaces**
- **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:



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 (ybackground-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 paddingnecesario 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:

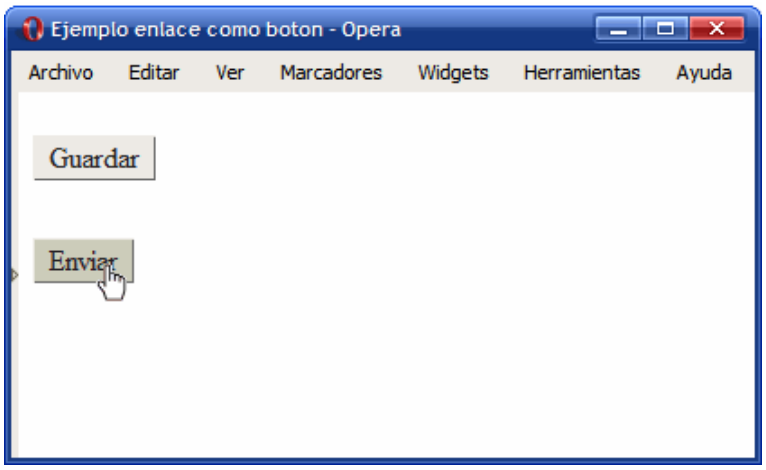
Las reglas CSS necesarias se muestran a continuación:

```
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:



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

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

a.boton {
        text-decoration: none;
        background: #EEE;
        color: #222;
        border: 1px outset #CCC;
        padding: .1em .5em;
}

a.boton:hover {
        background: #CCB;
}

a.boton:active {
        border: 1px inset #000;
}
```

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

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

Capítulo 8. Imágenes

- **8.1. Estilos básicos**
- **8.1.1. Establecer la anchura y altura de las imágenes**

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

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

```
<img id="destacada" src="imagen.png" />
```

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

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

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

```
<img src="imagen.png" width="120" height="250" />
```

- **8.1.2. Eliminar el borde de las imágenes con enlaces**

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

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

Capítulo 9. Listas

- **9.1. Estilos básicos**
- **9.1.1. Viñetas personalizadas**

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

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

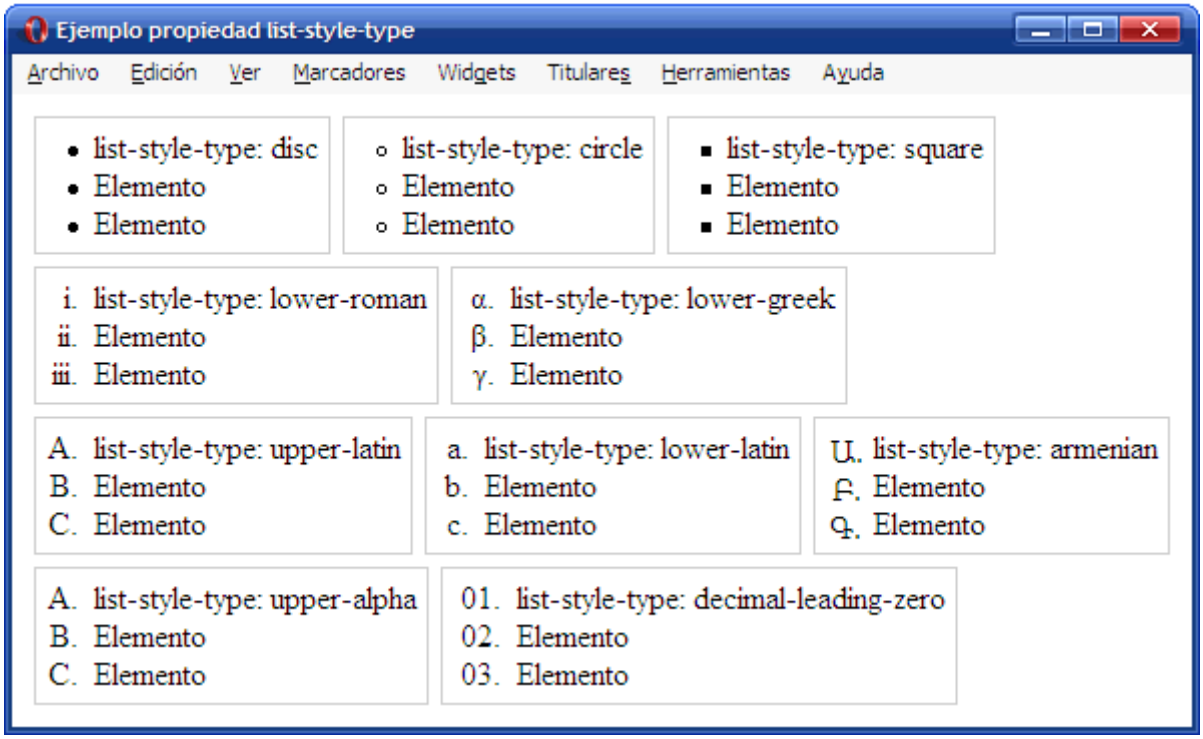
Propiedad	**list-style-type**
Valores	disc | circle | square | decimal | decimal-leading-zero | lower-roman | upper-roman | lower-greek | lower-latin | upper-latin | armenian | georgian | lower-alpha | upper-alpha | none | inherit
Se aplica a	Elementos de una lista
Valor inicial	disc
Descripción	Permite 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:



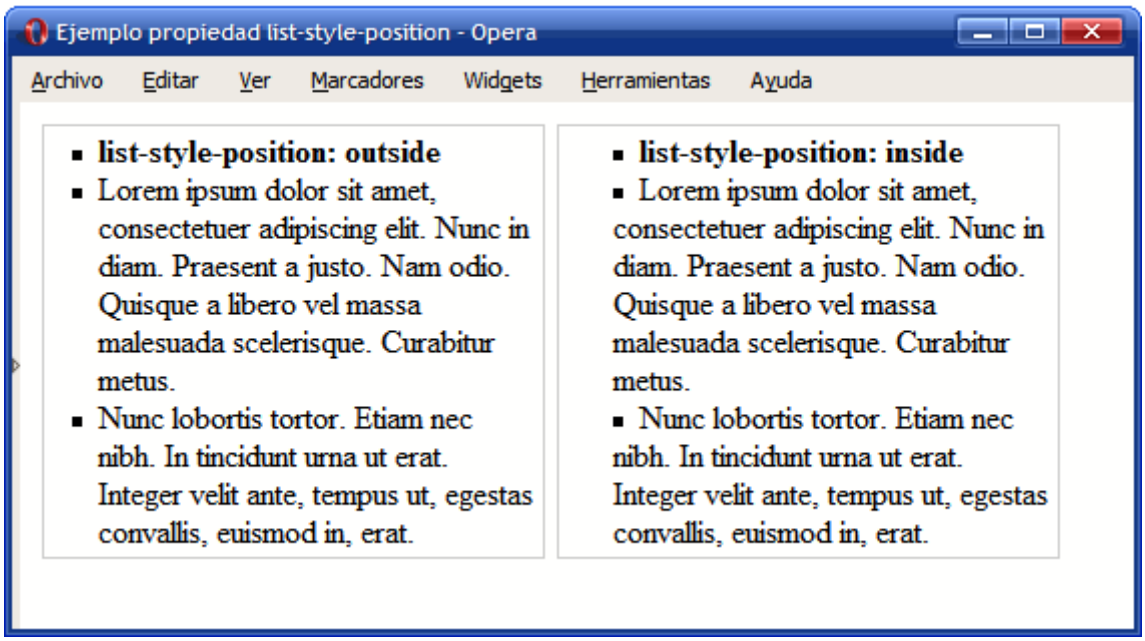
El código CSS de algunas de las listas del ejemplo anterior se muestra a continuación:

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

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

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

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

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

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



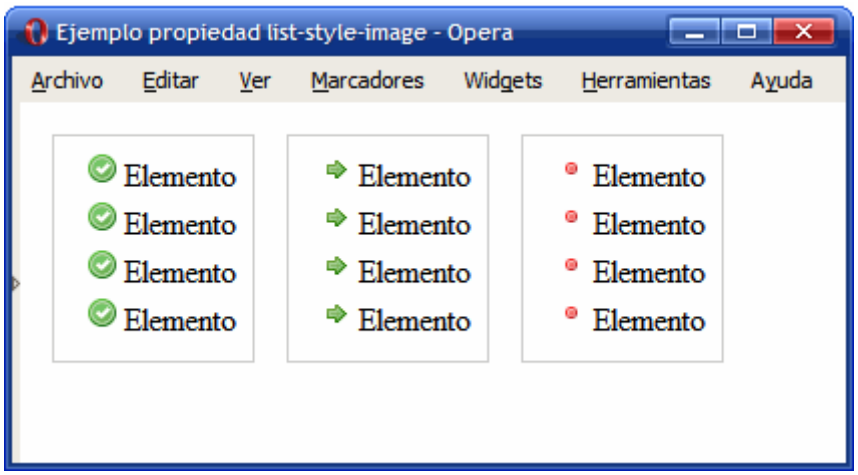
Utilizando las propiedades anteriores (list-style-type y list-style-position), se puede seleccionar el tipo de viñeta y su posición, pero no es posible personalizar algunas de sus características básicas como su color y tamaño.

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

Propiedad	**list-style-image**
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:



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

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

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

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

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

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

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

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

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

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

- **9.1.2. Menú vertical**

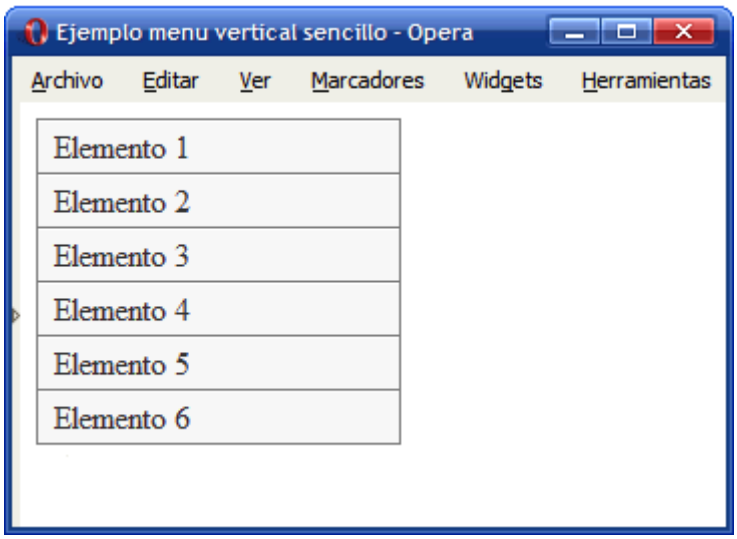
Los sitios web correctamente diseñados emplean las listas de elementos para crear todos sus menús de navegación. Utilizando la etiqueta <ul> de HTML se agrupan todas las opciones del menú y haciendo uso de CSS se modifica su aspecto para mostrar un menú horizontal o vertical.

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

Lista de enlaces original:

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

Aspecto final del menú vertical:



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; }
```

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

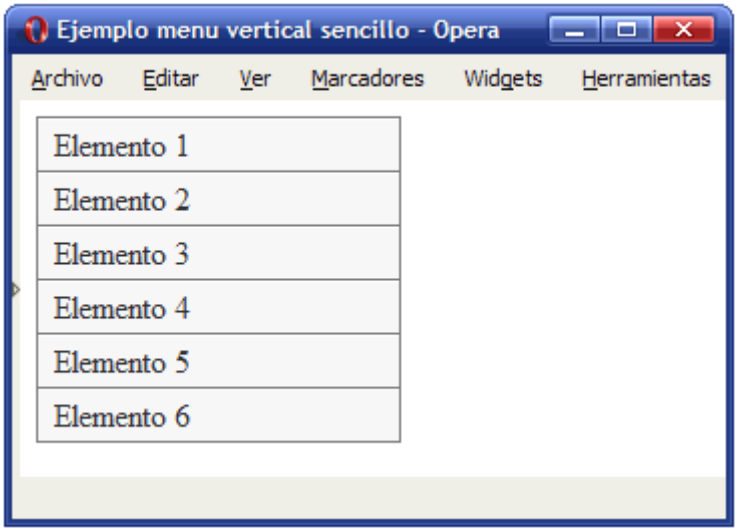
```
ul.menu {

 list-style: none;

 margin: 0;

 padding: 0;

 width: 180px;

}
```

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

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

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



Los tipos de menús verticales que se pueden definir mediante las propiedades CSS son innumerables.

- **9.2. Estilos avanzados de listas**
- **9.2.1. Menú horizontal básico**

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

Código HTML del menú horizontal:

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

Aspecto final del menú horizontal:

Elemento 1	Elemento 2	Elemento 3	Elemento 4	Elemento 5

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

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

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

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

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

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

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

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

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

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

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

Elemento 1	Elemento 2	Elemento 3	Elemento 4	Elemento 5

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

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

Elemento 1	Elemento 2	Elemento 3	Elemento 4	Elemento 5

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

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

Elemento 1	Elemento 2	Elemento 3	Elemento 4	Elemento 5

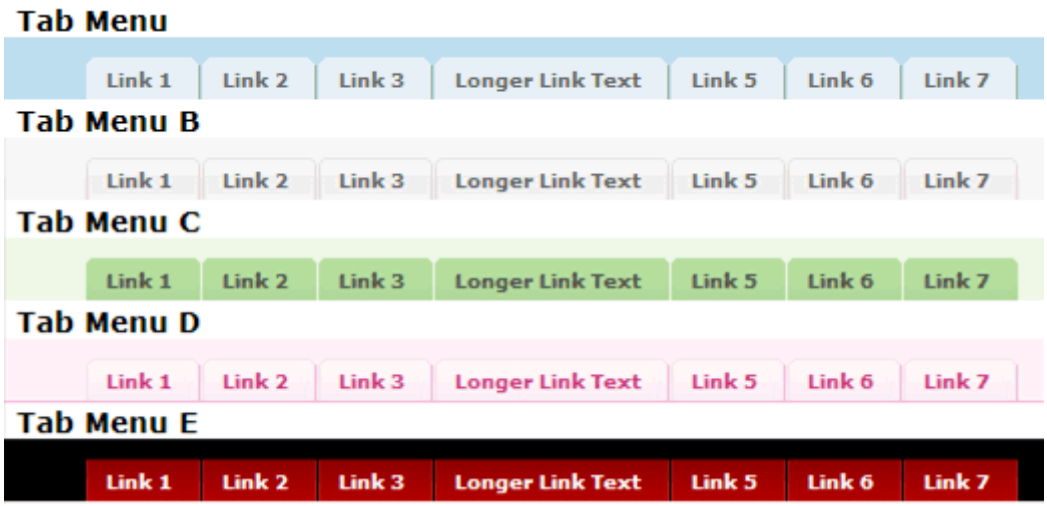
El código CSS final se muestra a continuación:

```
ul.menu {
 background: #F4F4F4;
 border: 1px solid #7C7C7C;
 list-style: none;
 margin: 0;
 padding: 0;
 overflow: hidden;
}
ul.menu li {
 float: left;
 width: 20%;
}
```

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

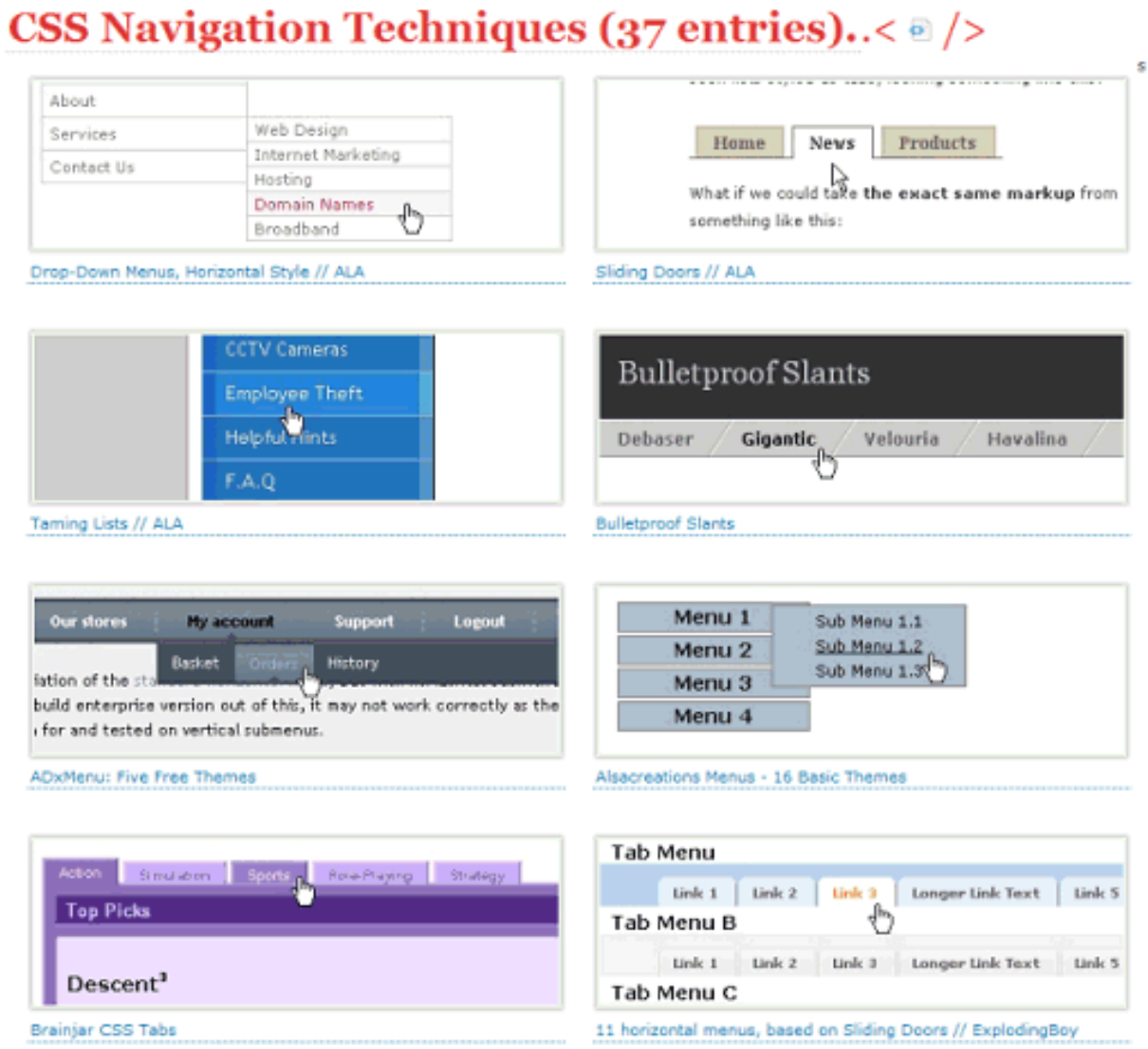
- **9.2.2. Menús avanzados**

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:



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

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



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

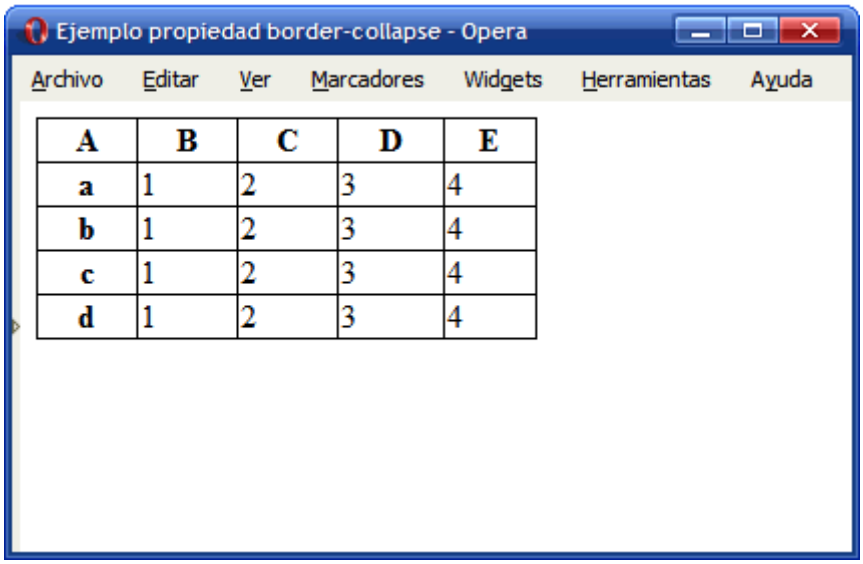
- **10.1. Estilos básicos**

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

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

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

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



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

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

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

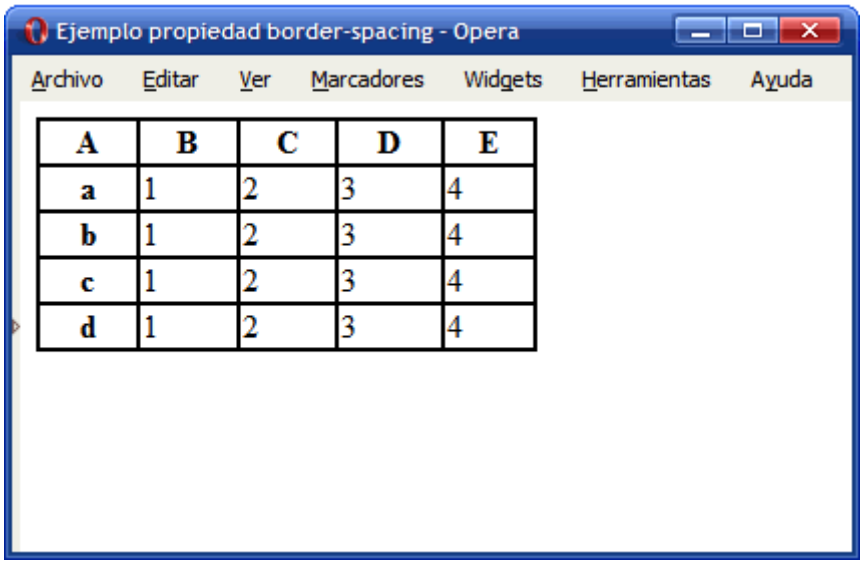
Aunque parece sencillo, el mecanismo que utiliza el modelo collapse es muy complejo, ya que cuando los bordes que se fusionan no son exactamente iguales, debe tener en cuenta la anchura de cada borde, su estilo y el tipo de celda que contiene el borde (columna, fila, grupo de filas, grupo de columnas)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

Si se opta por el modelo separate (que es el que se aplica si no se indica lo contrario) se puede utilizar la propiedad border-spacing para controlar la separación entre los bordes de cada celda.

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

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

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



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

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

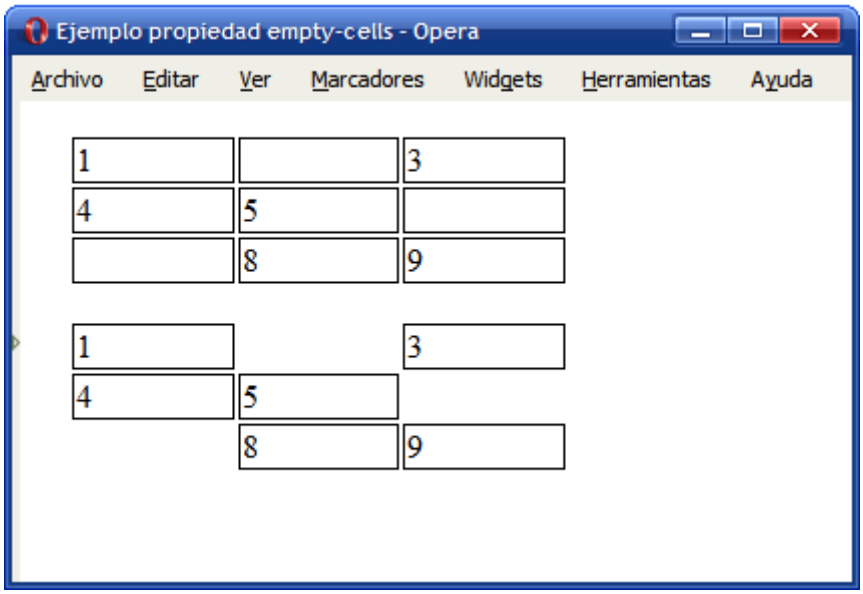
- **10.2. Estilos avanzados de tablas**

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 propiedadempty-cells. Esta propiedad sólo se aplica cuando el modelo de bordes de la tabla es de tiposeparate.

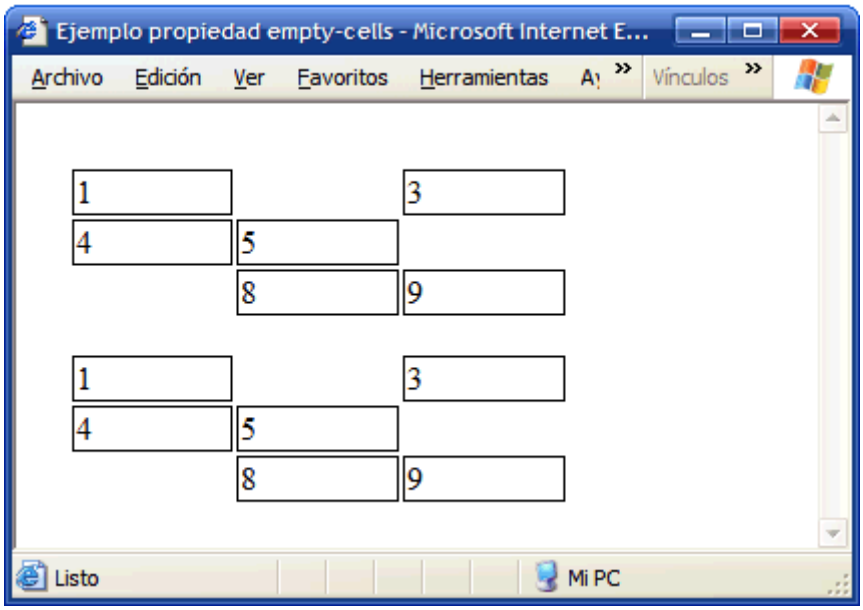
Propiedad	**empty-cells**
Valores	show | hide | inherit
Se aplica a	Celdas de una tabla
Valor inicial	show
Descripción	Define el mecanismo utilizado para el tratamiento de las celdas vacías de una tabla

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

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



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



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

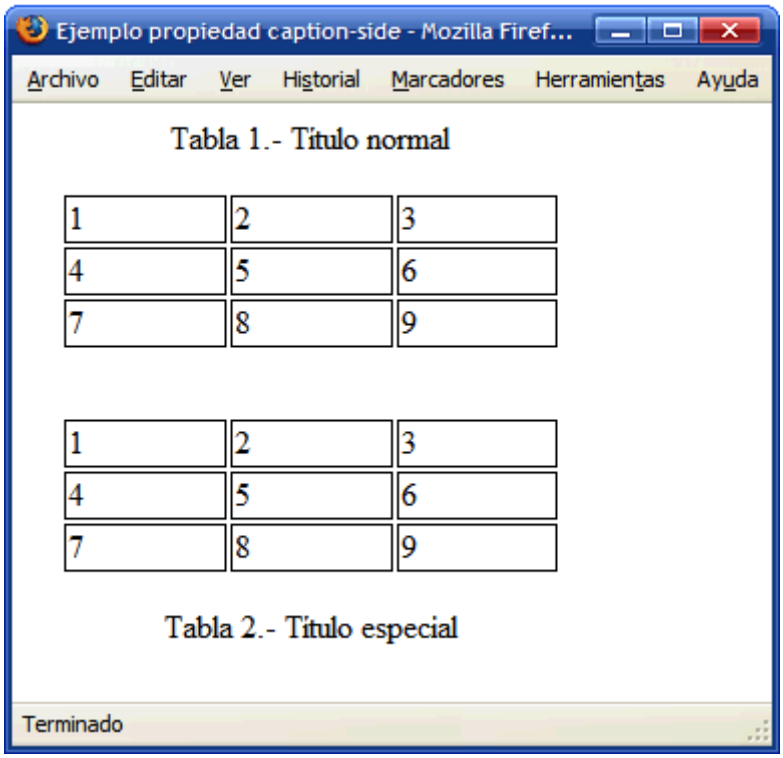
Propiedad	**caption-side**
Valores	top | bottom | inherit
Se aplica a	Los elementos caption
Valor inicial	top
Descripción	Establece la posición del título de la tabla

El valor bottom indica que el título de la tabla se debe mostrar debajo de los contenidos de la tabla. Si también se quiere modificar la alineación horizontal del título, debe utilizarse la propiedad text-align.

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

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

<table class="especial" summary="Tabla genérica">
	<caption>Tabla 2.- Título especial</caption>
	<tr>
		<td>1</td>
		<td>2</td>
		<td>3</td>
	</tr>
	...
</table>
```

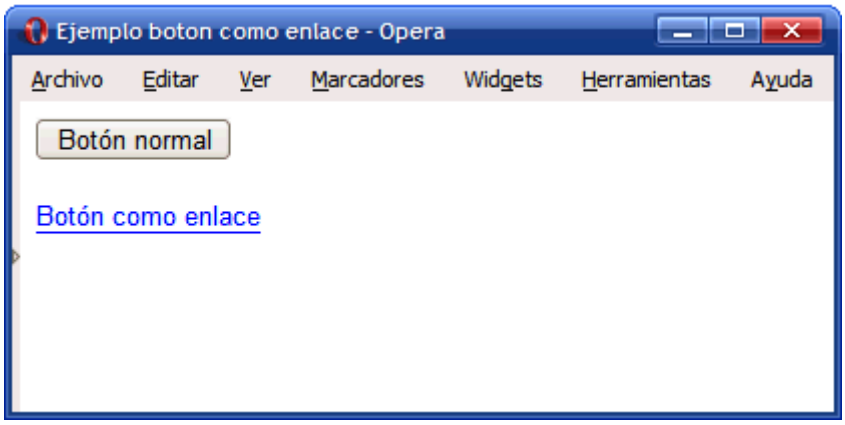
La siguiente imagen muestra el resultado de visualizar el ejemplo anterior en cualquier navegador:



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.



Las reglas CSS del ejemplo anterior son las siguientes:

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

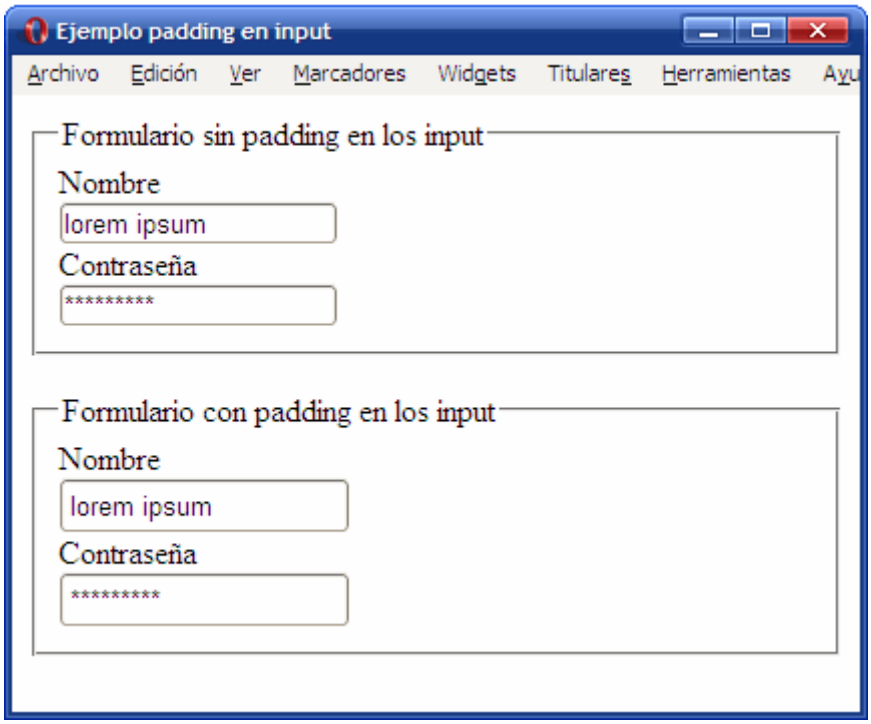
```
<input type="button" value="Botón normal" />
```

```
<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:

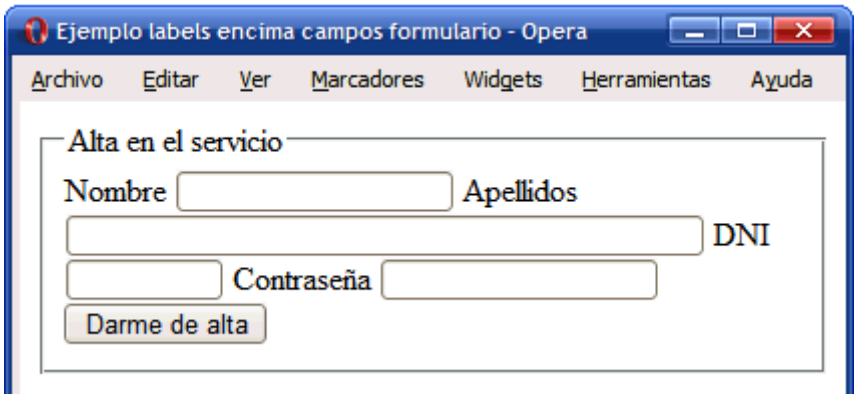


La regla CSS necesaria para mejorar el formulario es muy sencilla:

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

- **11.1.3. Labels alineadas y formateadas**

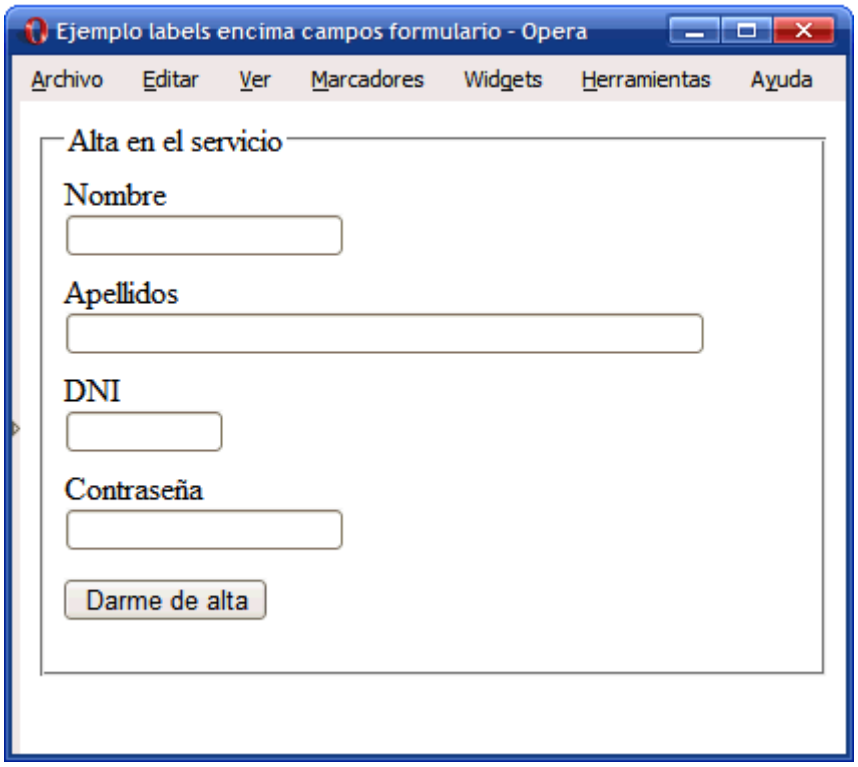
Los elementos <input> y <label> de los formularios son elementos en línea, por lo que el aspecto que muestran los formularios por defecto, es similar al de la siguiente imagen:



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:



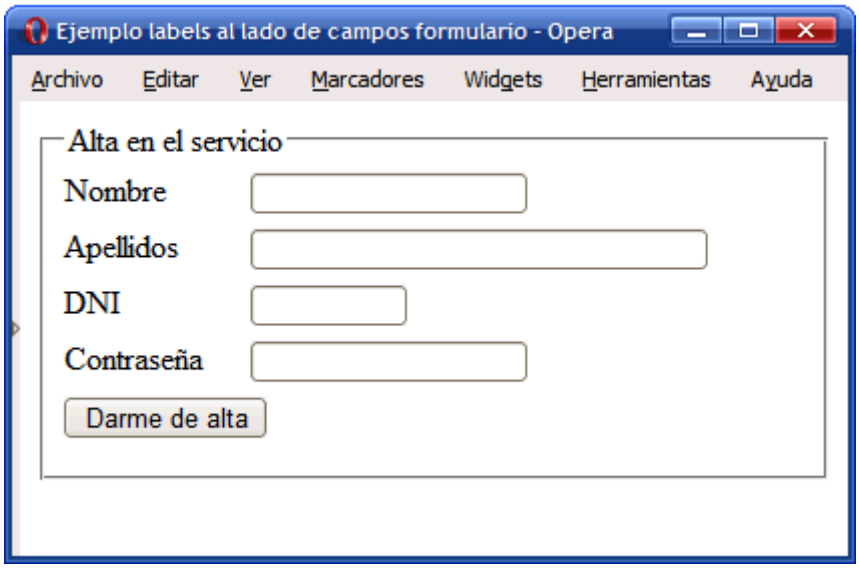
En primer lugar, se muestran los elementos <label> como elementos de bloque, para que añadan una separación para cada campo del formulario. Además, se añade un margen superior para no mostrar juntas todas las filas del formulario:

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

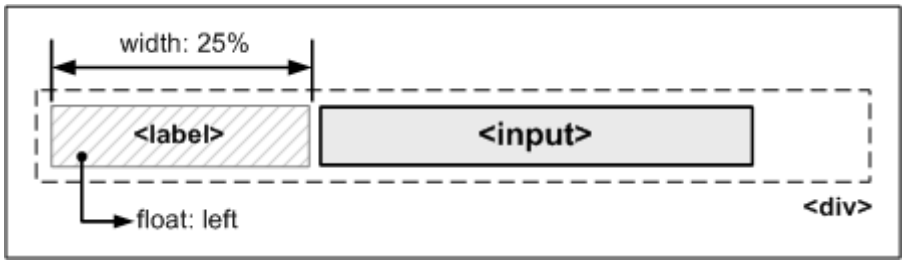
El botón del formulario también se muestra como un elemento de bloque y se le añade un margen para darle el aspecto final deseado:

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

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



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



Por tanto, en el código HTML del formulario anterior se añaden los elementos :

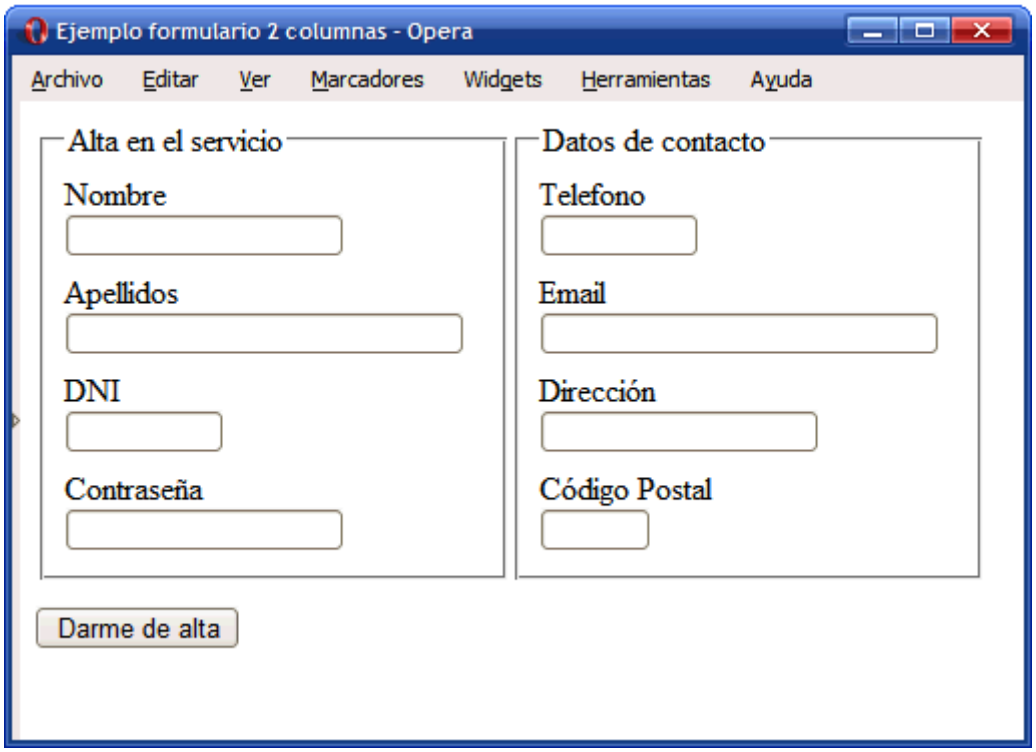
```
<form>
        <fieldset>
        <legend>Alta en el servicio</legend>

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

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

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

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

- **11.2. Estilos avanzados**
- **11.2.1. Formulario en varias columnas**

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



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

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

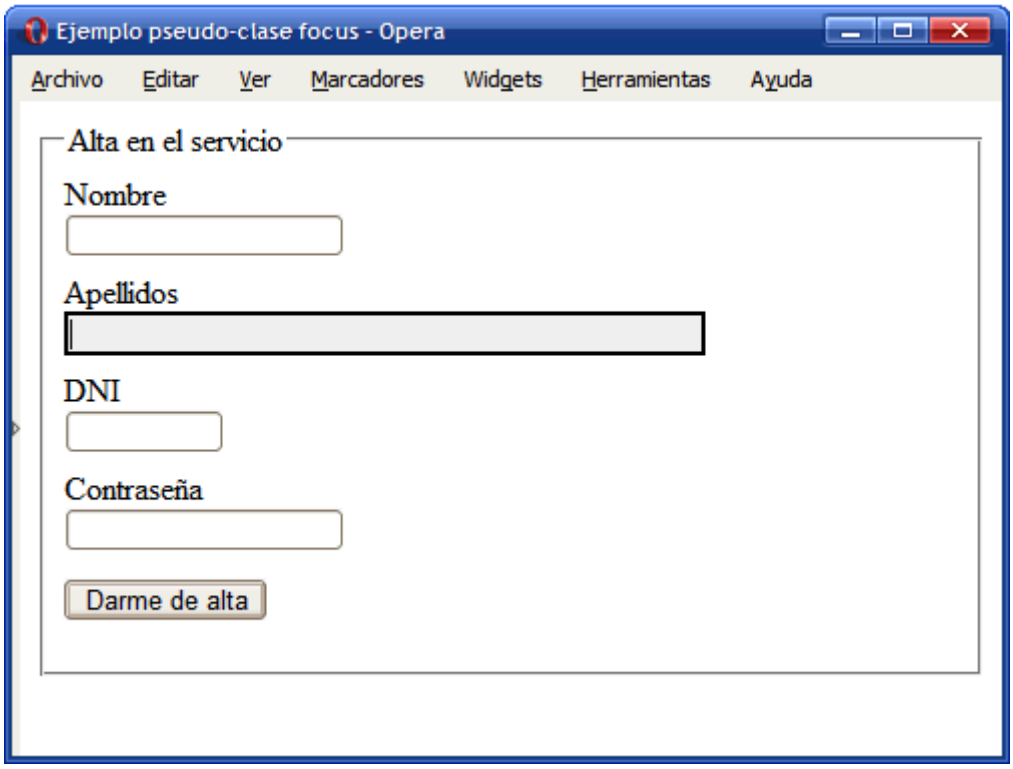
```
<form>
        <fieldset>
                ...
        </fieldset>

        ...
</form>
```

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

- **11.2.2. Resaltar el campo seleccionado**

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

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



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

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

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

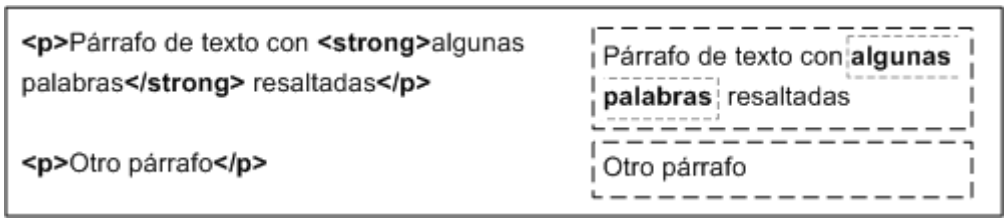
FIN

- Modelo de cajas (http://librosweb.es/css/).
- Posicionamiento y visualización (http://librosweb.es/css/).

Capítulo 4. Modelo de cajas

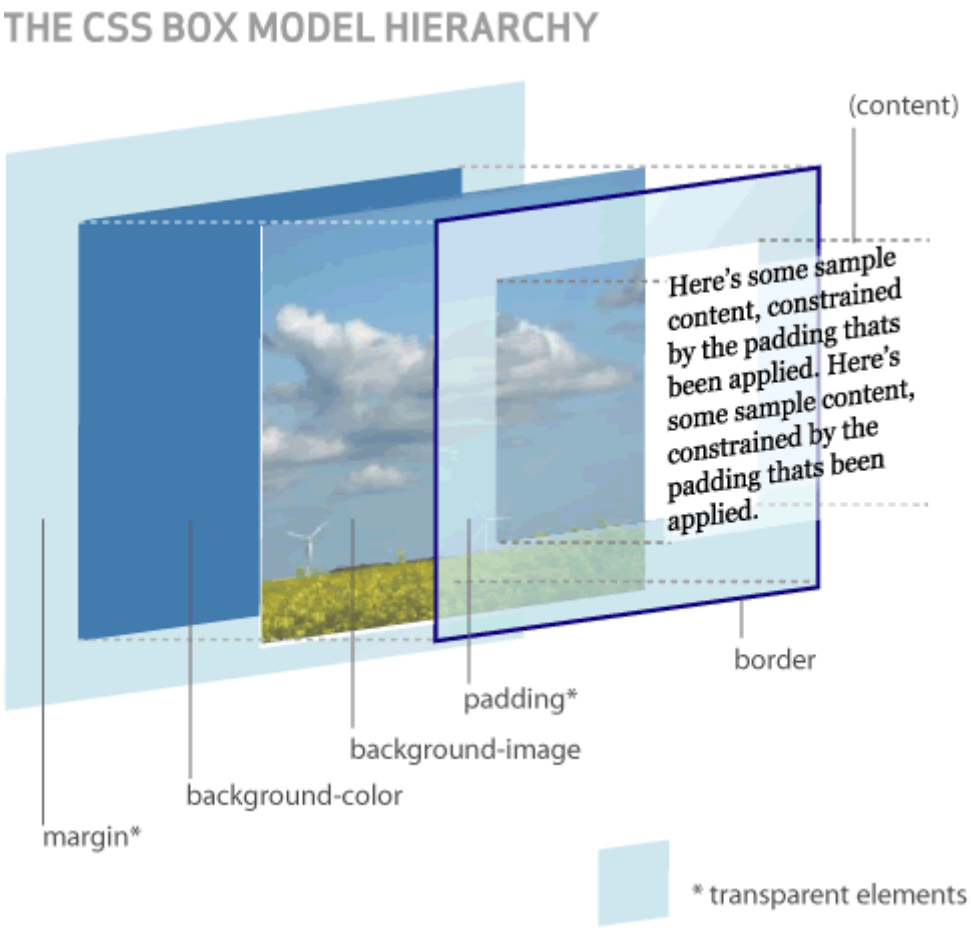
El modelo de cajas o *"box model"* es seguramente la característica más importante del lenguaje de hojas de estilos CSS, ya que condiciona el diseño de todas las páginas web. El modelo de cajas es el comportamiento de CSS que hace que todos los elementos de las páginas se representen mediante cajas rectangulares.

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



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

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



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

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

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

Si una caja define tanto un color como una imagen de fondo, la imagen tiene más prioridad y es la que se visualiza. No obstante, si la imagen de fondo no cubre totalmente la caja del elemento o si la imagen tiene zonas transparentes, también se visualiza el color de fondo. Combinando imagenes transparentes y colores de fondo se pueden lograr efectos gráficos muy interesantes.

- **4.1. Anchura y altura**
- **4.1.1. Anchura**

Con box-sizing: content-box (el valor por defecto) las propiedades width y height afectan sólo al contenido de la caja. Con box-sizing: border-box, width y height hacen referencia al tamaño del contenido, el padding y el borde.

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

Propiedad	**width**
Valores	unidad de medida | porcentaje | auto | inherit
Se aplica a	Todos los elementos, salvo los elementos en línea que no sean imágenes, las filas de tabla y los grupos de filas de tabla
Valor inicial	auto
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 lateral:

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

```
<div id="lateral">
                                                                box-sizing: border-box;
        ...
</div>
```

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

- **4.1.2. Altura**

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

Propiedad	**height**
Valores	unidad de medida | porcentaje | auto | inherit
Se aplica a	Todos los elementos, salvo los elementos en línea que no sean imágenes, las columnas de tabl y los grupos de columnas de 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 de cabecera:

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


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

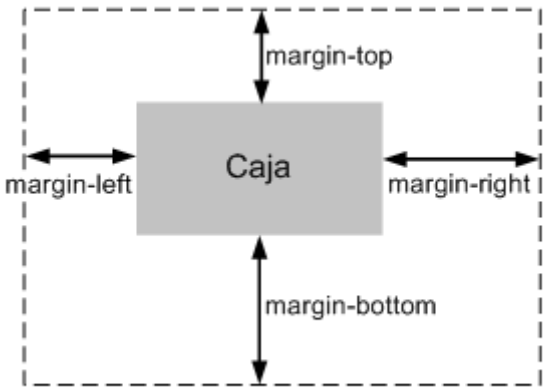
CSS define otras dos propiedades relacionadas con la altura de los elementos: min-height y max-height, que se verán más adelante.

- **4.2. Margen y relleno**
- **4.2.1. Margen**

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

Propiedades	**margin-top, margin-right, margin-bottom,margin-left**
Valores	unidad de medida | porcentaje | auto | inherit
Se aplica a	Todos los elementos, salvo margin-top y margin-bottom que sólo se aplican a los elementos de bloque y a las imágenes
Valor inicial	0
Descripción	Establece cada uno de los márgenes horizontales y verticales de un elemento

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



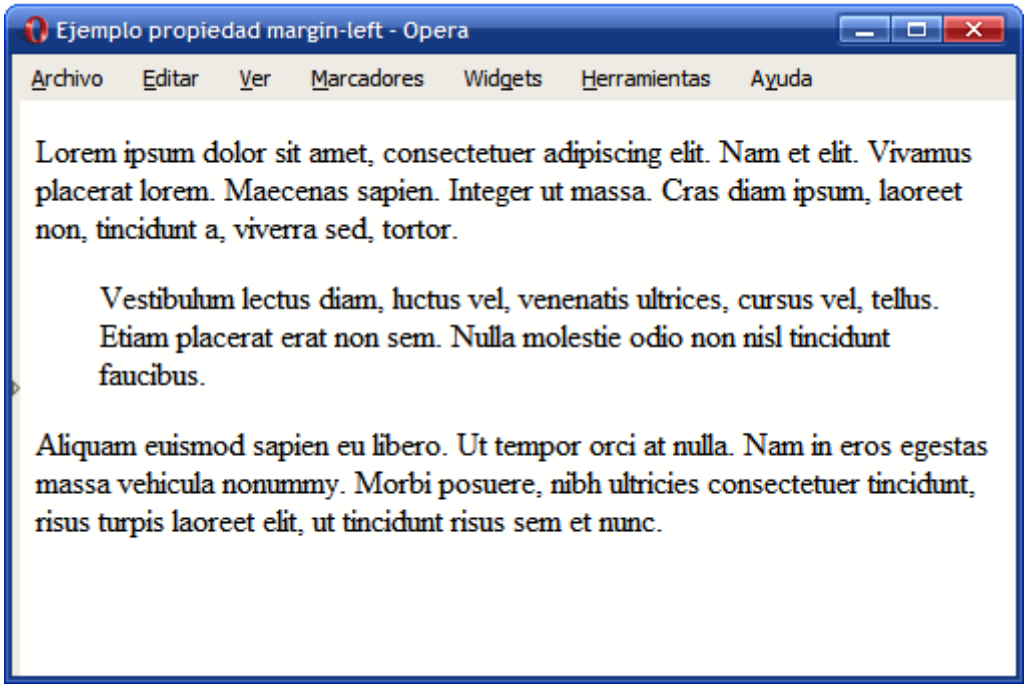
Las unidades más utilizadas para indicar los márgenes de un elemento son los píxeles (cuando se requiere una precisión total), los em (para hacer diseños que mantengan las proporciones) y los porcentajes (para hacer diseños líquidos o fluidos).

El siguiente ejemplo añade un margen izquierdo al segundo párrafo:

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

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

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

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

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



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

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

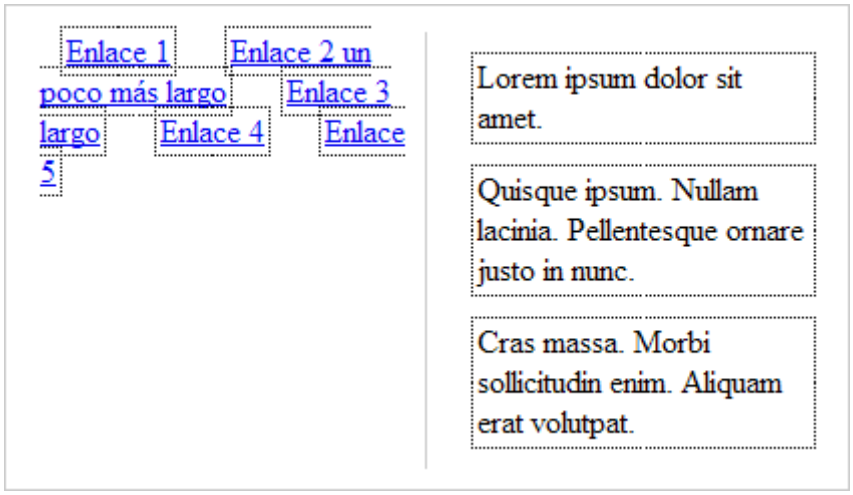


Figura 4.6 Los márgenes verticales sólo se aplican a los elementos de bloque e imágenes

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

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

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

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

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

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

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

Código CSS original:

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

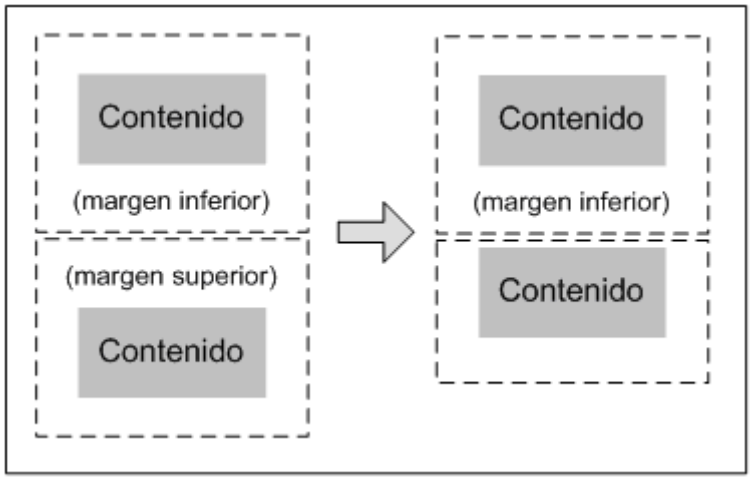
Alternativa directa:

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

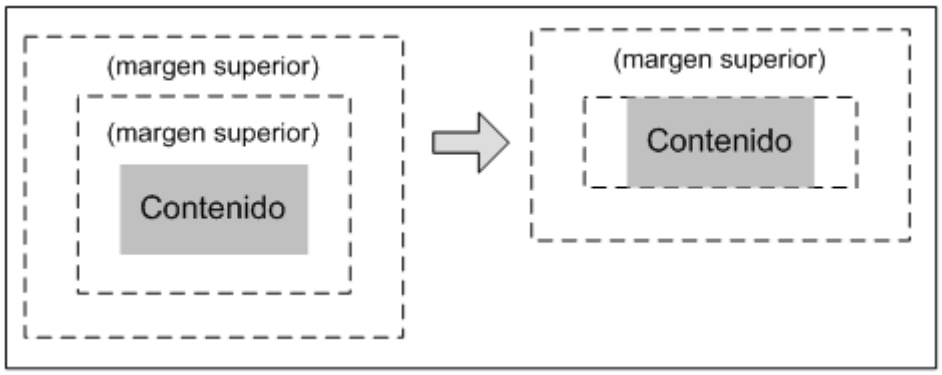
Otra alternativa:

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

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



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:



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

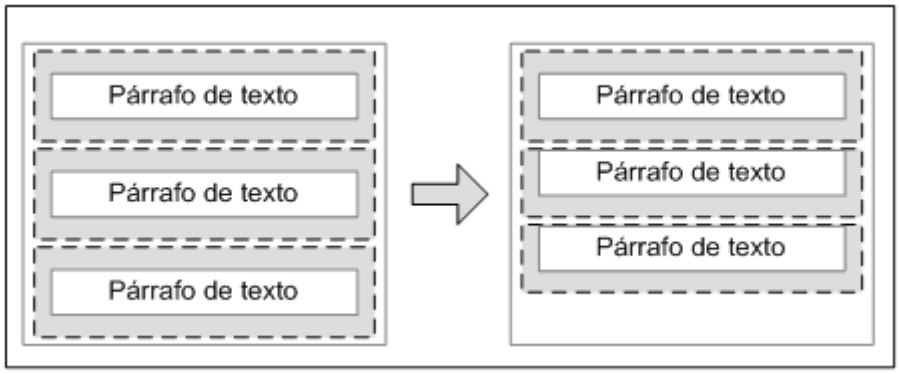


Figura 4.9 Motivo por el que se fusionan automáticamente los márgenes verticales

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

- **4.2.2. Relleno**

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

Propiedades	**padding-top, padding-right, padding-bottom,padding-left**
Valores	unidad de medida | porcentaje | inherit
Se aplica a	Todos los elementos excepto algunos elementos de tablas como grupos de cabeceras y grupos de pies de tabla
Valor inicial	0
Descripción	Establece cada uno de los rellenos horizontales y verticales de un elemento

Cada una de estas propiedades establece la separación entre el contenido y los bordes laterales de la caja del elemento:

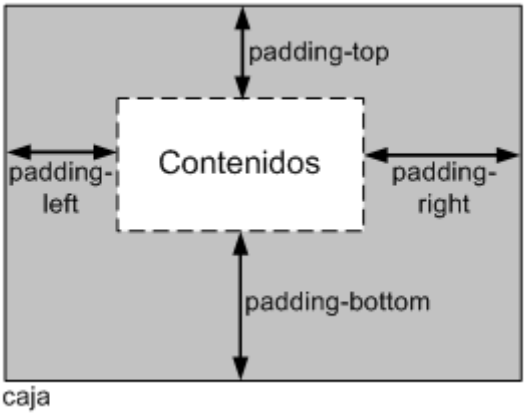


Figura 4.10 Las cuatro propiedades relacionadas con los rellenos

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

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

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 = 2 em */

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

- **4.3. Bordes**

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

- **4.3.1. Anchura**

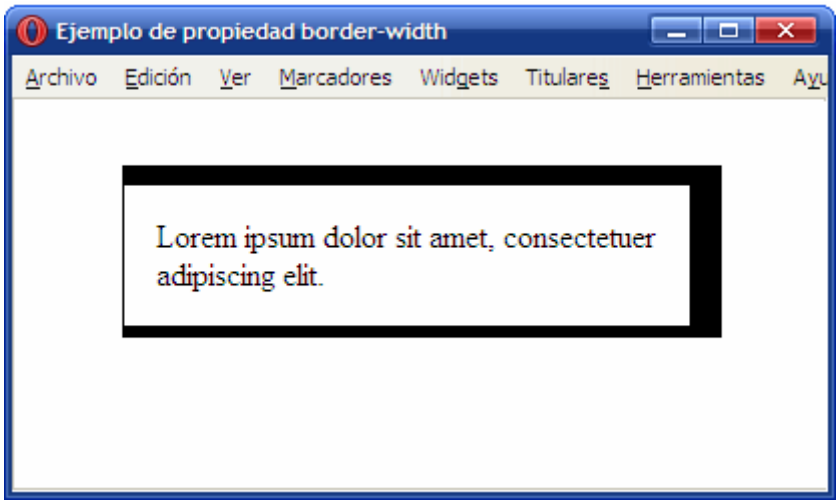
La anchura de los bordes se controla con las cuatro propiedades siguientes:

Propiedades	**border-top-width, border-right-width, border-bottom-width, border-left-width**
Valores	(unidad de medida | thin | medium | thick) | inherit
Se aplica a	Todos los elementos
Valor inicial	Medium
Descripción	Establece la anchura de cada uno de los cuatro bordes de los elementos

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

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

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



Las reglas CSS utilizadas se muestran a continuación:

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

Si se quiere establecer de forma simultánea la anchura de todos los bordes de una caja, es necesario utilizar una propiedad *"shorthand"* llamada border-width:

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

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

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

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

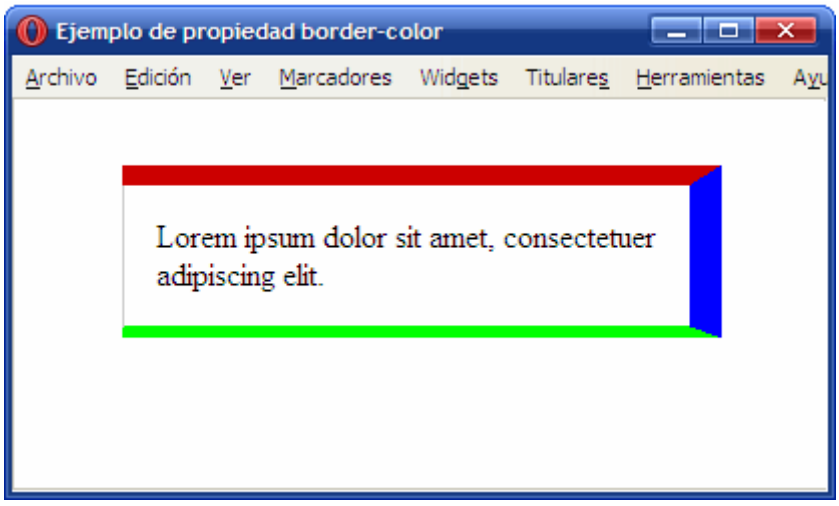
Si se indican tres valores, el primero se aplica al borde superior, el segundo se aplica al borde izquierdo y derecho y el tercer valor se aplica al borde inferior. Si se indican los cuatro valores, el orden de aplicación es superior, derecho, inferior e izquierdo.

- **4.3.2. Color**

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

Propiedades	**border-top-color, border-right-color, border-bottom-color, border-left-color**
Valores	color | transparent | inherit
Se aplica a	Todos los elementos
Valor inicial	-
Descripción	Establece el color de cada uno de los cuatro bordes de los elementos

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



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

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

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

Propiedad	**border-color**
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 las de la propiedad border-width.

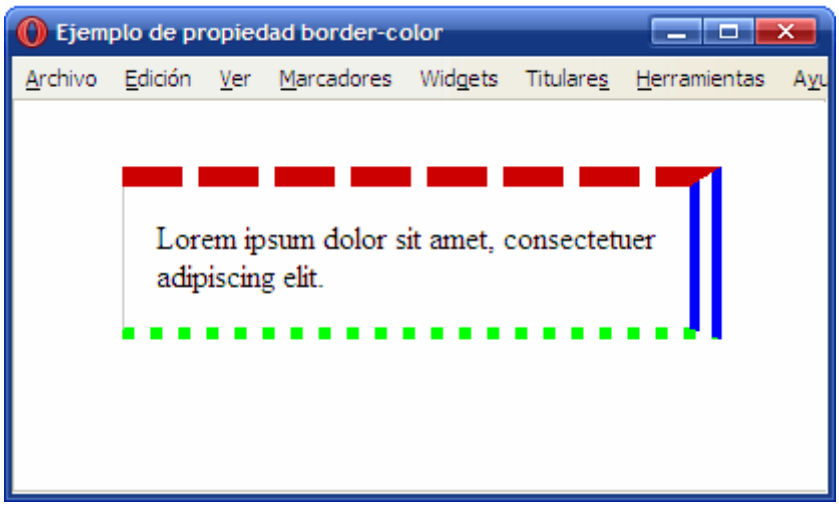
- **4.3.3. Estilo**

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

Propiedades	**border-top-style, border-right-style, border-bottom-style, border-left-style**
Valores	none | hidden | dotted | dashed | solid | double | groove | ridge | inset | outset | inherit
Se aplica a	Todos los elementos
Valor inicial	none
Descripción	Establece el estilo de cada uno de los cuatro bordes de los elementos

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

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



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

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

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

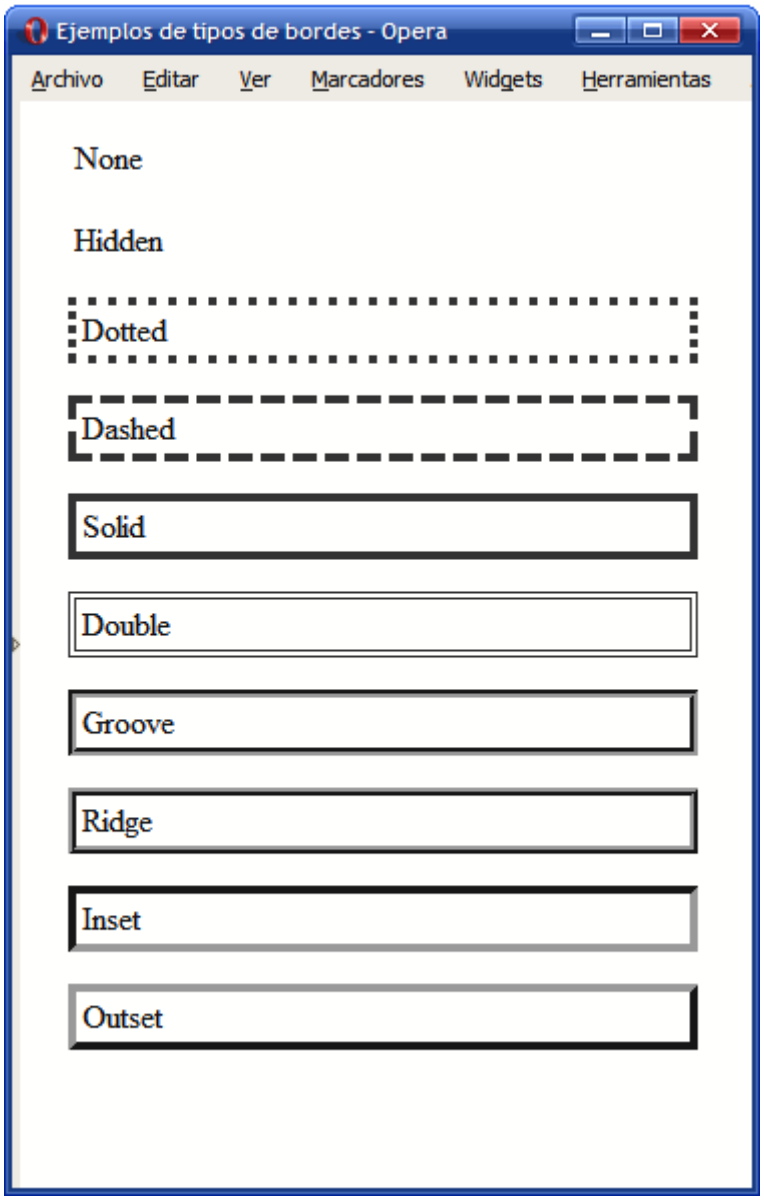


Figura 4.14 Tipos de bordes definidos por CSS

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

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

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

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

- **4.3.4. Propiedades shorthand**

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

Propiedades	**border-top, border-right, border-bottom, border-left**
Valores	(unidad de medida_borde || color_borde || estilo_borde) | inherit
Se aplica a	Todos los elementos
Valor inicial	-
Descripción	Establece el estilo completo de cada uno de los cuatro bordes de los elementos

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

- <medida_borde>: una medida CSS o alguna de las siguientes palabras clave: thin, medium,thick.
- <color_borde>: un color de CSS o la palabra clave transparent
- <estilo_borde>: una de las siguientes palabras clave: none, hidden, dotted, dashed, solid,double, groove, ridge, inset, outset.

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

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

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

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

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

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

Las siguientes reglas CSS son equivalentes:

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

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

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

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

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

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

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

- **4.5. Fondos**

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

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

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

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

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

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

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

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

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

Reglas CSS

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

Resultado

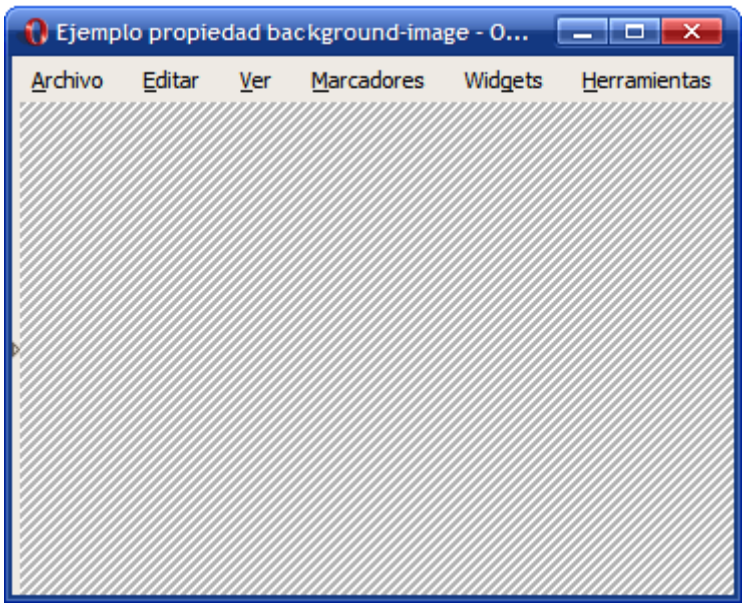


Figura 4.19 Página con una imagen de fondo

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

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

Propiedad	**background-repeat**
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

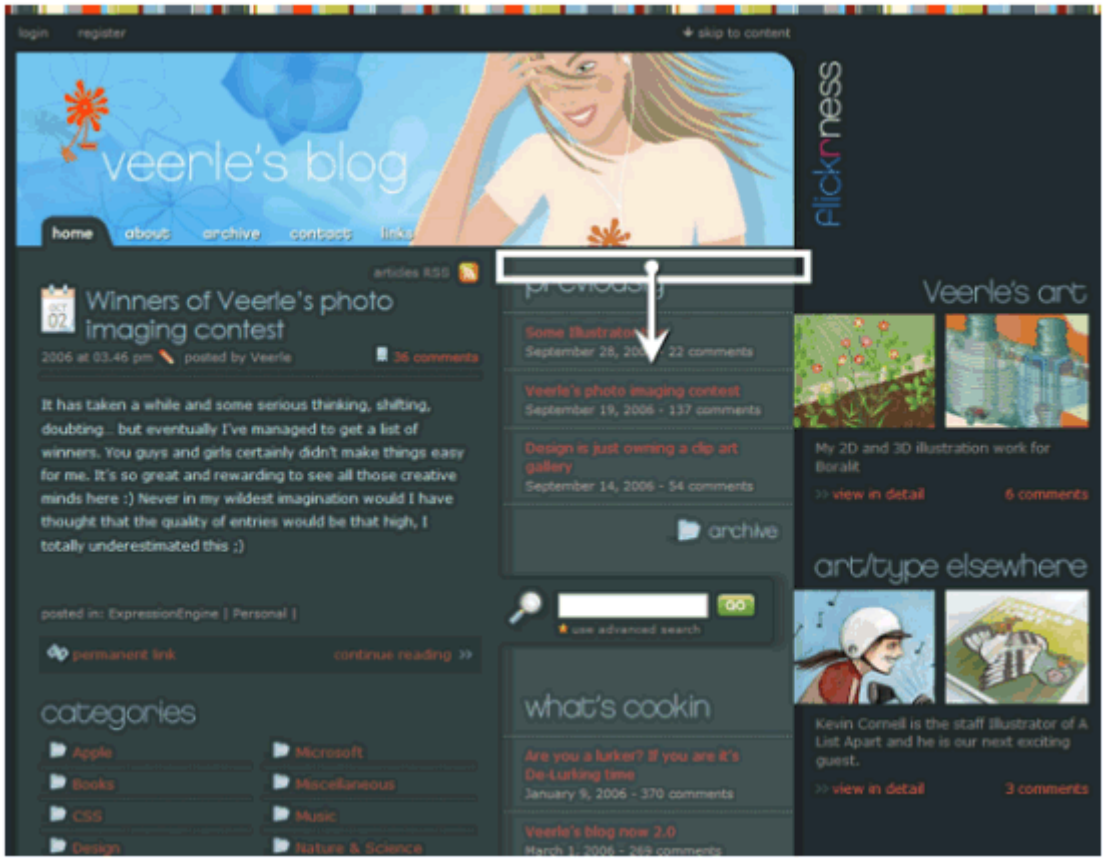
El valor repeat indica que la imagen se debe repetir en todas direcciones y por tanto, es el comportamiento por defecto. El valor no-repeat muestra una sola vez la imagen y no se repite en ninguna dirección. El valor repeat-x repite la imagen sólo horizontalmente y el valor repeat-y repite la imagen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

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:



Las reglas CSS definidas para esa columna de contenidos son:

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

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

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

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

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

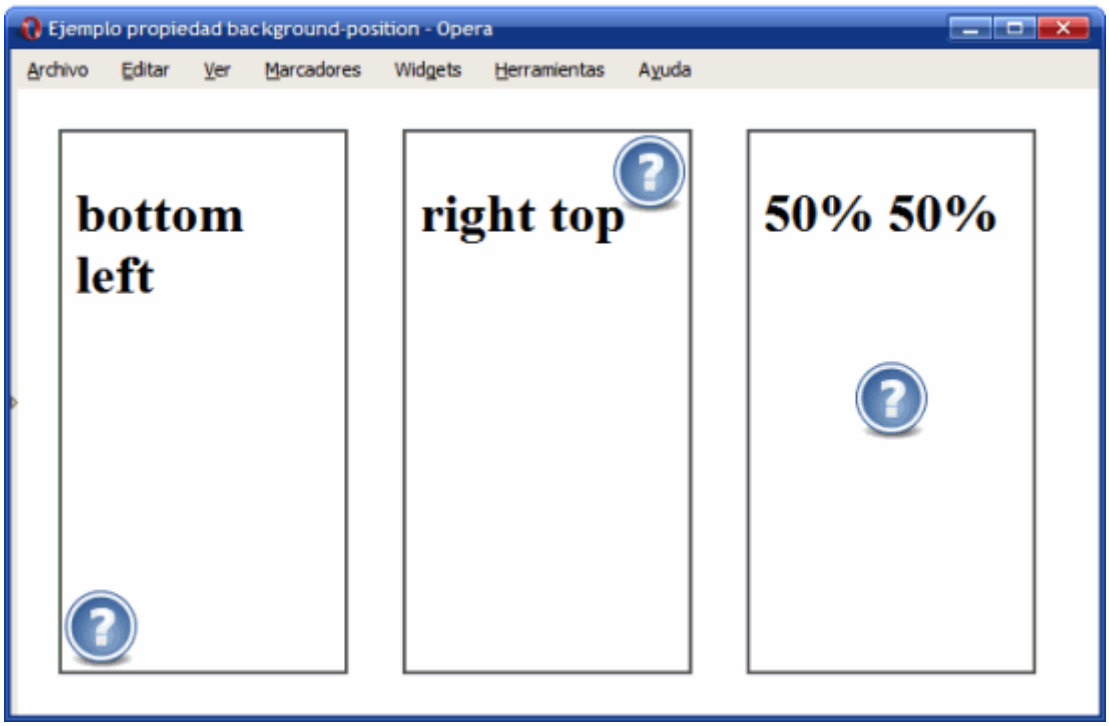
Cuando se utilizan porcentajes, su interpretación no es intuitiva. Si el valor de la propiedadbackground-position se indica mediante dos porcentajes x% y%, el navegador coloca el punto (x%, y%) de la imagen de fondo en el punto (x%, y%) del elemento.

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

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

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

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



Las reglas CSS del ejemplo anterior se muestran a continuación:

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

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

<div id="caja1"><h1>bottom left</h1></div>
<div id="caja2"><h1>right top</h1></div>
<div id="caja3"><h1>50% 50%</h1></div>
```

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

Propiedad	**background-attachment**
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.

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

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

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

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

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

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

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

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

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

background: none;

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

Capítulo 5. Posicionamiento y visualización

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

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

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

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

- **5.1. Tipos de elementos**

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

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

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

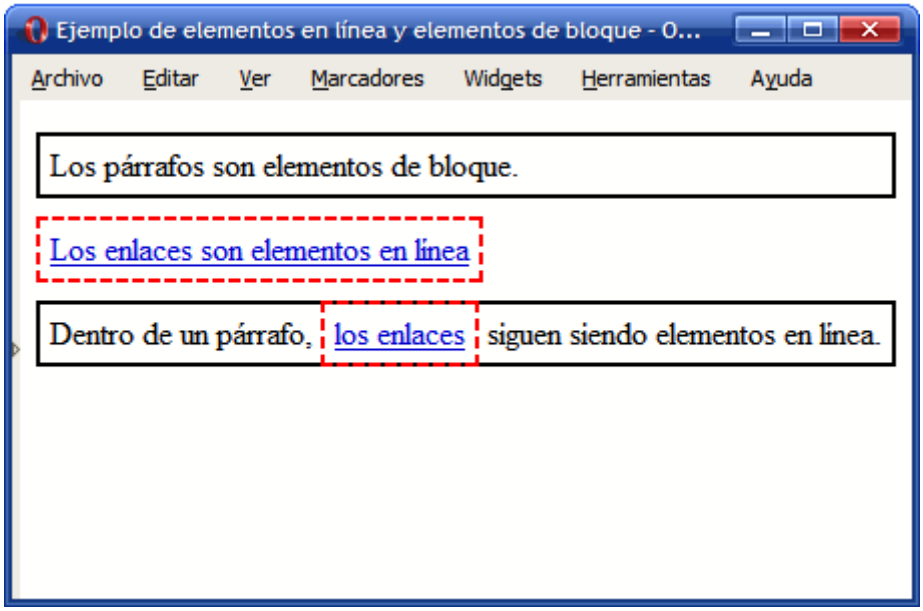


Figura 5.1 Cajas creadas por los elementos de línea y los elementos de bloque

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

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

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

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

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

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

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

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

- **5.2. Posicionamiento**

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

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

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

- Posicionamiento normal o estático: se trata del posicionamiento que utilizan los navegadores si no se indica lo contrario.

- Posicionamiento relativo: variante del posicionamiento normal que consiste en posicionar una caja según el posicionamiento normal y después desplazarla respecto de su posición original.

- Posicionamiento absoluto: la posición de una caja se establece de forma absoluta respecto de su elemento contenedor y el resto de elementos de la página ignoran la nueva posición del elemento.

- Posicionamiento fijo: variante del posicionamiento absoluto que convierte una caja en un elemento inamovible, de forma que su posición en la pantalla siempre es la misma independientemente del resto de elementos e independientemente de si el usuario sube o baja la página en la ventana del navegador.

- Posicionamiento flotante: se trata del modelo más especial de posicionamiento, ya que desplaza las cajas todo lo posible hacia la izquierda o hacia la derecha de la línea en la que se encuentran.

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

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

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

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

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

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

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

Propiedades	**top, right, bottom, left**
Valores	unidad de medida | porcentaje | auto | inherit
Se aplica a	Todos los elementos posicionados
Valor inicial	auto
Descripción	Indican el desplazamiento horizontal y vertical del elemento respecto de su posición original

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

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

- **5.3. Posicionamiento normal**

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

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

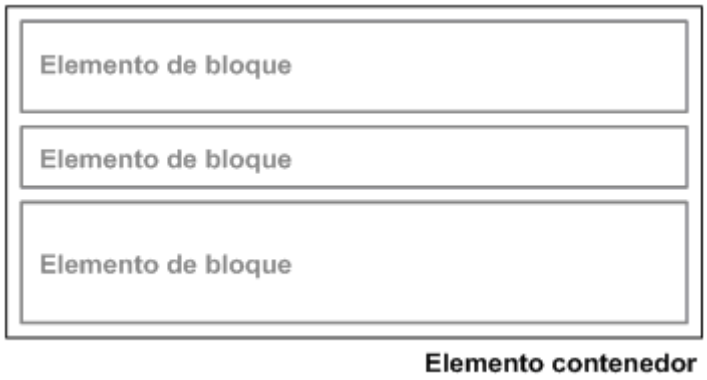


Figura 5.2 Posicionamiento normal de los elementos de bloque

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

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

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

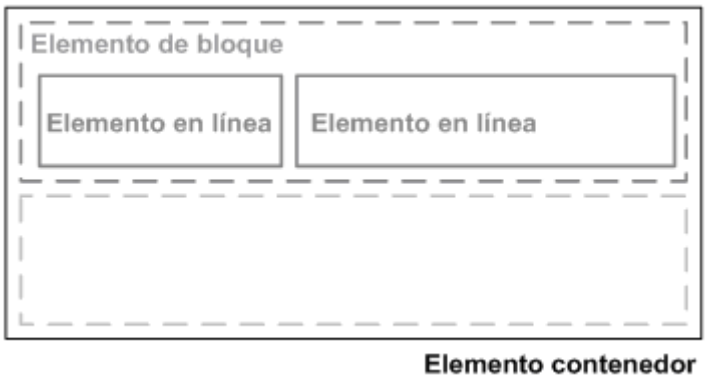


Figura 5.3 Posicionamiento normal de los elementos en línea

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

- **5.4. Posicionamiento relativo**

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

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

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

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

Por tanto, la propiedad top se emplea para mover las cajas de forma descendente, la propiedadbottom mueve las cajas de forma ascendente, la propiedad left se utiliza para desplazar las cajas hacia la derecha y la propiedad right mueve las cajas hacia la izquierda. Este comportamiento parece poco intuitivo y es causa de errores cuando se empiezan a diseñar páginas con CSS. Si se utilizan valores negativos en las propiedades top, right, bottom y left, su efecto es justamente el inverso.

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

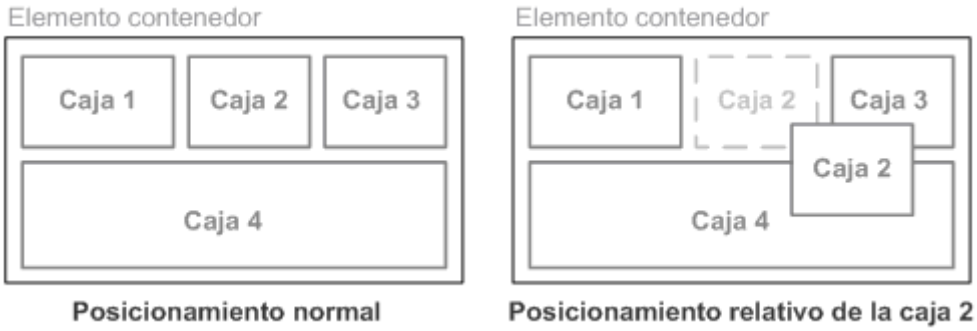


Figura 5.4 Ejemplo de posicionamiento relativo de un elemento

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

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

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

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

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

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

El siguiente ejemplo muestra tres imágenes posicionadas de forma normal:

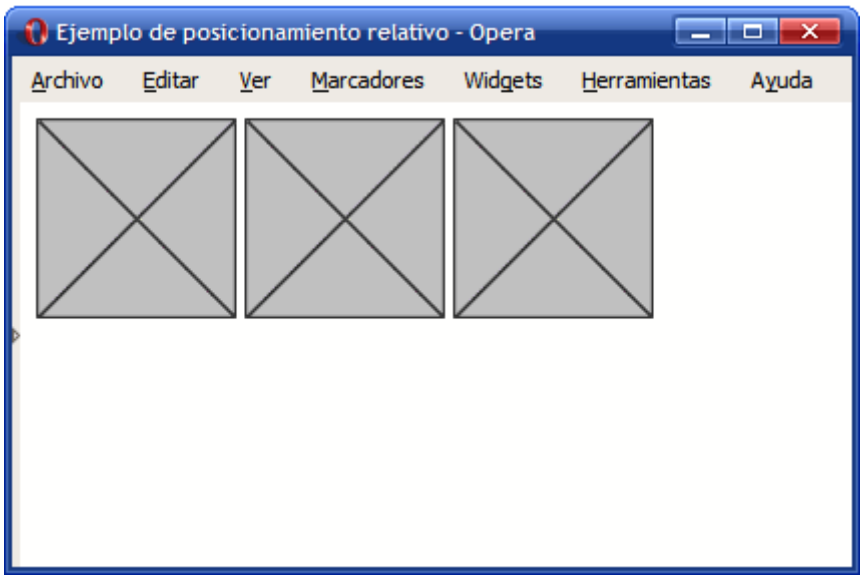


Figura 5.5 Elementos posicionados de forma normal

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

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

<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:

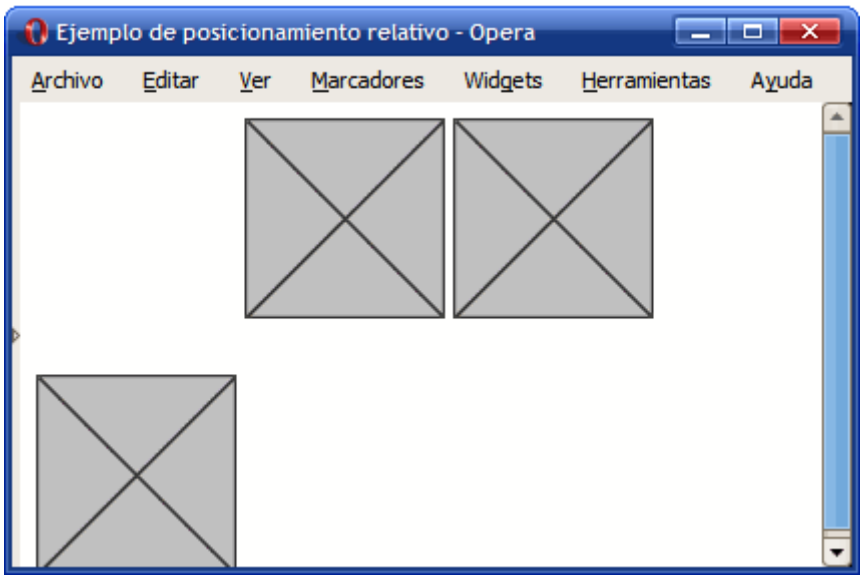


Figura 5.6 Elemento posicionado de forma relativa

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

- **5.5. Posicionamiento absoluto**

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

Cuando una caja se posiciona de forma absoluta, el resto de elementos de la página se ven afectados y modifican su posición. Al igual que en el posicionamiento relativo, cuando se posiciona de forma absoluta una caja es probable que se produzcan solapamientos con otras cajas.

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

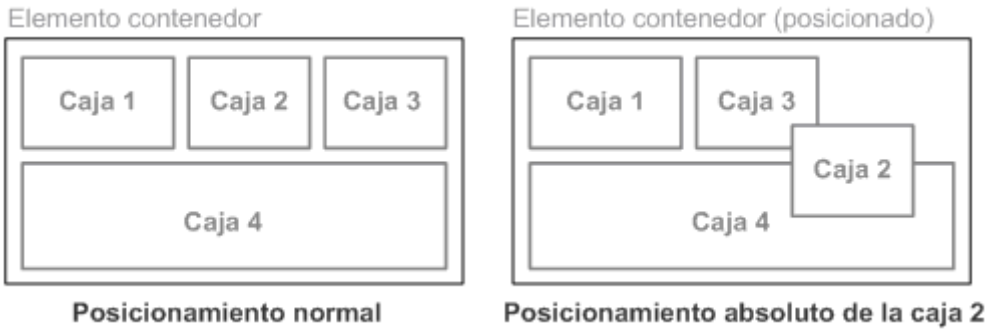


Figura 5.7 Ejemplo de posicionamiento absoluto de un elemento

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

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

Por otra parte, el desplazamiento de una caja posicionada de forma absoluta se controla mediante las propiedades top, right, bottom y left. A diferencia del posicionamiento relativo, la interpretación de los valores de estas propiedades depende del elemento contenedor de la caja posicionada.

Determinar la referencia utilizada para interpretar los valores de top, right, bottom y left de una caja posicionada de forma absoluta es un proceso complejo que se compone de los siguientes pasos:

- Se buscan todos los elementos contenedores de la caja hasta llegar al elemento <body> de la página.
- Se recorren todos los elementos contenedores empezando por el más cercano a la caja y llegando hasta el <body>
- El primer elemento contenedor que esté posicionado de cualquier forma diferente a position: static se convierte en la referencia que determina la posición de la caja posicionada de forma absoluta.
- Si ningún elemento contenedor está posicionado, la referencia es la ventana del navegador, que no debe confundirse con el elemento <body> de la página.

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

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

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

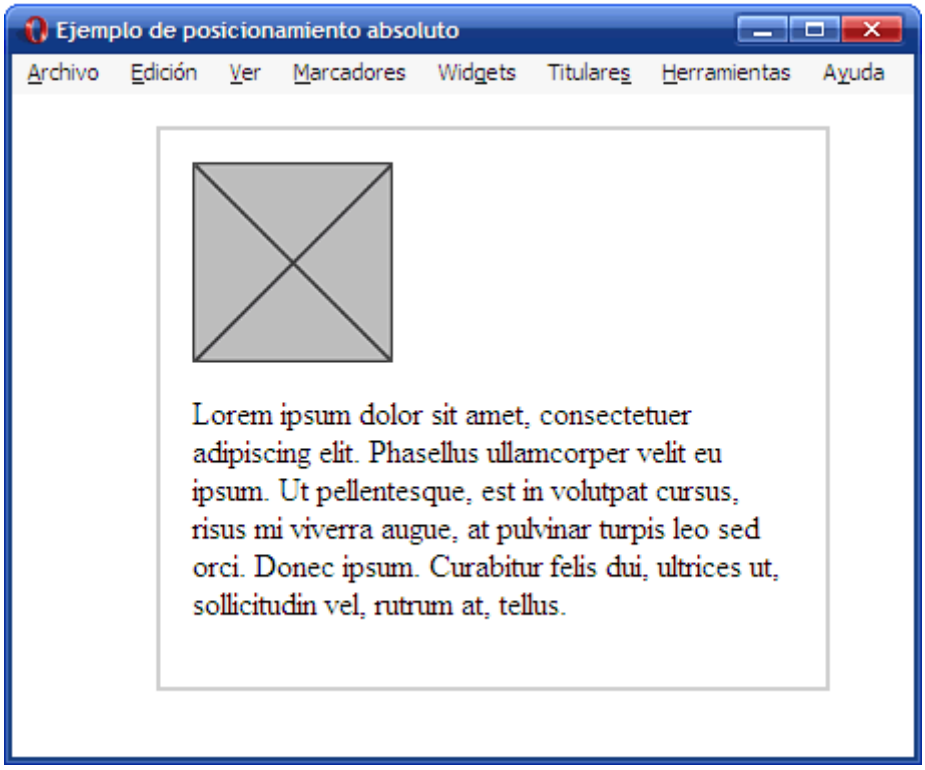


Figura 5.8 Situación original antes de modificar el posicionamiento

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

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

<div>
        <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>
```

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

```
div img {
        position: absolute;
        top:      50px;
        left: 50px;
}
```

El resultado visual se muestra en la siguiente imagen:

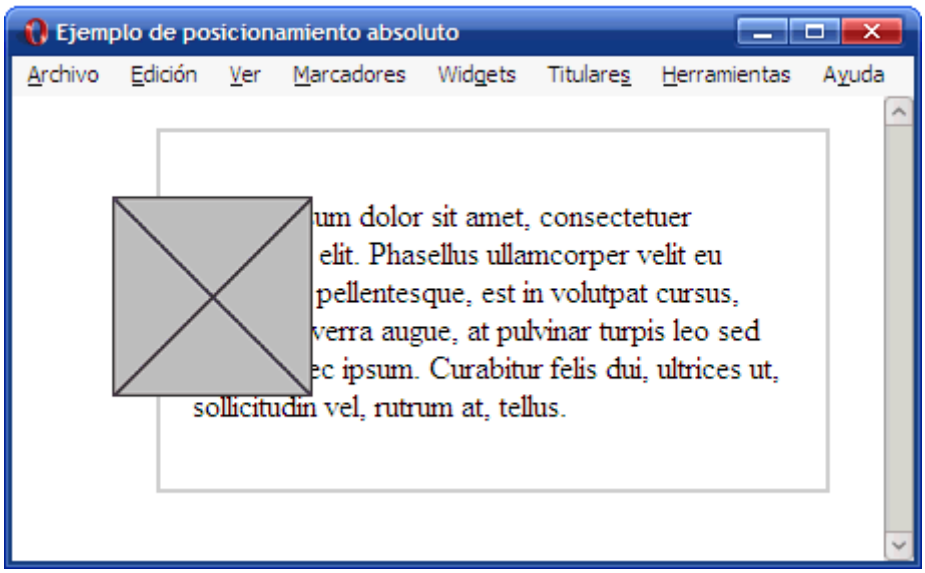


Figura 5.9 Imagen posicionada de forma absoluta

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

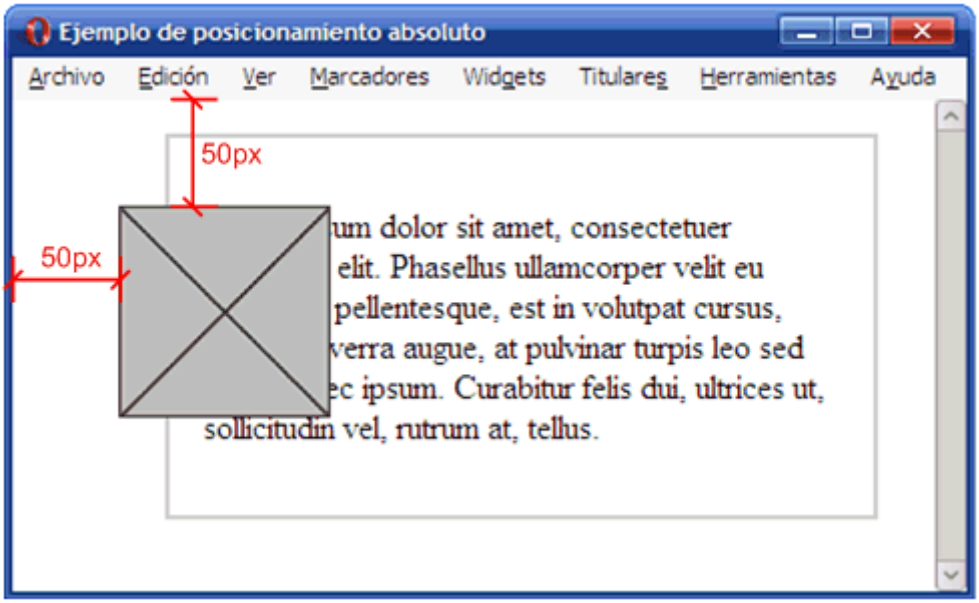


Figura 5.10 La referencia del posicionamiento absoluto es la ventana del navegador

Para posicionar la imagen de forma absoluta, el navegador realiza los siguientes pasos:

1. Obtiene la lista de elementos contenedores de la imagen: y <body>.

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

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

A continuación, se modifica el ejemplo anterior posicionando de forma relativa el elemento que contiene la imagen y el párrafo. La única propiedad añadida al es position: relative por lo que el elemento contenedor se posiciona pero no se desplaza respecto de su posición original:

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

```
div img {
        position: absolute;
        top: 50px;
        left: 50px;
}
```

La siguiente imagen muestra el resultado obtenido:

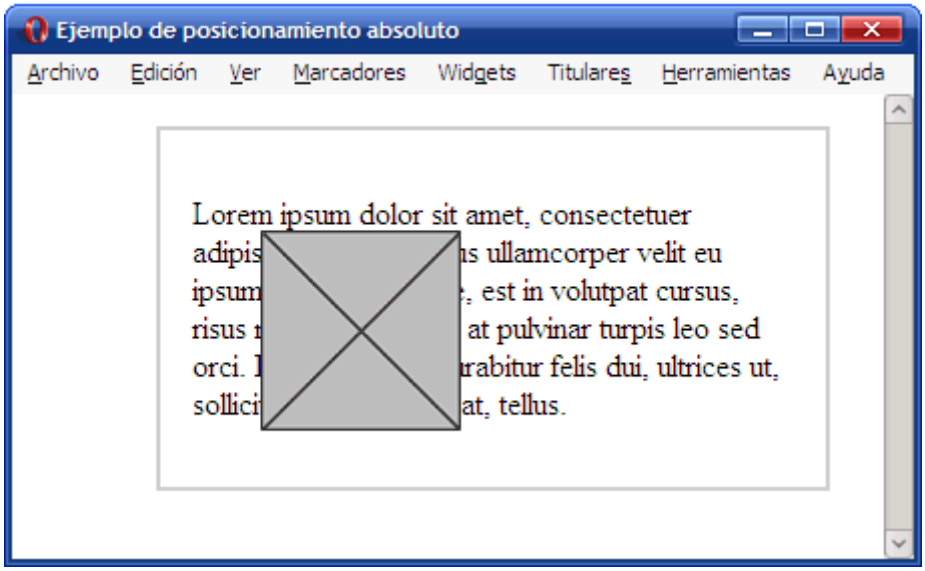


Figura 5.11 Imagen posicionada de forma absoluta

En este caso, como el elemento contenedor de la imagen está posicionado, se convierte en la referencia para el posicionamiento absoluto. El resultado es que la posición de la imagen es muy diferente a la del ejemplo anterior:

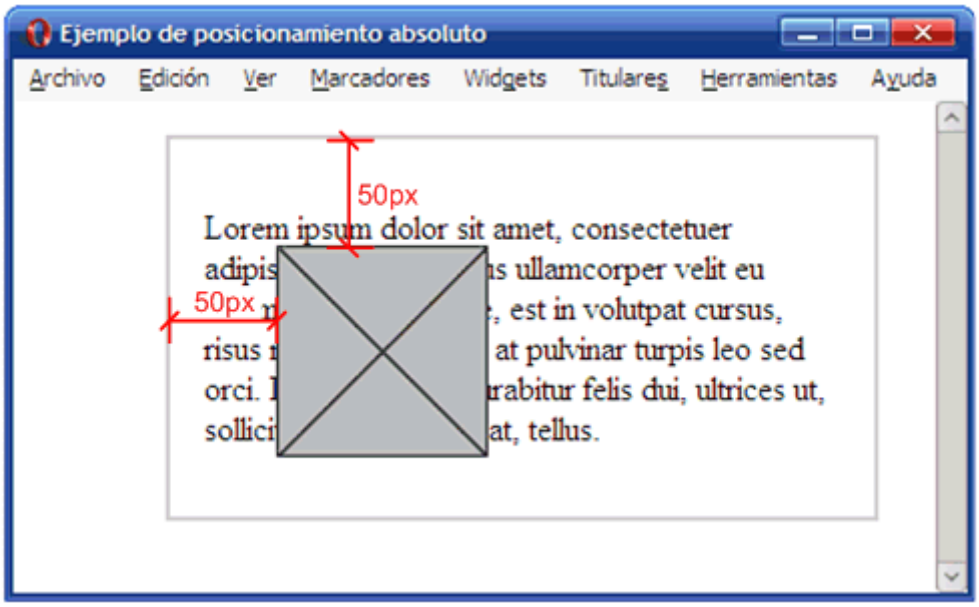


Figura 5.12 La referencia del posicionamiento absoluto es el elemento contenedor de la imagen

Por tanto, si se quiere posicionar un elemento de forma absoluta respecto de su elemento contenedor, es imprescindible posicionar este último. Para ello, sólo es necesario añadir la propiedad position: relative, por lo que no es obligatorio desplazar el elemento contenedor respecto de su posición original.

- **5.6. Posicionamiento fijo**

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

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

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

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

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

- **5.7. Posicionamiento flotante**

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

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

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

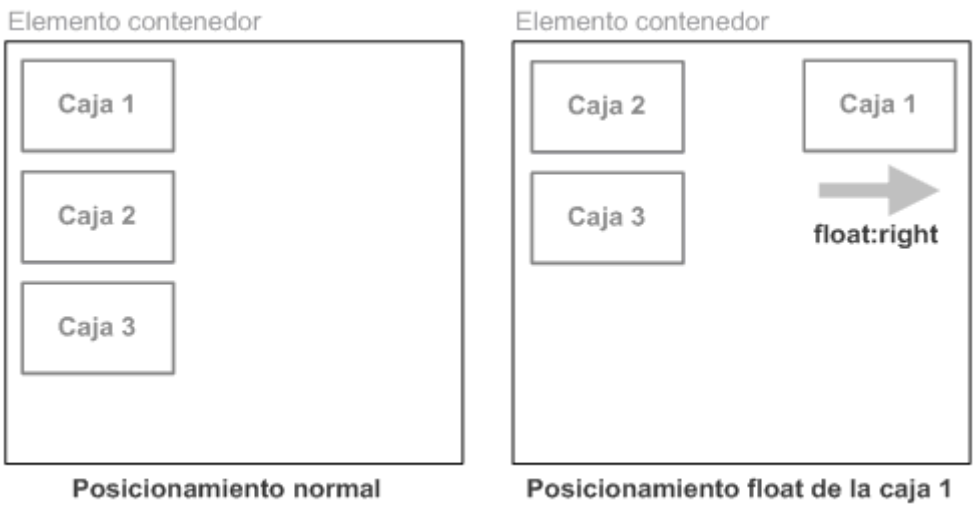


Figura 5.13 Ejemplo de posicionamiento float de una caja

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

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

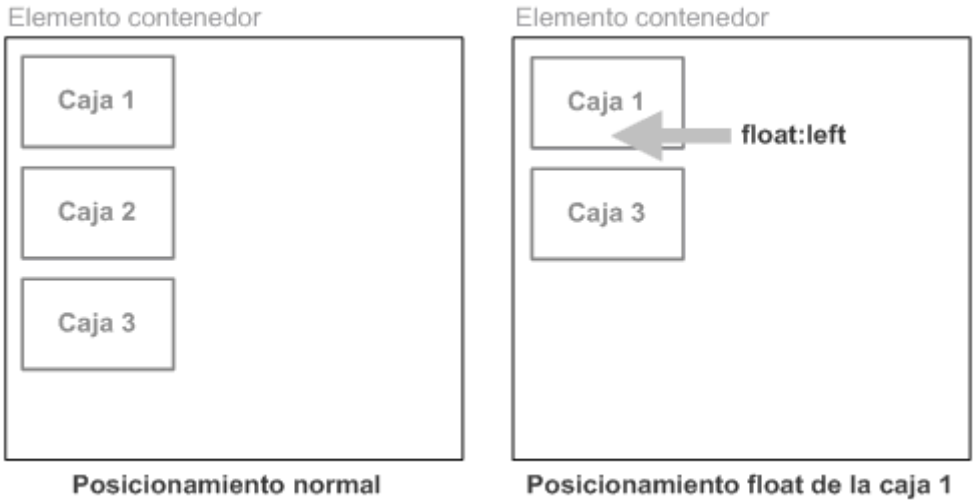


Figura 5.14 Ejemplo de posicionamiento float de una caja

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

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

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

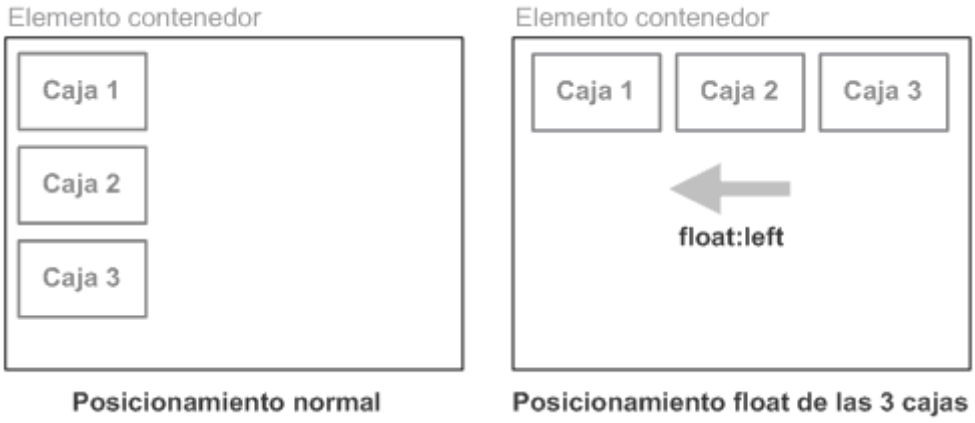


Figura 5.15 Ejemplo de posicionamiento float de varias cajas

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

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

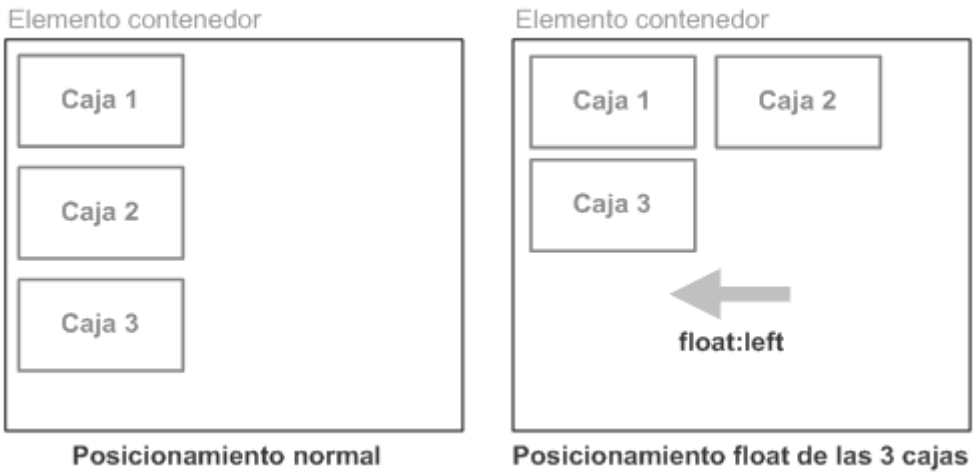


Figura 5.16 Ejemplo de posicionamiento float cuando no existe sitio suficiente

Las cajas flotantes influyen en la disposición de todas las demás cajas. Los elementos en línea*hacen sitio* a las cajas flotantes adaptando su anchura al espacio libre dejado por la caja desplazada. Los elementos de bloque no les hacen sitio, pero sí que adaptan sus contenidos para que no se solapen con las cajas flotantes.

La propiedad CSS que permite posicionar de forma flotante una caja se denomina float:

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

Si se indica un valor left, la caja se desplaza hasta el punto más a la izquierda posible en esa misma línea (si no existe sitio en esa línea, la caja baja una línea y se muestra lo más a la izquierda posible en esa nueva línea). El resto de elementos adyacentes se adaptan y *fluyen* alrededor de la caja flotante.

El valor right tiene un funcionamiento idéntico, salvo que en este caso, la caja se desplaza hacia la derecha. El valor none permite anular el posicionamiento flotante de forma que el elemento se muestre en su posición original.

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

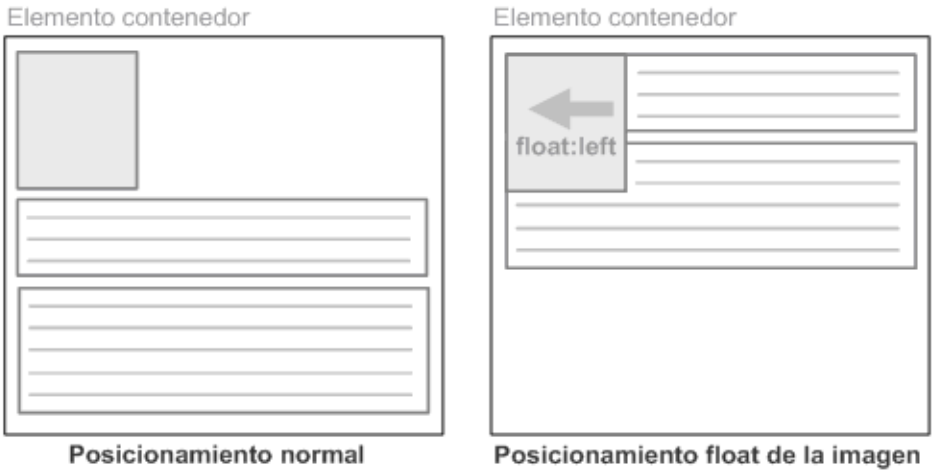


Figura 5.17 Elementos que fluyen alrededor de un elemento posicionado mediante float

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

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

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

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

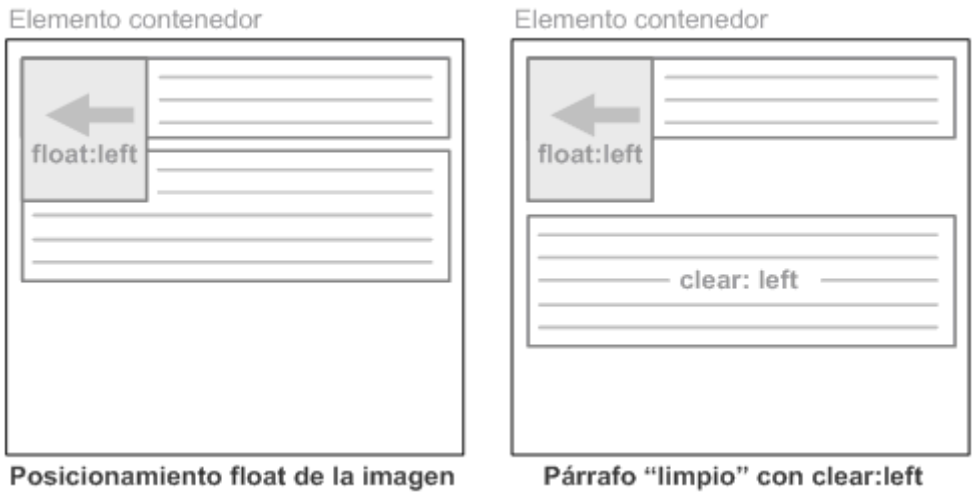


Figura 5.18 Forzando a que un elemento no fluya alrededor
de otro elemento posicionado mediante float

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

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

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

Propiedad	**clear**
Valores	none | left | right | both | inherit
Se aplica a	Todos los elementos de bloque
Valor inicial	none
Descripción	Indica el lado del elemento que no debe ser adyacente a ninguna caja flotante

La propiedad clear indica el lado del elemento HTML que no debe ser adyacente a ninguna caja posicionada de forma flotante. Si se indica el valor left, el elemento se desplaza de forma descendente hasta que pueda colocarse en una línea en la que no haya ninguna caja flotante en el lado izquierdo.

La especificación oficial de CSS explica este comportamiento como *"un desplazamiento descendente hasta que el borde superior del elemento esté por debajo del borde inferior de cualquier elemento flotante hacia la izquierda"*.

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

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

Como se verá más adelante, la propiedad clear es imprescindible cuando se crean las estructuras de las páginas web complejas.

Si se considera el siguiente código CSS y HTML:

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

.derecha   { float: right; }
.izquierda { float: left;        }
<div id="paginacion">
        <span class="izquierda">&laquo; Anterior</span>
        <span class="derecha">Siguiente &raquo;</span>
</div>
```

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

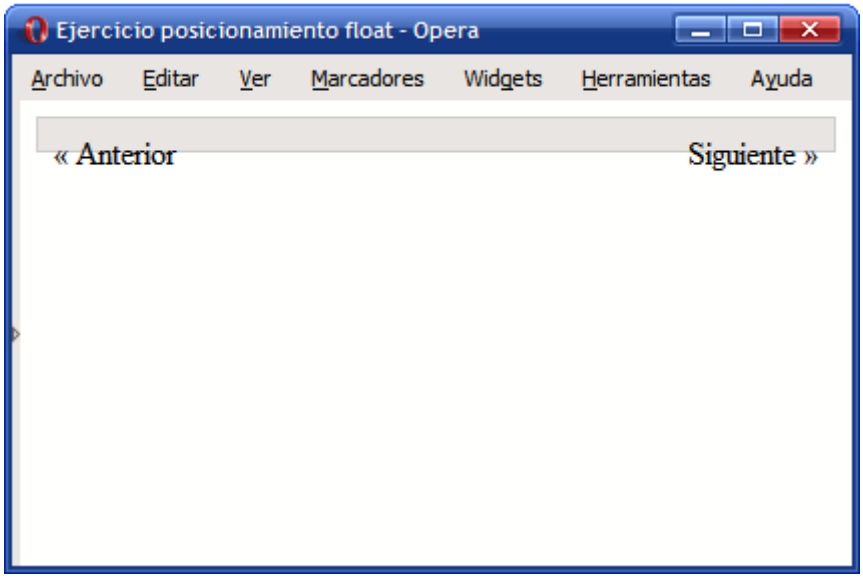


Figura 5.19 Visualización incorrecta de dos elementos posicionados mediante float

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

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

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

.derecha { float: right; }
.izquierda { float: left; }
```

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

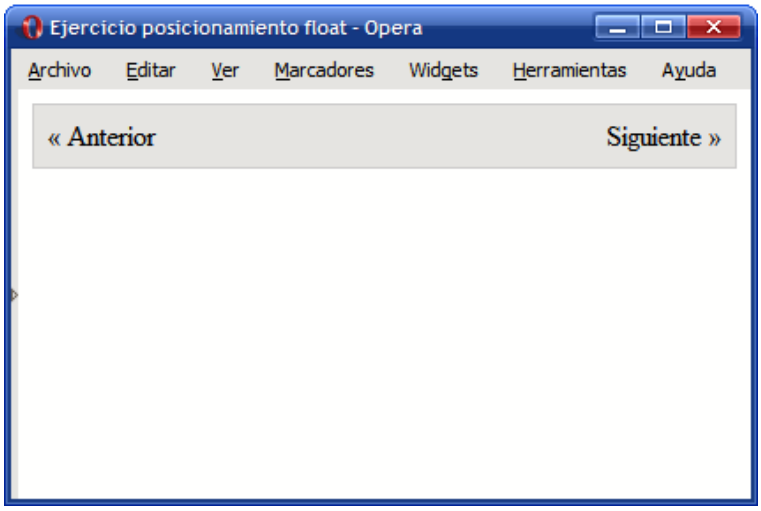


Figura 5.20 Visualización correcta de dos elementos posicionados mediante float

- **5.8. Visualización**

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

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

- **5.8.1. Propiedades display y visibility**

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

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

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

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

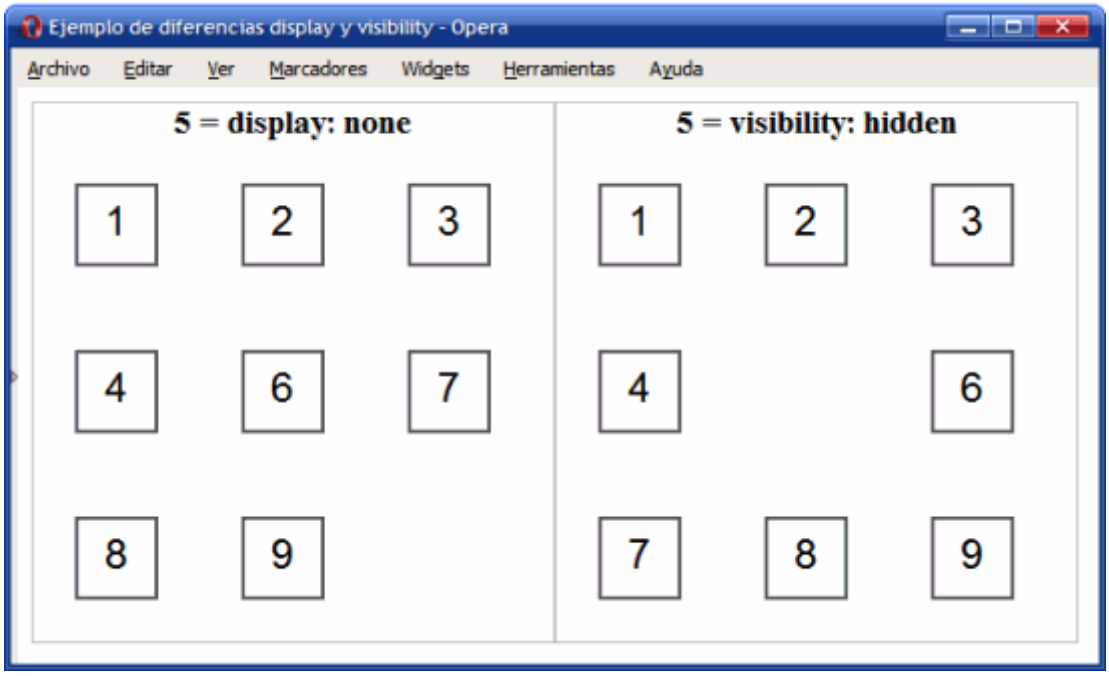


Figura 5.21 Diferencias visuales entre las propiedades display y visibility

En general, cuando se oculta un elemento no es deseable que siga ocupando sitio en la página, por lo que la propiedad display se utiliza mucho más que la propiedad visibility.

A continuación se muestra la definición completa de la propiedad display:

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

Las posibilidades de la propiedad display son mucho más avanzadas que simplemente ocultar elementos. En realidad, la propiedad display modifica la forma en la que se visualiza un elemento.

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

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

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

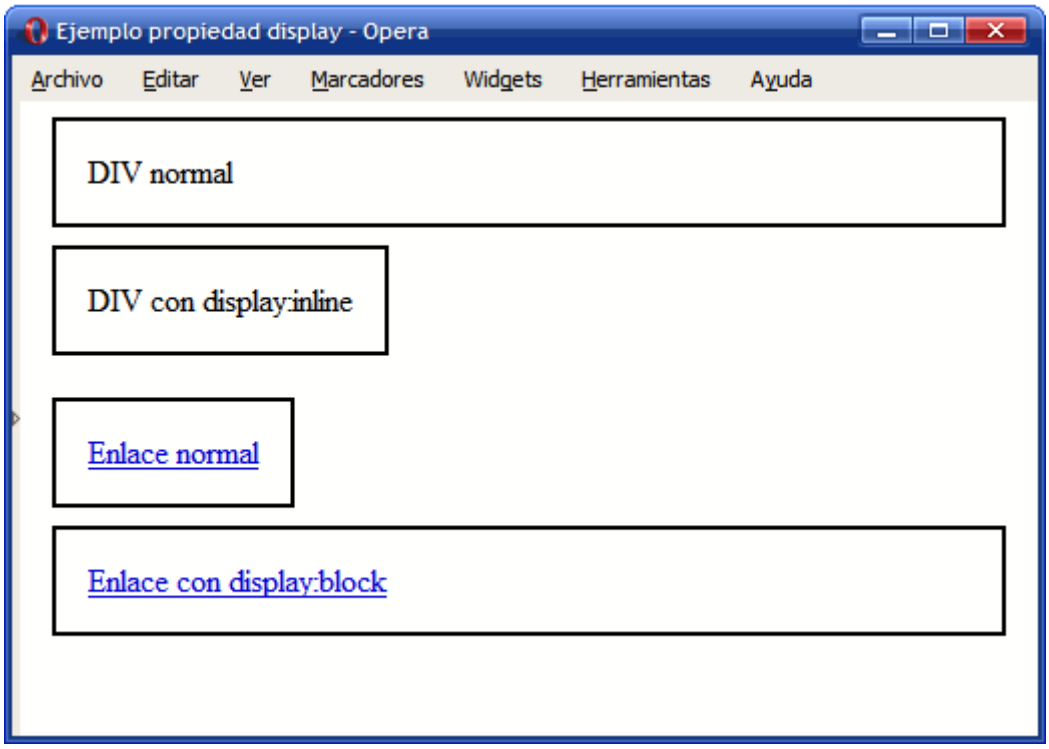


Figura 5.22 Ejemplo de propiedad display

Las reglas CSS del ejemplo anterior son las siguientes:

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

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


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

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

Como se verá más adelante, la propiedad display: inline se puede utilizar en las listas (<ul>,<ol>) que se quieren mostrar horizontalmente y la propiedad display: block se emplea frecuentemente para los enlaces que forman el menú de navegación.

Por su parte, la definición completa de la propiedad visibility es mucho más sencilla:

Propiedad	**visibility**
Valores	visible | hidden | collapse | inherit
Se aplica a	Todos los elementos
Valor inicial	visible
Descripción	Permite hacer visibles e invisibles a los elementos

Las posibilidades de la propiedad visibility son mucho más limitadas que las de la propiedaddisplay, ya que sólo permite hacer visibles o invisibles a los elementos de la página.

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

Por último, el valor collapse de la propiedad visibility sólo se puede utilizar en las filas, grupos de filas, columnas y grupos de columnas de una tabla. Su efecto es similar al de la propiedad display, ya que oculta completamente la fila y/o columna y se pueden mostrar otros contenidos en ese lugar. Si se utiliza el valor collapse sobre cualquier otro tipo de elemento, su efecto es idéntico al valorhidden.

- **5.8.2. Relación entre display, float y position**

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

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

2. Si position vale absolute o fixed, la caja se posiciona de forma absoluta, se considera quefloat vale none y la propiedad display vale block tanto para los elementos en línea como para los elementos de bloque. La posición de la caja se determina mediante el valor de las propiedades top, right, bottom y left.

3. En cualquier otro caso, si float tiene un valor distinto de none, la caja se posiciona de forma flotante y la propiedad display vale block tanto para los elementos en línea como para los elementos de bloque.

- **5.8.3. Propiedad overflow**

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

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

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

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

Los valores de la propiedad overflow tienen el siguiente significado:

- visible: el contenido no se corta y se muestra sobresaliendo la zona reservada para visualizar el elemento. Este es el comportamiento por defecto.

- hidden: el contenido sobrante se oculta y sólo se visualiza la parte del contenido que cabe dentro de la zona reservada para el elemento.

- scroll: solamente se visualiza el contenido que cabe dentro de la zona reservada para el elemento, pero también se muestran barras de *scroll* que permiten visualizar el resto del contenido.

- auto: el comportamiento depende del navegador, aunque normalmente es el mismo que la propiedad scroll.

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

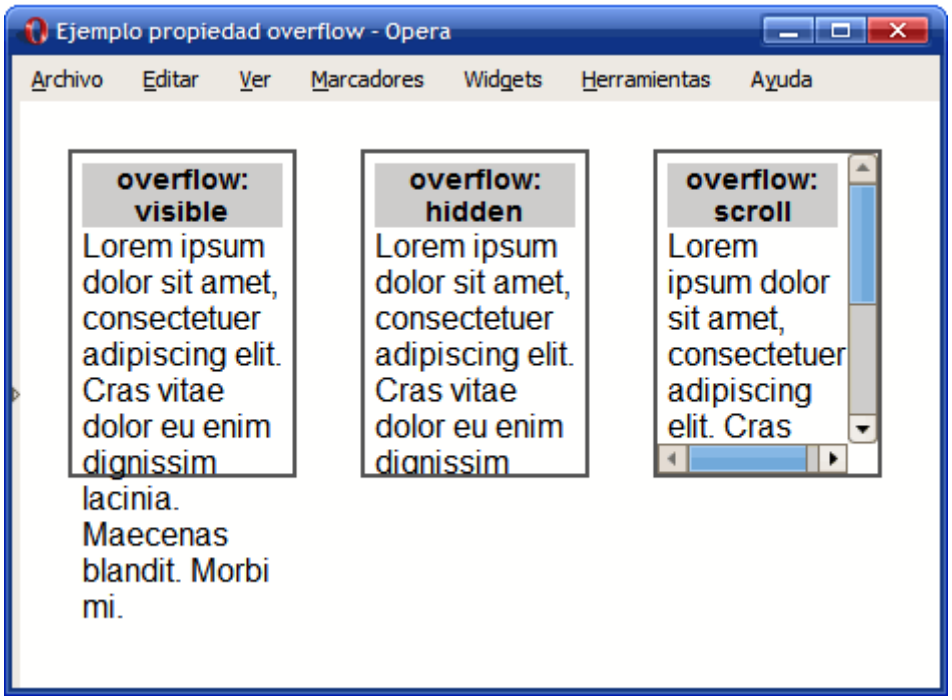


Figura 5.23 Ejemplo de propiedad overflow

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

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

```
<div><h1>overflow: visible</h1> Lorem ipsum dolor sit amet, 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>
```

- **5.8.4. Propiedad z-index**

Además de posicionar una caja de forma horizontal y vertical, CSS permite controlar la posición tridimensional de las cajas posicionadas. De esta forma, es posible indicar las cajas que se muestran delante o detrás de otras cajas cuando se producen solapamientos.

La posición tridimensional de un elemento se establece sobre un tercer eje llamado Z y se controla mediante la propiedad z-index. Utilizando esta propiedad es posible crear páginas complejas con varios niveles o capas.

A continuación se muestra la definición formal de la propiedad z-index:

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

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

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

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

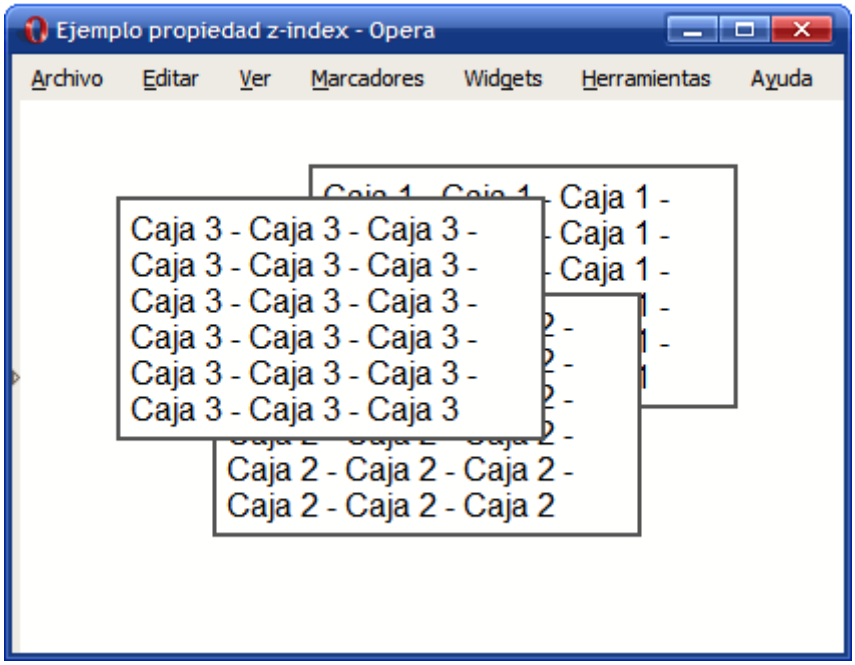


Figura 5.24 Ejemplo de propiedad z-index

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

```
div { position: absolute; }

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

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

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

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

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

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

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

FIN

- Diseño de estilos para diferentes dispositivos.
- Buenas prácticas en el uso de hojas de estilo.

INICIO

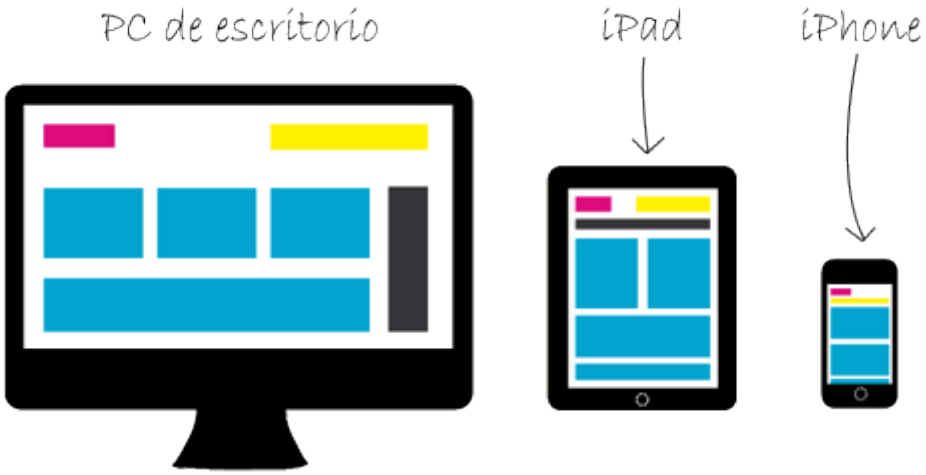
- **Media Queries en CSS3**

El concepto de media queries lo podemos definir en español como “consulta a medios”, y eso es lo que realmente trabajamos con esta funcionalidad de css3. Hoy en día vemos la gran evolución de la comunicación, en especial la web, últimamente se ha incrementado de una manera considerable la manera en que personas acceden a la web, hoy se puede acceder desde televisores, computadores, celulares, tabletas e incluso consolas de videojuegos. Lo que se convierte en un reto para el diseñador o desarrollador web, hacer que la web se visualice bien en los diferentes dispositivos y diferentes resoluciones de pantalla, desde un Smartphone hasta un televisor de gran resolución.

Esto lo podemos resolver gracias a los media queries, que nos permiten personalizar los estilos basándonos en las características del dispositivo del visitante de nuestra web.

Soporte de media queries en los diferentes navegadores.

Visualización de un diseño web utilizando media queries

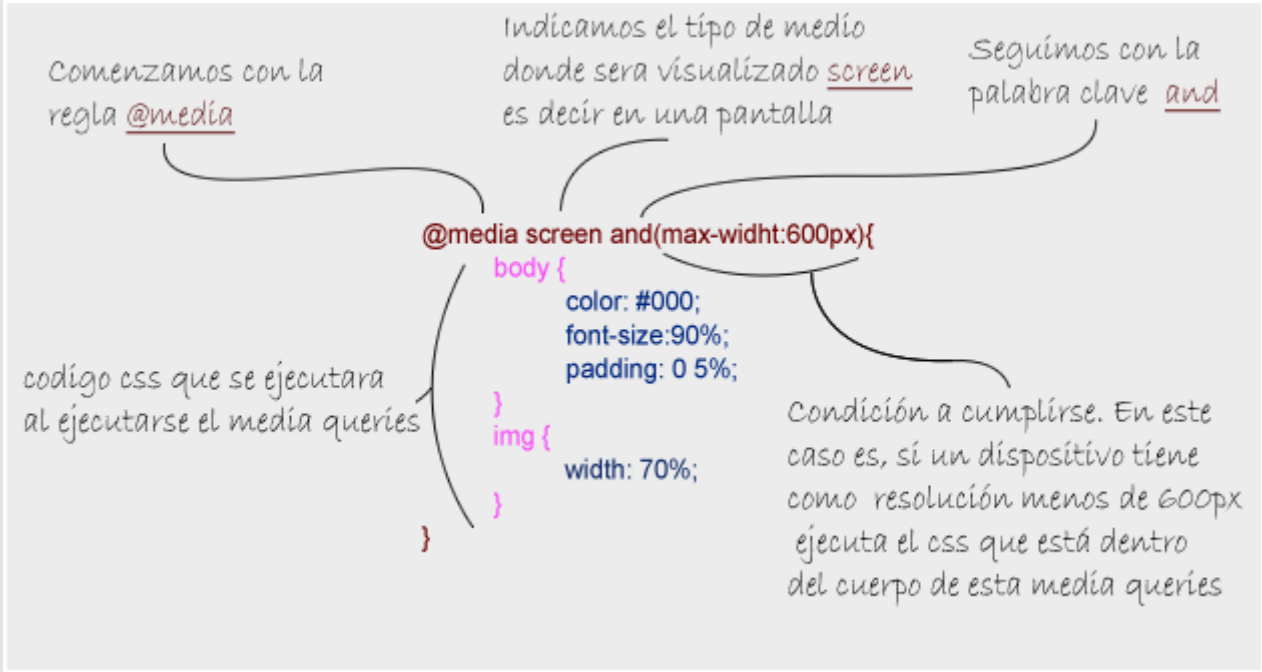


Pero veamos cómo podemos añadir esos estilos condicionales y la consulta queries que si se da la condición desatara dichos estilos.

Una de las formas para hacer esto es añadiendo la regla dentro de nuestro css principal o también creando una hoja independiente y luego adjuntarla. Veamos la regla condicional que en caso de cumplirse ejecutara ciertos estilos css.

```
@media screen and(max-widht:600px){
	body {
		color: #000;
		font-size:90%;
		padding: 0 5%;
	}
}
```

Analicemos la sintaxis de la ccs media querie.



Lo que en este caso debemos de tener presente es la función o condicional que al cumplirse desatara y ejecuta el css preparado para el dispositivo en cuestión.

En el caso anterior era max-widht:600px, es decir cualquier persona que habrá la web desde un dispositvo que la resolución sea menor de 600px automáticamente ejecutara el css que está en su cuerpo. Ahora, hay muchas más opciones de condicionales que funcionan con media queries, y es lo que denominamos los **Media features**, miremos cuales podemos utilizar para ejecutar css condicionales.

1 – width

```
@media screen and (min-width: 400px) and (max-width: 700px) { … }
```

media queries que funcionara si la web se abre en un dispositivo entre los rangos de 400 y 700 px. max-widht es el acho maximo y min-width el ancho minimo.

2 – height

```
@media screen and (height: 700px) { … }
```

Con esta función especificamos la altura de visualización del dispositivo, acepta los prefijos min/max.

3 – device-width

```
@media screen and (device-width: 800px) { … }
```

Especificamos el ancho del área de representación, la anchura de la hoja web, acepta los prefijos min/max.

4 – device-height

```
@media screen and (device-height:600px) { … }
```

Especificamos la altura del área de representación, la altura de la hoja web, acepta los prefijos min/max.

5 – orientation

```
@media all and (orientation:portrait) { … }
@media all and (orientation:landscape) { … }
```

Especificamos la Orientación del dispositivo, podemos tener como valores **portrait** o **landscape**. La orientación es **portrait** cuando el media feature width es igual o mayor a la media feature height, de lo contrario la orientación es **landscape.**

6 – aspect-ratio

```
@media screen and (aspect-ratio: 1280/720)
```

Especificamos la relación del dispositivo entre la media feature width y la media feature height, acepta los prefijos min/max.

7 – device-aspect-ratio

```
@media screen and (device-aspect-ratio: 1280/720) { … }
```

Especificamos la relación del dispositivo entre la media feature device-width y la media feature device-height, acepta los prefijos min/max.

8 – color

```
@media all and (min-color: 2) { … }
```

especificamos el numero de bits por componentes del color,

La anterior media queries aplica para los dispositivos con 2 o más bits como componentes de color, acepta los prefijos min/max.

9 – color-index

```
@media all and (min-color-index: 1) { … }
```

Con esto especificamos el número de entradas en la tabla de conversión de color del dispositivo de salida, si el dispositivo no utiliza una tabla de conversión de color, su valor será cero, acepta los prefijos min/max.

10 – monochrome

```
@media all and (min-monochrome: 2) { … }
```

Básicamente con esta función especificamos que el dispositivo es a blanco y negro, y lo especificamos por el número de bits por pixel, acepta los prefijos min/max.

```
@media all and (monochrome) { … }
@media all and (min-monochrome: 1) { … }
```

En el anterior código especificamos 2 maneras de aplicar los css especiales por media queries a todos los dispositivos monochrome es decir a blanco y negro.

11 – resolution

```
@media print and (min-resolution: 300dpi) { … }
```

Especificamos la densidad de pixeles de un dispositivo de salida, acepta los prefijos min/max.

Por ejemplo en el código anterior especificamos los css para los dispositivos con una resolución mínima de 300 ppp “puntos por pulgadas”

12 – scan

```
@media tv and (scan: progressive) { … }
/* o */
@media tv and (scan: interlace) { … }
```

Especificamos los css para los procesos de escaneados utilizados por los dispositivos de salida de televisión, los valores pueden ser progressive o interlace.

———————-

En resumen.

En css 2.1 utilizamos los media types screen y print para aplicar estilos diferentes a la web, ahora en CSS3 tenemos la opción de Media queries, que es mucho más abundante en funcionalidades, ya que nos va a permitir crear estilos css alternos para los diferentes dispositivos que pueden conectarse a internet y visualizar una web, ya no hay motivos para no hacer que nuestra web se vea bien en todas las pantallas.

Para más información sobre este tema: http://www.w3.org/TR/css3-mediaqueries/

Capítulo 2. Buenas prácticas

- **Inicializar los estilos**

Cuando los navegadores muestran una página web, además de aplicar las hojas de estilo de los diseñadores, siempre aplican otras dos hojas de estilos: la del navegador y la del usuario.

La hoja de estilos del navegador se utiliza para establecer el estilo inicial por defecto a todos los elementos HTML: tamaños de letra, decoración del texto, márgenes, etc. Esta hoja de estilos siempre se aplica a todas las páginas web, por lo que cuando una página no incluye ninguna hoja de estilos propia, el aspecto con el que se muestra en el navegador se debe a esta hoja de estilos del navegador.

Por su parte, la hoja de estilos del usuario es la que puede aplicar el usuario mediante su navegador. Aunque la inmensa mayoría de usuarios no utiliza esta característica, en teoría es posible que los usuarios establezcan el tipo de letra, color y tamaño de los textos y cualquier otra propiedad CSS de los elementos de la página que muestra el navegador.

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

Figura 2.1 Orden en el que se aplican las diferentes hojas de estilos

Por tanto, las reglas que menos prioridad tienen son las del CSS 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.

NOTA CSS define la palabra reservada !important para controlar la prioridad de las declaraciones de las diferentes hojas de estilos. Las reglas CSS que incluyen la palabra!important tienen prioridad sobre el resto de las reglas CSS, independientemente del orden en el que se incluyan o definan las reglas.

En caso de igualdad, las reglas !important de los usuarios son más importantes que las reglas!important del diseñador. Gracias a esta característica, si un usuario sufre deficiencias visuales, puede crear una hoja de estilos CSS con reglas de tipo !important con la seguridad de que el navegador siempre aplicará esas reglas por encima de cualquier otra regla definida por los diseñadores.

El principal problema de las hojas de estilo de los navegadores es que los valores que aplican por defecto son diferentes en cada navegador. Aunque todos los navegadores coinciden en algunos valores importantes (tipo de letra serif, color de letra negro, etc.) presentan diferencias en valores tan importantes como los márgenes verticales (margin-bottom y margin-top) de los títulos de sección (<h1>, ... <h6>), la tabulación izquierda de los elementos de las listas (margin-left opadding-left según el navegador) y el tamaño de línea del texto (line-height).

A continuación se muestra el código HTML de una página de ejemplo y seguidamente, una imagen que muestra cómo la visualizan los navegadores Internet Explorer y Firefox:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>Reset</title>
</head>
```

```
<body>

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

<h2>Consectetuer adipiscing elit</h2>

<p>Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh euismod tincidunt ut consectetuer adipiscing elit</p>

<ul>

        <li>Elemento 1</li>

        <li>Elemento 2</li>

        <li>Elemento 3</li>

</ul>

<table summary="Lorem Ipsum">

        <caption>Lorem Ipsum</caption>

        <tr>

                <th>Celda 1-1</th>

                <th>Celda 1-2</th>

        </tr>

        <tr>

                <td>Celda 2-1</td>

                <td>Celda 2-2</td>

        </tr>

</table>

</body>

</html>
```

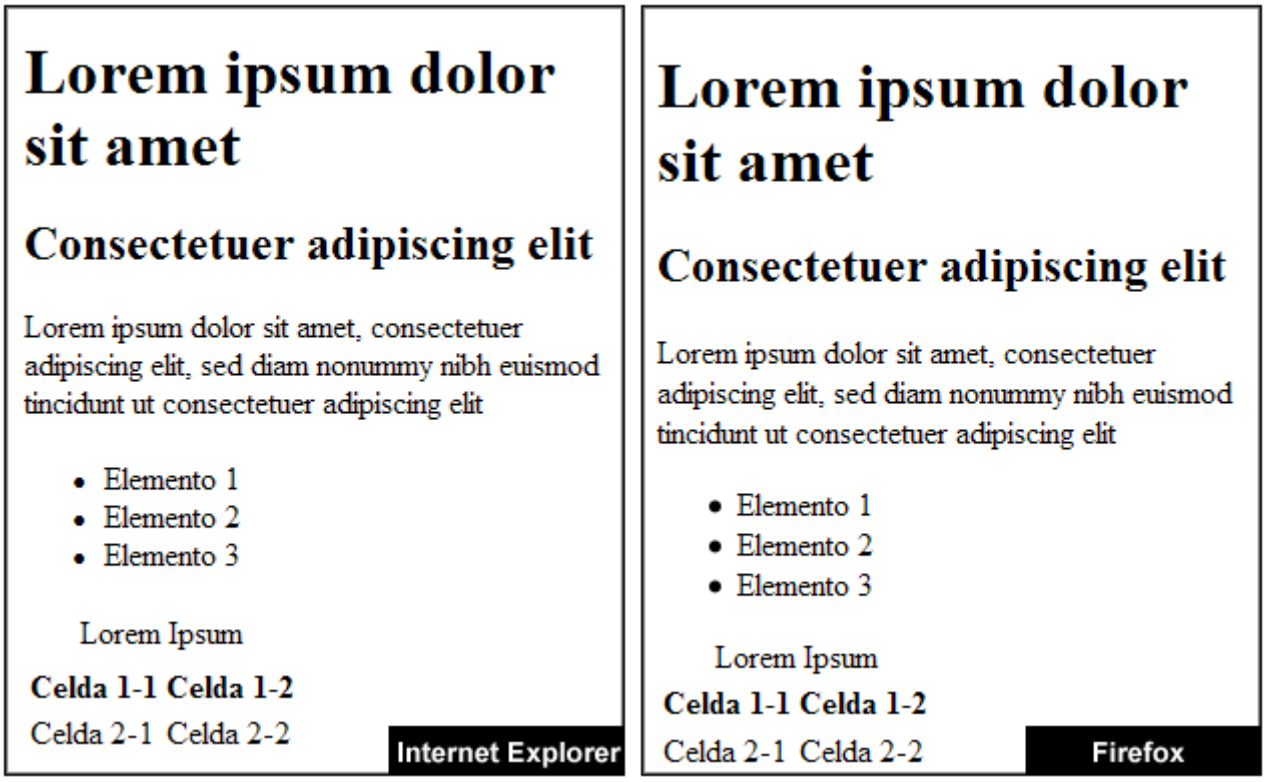


Figura 2.2 Visualización de una misma página en los navegadores Internet Explorer y Firefox

Como todas las hojas de estilo de los navegadores son diferentes, cuando un diseñador prueba sus estilos sobre diferentes navegadores, es común encontrarse con diferencias visuales apreciables. La solución a este problema es muy sencilla y consiste en borrar o *resetear* los estilos que aplican por defecto los navegadores.

Una de las formas más sencillas de neutralizar algunos de los estilos de los navegadores consiste en eliminar el margen y relleno a todos los elementos de la página para establecerlos posteriormente de forma individual:

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

Aunque la regla CSS anterior se ha utilizado desde hace muchos años, se trata de una solución muy rudimentaria y limitada. La solución completa consiste en crear una hoja de estilos CSS que neutralice todos los estilos que aplican por defecto los navegadores y que pueden afectar al aspecto visual de las páginas. Este tipo de hojas de estilos se suelen llamar *"reset CSS"*.

A continuación se muestra la hoja de estilos reset.css propuesta por el diseñador Eric Meyer:

```
/* v1.0 | 20080212 */

html, body, div, span, applet, object, iframe,
h1, h2, h3, h4, h5, h6, p, blockquote, pre,
a, abbr, acronym, address, big, cite, code,
del, dfn, em, font, img, ins, kbd, q, s, samp,
small, strike, strong, sub, sup, tt, var,
b, u, i, center,
dl, dt, dd, ol, ul, li,
fieldset, form, label, legend,
table, caption, tbody, tfoot, thead, tr, th, td {
        margin: 0;
        padding: 0;
        border: 0;
        outline: 0;
        font-size: 100%;
        vertical-align: baseline;
        background: transparent;
}
```

```
body {
	line-height: 1;
}
ol, ul {
	list-style: none;
}
blockquote, q {
	quotes: none;
}
blockquote:before, blockquote:after,
q:before, q:after {
	content: '';
	content: none;
}

/* No olvides definir estilos para focus */
:focus {
	outline: 0;
}

/* No olvides resaltar de alguna manera el texto insertado/borrado */
ins {
	text-decoration: none;
}
del {
	text-decoration: line-through;
}
```

```
/* En el código HTML es necesario añadir cellspacing="0" */
table {
        border-collapse: collapse;
        border-spacing: 0;
}
```

El propio Eric Meyer recuerda que la hoja de estilos anterior es sólo un punto de partida que debe ser adaptado por cada diseñador hasta obtener los resultados deseados. Utilizar una hoja de estilos de tipo *reset* es una de las buenas prácticas imprescindibles para los diseñadores web profesionales.

- **2.2. Comprobar el diseño en varios navegadores**
- **2.2.1. Principales navegadores**

Una de las prácticas imprescindibles de los diseñadores web profesionales consiste en probar su trabajo en varios navegadores diferentes. De esta forma, el diseñador puede descubrir los errores de su trabajo y también los errores causados por los propios navegadores.

El número de navegadores y versiones diferentes que se deben probar depende de cada diseñador. En el caso ideal, el diseñador conoce estadísticas de uso de los navegadores que utilizan los usuarios para acceder al sitio o aplicación web que está diseñando. Una buena práctica consiste en probar los diseños en aquellos navegadores y versiones que sumen un 90% de cuota de mercado.

Cuando no se dispone de estadísticas de uso o el diseño es para un sitio web completamente nuevo, se debe probar el diseño en los navegadores más utilizados por los usuarios en general. Aunque no existe ninguna estadística completamente fiable y los resultados varían mucho de un país a otro, en general los siguientes navegadores y versiones suman más del 90% de cuota de mercado: Internet Explorer 6, Internet Explorer 7, Firefox 2, Firefox 3, Safari 3, Opera 9 y Google Chrome 1.

En primer lugar, los diseñadores web profesionales disponen de todos los navegadores principales instalados en sus equipos de trabajo. Por lo tanto, si no lo has hecho ya, descarga e instala los siguientes navegadores:

- Firefox: disponible para sistemas operativos Windows, Mac, Linux y en más de 45 idiomas.

- Opera: disponible para sistemas operativos Windows, Mac, Linux y en múltiples idiomas.
- Safari: disponible solamente para sistemas operativos Windows y Mac.
- Google Chrome: disponible en más de 40 idiomas y para Windows, Mac, Linux.

Respecto al navegador Internet Explorer, la mayoría de diseñadores trabajan en entornos Windows en los que ya está instalado por defecto. Si tienes la fortuna de trabajar con un sistema operativo tipo Linux, puedes instalar varias versiones de Internet Explorer mediante la aplicación IEs4Linux. También es posible instalar varias versiones de Internet Explorer en los sistemas operativos Mac OS X gracias a la aplicación ies4osx.

- **2.2.2. Probar el diseño en todos los navegadores**

En algunas ocasiones no es suficiente con probar los diseños sólo en los navegadores más utilizados, ya que el cliente quiere que su sitio o aplicación web se vea correctamente en muchos otros navegadores. Por otra parte, no es práctico que los diseñadores prueben sus trabajos en decenas de navegadores y versiones para cada sistema operativo.

Afortunadamente, existe una aplicación web gratuita que permite solucionar este problema. La aplicación Browsershots prueba la página indicada en varias versiones diferentes de cada navegador, crea una imagen de cómo se ve la página en cada uno de ellos y nos muestra esa imagen.

Aunque el proceso es lento y mucho menos flexible que probar la página en nuestros propios navegadores, el resultado permite al diseñador comprobar si su trabajo se ve correctamente en multitud de navegadores y sistemas operativos. En la actualidad, Browsershots comprueba el aspecto de las páginas en 4 sistemas operativos y 72 navegadores diferentes.

- **2.2.3. Integrar Internet Explorer en Firefox**

Algunos diseñadores prueban continuamente sus diseños en los navegadores Internet Explorer y Firefox y después los comprueban en el resto de navegadores para corregir los últimos errores. Si este es tu caso, puedes mejorar tu productividad gracias a una extensión de Firefox.

IE Tab es una extensión que se instala en el navegador Firefox y hace que Internet Explorer se integre en Firefox. Una vez instalada, esta extensión permite ver las páginas con Internet Explorer dentro de Firefox.

Aunque resulta sorprendente, IE Tab hace que las páginas en Firefox se puedan ver mediante Internet Explorer, de forma que los diseñadores no tienen que cambiar constantemente de navegador y por tanto aumenta considerablemente su productividad.

- **2.2.4. Diferentes versiones de Internet Explorer**

El principal problema de los diseñadores web que quieren probar su trabajo en diferentes navegadores y versiones es la imposibilidad de instalar varias versiones del navegador Internet Explorer en el mismo sistema operativo.

Aunque la empresa Microsoft, creadora de Internet Explorer, sigue sin resolver este problema, se han publicado soluciones no oficiales para disponer de varias versiones de Internet Explorer en el mismo sistema operativo.

La primera solución propuesta fue el *Browser Archive*, un repositorio de navegadores desde el que se pueden descargar versiones antiguas de decenas de navegadores diferentes, entre ellos Internet Explorer. Lamentablemente hace mucho tiempo que no se añaden nuevas versiones, por lo que la última versión disponible de Internet Explorer es la 6.

Más recientemente se ha presentado IETester, una aplicación descargable gratuitamente y que permite disponer de Internet Explorer 5.5, 6, 7 y 8 en un mismo sistema operativo. De esta forma, IETester es una de las herramientas imprescindibles de los diseñadores web profesionales.

- **2.3. Mejora progresiva**

La mejora progresiva (*"progressive enhancement"*) es uno de los conceptos más importantes del diseño web y a la vez uno de los más desconocidos. Su origen proviene de su concepto contrario, la degradación útil o *"graceful degradation"*.

La degradación útil es un concepto propuesto hace décadas por el psicólogo inglés David Courtenay Marr. Aplicada al diseño web, la degradación útil significa que un sitio web sigue funcionando correctamente cuando el usuario accede con un navegador limitado o antiguo en el que no funcionan las características más avanzadas.

La mejora progresiva toma ese concepto y lo aplica de forma inversa. En el diseño web, la mejora progresiva significa que el sitio web dispone de características más avanzadas cuanto más avanzado sea el navegador con el que accede el usuario.

Muchos diseñadores web y muchos de sus clientes están obsesionados con que sus diseños se vean exactamente igual en cualquier versión de cualquier navegador. Aunque resulta prácticamente imposible conseguirlo, este tipo de diseñadores prefiere sacrificar cualquier característica interesante de CSS de manera que las páginas se vean igual en cualquier navegador.

La mejora progresiva propone que el diseño web se realice de la siguiente manera:

- En primer lugar, el diseño web debe permitir el acceso completo y correcto a toda la información de la página independientemente del tipo de navegador utilizado por el usuario.
- Después de cumplir el requisito anterior, se deben utilizar las características más modernas de CSS 2 e incluso de CSS 3, aunque sólo los usuarios con navegadores más modernos sean capaces de disfrutarlas.

La técnica de la mejora progresiva es mucho mejor que las soluciones alternativas que utilizan algunos diseñadores, como por ejemplo:

- Utilizar sólo las características de CSS que soporte correctamente el navegador obsoleto Internet Explorer 6, porque un gran número de usuarios siguen utilizándolo.
- Utilizar sólo las características de CSS que soporten correctamente navegadores limitados como Internet Explorer 7, ya que es el navegador más utilizado por los usuarios.
- Olvidarse completamente de navegadores limitados como Internet Explorer 6 y 7, diseñando los sitios web sólo para los navegadores más modernos.

A continuación se muestra la mejora progresiva en la práctica mediante un ejemplo publicado en el artículo *Progressive Enhancement with CSS 3: A better experience for modern browsers*.

El propósito del ejemplo es crear un menú de navegación que se ve más bonito cuanto más moderno sea tu navegador. Como es habitual, el código HTML del menú se basa en una lista de tipo <ul>:

```
<ul>
        <li><a href="">Lorem Ipsum</a></li>
        <li><a href="">Lorem Ipsum</a></li>
        <li><a href="">Lorem Ipsum</a></li>
        <li><a href="">Lorem Ipsum</a></li>
        <li><a href="">Lorem Ipsum</a></li>
</ul>
```

El primer paso consiste en aplicar los estilos CSS básicos que interpretan correctamente todas las versiones de todos los navegadores. Aunque estos estilos hacen que el menú tenga un aspecto muy básico, permiten el acceso correcto a todos los contenidos.

```
ul {
        background-color: blue;
        border-bottom: 1px dotted #999;
        list-style: none;
        margin: 15px;
        width: 150px;
        padding-left: 0;
}

li {
        background-color: #FFF;
        border: 1px dotted #999;
        border-bottom-width: 0;
        font: 1.2em/1.333 Verdana, Arial, sans-serif;
}
li a {
        color: #000;
        display: block;
        height: 100%;
        padding: 0.25em 0;
        text-align: center;
        text-decoration: none;
}
li a:hover { background-color: #EFEFEF; }
```

Las reglas CSS anteriores hacen que el menú de navegación tenga el siguiente aspecto en cada navegador:

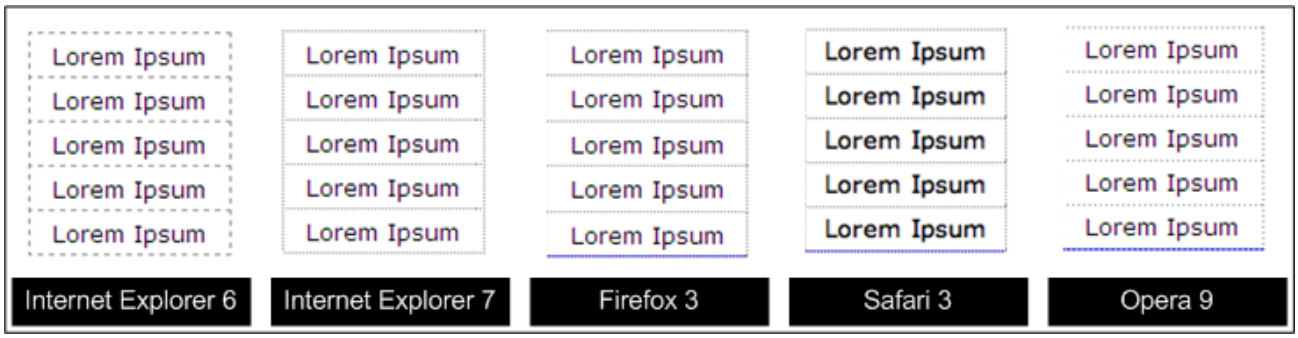


Figura 2.3 Aspecto del menú en los navegadores Internet Explorer 6 y 7, Firefox 3, Safari 3 y Opera 9

Como se ve en la imagen anterior, incluso con unos estilos CSS tan básicos se producen diferencias visuales entre los navegadores. El motivo es que Internet Explorer 6 no es capaz de mostrar un borde punteado de 1px de anchura y lo sustituye por un borde discontinuo.

El siguiente paso consiste en utilizar el selector de hijos (uno de los selectores avanzados de CSS) para añadir nuevos estilos:

```
body > ul { border-width: 0; }
ul > li {
  border: 1px solid #FFF;
  border-width: 1px 0 0 0;
}
li > a {
  background-color: #666;
  color: white;
  font-weight: bold;
}
li:first-child a { color: yellow; }
li > a:hover{ background-color: #999; }
```

Ahora el primer elemento del menú de navegación se muestra con otro estilo y cuando el usuario pasa su ratón por encima de un elemento se muestra destacado:

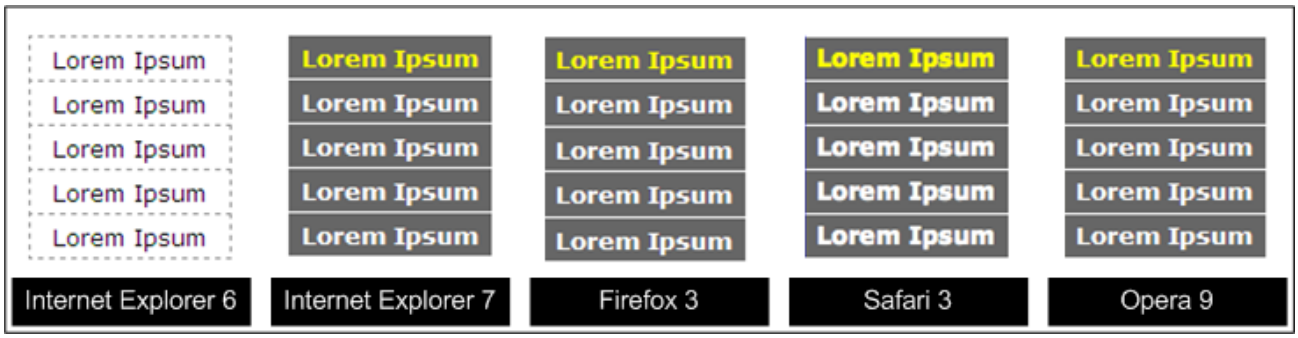


Figura 2.4 Aspecto del menú en los navegadores Internet Explorer 6 y 7, Firefox 3, Safari 3 y Opera 9

El navegador Internet Explorer 6 se queda atrás y sigue mostrando el menú con el mismo aspecto, ya que no es capaz de entender el selector de hijos. La mejora progresiva permite que los usuarios de Internet Explorer 6 sigan accediendo a todos los contenidos y que el resto de usuarios vean un menú más avanzado.

A continuación se modifica la opacidad de los elementos del menú, de forma que el elemento seleccionado se vea más claramente:

```
li { opacity: 0.9; }

li:hover{ opacity: 1; }
```

En esta ocasión, el navegador que se queda atrás es Internet Explorer 7, ya que no incluye soporte para esa propiedad de CSS. En el resto de navegadores se muestra correctamente el efecto:

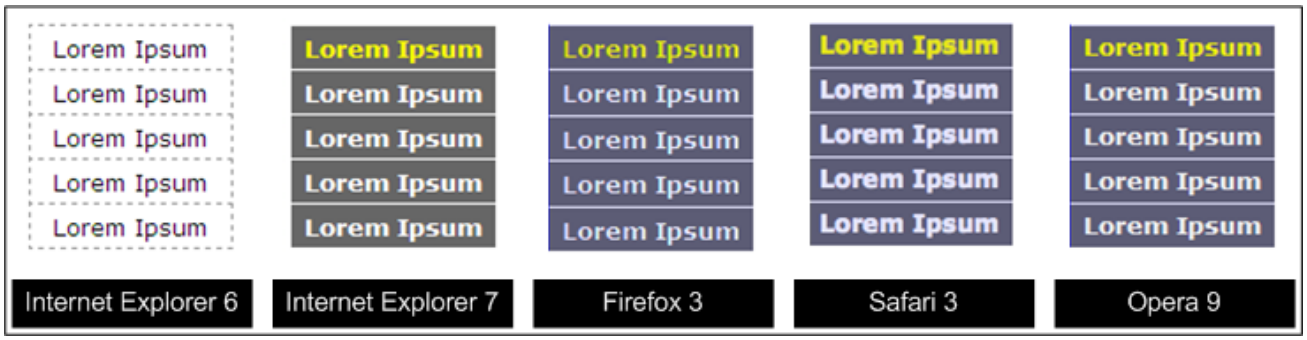


Figura 2.5 Aspecto del menú en los navegadores Internet Explorer 6 y 7, Firefox 3, Safari 3 y Opera 9

Otra posible mejora del menú consiste en añadir una sombra al texto del elemento seleccionado mediante la propiedad text-shadow de CSS:

```
li a:hover { text-shadow: 2px 2px 4px #333; }
```

Solamente los navegadores Safari y Opera soportan la propiedad text-shadow, por lo que el navegador Firefox se queda atrás y no muestra esta mejora:

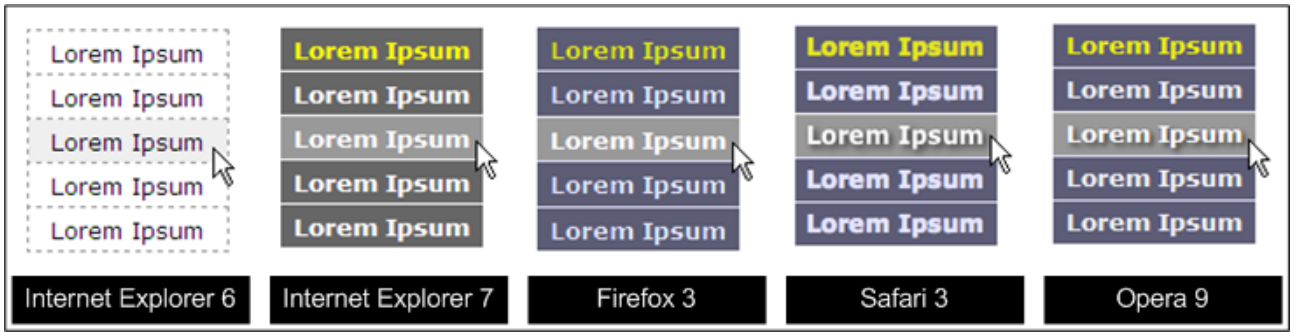


Figura 2.6 Aspecto del menú en los navegadores Internet Explorer 6 y 7, Firefox 3, Safari 3 y Opera 9

Por último, utilizando los selectores de CSS 3 se va a alternar el color de fondo de los elementos del menú para mejorar su visualización:

```
li:nth-child(2n+1) a {
        background-color: #333;
}

li:nth-child(n) a:hover {
        background-color: #AAA;
        color: #000;
}

li:first-child > a:hover{ color: yellow; }
```

Solamente los navegadores Opera y Safari incluyen todos los selectores de CSS 3, por lo que el resultado final del menú en cada navegador es el que muestra la siguiente imagen:

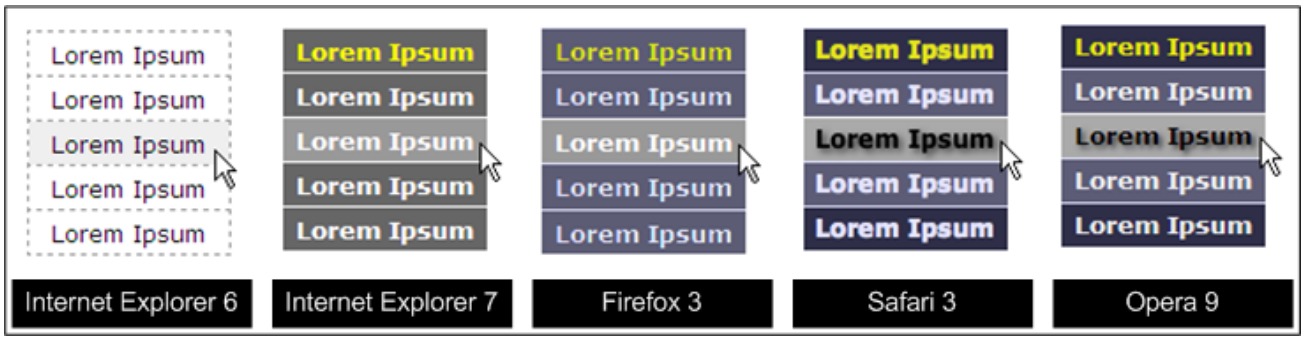


Figura 2.7 Aspecto del menú en los navegadores Internet Explorer 6 y 7, Firefox 3, Safari 3 y Opera 9

Como se ha visto en este ejemplo, la mejora progresiva permite aprovechar todas las posibilidades de CSS sin perjudicar a los navegadores obsoletos o limitados. Los usuarios de Internet Explorer 6 visualizan un menú muy básico adecuado para un navegador obsoleto, pero que les permite el acceso a todos los contenidos. Los usuarios de Internet Explorer 7 visualizan un menú normal adecuado a las limitaciones de su navegador pero que también permite el acceso a todos los contenidos. Por último, los usuarios de los navegadores más avanzados (Opera y Safari) visualizan un menú avanzado que aprovecha todas las características disponibles en CSS.

- **2.4. Depuración**

Inevitablemente, todos los diseñadores web cometen errores en los trabajos que realizan. En la mayoría de las ocasiones, los errores se descubren al probar el diseño en diferentes navegadores. Además de mostrar los errores, los principales navegadores disponen de herramientas que permiten descubrir de forma sencilla la causa concreta del error.

Antes de que existieran estas herramientas avanzadas, el trabajo del diseñador era mucho más complicado, ya que no era fácil descubrir la causa exacta del error entre todas las posibles:

- El selector está mal escrito.
- Las propiedades están mal escritas o tienen valores no permitidos.
- Otros selectores tienen más prioridad y están sobreescribiendo una propiedad y/o valor.
- Las reglas y valores están bien escritos, pero los elementos no ocupan el espacio que a simple vista parece que están ocupando en la pantalla.
- El navegador tiene un error que impide mostrar correctamente la página.

Los diseñadores web idearon hace mucho tiempo soluciones ingeniosas para cada uno de los problemas anteriores. En primer lugar, cuando no se está seguro de si todas las reglas CSS están bien escritas, lo mejor es validar la hoja de estilos utilizando el validador CSS del W3C.

Una vez descartado el error de sintaxis, el siguiente problema a resolver es por qué una regla CSS no se aplica correctamente a un elemento. Una estrategia muy utilizada consistía en añadir alguna propiedad que sea visualmente significativa para comprobar si realmente el selector se está aplicando. Poner todo el texto del elemento en negrita, aumentar mucho su tamaño de letra y cambiar el color de fondo eran algunas de las estrategias habituales. Cuando lo anterior no resultaba, se utilizaba directamente la palabra reservada !important para aumentar la prioridad de esa propiedad CSS.

Otro de los problemas habituales en el diseño web está relacionado con el espacio que ocupa cada elemento en pantalla. Como los elementos por defecto no muestran ningún borde y su color de fondo es transparente, no es posible conocer a simple vista el espacio que ocupa cada elemento. Por lo tanto, cuando se posicionan elementos de forma flotante o cuando se establecen márgenes, rellenos, alturas y anchuras máximas/mínimas, no es posible visualizar si el navegador está mostrando correctamente todos los elementos.

Para solucionar este problema la técnica habitual consistía en añadir un borde visible a los elementos. Como los bordes visibles ocupan sitio en pantalla, el problema de esta solución es que modifica el propio diseño. La alternativa consistía en añadir un color de fondo diferente para cada elemento. Otra posible alternativa es el uso de la propiedad outline de CSS, que añade un perfil en el contorno de un elemento pero no ocupa sitio y por tanto no modifica el diseño de la página.

Los navegadores modernos como Safari, Opera y Firefox incluyen el soporte de la propiedadoutline, mientras que el navegador Internet Explorer 7 no es capaz de mostrar perfiles en los elementos de la página. El diseñador Chris Page ha publicado un artículo llamado *A Handy CSS Debugging Snippet* en el que muestra unas reglas CSS que hacen uso de outline y permiten depurar fácilmente cualquier diseño sin utilizar herramientas avanzadas:

```
* { outline: 2px dotted red }

* * { outline: 2px dotted green }

* * * { outline: 2px dotted orange }

* * * * { outline: 2px dotted blue }

* * * * * { outline: 1px solid red }

* * * * * * { outline: 1px solid green }

* * * * * * * { outline: 1px solid orange }

* * * * * * * * { outline: 1px solid blue }
```

2.4.1. Firebug

Al margen de soluciones manuales y técnicas más o menos ingeniosas, los diseñadores web profesionales de hoy en día utilizan herramientas avanzadas para averiguar con precisión la causa de los errores de diseño. De todas las herramientas disponibles, la mejor con mucha diferencia esFirebug, una extensión del navegador Firefox.

Firebug dispone de todas las utilidades que necesitan los diseñadores y programadores web en su trabajo. Firebug es una herramienta gratuita, completa, fácil de utilizar y para la que se publican nuevas versiones de forma continua.

Después de instalar Firebug, en la esquina inferior derecha del navegador Firefox aparece un nuevo icono. Para acceder a Firebug, se puede pinchar con el ratón sobre ese icono o se puede pulsar directamente la tecla F12 después de cargar la página que se quiere depurar.

Firebug muestra su información dividida en los siguientes paneles: * Consola: muestra mensajes de error, notificaciones y otros tipos de mensajes. No es muy útil para los diseñadores web. * HTML: muestra el código HTML de la página y permite seleccionar los elementos, modificar sus contenidos y ver las reglas CSS que se le están aplicando. * CSS: muestra todas las hojas de estilos incluidas en la página y permite modificar sus valores. * Guión y DOM: paneles relacionados con la programación JavaScript. No son muy útiles para los diseñadores web. * Red: muestra toda la información de todos los elementos descargados por la página (HTML, JavaScript, CSS, imágenes).

El primero de los paneles importantes para los diseñadores web es el panel HTML:

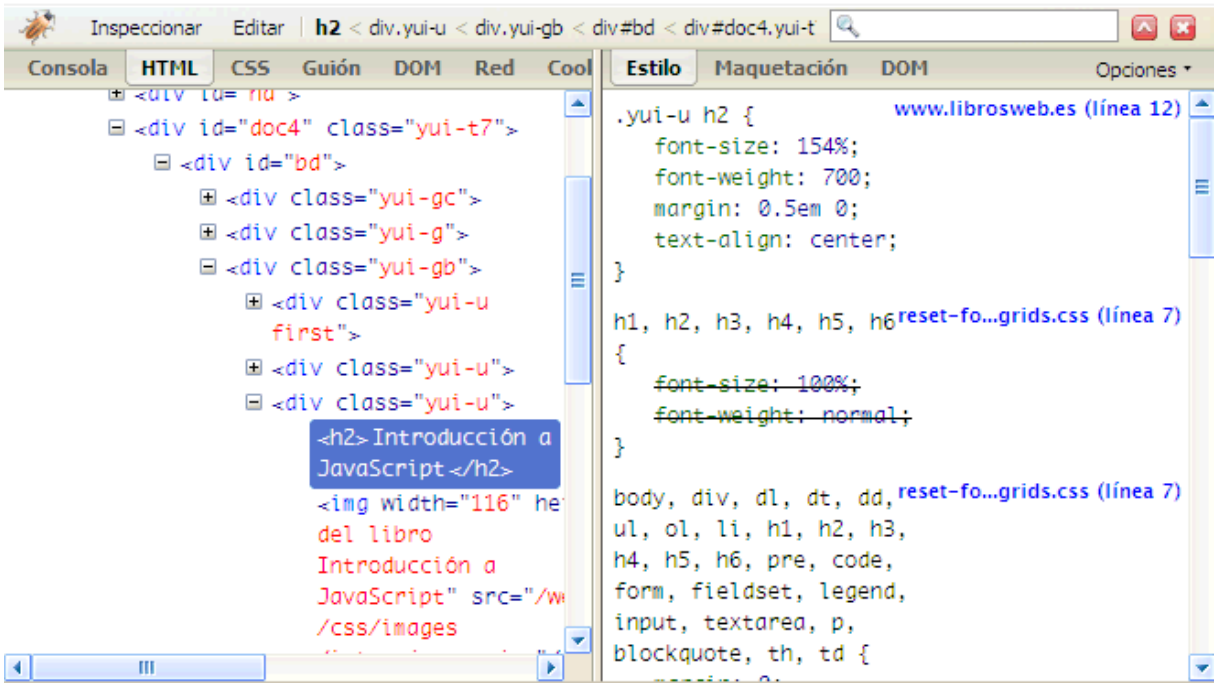


Figura 2.8 Panel HTML de Firebug

El panel HTML es el más utilizado por los diseñadores web, ya que muestra toda la información de la página relacionada con HTML y CSS. En la parte izquierda del panel se muestra el código HTML de la página y en la parte derecha la información CSS.

Si se pulsa el botón "Inspeccionar" de la parte superior de Firebug, es posible seleccionar con el ratón un elemento de la página web. Después de pinchar sobre cualquier elemento, en la parte izquierda se muestra el código HTML de ese elemento y en la parte derecha todas las reglas CSS que se le aplican.

Si se pulsa sobre el contenido de un elemento en la parte izquierda del panel HTML, se puede modificar su valor y ver el resultado en tiempo real sobre la propia página web. Desde este mismo panel también es posible eliminar el elemento de la página.

La parte derecha del panel HTML es la más útil, ya que siempre muestra todas las reglas CSS que se aplican a un elemento de la página. Gracias a esta información es imposible dudar si un selector está bien escrito o si una regla CSS realmente se está aplicando a un elemento.

Además, como normalmente varias reglas CSS diferentes aplican valores diferentes a las mismas propiedades de un mismo elemento, Firebug muestra tachadas todas las propiedades que en teoría se deben aplicar al elemento pero que no lo hacen porque existen otras reglas CSS con más prioridad.

La parte derecha del panel HTML incluye otras utilidades interesantes como cuando se pasa el ratón por encima de un color definido en formato hexadecimal y que hace que se vea realmente cuál es el color. Igualmente, al pasar el ratón por encima de una url() utilizada para incluir una imagen, Firebug muestra de qué imagen se trata. Las reglas CSS que se muestran en la parte derecha del panel HTML también se pueden modificar, eliminar y bloquear temporalmente. También es posible añadir nuevas propiedades a cualquier regla CSS.

Firebug muestra por defecto el valor de las reglas CSS tal y como se han establecido en las hojas de estilos. Sin embargo, muchas veces estos valores originales no son prácticos. ¿cuál es el tamaño de letra de un elemento con font-size: 1em? Sin más información es imposible saberlo. ¿cuál es la anchura en píxeles de un elemento con la propiedad width: 60%? Imposible saberlo sin conocer las anchuras de sus elementos contenedores.

Por todo ello, Firebug permite mostrar los valores que realmente utiliza Firefox para dibujar cada elemento en pantalla. Pulsando sobre el texto Opciones de la parte derecha del panel HTML, se puede activar la opción Show Computed Style:

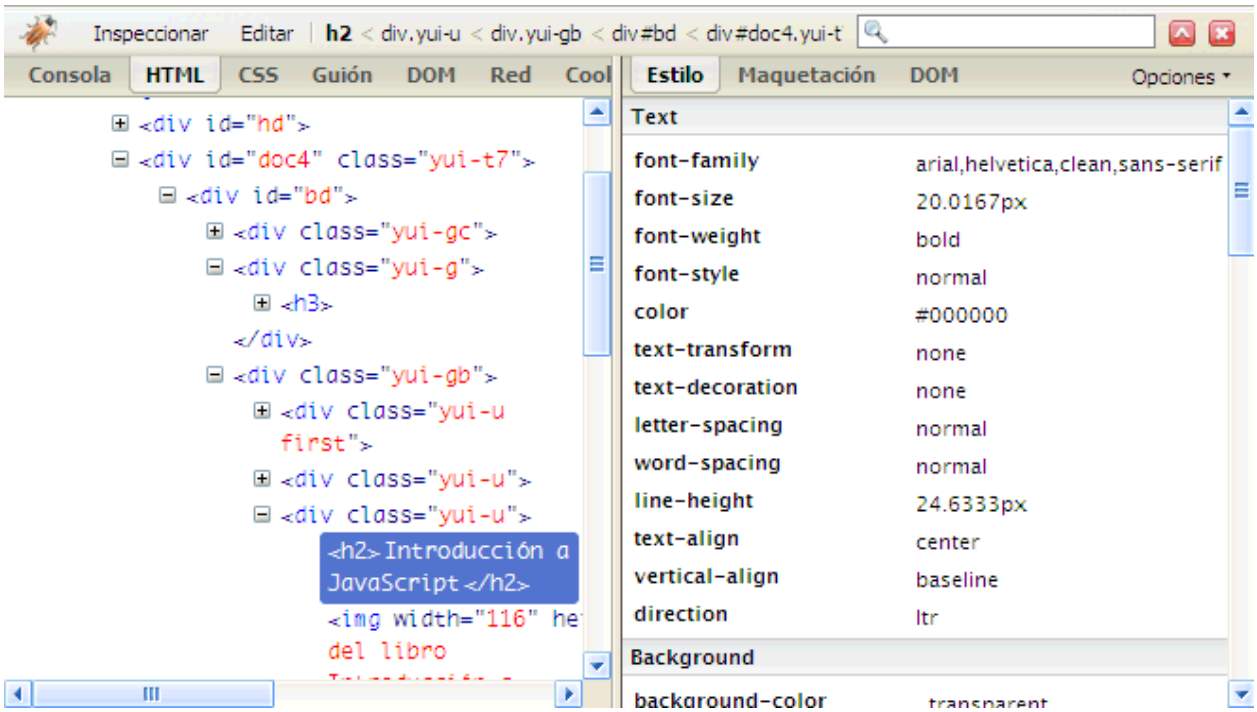


Figura 2.9 Panel HTML de Firebug con la opción Show Computed Style

Después de activar esta opción, los tamaños de letra y anchuras se muestran en píxeles y se muestra el valor de todas las propiedades CSS del elemento, independientemente de si se han establecido de forma explícita o de si se trata de los valores por defecto que aplica el navegador.

Otra de las utilidades más interesantes del panel HTML es la información sobre la maquetación del elemento, que se puede mostrar pinchando sobre la pestaña Maquetación de la parte derecha del panel:

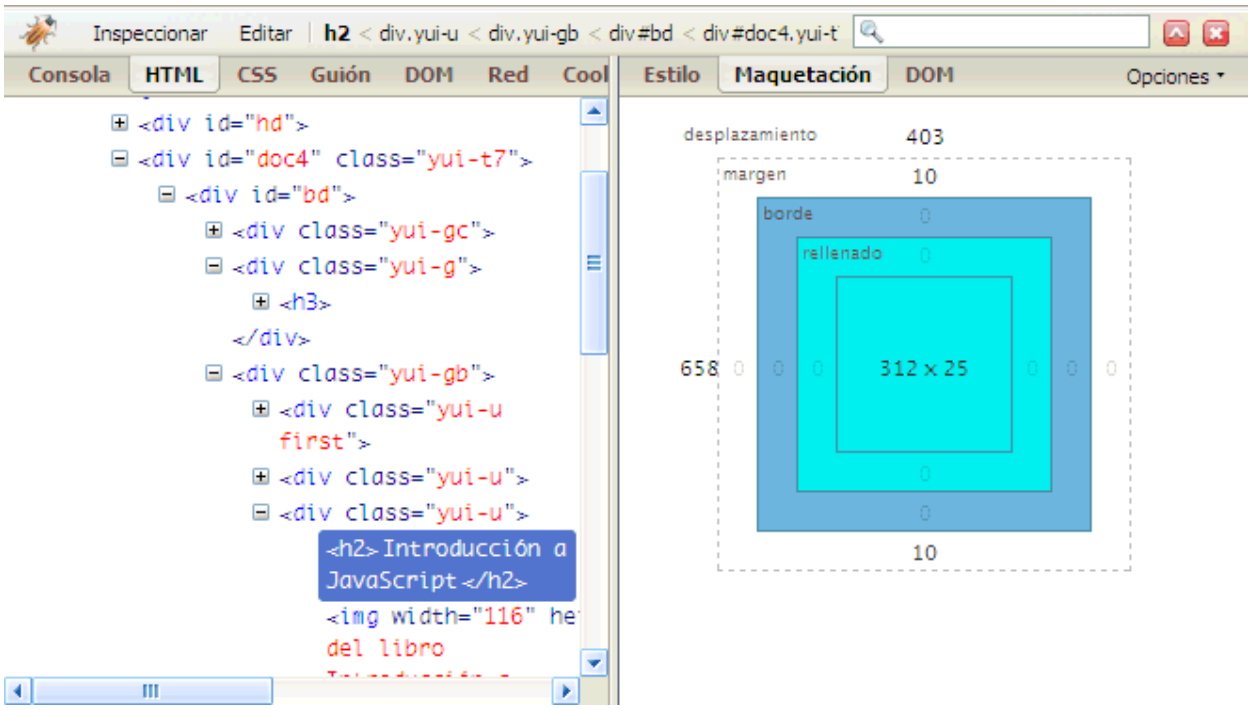


Figura 2.10 Pestaña Maquetación del panel HTML de Firebug

La opción Maquetación muestra la información completa del *"box model"* o modelo de cajas de un elemento: anchura, altura, rellenos, bordes y márgenes.

El otro panel más utilizado por los diseñadores web es el panel CSS, que muestra el contenido de todas las hojas de estilos que se están aplicando en la página y permite realizar cualquier modificación sobre cualquier regla CSS viendo el resultado en tiempo real en la propia página:

Figura 2.11 Panel CSS de Firebug

Por último, el panel Red de Firebug permite ver toda la información sobre todos los elementos que se descarga el navegador para mostrar la página:

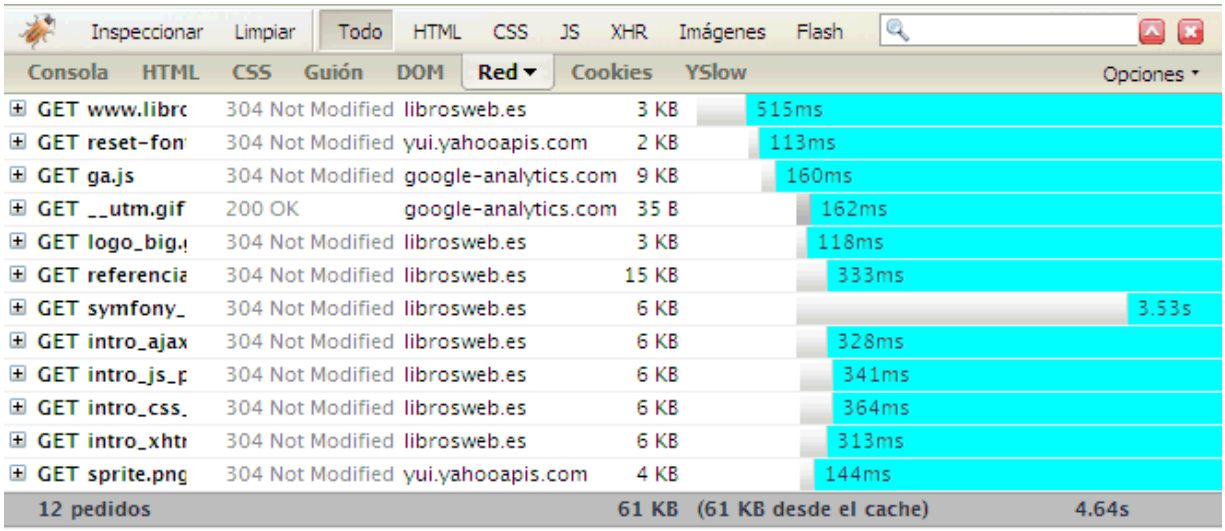


Figura 2.12 Panel Red de Firebug

Desde el punto de vista del diseñador, este panel se puede utilizar para mejorar el rendimiento de la página reduciendo el número de archivos CSS, reduciendo el número de imágenes de adorno mediante los *sprites CSS*, reduciendo el número de peticiones HTTP realizadas al servidor y reduciendo el tamaño de los archivos.

2.4.2. Otras herramientas de depuración

Firebug es la mejor herramienta para depurar el diseño de los sitios web, pero sólo está disponible para el navegador Firefox. Como la mayoría de errores en el diseño web sólo se producen en los navegadores de la familia Internet Explorer, Firebug no se puede utilizar.

Afortunadamente, los creadores de Firebug han publicado una versión reducida y simplificada de Firebug compatible con el resto de navegadores. La versión reducida se denomina Firebug Lite y requiere el uso de JavaScript. Aunque se puede descargar Firebug Lite para utilizarlo desde nuestros propios servidores, la forma más sencilla de probarla es añadir el siguiente código en la página que se quiere depurar:

```
<script type="text/javascript" src="http://getfirebug.com/releases/lite/1.2/firebug-lite-compressed.js"></script>
```

El código anterior se puede colocar en cualquier zona de la página, aunque normalmente se incluye dentro de la sección <head>.

Por último, como las empresas que desarrollan los navegadores consideran que Firebug es insuperable, desde hace un tiempo se dedican a copiar todas sus características. Internet Explorer 8, Safari 3 y Opera 9 disponen de herramientas de depuración que son una réplica de Firebug.

Internet Explorer 8 ha denominado a su herramienta *Developer Tools*, mientras que Opera la ha denominado *Dragonfly* y en Safari está disponible desde el menú Desarrollo.

FIN