

Certificado Programador CSS3 para móviles.



Manual del participante

2014



Índice

Índice	2
Formación de la comunidad de aprendizaje	3
Objetivo del curso	4
Objetivo	4
Requerimientos de Software.....	4
Unidad 1 Introducción.....	5
Hojas de estilo	5
Mostrar contenido de diversas formas	5
HTML, XHTML y CSS.....	18
Unidad 2 Estructura del CSS	20
Sintaxis	20
Inclusión de CSS en un HTML.....	21
Incluir CSS en los elementos HTML.....	24
Unidad 3 Elementos básico.....	26
Unidad 4 Colores y Fondos	38
Unidad 5 Texto	44
Unidad 6 Fuentes.....	59
Unidad 7 Ligas	70
Tablas.....	77
Formas	85
Modelo de Caja	93
Unidad 8 CSS 3	134
Conclusiones finales	138
Unidades de medida	139
Unidades relativas.....	139
Unidades absolutas	142
Porcentajes.....	143
El valor inherit	144
Fuentes de consulta	146
Bibliografía	146
Cibergrafía	146

Formación de la comunidad de aprendizaje

La formación de la comunidad de aprendizaje es un proceso que debe llevarse a cabo para iniciar cada uno de nuestros cursos.

Su finalidad es crear un clima propicio para la celebración de la actividad instruccional, es decir, generar un entendimiento previo entre el instructor y los participantes sobre los temas que se desarrollarán durante ésta, así como las estrategias educativas que se llevarán a cabo para lograr un mejor aprendizaje.

Un adecuado manejo de la comunidad de aprendizaje es un elemento fundamental para garantizar la satisfacción de uno de los clientes involucrados en la impartición de los cursos: **los participantes.**

Presentación del Instructor:

- ★ Nombre, profesión, años de experiencia como instructor, experiencia en la impartición del curso, o cursos similares o relacionados.

Alineación de expectativas:

- ★ El instructor recabará las expectativas de los participantes respecto al curso, con el fin de dejarles claro el objetivo del mismo.
- ★ En caso de que alguna expectativa no coincida con los temas que el curso contiene, el instructor dejará claro cuáles de las expectativas expresadas no serán cubiertas con el curso y porqué.
- ★ Las expectativas alineadas serán anotadas en hojas de rotafolio para su revisión al término del curso.
- ★ Durante el desarrollo del curso el instructor deberá cubrir las expectativas alineadas.

Presentación del objetivo del curso:

- ★ El instructor presentará a los participantes el objetivo del curso, aclarando dudas al respecto si las hubiese.

Reglas de oro:

- ★ El instructor promoverá el establecimiento de reglas por parte de los participantes que se observarán a través del curso; por lo que puede proponer: tiempo de tolerancia para iniciar las sesiones, respeto hacia los compañeros, participación de todos en técnicas y ejercicios grupales, etc.; se incluirán todos los puntos que los participantes consideren pertinentes.
- ★ Se anotarán los acuerdos en hojas de rotafolio y se colocarán en un espacio en el que sean visibles a lo largo de todo el curso.

Cumplimiento de expectativas

- ★ Al finalizar el curso el instructor deberá llevar a cabo una revisión de las expectativas alineadas que se anotaron en hojas de rotafolio al inicio del curso
- ★ Se revisará cada una de las expectativas alineadas palomeando las que hayan sido cumplidas, y el instructor explicará de qué manera se llevó a cabo tal cumplimiento.

Objetivo del curso

Objetivo

Al final del curso el asistente será capaz de presentar contenidos en línea basados en HTML, aprovechando las ventajas que ofrece el uso de las hojas de estilo mediante las especificaciones CSS2 y la nueva CSS3.

Esta certificación está enfocada a aquellos interesados en aprovechar la potencia de las hojas de estilo para desplegar información basada en HTML de diversos modos gráficos sin necesidad de modificar el contenido.

Requerimientos de Software

- ✓ Un editor de textos planos (como [Notepad++](#)/[Sublime Text 2](#) para Windows, [gedit](#)/[Kate](#) para Linux o [TextMate](#) para Mac OS X);

Unidad 1 Introducción

Hojas de estilo

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

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

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

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

Mostrar contenido de diversas formas

Dentro de este apartado, veremos la forma en la que podemos presentar la información en diferentes formas, en donde de inicio, comenzaremos con la presentación de la información de una página web en formato de HTML puro, para después incluir un poco de "magia" con CSS.

Lo primero que vamos hacer en este apartado es el código **HTML5** para lo que crearemos un documento en blanco con el nombre de "index.html" y luego los **estilos CSS3**, bueno empezamos:

Escribimos la cabecera de la página

```
<!DOCTYPE html/>
<html lang="es">
<head>
<meta charset="utf-8" />
<meta name="description"content="Curso de HTML5" />
<title>Curso de HTML5 y CSS3 </title>
<link rel="stylesheet" href=" stylo.css" />
</head>
```

Bueno esta es la parte de la cabecera de nuestra web ahora le explico cómo funciona primero pusimos la etiqueta `<!DOCTYPE html/>` esto significa que empezamos a programar nuestra web con **HTML5**.

Luego escribimos la etiqueta de apertura `<html lang="es">` y le establecemos el lenguaje español en nuestro caso esto es para que nos res conozca los caracteres latinos y no tengamos problemas con esos signos raros.

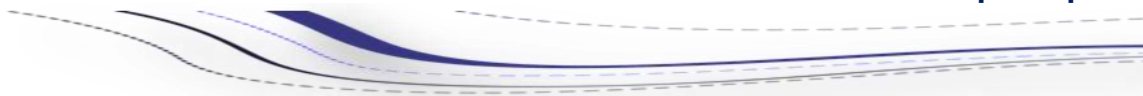
Luego abrimos y cerramos la etiqueta `<head> </head>` y en ella escribimos el `<meta charset="utf-8" />` para los signos de puntuación también una meta con la descripción de nuestra página web que es esta `<meta name="description"content="Curso de HTML5" />`

También incluimos el titulo de nuestra página con la etiqueta

```
<title>El titulo de tu web </title>
```

Una vez añadida la etiqueta title enlazamos una hoja de estilo con el nombre “stylo.css” en la cual estará el código **CSS3** que vamos a crear para maquetar nuestra web `<link rel="stylesheet" href=" stylo.css" />` luego cerramos la etiqueta `</head>` y continuamos con el cuerpo de la página, y estaría estructurado de la siguiente manera solo tenemos que agregar el siguiente código.

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



Con esto ya tenemos el cuerpo de nuestra página programada en HTML ahora empezamos a introducir el contenido con etiquetas **HTML5** obvio que pensaban que era con HTML4 bueno lo primero que creamos es la cabecera de nuestra página con la etiqueta `<header></header>`

Y dentro de este header creamos un Título con el máximo tamaño (`<h1>`) e introducimos un texto para describirlo, en este caso nuestra cabecera sería la siguiente.

```
<header>
<h1>Esta es la cabecera de nuestro sitio web</h1>
</header>
```

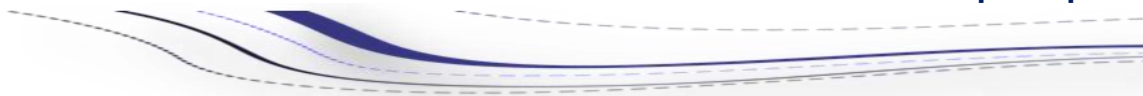
Ahora creamos una etiqueta `<nav></nav>` para la botonera y quedaría así

```
<nav>
Esta es nuestra botonera
</nav>
```

Ahora creamos el cuerpo del la mayor parte del contenido de nuestra web para esto creamos una etiqueta `<section></section>` y dentro de esta metemos otro `section` parecido pero con `id` diferentes y también agregamos las etiquetas `<article></article>` y `<aside></aside>`

En este caso nuestro código **HTML5** quedaría así

```
<section id="contenedor">
<section id="principal">
Esto es una seccion
<article>
Esto es un article
</article>
</section>
<aside>
Esto es un aside
</aside>
</section>
```



Ahora lo que nos falta es el pie de página y para esto utilizamos las etiquetas `</footer></footer>`

Que el código sería el siguiente

```
<footer>
Este es el pie de nuestra página
</footer>
```

Bueno con todo esto explicado ahora les pongo el código **HTML5** completo para que lo estudien y vean como está estructurado y para pasar a **maquetar con CSS3**.

Ahora procedemos a maquetar nuestra **página HTML5** con puro **código CSS3** y que todo se ajuste a cualquier pantalla donde se esté visualizando.

Bueno empecemos lo primero que vamos a definir es el atributo de todo el HTML para esto lo primero que hacemos es un `“*”` y entre llaves ponemos el código que queremos establecerle al documento completo en este caso yo solo les voy a establecer dos valores que es el margen y el padding

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

Ahora establecemos la tipografía de nuestro sitio web para eso tenemos que utilizar la el atributo de CSS3 que es `@font-face` y lo que les vamos a establecer a este atributo son 2 mandatos que el tipo de letras y el estilo sea normal

```
@font-face {
font-weight:normal;
font-style:normal;
}
```

Si has hecho todo bien nuestro índice tendría que estar quedando así



De lo contrario revisa el código para asegurarnos de que no exista error. Bueno una vez con esto así en caso de que no hayas cometido ningún error proseguimos.

Y ahora continuamos con el siguiente bloque de **código CSS3** que corresponde al `body` en este caso al `body` lo primero que les vamos a decir es que tenga un background blanco un color de letras naranja una tipografía o tipo de letra Helvética, Verdana y un tamaño de letras de 2em entonces nuestro **código CSS3** sería el siguiente

```
body{
background: #FFF;
color:#F60;
font-family:Helvetica, Verdana;
font-size: 2em;
}
```

Ahora lo que hacemos es que al `h1` que está dentro del `<header>` es establecerles los parámetros de cómo queremos que sea para estos le ponemos el siguiente **código CSS3**.

```
h1{
color:#F60;
margin:0.25em auto;
text-align:center;
text-shadow: 5px 5px 10px rgba(255,255,255,0.5);
}
```

En este código lo primero que establecimos fue el color de las letras del `<h1>` en este caso (`color:#F60;`) que corresponde al naranja, luego el `margin` es espacio que va a tener de los demás elementos. También establecimos el `text-align:center;` esto es para alinear el texto

al centro y algo muy importante que no se puede quedar es el atributo (`text-shadow:`) que lo que nos proporciona es la sombra a las letras.

Como ya agregamos los estilos del `<body>` y del `<h1>` que está dentro del `<header>` nuestra página tendría que estar quedando de la siguiente manera.



Si se fijan nuestro documento HTML a cambiado mucho y en el `h1` se nota una sombra la cual la establecimos con el atributo (`text-shadow:`) si tu imagen va quedando igual que esta sigamos adelante, en caso contrario te recomiendo volver atrás y revisar el código para ver en que hemos fallado.

Bueno seguimos maquetando nuestro **index.html** ahora vamos a establecer el color de fondo de la cabecera que en este caso sería nuestra etiqueta `<header></header>` para ello vamos a ponerle un `background:#DEDEDE;` que corresponde a un gris también un `border-radius:0.25em;` para ponerles los bordes medio redondeados en `margin` que va a tener de los otros elementos donde les ponemos (`margin:0.25em auto`) con esto les indicamos al `<header>` que tome un `margin` de arriba de 0.25em y a los lados se centre con el atributo (`auto`).

Y algo nuevo que incorpora CSS3 para que nuestros sitios puedan ser auto ajustables es el (`max-width`) y el (`min-height`) con estos atributos no le establecemos márgenes variables a nuestros documentos HTML donde les indico que le tamaño máximo o `max-width` va hacer el que queramos y el tamaño mínimo `min-height` también va hacer el que le establezcamos, o sea ya nuestros documentos no van a tener un tamaño fijo si no que van a varias dependiendo el tamaño de la pantalla que los este visualizando.

Con esto claro solo queda establecer el `padding` y alinear el texto lo cual lo hacemos con las etiquetas (`padding:0.25em;`) y (`text-align:center;`)

Con esto el código para nuestro `<header>` sería el siguiente.

```
header {
background:#DEDEDE;
border-radius:0.5em;
margin: 0.5em auto;
max-width:960px;
padding:0.25em;
text-align:center;
}
```

Ahora si visualizamos nuestro documento tendría que tener el siguiente aspecto



Bueno si todo va bien seguimos maquetando nuestro sitio, y se nos presenta el siguiente problema si se han fijado la botonera y el pie de página vendrían siendo algo parecido a la etiqueta `<header> </header>` si te lo imaginaste estas en lo correcto y para ahorrar tiempo y en escribir todo este código nuevamente solo tendremos que agregar a la etiqueta `header` las etiquetas `nav` y `footer` que quedaría de la siguiente manera.

```
header ,nav,footer{
background:#DEDEDE;
border-radius:0.5em;
margin: 0.5em auto;
max-width:960px;
padding:0.25em;
text-align:center;
}
```

Si visualizamos nuestro documento esto quedaría así.



Si vemos a nuestro sitio ya le falta muy poco por maquetar solo el `<section></section>` y el `<aside></aside>` lo cual vamos a explicar de inmediato.

Si nos fijamos en el index creamos un `<section>` con un ID contenedor o sea este `<section id="contenedor">` y otro `<section id="principal">` bueno como como tenemos identificados nuestros section lo que vamos hacer es crear en la hoja de estilo lo estilos correspondiente pero en este caso vamos a ponerle una almohadilla o gato (#) al cual vamos hacer referencias de la siguiente manera `section#contenedor` y `section#principal`.

Primero escribimos el `section#contenedor` y les ponemos un margin de 0.2em y auto a los lados para que se centre, luego `max-width:960px;` les decimos que el tamaño máximo que va a tener este section es de 960px, les ponemos un padding de 0 y un `text-align: center` para que el contenido se centre.

En conclusión nuestro código CSS3 para el section contenedor sería el siguiente

```
section#contenedor{
margin:0.2em auto;
max-width:960px;
padding:0;
text-align:center;
}
```

Ahora seguimos con el `section#principal`. A este le decimos que tenga un `background` o color de fondo un `border-radius` para que tenga esquinas redondeadas, y otro atributo que añada CSS 3 es `display:inline-block;` esto le indica a nuestro `section` que todo lo que

venga dentro del será visto como bloques y les añadimos el atributo `vertical-align:top` este nuevo atributo de CSS3 viene a sustituir al `float` del css anterior esto le indica a nuestro section que alinee todo verticalmente y que sea en la parte superior.

Luego les indicamos que tome un tamaño de un 65% del section contenedor lo otro ya lo sabemos por eso no lo explico y les pongo el siguiente código.

Si notan que puse el `<aside>` en este código es porque quiero que tenga la misma forma que el section principal solo que a este les vamos a dar un tamaño más pequeño en la parte de abajo

```
section#principal, aside{
  background:#F1F1F1;
  border-radius:0.5em;
  display: inline-block;
  margin:0.25em auto;
  max-width:960px;
  min-height:200px;
  padding:0.25em;
  text-align:center;
  vertical-align:top;
  width:65%;
}
```

Si han hecho todo bien nuestro documento iría quedando de la siguiente manera:



Como le explique arriba tenemos que establecerle el porcentaje que va a tomar el `aside` del `section` en este caso con un `30%` quedaría bien mas debajo de ese código escribimos el siguiente

```
aside{
width:30%;
}
```

Ahora si actualizamos nuestro navegador todo se vería de la siguiente manera.



Bueno ahora solo nos falta darle estilo al `article`, este será sencillo solo tenemos que ponerle un color de fondo distinto al fondo actual para ellos les ponemos un `background:#000`; que corresponde al negro, luego un `border-radius` para las esquinas curvas y les ponemos un alto mínimo con el atributo `min-height`: de `140px`; y un `padding`: de `0.25em` con esto quedaría listo nuestro `article`.

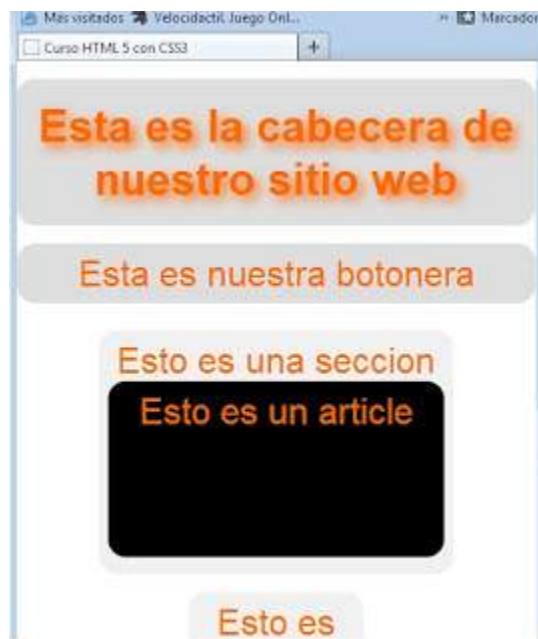
```
article{
background:#011E30;
border-radius: 0.5em;
min-height:140px;
padding:0.25em;
}
```

Si has copiado bien todo el código esta tendría que ser la imagen final de nuestro documento.



Ahora si te fijas en el navegador reduciendo el tamaño de la ventana veras como todo se ajusta al tamaño de la ventana, esto lo obtuvimos porque trabajamos con los tamaños en (em) y no en (px) también porque los atributos max-width y min-height.

Bueno aquí les dejo una toma de mi navegador reducido a lo que simula una pantalla de un celular.



Aquí les dejo los códigos del ejemplo completo.

index.html

```
<!DOCTYPE html/>
<html lang="es">
<head>
  <meta charset="utf-8" />
```



```

<meta name="description"content="Practica de html 5" />
<title>Curso de HTML5 y CSS3   </title>
<link rel="stylesheet" href="stylo.css" />
</head>
<body>
  <header>
    <h1>Esta es la cabecera de nuestro sitio web</h1>
  </header>
  <nav>
    Esta es nuestra botonera
  </nav>
  <section id="contenedor">
    <section id="principal">
      Esto es una seccion
      <article>
        Esto es un article
      </article>
    </section>
    <aside>
      Esto es un aside
    </aside>
  </section>
  <footer>
    Este es el pie de nuestra p&aacute;gina
  </footer>
</body>
</html>

```

stylo.css

```

*{
margin: 0;
padding: 0;
}
@font-face {
  font-weight:normal;
  font-style:normal;
}
body{
  background: #FFF;
  color:#F60;
  font-family:Helvetica, Verdana;
  font-size: 2em;

```



```
}  
h1{  
  color:#F60;  
  margin:0.25em auto;  
  text-align:center;  
  font-size:1.25em;  
  text-shadow: 5px 5px 10px rgba(255,255,255,0.5);  
}  
header,nav,footer {  
  background:#DEDEDE;  
  border-radius:0.5em;  
  margin: 0.5em auto;  
  max-width:960px;  
  padding:0.25em;  
  text-align:center;  
}  
section#contenedor{  
  margin:0.2em auto;  
  max-width:960px;  
  padding:0;  
  text-align:center;  
}  
section#principal, aside{  
  background:#F1F1F1;  
  border-radius:0.5em;  
  display: inline-block;  
  margin:0.25em auto;  
  max-width:960px;  
  min-height:200px;  
  padding:0.25em;  
  text-align:center;  
  vertical-align:top;  
  width:65%;  
}  
article{  
  background:#000;  
  border-radius: 0.5em;  
  min-height:140px;  
  padding:0.25em;  
}  
aside{  
  width:30%;  
}
```

HTML, XHTML y CSS

HTML y XHTML

El lenguaje XHTML es muy similar al lenguaje HTML. De hecho, XHTML no es más que una adaptación de HTML al lenguaje XML. Técnicamente, HTML es descendiente directo del lenguaje SGML, mientras que XHTML lo es del XML (que a su vez, también es descendiente de SGML).

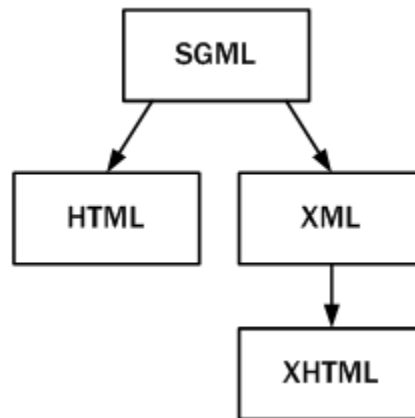


Figura 1.1 Esquema de la evolución de HTML y XHTML

Las páginas y documentos creados con XHTML son muy similares a las páginas y documentos HTML. Las discusiones sobre si HTML es mejor que XHTML o viceversa son recurrentes en el ámbito de la creación de contenidos web, aunque no existe una conclusión ampliamente aceptada.

Hace aproximadamente dos años, Los diseñadores se veían obligados a seleccionar entre HTML 4.01 y XHTML 1.0, la mayoría de diseñadores escogían XHTML. Sin embargo, desde que los navegadores han adoptado las especificaciones de HTML5, han estado cambiando su elección, para aprovechar las bondades de HTML5, aunque no todas las propiedades hayan sido aceptadas oficialmente

HTML y CSS

Originalmente, las páginas HTML sólo incluían información sobre sus contenidos de texto e imágenes. Con el desarrollo del estándar HTML, las páginas empezaron a incluir también información sobre el aspecto de sus contenidos: tipos de letra, colores y márgenes.

La posterior aparición de tecnologías como JavaScript, provocaron que las páginas HTML también incluyeran el código de las aplicaciones (llamadas *scripts*) que se utilizan para crear páginas web dinámicas.

Incluir en una misma página HTML los contenidos, el diseño y la programación complica en exceso su mantenimiento. Normalmente, los contenidos y el diseño de la página web son responsabilidad de diferentes personas, por lo que es conveniente separarlos.

CSS es el mecanismo que permite separar los contenidos definidos mediante HTML/XHTML y el aspecto que deben presentar esos contenidos:

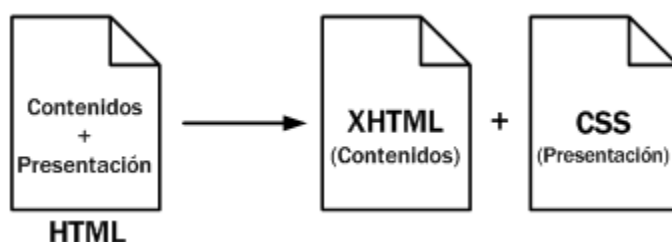


Figura 1.2 Esquema de la separación de los contenidos y su presentación

Una ventaja añadida de la separación de los contenidos y su presentación es que los documentos XHTML creados son más flexibles, ya que se adaptan mejor a las diferentes plataformas: pantallas de ordenador, pantallas de dispositivos móviles, impresoras y dispositivos utilizados por personas discapacitadas.

De esta forma, utilizando exclusivamente XHTML se crean páginas web "*feas*" pero correctas. Aplicando CSS, se pueden crear páginas "*bonitas*" a partir de las páginas XHTML correctas.

Unidad 2 Estructura del CSS

Sintaxis

Sintaxis de la definición de cada propiedad CSS

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

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

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

Si el valor permitido se indica como una sucesión de valores separados por una barra simple (carácter `|`) el valor de la propiedad debe tomar uno y sólo uno de los valores indicados. Por ejemplo, la notación `<porcentaje> | <medida> | inherit` indica que la propiedad solamente puede tomar como valor la palabra reservada `inherit` o un porcentaje o una medida.

Si el valor permitido se indica como una sucesión de valores separados por una barra doble (símbolo `||`) el valor de la propiedad puede tomar uno o más valores de los indicados y en cualquier orden.

Por ejemplo, la notación `<color> || <estilo> || <medida>` indica que la propiedad puede tomar como valor cualquier combinación de los valores indicados y en cualquier orden. Se podría establecer un color y un estilo, solamente una medida o una medida y un estilo. Además, el orden en el que se indican los valores es indiferente. Opcionalmente, se pueden utilizar paréntesis para agrupar diferentes valores.

Por último, en cada valor o agrupación de valores se puede indicar el tipo de valor: opcional, obligatorio, múltiple o restringido.

El carácter `*` indica que el valor ocurre cero o más veces; el carácter `+` indica que el valor ocurre una o más veces; el carácter `?` indica que el valor es opcional y por último, el carácter `{número_1, número_2}` indica que el valor ocurre al menos tantas veces como el valor indicado en `número_1` y como máximo tantas veces como el valor indicado en `número_2`.

Por ejemplo, el valor `[<family-name> ,]*` indica que el valor de tipo `<family_name>` seguido por una coma se puede incluir cero o más veces. El valor `<url>? <color>` significa que la URL es opcional y el color obligatorio y en el orden indicado. Por último, el valor `[<medida> | thick | thin] {1,4}` indica que se pueden escribir entre 1 y 4 veces un valor que sea o una medida o la palabra `thick` o la palabra `thin`.

No obstante, la mejor forma de entender la notación formal para las propiedades de CSS es observar la definición de cada propiedad y volver a esta sección siempre que sea necesario.

Inclusión de CSS en un HTML

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.

Incluir CSS en el mismo documento HTML

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

Ejemplo:

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



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

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

El principal inconveniente es que si se quiere hacer una modificación en los estilos definidos, es necesario modificar todas las páginas que incluyen el estilo que se va a modificar. Los ejemplos mostrados en este libro utilizan este método para aplicar CSS al contenido HTML de las páginas. De esta forma el código de los ejemplos es más conciso y se aprovecha mejor el espacio.

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:

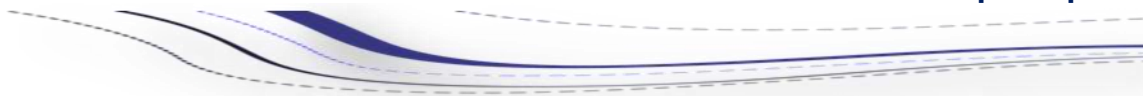
```
p { color: black; font-family: Verdana; }
```

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");
```

Incluir CSS en los elementos HTML

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

Ejemplo:

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



```
</head>

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

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

Unidad 3 Elementos básico

¿Qué es una regla CSS?

Es necesario entender que, sin los estilos que proporciona CSS; un documento HTML no es más que texto plano. Es decir, no existen colores, jerarquización de textos, imágenes de fondo, sombras, redondeados, etc.

Se le llama “regla CSS”, porque al escribir un estilo nuevo, en verdad estamos ordenándole al explorador que, si los selectores que especificamos en el CSS se cumplen, esos objetos seleccionados **deberán obligatoriamente** cambiar su estilo inicial, por el que nosotros hemos definido.

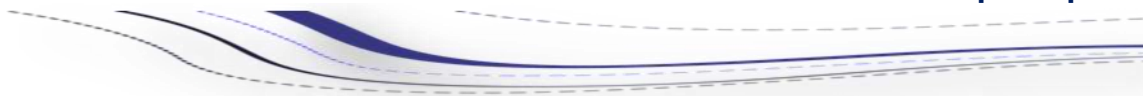
Así, si especificamos que todos los párrafos dentro de un documento deben ser de color gris claro; entonces, cuando un usuario ingrese a la página web, todos los párrafos correspondientes tomarán los estilos definidos en el .CSS.

La composición básica de una regla CSS.

Una regla CSS se compone de un selector y una declaración. El selector es el objeto HTML al cual estamos refiriéndonos con nuestra regla CSS. La declaración –por su parte– es donde definimos que propiedad queremos que se vea afectada por nuestra regla (por ejemplo: color, tamaño, bordes, etc.); y el valor que queremos que tal propiedad tenga.

A continuación, una imagen que puede explicar lo anterior de manera mucho más gráfica:





Como puedes ver en la imagen anterior, para hacer una nueva regla CSS, es necesario seguir una *nomenclatura* específica. Primero, siempre se parte con el selector, luego se abren la declaración con “{“; se define la propiedad; se ponen dos puntos “:”; se especifica el valor numérico o textual de la propiedad que acabamos de nombrar; y se cierra con un punto-coma “;”. Desde aquí, se pueden seguir agregando más reglas de la siguiente forma:

```
h1 {color:#ccc; font-size:12px; background:#000;}.
```

Es decir, basta solo con seguir declarando estilos para las distintas propiedades que queramos cambiar, separándolas con punto y coma “;”, y –finalmente– cerrando todo el conjunto con un corchete “}”.

Medios CSS

CSS tiene una característica genial, la cual nos permite entregar estilos específicos para distintos tipos de dispositivos y pantallas. Estos son los “Medios CSS”.

Por dar un ejemplo del funcionamiento de los Medios CSS: imagínate que tu sitio web provee a los internautas con letras de sus canciones favoritas. Sería lógico pensar que – en algún momento– algún usuario intentará imprimir algunas de las lyrics desde tu sitio web. En tal momento, todo aquel esfuerzo que hiciste por darle colores, tamaños, animaciones, etc. a tu sitio web; no valdrá la pena, pues el usuario no querrá gastar tinta en aquellos detalles.

Es aquí donde los medios CSS entran en juego, pues nos permiten entregar estilos únicos, según se esté viendo en una pantalla, en un proyector, en una impresora, en un teléfono, etc.

A continuación se entrega una pequeña tabla con los distintos Medios CSS, y su explicación correspondiente.

Medio	Descripción
all	Las reglas CSS se aplicarán a cualquier dispositivo.
Braille	Estas reglas solo serán visibles por dispositivos táctiles basado en el sistema Braille
embosed	En este caso, para las impresoras Braille
handheld	Dispositivos de mano, como celulares, PDA, etc. Sin embargo, hoy en día –con CSS3 se utilizan otras reglas–. Las explicaremos con más detalle en un tutorial futuro.
print	Impresoras, y exploradores que estén en modo “vista previa para imprimir”
projection	Para proyectores digitales.
screen	Estos estilos serán válidos en cualquier pantalla de computador.
speech	Estos son para dispositivos especializados en personas discapacitadas
tty	Dispositivos textuales limitadas (teletipos, terminales de texto, etc.)
tv	Televisores y dispositivos con baja resolución.

Existen muchos medios CSS, sin embargo, lo más probable es que los que utilices más en un futuro sean “**screen**” y “**print**”. Además, con las nuevas especificaciones CSS3, se están generando muchos cambios en los Medios CSS, y están apareciendo unos nuevos tipos de medios, llamados “Media Query”.

¿Cómo llamamos a los “medios CSS” en nuestro documento?

Podemos utilizar la regla `@media`, dentro del archivo CSS, de la siguiente forma:

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

Como puedes ver, dentro del mismo archivo `.css`, podemos seccionar por dispositivo, como se mostrarán nuestros estilos; pero también hay otra forma. Podemos utilizar los llamados “import”, sin embargo, no se recomienda utilizar esta forma, a menos de que se sepa expresamente lo que se está haciendo:

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

Estos `@import` también van dentro del archivo CSS contenedor. Lo que `@import` hace exactamente, es buscar el archivo `.css` que hemos definido, para luego importar los estilos contenidos dentro de tal archivo, al CSS actual (al que hace el llamado `@import` propiamente tal).

Finalmente, podemos definirlo en el header del documento madre (y no en los CSS), de la siguiente forma:

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

Comentarios

Los comentarios son pequeños “trozos de texto”, dejados por el desarrollador o diseñador; para su propio entendimiento del documento. Esto es especialmente útil cuando se trabaja con documentos grandes; pues permite seccionar nuestros estilos de manera más legible.

Por ejemplo, podríamos “separar” mediante comentarios el área de estilos que se corresponden solamente con los comentarios de nuestra web, y otra área dentro del CSS separada con comentarios que especifiquen que son para nuestras portadas principales.

De esta forma, podemos llevar un orden y jerarquías y así no perdernos. O –incluso– si trabajamos con otros diseñadores, al menos le aliviaremos el trabajo de entender nuestro desarrollo.

- Los comentarios comienzan con `/*`
- Terminan con `*/`
- No se pueden poner comentarios dentro de otro comentario.
- Los comentarios pueden ocupar todas las líneas que el diseñador estime necesario, pero deben estar contenidas dentro de las etiquetas `“/*”` y `“*/”`

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

Uso de Selectores básicos

En párrafos anteriores pudimos ver que una regla CSS se compone de un selector y una declaración. El selector es imprescindible, pues es quien especificará a que objetos de nuestro documento, se le aplicarán los estilos deseados.

En palabras simples: la declaración es “lo que va a suceder”, y el selector es “a quien le va a suceder”. Siguiendo esta lógica, una declaración simple podría ser que “el o los objetos deseados, tengan un fondo de color rojo”, y el selector podría encargarse de que esta regla se aplique a todos los párrafos de nuestro documento. De esta forma, todos los objetos de párrafo en nuestro documento, tendrán fondo rojo.

Las reglas CSS son bastante flexibles. Una misma regla puede ser aplicada a infinitos selectores, y a un mismo selector se le pueden aplicar todas las reglas CSS que necesitemos. Incluso podemos definir que una regla se aplique a **todo** nuestro documento, es decir, a todos los objetos que compongan nuestra página web. Esto es especialmente útil, por ejemplo, para elementos que se vayan a repetir en nuestro sitio. Por ejemplo: quizá queremos que todos los links de nuestra web sean de color verde, para diferenciarlo del resto del párrafo. Para esto, aplicaríamos una regla “color:green”, a un selector “a”, el cual vendría a especificar “todos los anchors (links) del documento”:

```
a {color:green;}
```

Tipos de selectores.

CSS define casi una docena de selectores de donde elegir, pero –desafortunadamente–, internet Explorer, uno de los exploradores más utilizado por los internautas (incluso hoy en día), no soporta muchos de estos. Sin embargo, la mayoría de exploradores más modernos si los soportan (véase Firefox, Google Chrome, Safari y Opera, entre otros).

Selector universal (*)

Este selector se encarga de entregar estilos a **cualquier** objeto que encuentre en su camino. Se utiliza mediante un asterisco (*), y no se suele utilizar con frecuencia, pues es poco probable que –efectivamente– utilizemos un tipo de estilos para tantos selectores a la vez.

Sin embargo, existen un par de situaciones en las que sí podrían ser de utilidad. Por ejemplo: quizá querríamos que todos los elementos de nuestro sitio web partieran con un tamaño base de “13px” (trece pixeles), como una forma de estandarizar tamaños, y luego –solo si lo necesitamos– cambiar los tamaños de objetos únicos. Esto lo llevaríamos a cabo de la siguiente forma:

```
body * {font-size:13px}
```

En el ejemplo anterior, estamos especificando que dentro del body de nuestro documento, todos los objetos tengan un estilo base de “font-size:13px”, o tamaño de tipografía de trece pixeles.

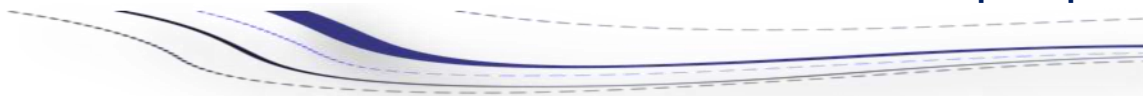
Selector de tipo, o etiqueta.

Este es el tipo más común de selector. Se utiliza para seleccionar para entregarle estilos a cualquier objeto que coincida con la etiqueta HTML que especifiquemos. Por ejemplo, para darle estilos a todos los párrafos, utilizaríamos lo siguiente:

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

Sin embargo, ¿Qué pasa cuando queremos darle un mismo estilo a varias etiquetas HTML al mismo tiempo? ¿Acaso tendríamos que repetir la etiqueta tantas veces, según etiquetas haya? Afortunadamente, CSS nos ofrece la posibilidad de “encadenar” estilos, a varios elementos al mismo tiempo, de la siguiente forma:

```
h1, h2, h3, p {color:black;}
```

Como pueden ver, le estamos dando el mismo estilo “color negro” a todos los elementos h1, h2, h3 y p. La forma de realizarlo, es –simplemente– separando los selectores por una coma (,).

Selector descendiente.

El selector descendiente se utiliza para encontrar objetos dentro de otros objetos. Por ejemplo, si quisiéramos darle un estilo a cualquier link dentro de algún párrafo, usaríamos lo siguiente:

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

Es decir, se inicia la selección con el “elemento HTML padre” (el que contiene a otro objeto), y luego –separados por un espacio–, el “elemento HTML hijo” (el objeto contenido). Claro que esta regla no discrimina si el objeto es hijo directo, o no, del elemento padre; por lo que todos los links que haya dentro de un p se verán afectados por el estilo, sin importar que hayan links dentro de otros links, etc.

Por ejemplo: quizá queremos que solo el primer link dentro de nuestro párrafo sea de color negro, pero que el resto sea de color rojo. Siguiendo la lógica anterior, quizá podríamos hacer algo como esto:

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

Desafortunadamente, esto es incorrecto, pues lo único que sucederá, será que el segundo estilo “sobre-escribirá” al primero, al tratarse de la misma regla, aplicada a los mismos objetos.

CSS no está discriminando si nos referimos al primero, o segundo, o tercer link dentro de un párrafo, sino que entiende que (los estilos) serán aplicados a todos y cada uno de los links que estén dentro de un párrafo. Claro que es posible ser más específico a la hora de dar estilos a nuestro documento, para que no ocurra lo anterior; pero lo iremos viendo de a poco.

Primero, es necesario que entiendas que es posible utilizar varios selectores descendientes seguidos, sin problemas. Por ejemplo, quizá nuestra regla CSS se aplicará a cualquier elemento link, que esté dentro de un li, el cual –a su vez– esté dentro de un ul, el cual –también– esté dentro de un div. La regla quedaría de la siguiente forma:

```
div ul li a {color:black;}
```

No debe confundirse lo anterior, con la separación de elementos HTML con una coma (,). Si se separan con una coma, estamos diciéndole a CSS, que el estilo se aplicará a todos los elementos separados por coma. Pero si lo separamos con espacios, le estamos diciendo que los estilos deben afectar a los elementos dentro de otros objetos HTML. Eje:

```
/* La sig. regla se aplica a todo link que esté dentro de algún li, que
esté dentro de un ul, que esté dentro de un div: */
div ul li a {color:black;}
/* Pero la siguiente regla se aplica a todos los elementos div, a todos los
elementos ul, a cada elemento li, y a todos los links de nuestro documento:
*/
div, ul, li, a {color:black;}
```

Podemos combinar selectores explicados anteriormente, con este. Por ejemplo, en el siguiente ejemplo veremos una forma de ser más específicos utilizando el selector universal (*):

```
/* El siguiente ejemplo se aplica a todos los links dentro de un párrafo */
p a {color:black;}

/* Pero el siguiente ejemplo solo aplicará el estilo al segundo párrafo,
pues estamos especificando que se le den estilos a "todo link, dentro de
cualquier elemento HTML, dentro de un párrafo. En este caso, el único que
cumple la regla es el segundo párrafo (a continuación): */
p * a {color:black;}

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

Selector de clase (.)

Podemos ser todavía más específicos, si utilizamos clases para dar estilos a nuestro documento. Este es –probablemente– el selector más utilizado (y sobre-utilizado, según mi punto de vista), de las reglas CSS. Se utiliza mediante el símbolo **punto (.)**, seguido del nombre de la clase.

El selector de clase, nos permite darle estilos a todo elemento HTML que tenga el atributo “class” correspondiente. Por ejemplo: podríamos definir una clase llamada “link_verde”, la cual contendría el estilo “color:green;”. Ahora, solo bastaría con darle el atributo *class* correspondiente, al tag HTML que queramos que se vea afectado por tal estilo:

```
.verde {color:green;}  
< span >Texto 1< /span >  
< span >Texto 1< /span >  
< span class="verde" >Texto 1< /span >  
< span >Texto 1< /span >  
< span >Texto 1< /span >
```

En el ejemplo anterior, solo el tercer elemento span se verá afectado, pues es el único que tiene el atributo class:“verde”. También podemos ser más específicos, y dar distintos estilos a una misma clase, según el elemento que tenga el atributo:

```
p.verde {color:green;}  
div.verde {background:green;}
```

En el ejemplo anterior, estamos utilizando la misma clase, pero estamos generando dos estilos distintos. Esto es porque le estamos diciendo al explorador que, si ve un párrafo con la clase “verde”, le dé un color verde al texto. Pero si –en cambio– es un div el que tiene la clase “verde”; este cambiará su fondo (background) a verde, en vez del texto. ¡Todos gracias a los selectores de clase!

Para llevar a cabo lo anterior, es necesario entender que el selector de elemento HTML debe ir unido al selector de clase (es decir, sin separarlos por coma, o un espacio, pues quieren decir cosas totalmente distintas):

```
/* Selecciona cualquier párrafo que tenga clase "verde" */  
p.verde {color:green;}  
  
/* Selecciona objeto que tenga la clase "verde" dentro de algún párrafo del documento */  
p .verde {background:green;}  
  
/* Selecciona todos los párrafos, y todos los elementos HTML con la clase "verde" */  
p, .verde {background:green;}
```

Selectores de id (#):

Los selectores de id son similares a los de clase, pero se llaman mediante el uso del símbolo “gato” (#). La diferencia con “class”, recae en que solo se puede asignar a un elemento en todo nuestro documento. Es decir, no puede haber más de un elemento con una id específica. Es por eso que, este selector no es utilizado frecuentemente por los desarrolladores web profesionales actualmente.

Generalmente se utiliza id para generar funcionalidades complejas mediante código (por ejemplo, con javascript); sin embargo, CSS nos permite utilizarlo de todos modos.

El siguiente ejemplo le entregaría el estilo al elemento con la clase “unico” dentro de nuestro documento:

```
#unico {color:black;}
```

Selectores avanzados

Utilizando los selectores especificados hasta el momento, debería ser más que suficiente para diseñar un sitio web de complejidad media-básica. Sin embargo, para sitios web más complejos (entiéndase como portales, aplicaciones web, comunidades o foros web, etc.), quizá sea necesario tener más funcionalidades, y mejores formas de especificar estilos sin dejar espacio a la equivocación.

Para tales casos, utilizamos los selectores avanzados. Pero esto lo explicaremos más adelante.

Además, veremos en mayor profundidad, que es “Inheritance”, y porque es 100% necesario que lo dominemos, antes de adentrarnos en el uso de CSS para sitios más complejos.

Unidad 4 Colores y Fondos

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.

Palabras clave

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

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

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

Figura 3.1 Colores definidos mediante las palabras clave de CSS

La imagen anterior ha sido extraída de la [sección sobre colores de la especificación oficial de CSS](#).

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



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

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.

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%.

RGB hexadecimal

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

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

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

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

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

hexadecimal, se concatenan los valores hexadecimales de las componentes RGB en ese orden y se les añade el prefijo #. De esta forma, el color del ejemplo anterior es #4762B0 en formato RGB hexadecimal.

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

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

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

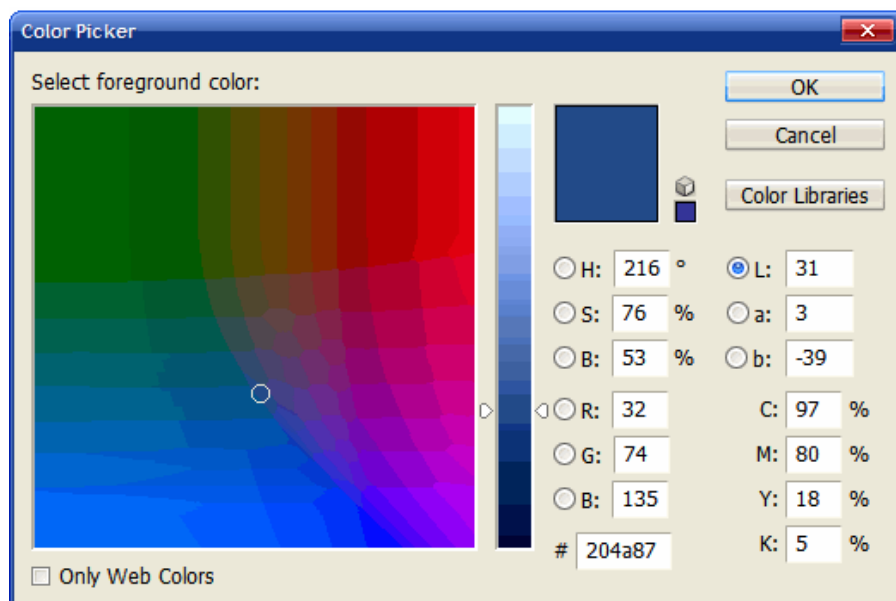



Figura 3.2 Herramienta de color de Photoshop para definir los colores según los modelos RGB, CMYK, Lab, HSB y RGB hexadecimal

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

#AAA = #AAAAAA

#FFF = #FFFFFF

#A0F = #AA00FF



#369 = #336699

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

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

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

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.

Colores web safe

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

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

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



auge en el uso de los dispositivos móviles hace que siga siendo un tema a considerar, ya que las pantallas de muchos móviles sólo pueden representar un número reducido de colores.

Consulta la [lista completa de colores web safe y sus valores hexadecimales](#).

Unidad 5 Texto

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

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

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

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

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

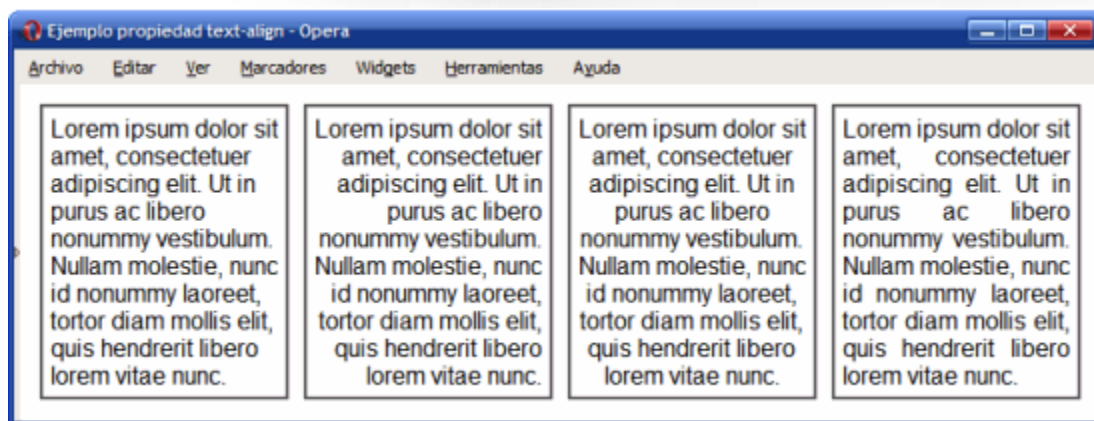


Figura 6.5 Ejemplo de propiedad text-align

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

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

Propiedad	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-height** permite indicar un número sin unidades que se interpreta como el múltiplo del tamaño de letra del elemento. Por lo tanto, estas tres reglas CSS son equivalentes:

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

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



Figura 6.6 Ejemplo de propiedad line-height

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

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

Propiedad	text-transform
Valor inicial	none
Descripción	Transforma el texto original (lo transforma a mayúsculas, a minúsculas, etc.)

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

La siguiente imagen muestra cada uno de los posibles valores:

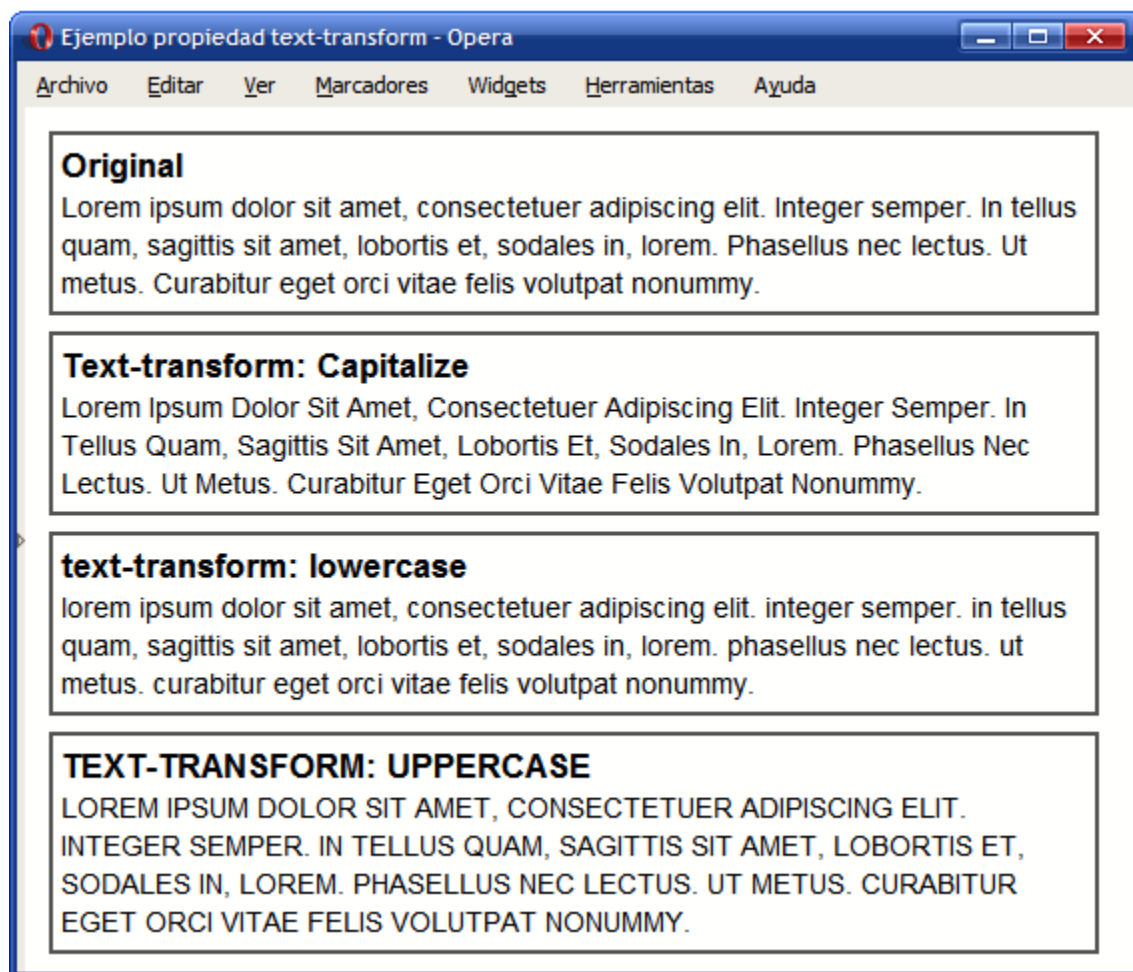


Figura 6.7 Ejemplo de propiedad text-transform



Las reglas CSS del ejemplo anterior se muestran a continuación:

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

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

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

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

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

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`:



Figura 6.8 Ejemplo de propiedad vertical-align

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

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

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

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

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

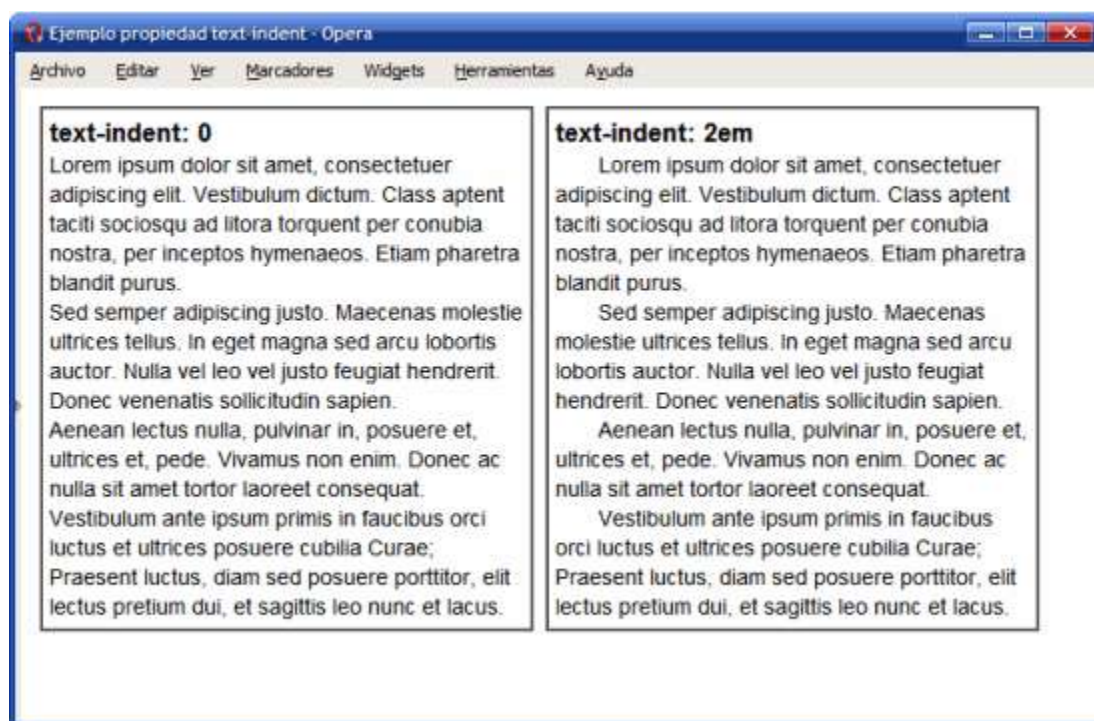


Figura 6.9 Ejemplo de propiedad text-indent

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

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

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

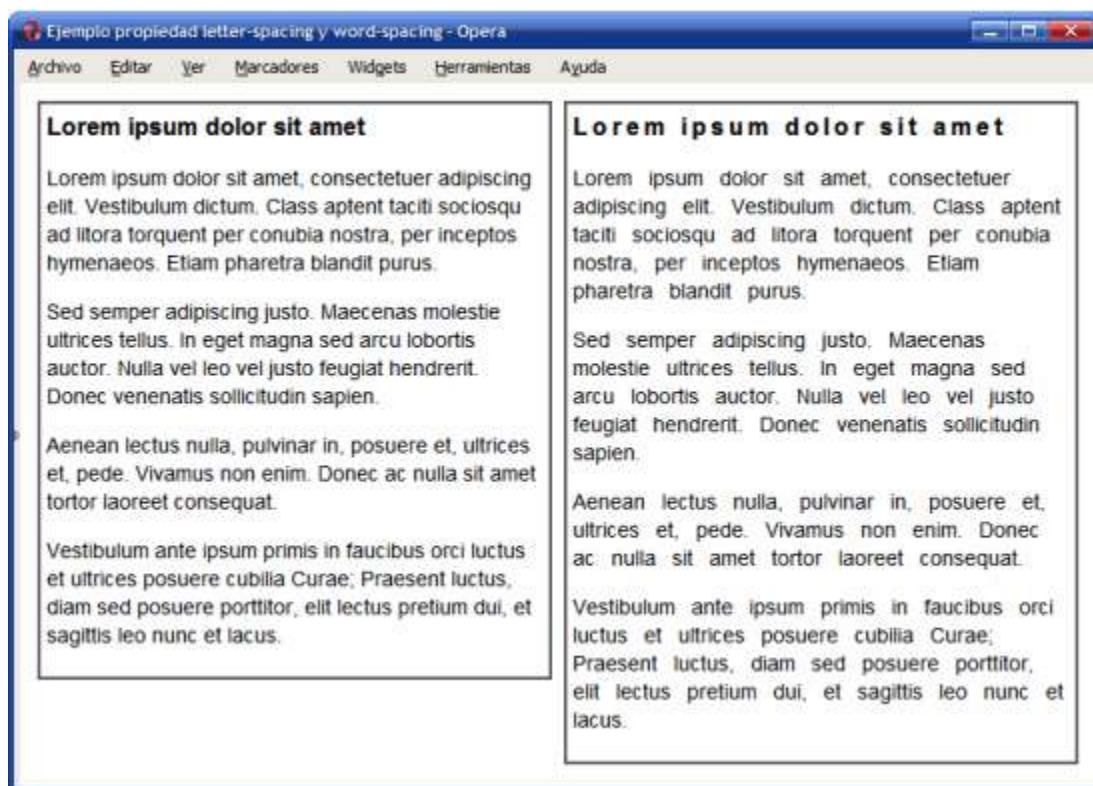


Figura 6.10 Ejemplo de propiedades `letter-spacing` y `word-spacing`

Las reglas CSS del ejemplo anterior se muestran a continuación:

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

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

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



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

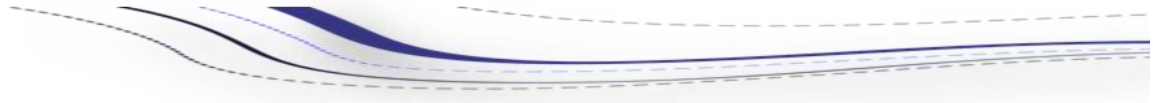
Como ya se sabe, el tratamiento que hace HTML de los espacios en blanco es uno de los aspectos más difíciles de comprender cuando se empiezan a crear las primeras páginas web. Básicamente, HTML elimina todos los espacios en blanco sobrantes, es decir, todos salvo un espacio en blanco entre cada palabra.

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

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

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





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

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

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

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



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

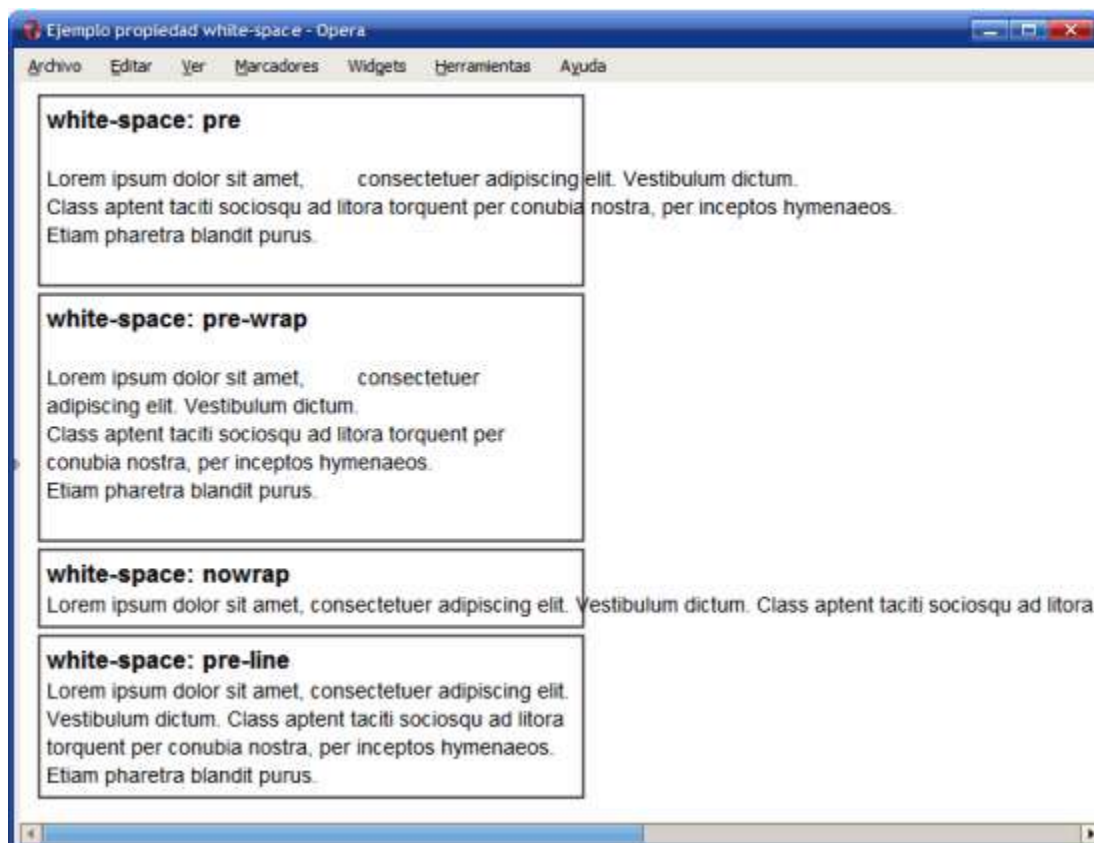


Figura 6.11 Ejemplo de propiedad `white-space`

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

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

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

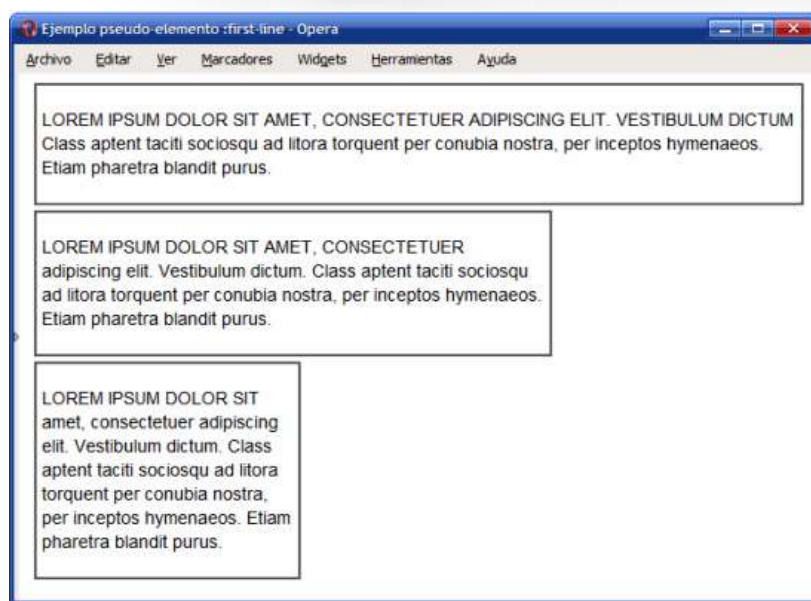


Figura 6.12 Ejemplo de pseudo-elemento first-line

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

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

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

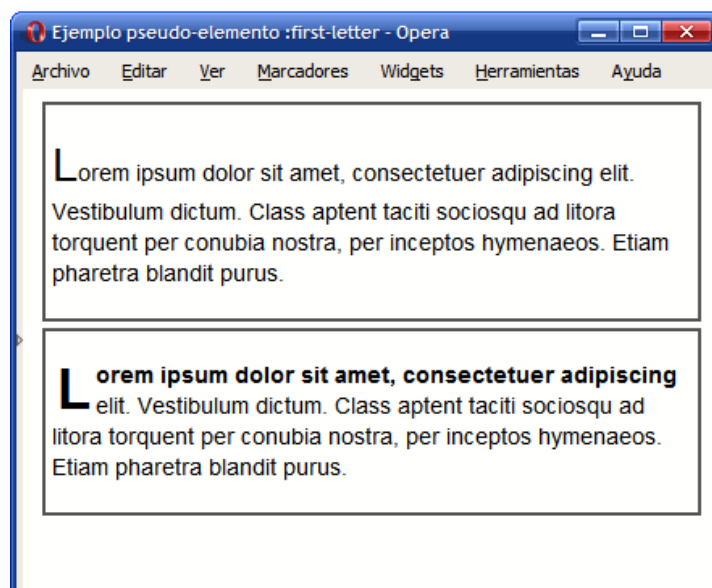


Figura 6.13 Ejemplo de pseudo-elemento first-letter

El código HTML y CSS se muestra a continuación:

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



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

Unidad 6 Fuentes

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

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

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

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

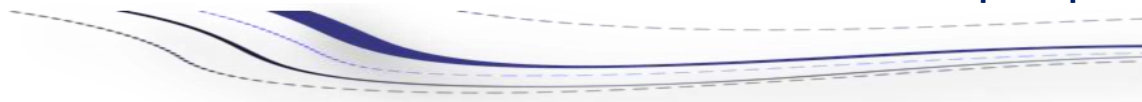
Para evitar el problema común de que el usuario no tenga instalada la fuente que quiere utilizar el diseñador, CSS permite indicar en la propiedad **font-family** más de un tipo de letra. El navegador probará en primer lugar con el primer tipo de letra indicado. Si el usuario la tiene instalada, el texto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

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

Para solucionar este problema se utilizan las familias tipográficas genéricas. Cuando la propiedad **font-family** toma un valor igual a **sans-serif**, el diseñador no indica al navegador que debe utilizar la fuente Arial, sino que debe utilizar *"la fuente que más se parezca a Arial de todas las que tiene instaladas el usuario"*. Por todo ello, el valor de **font-family** suele definirse como una lista de tipos de letra alternativos separados por comas. El último valor de la lista es el nombre de la familia tipográfica genérica que más se parece al tipo de letra que se quiere utilizar.

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

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



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

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

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

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

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.

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

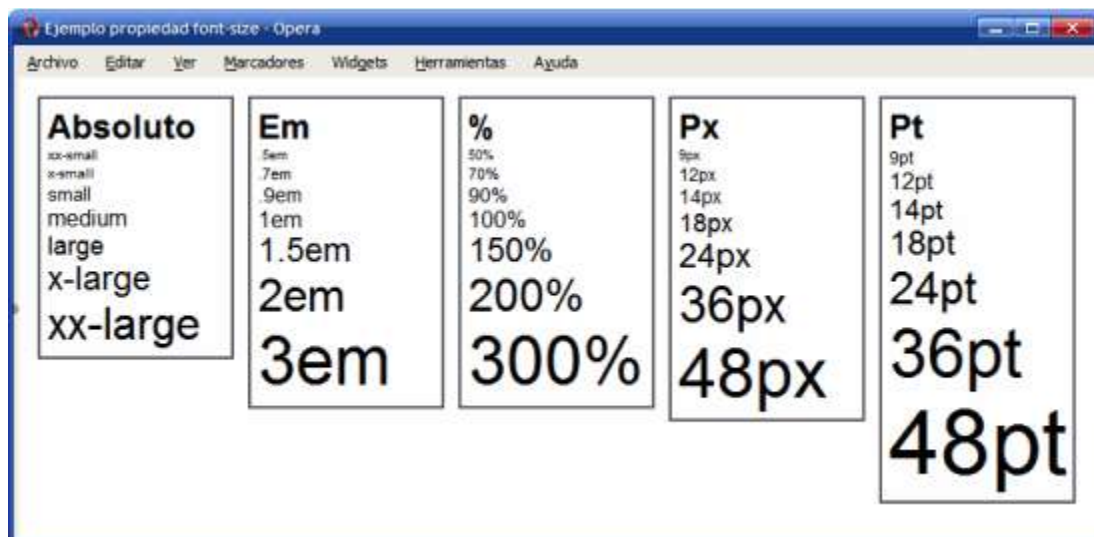


Figura 6.1 Comparación visual de las distintas unidades para indicar el tamaño del texto

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

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

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

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

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

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

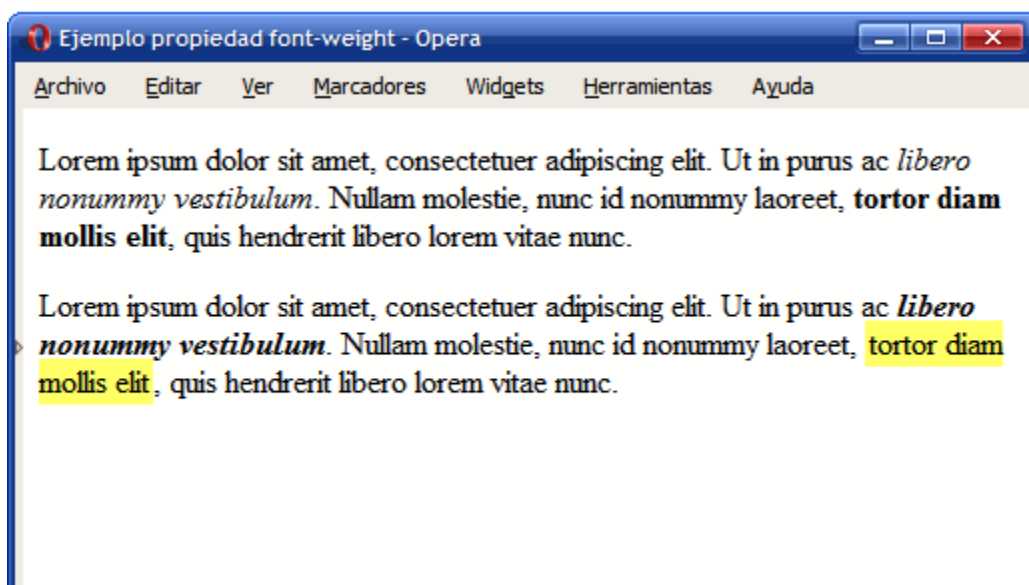


Figura 6.2 Ejemplo de propiedad font-weight

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

Las reglas CSS del ejemplo anterior se muestran a continuación:

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

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

Propiedad	font-style
Descripción	Establece el estilo de la letra utilizada para el texto

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

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

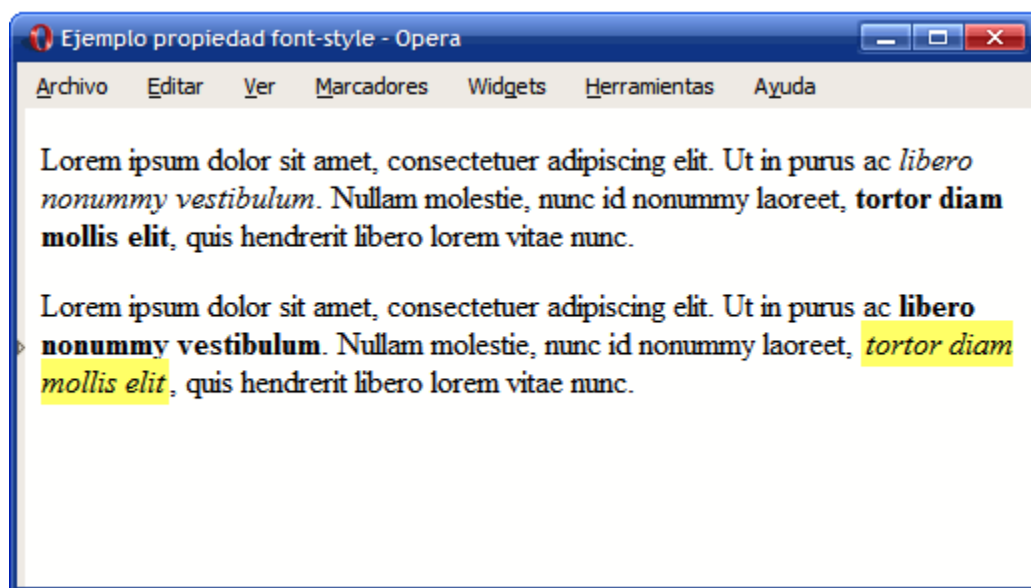



Figura 6.3 Ejemplo de propiedad font-style

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

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

```
#especial em {
  font-weight: bold;
  font-style: normal;
}
#especial strong {
```



```
font-weight: normal;  
font-style: italic;  
background-color:#FFFF66;  
padding: 2px;  
}
```

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

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

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

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



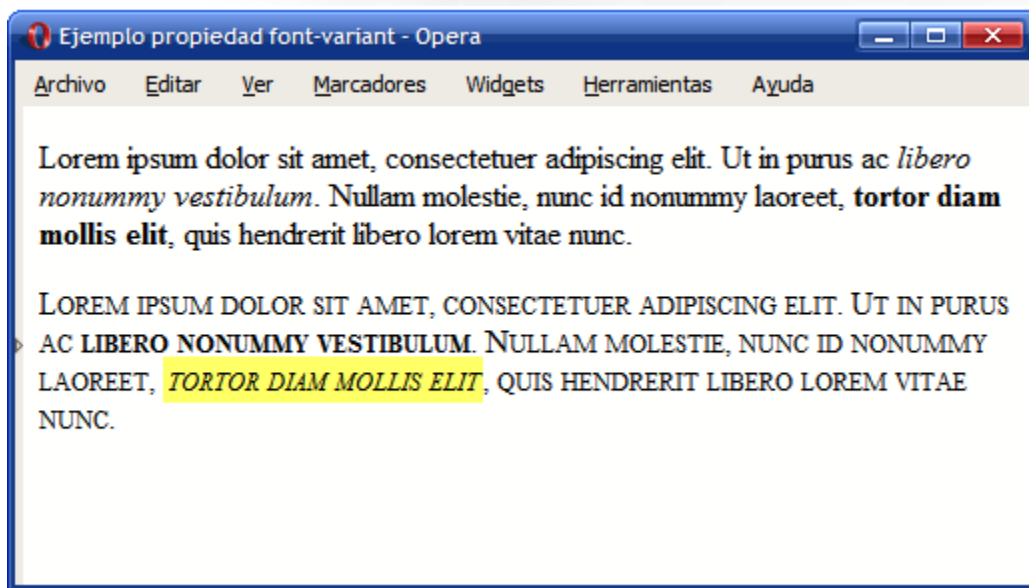


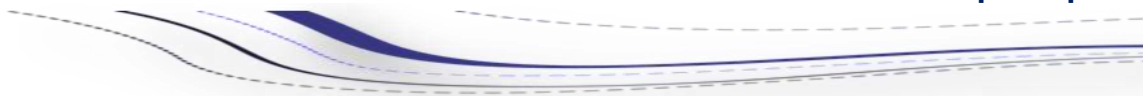
Figura 6.4 Ejemplo de propiedad font-variant

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

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

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

Propiedad	font
Valores	((font-style font-variant font-weight)? font-size (/ line-height)? font-family) caption icon menu message-box small-caption status-bar inherit
Se aplica a	Todos los elementos
Valor inicial	-



Propiedad	font
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.

Unidad 7 Ligas

Estilos básicos de Ligas

Tamaño, color y decoración

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



Figura 7.1 Ejemplo de enlaces con estilos diferentes

A continuación se muestran las reglas CSS del ejemplo anterior:

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

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

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

Pseudo-clases

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

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

- `:link`, aplica estilos a los enlaces que apuntan a páginas o recursos que aún no han sido visitados por el usuario.
- `:visited`, aplica estilos a los enlaces que apuntan a recursos que han sido visitados anteriormente por el usuario. El historial de enlaces visitados se borra automáticamente cada cierto tiempo y el usuario también puede borrarlo manualmente.
- `:hover`, aplica estilos al enlace sobre el que el usuario ha posicionado el puntero del ratón.
- `:active`, aplica estilos al enlace que está pinchando el usuario. Los estilos sólo se aplican desde que el usuario pincha el botón del ratón hasta que lo suelta, por lo que suelen ser unas pocas décimas de segundo.

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

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

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

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



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

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

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

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

/* Correcto: la pseudo-clase modifica el comportamiento del elemento <a> */
a:visited { ... }
```

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

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

Cuando se aplican varias pseudo-clases diferentes sobre un mismo enlace, se producen colisiones entre los estilos de algunas pseudo-clases. Si se pasa por ejemplo el ratón por encima de un enlace visitado, se aplican los estilos de las pseudo-clases `:hover` y

:visited. Si el usuario pincha sobre un enlace no visitado, se aplican las pseudo-clases **:hover**, **:link** y **:active** y así sucesivamente.

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

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

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

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

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

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

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

Estilos avanzados

Decoración personalizada

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

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

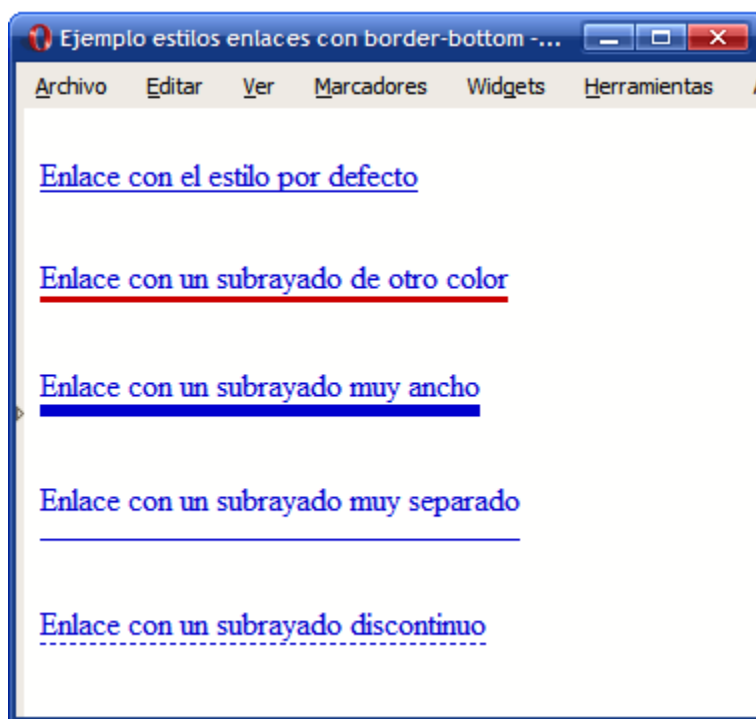


Figura 7.3 Enlaces con subrayado personalizado mediante la propiedad `border`

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

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

<a class="simple" href="#">Enlace con el estilo por defecto</a>
<a class="color" href="#">Enlace un subrayado de otro color</a>
<a class="ancho" href="#">Enlace con un subrayado muy ancho</a>
<a class="separado" href="#">Enlace con un subrayado muy separado</a>
<a class="discontinuo" href="#">Enlace con un subrayado discontinuo</a>
```

Imágenes según el tipo de enlace

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

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

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

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

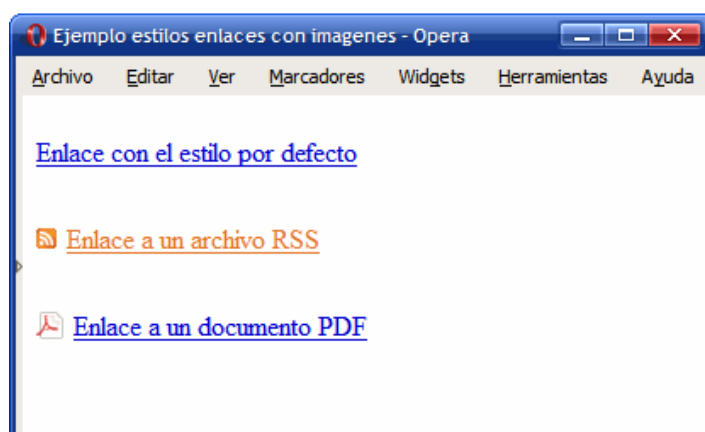


Figura 7.4 Personalizando el aspecto de los enlaces en función de su tipo

Las reglas CSS necesarias se muestran a continuación:

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

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

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

```
}
```

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

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

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

Mostrar los enlaces como si fueran botones

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

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

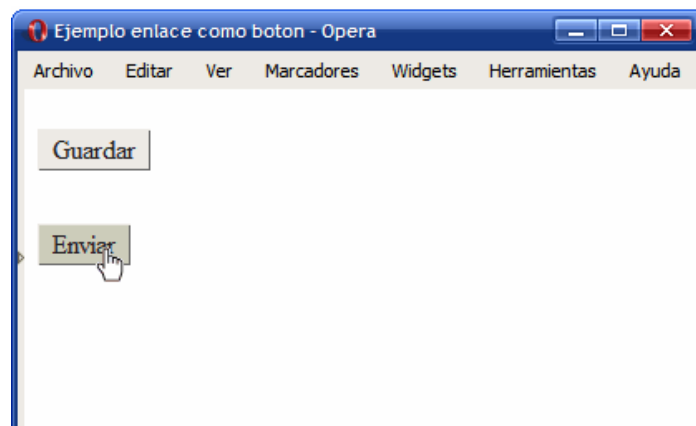


Figura 7.5 Mostrando un enlace como si fuera un botón

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

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

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

Tablas

Estilos básicos de Tablas

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

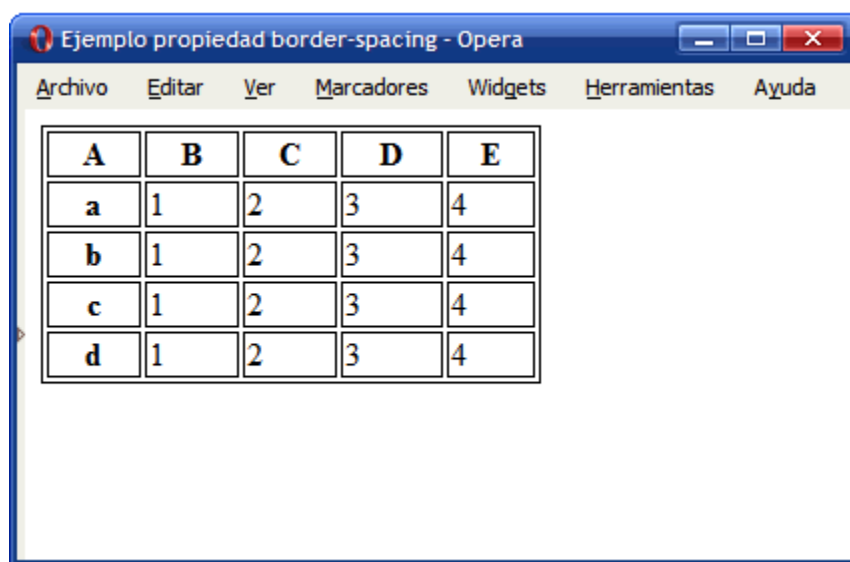


Figura 10.1 Aspecto por defecto de los bordes de una tabla

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

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

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

```

</tr>
...
</table>

```

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

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

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

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





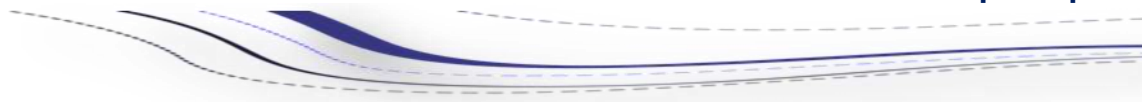
Figura 10.2 Ejemplo de propiedad border-collapse

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

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

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

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



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

Propiedad	<code>border-spacing</code>
Valores	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`:



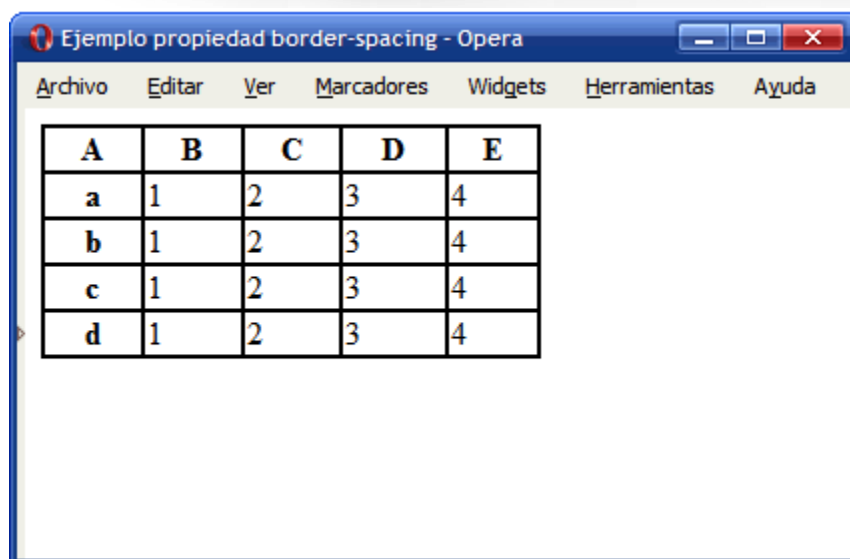


Figura 10.3 Ejemplo de propiedad border-spacing

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

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

Estilos avanzados

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

Propiedad	<code>empty-cells</code>
Valores	<code>show</code> <code>hide</code> inherit

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

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

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

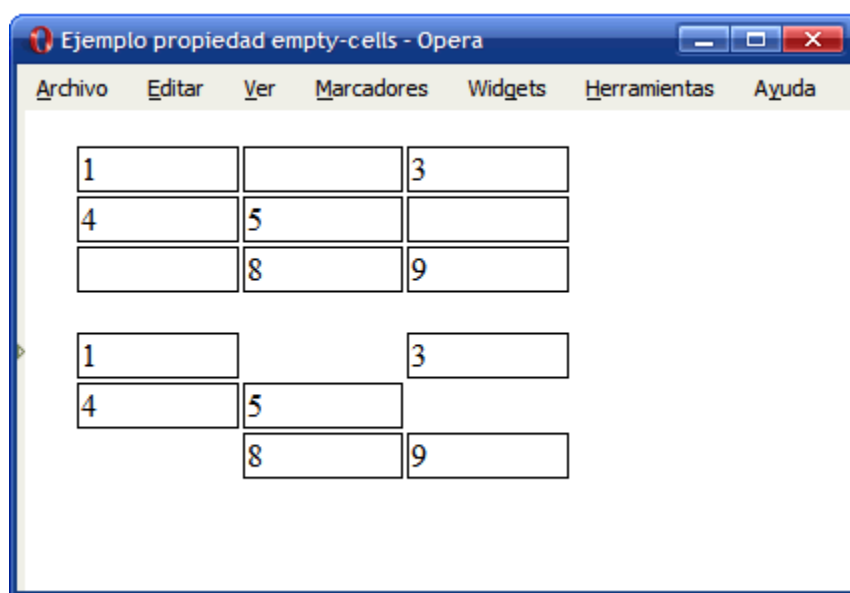


Figura 10.4 Ejemplo de propiedad empty-cells

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

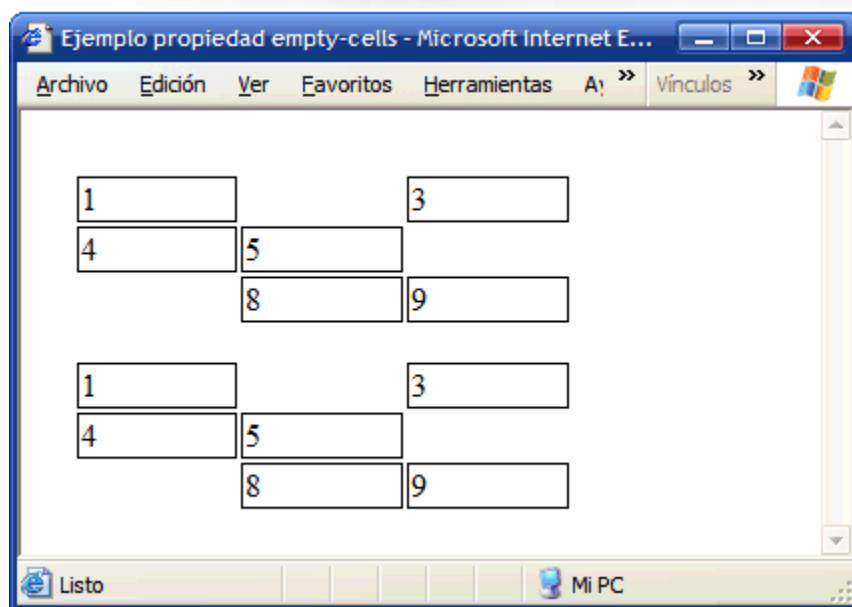


Figura 10.5 Internet Explorer no soporta la propiedad `empty-cells`

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

Propiedad	<code>caption-side</code>
Valores	<code>top</code> <code>bottom</code> inherit
Se aplica a	Los elementos <code>caption</code>
Valor inicial	<code>top</code>
Descripción	Establece la posición del título de la tabla

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

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

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

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

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

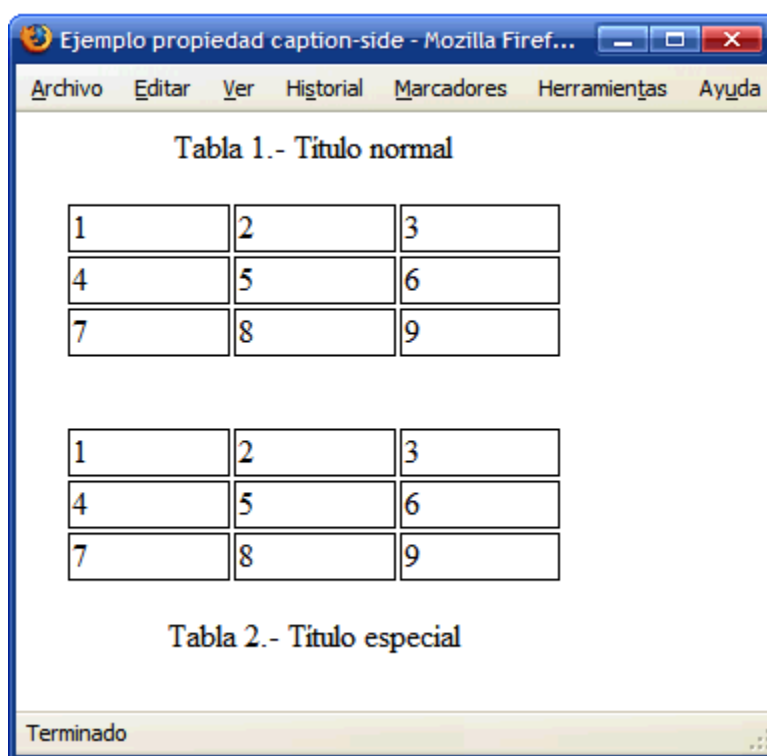


Figura 10.6 Ejemplo de propiedad `caption-side`

Formas

Estilos básicos de Formularios

Mostrar un botón como un enlace

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

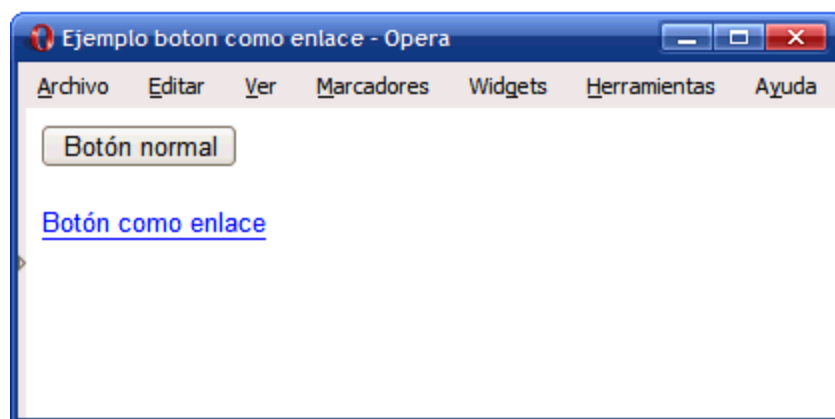


Figura 11.1 Mostrando un botón como si fuera un enlace

Las reglas CSS del ejemplo anterior son las siguientes:

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

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

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

Mejoras en los campos de texto

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

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

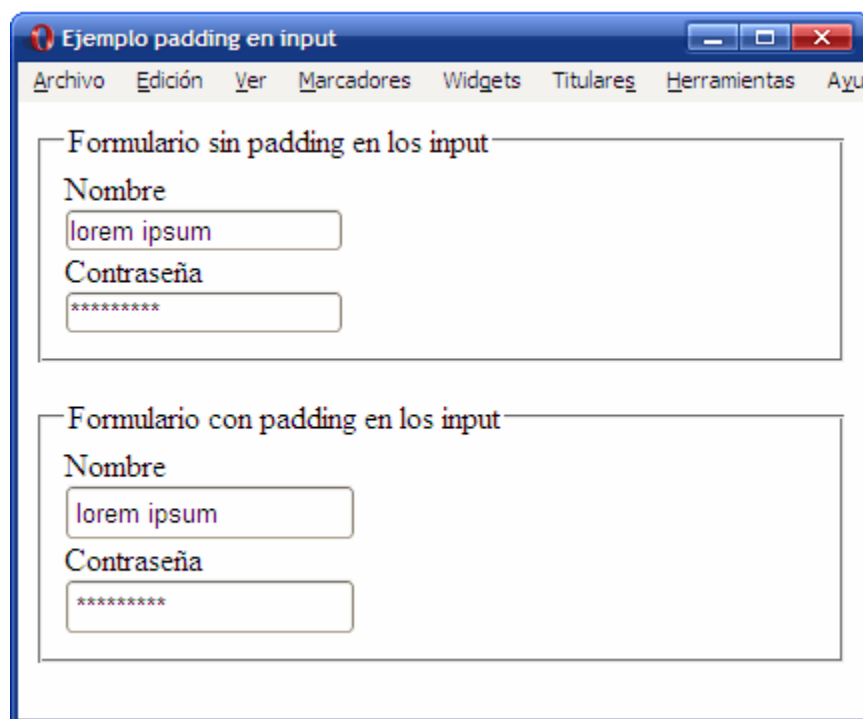


Figura 11.2 Mejorando el aspecto de los formularios gracias al padding

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

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

11.1.3. Labels alineadas y formateadas

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

Figura 11.3 Aspecto por defecto que muestran los formularios

El código HTML del ejemplo anterior es el siguiente:

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

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

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

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

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

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

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

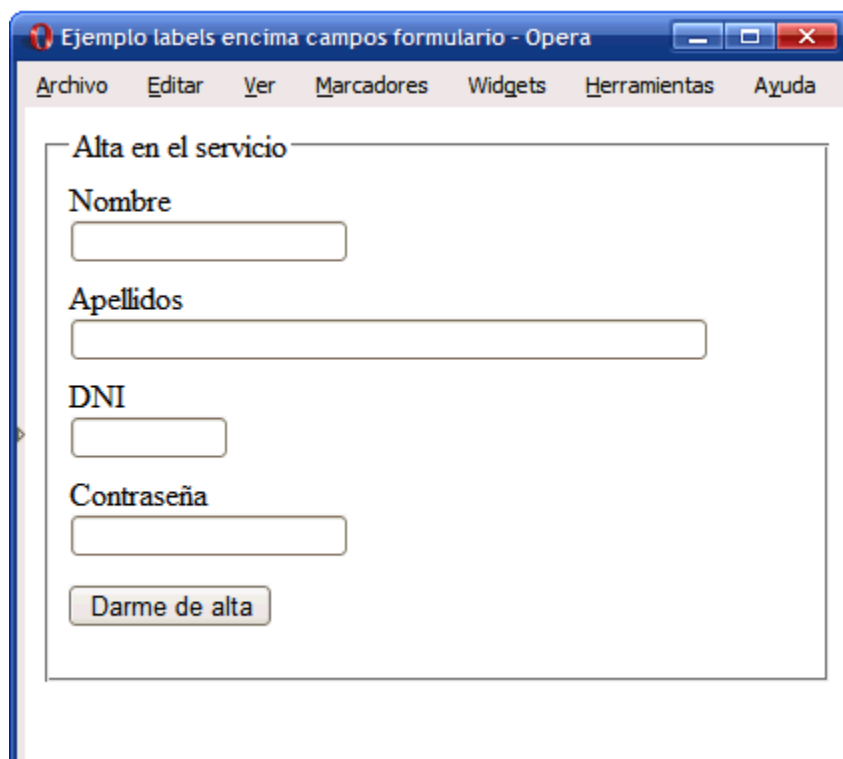


Figura 11.4 Mostrando las etiquetas label encima de los campos del formulario

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

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

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

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


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

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

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

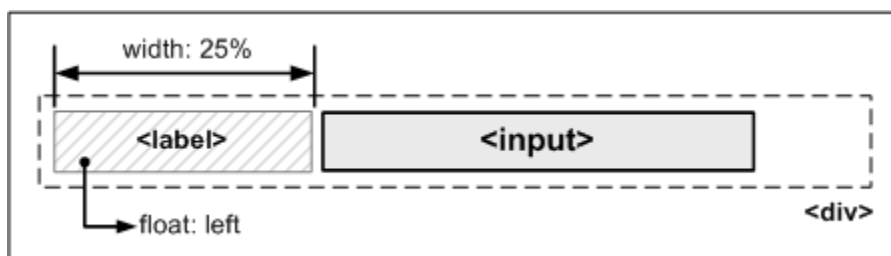
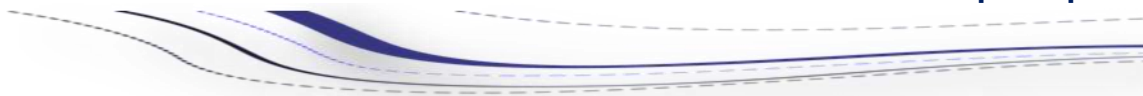


Figura 11.6 Esquema de la técnica de alineación de etiquetas label y campos de formulario

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

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

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



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

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

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

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

Estilos avanzados de Formularios

Formulario en varias columnas

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

Figura 11.7 Ejemplo de formulario a dos columnas

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

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

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

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

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

Resaltar el campo seleccionado

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

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

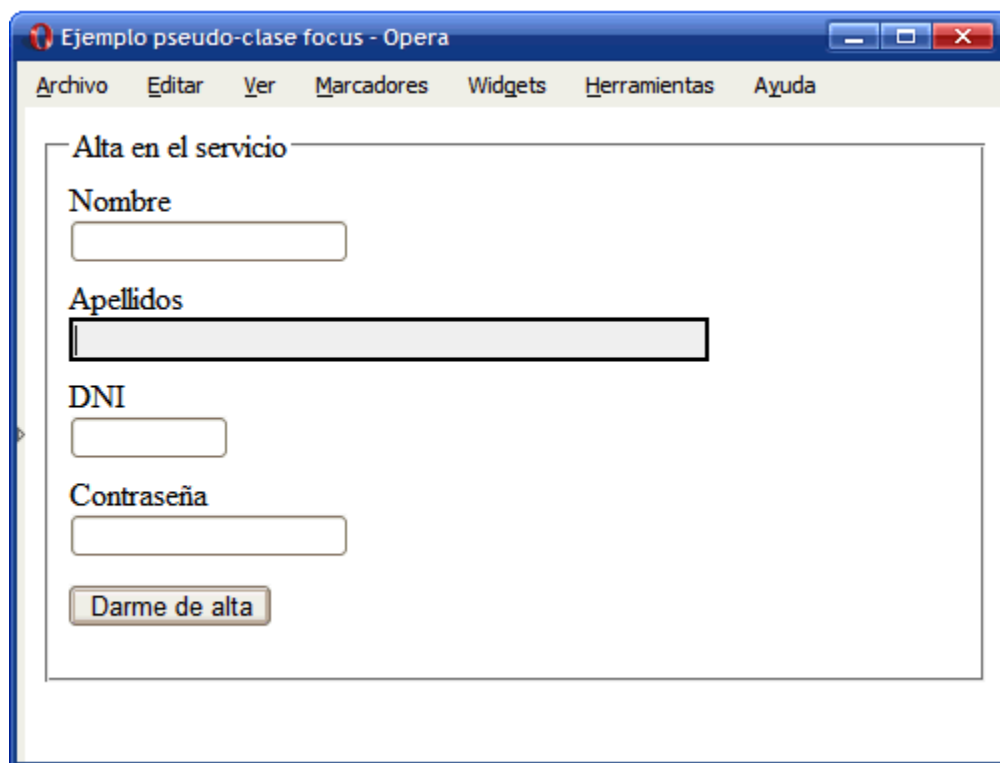


Figura 11.8 Ejemplo de pseudo-clase `:focus`

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

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

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

Modelo de Caja

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

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

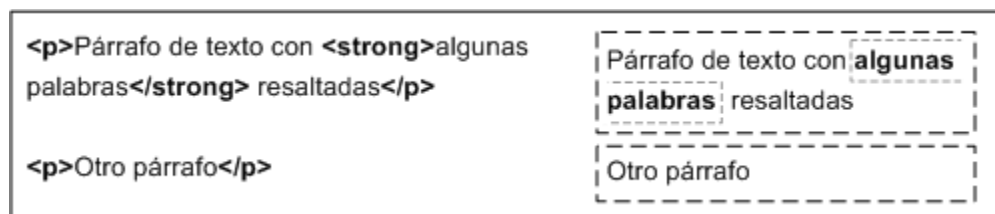


Figura 4.1 Las cajas se crean automáticamente al definir cada elemento HTML

Las cajas de las páginas no son visibles a simple vista porque inicialmente no muestran ningún color de fondo ni ningún borde. La siguiente imagen muestra las cajas que forman la página web de <http://www.alistapart.com/> después de forzar a que todas las cajas muestren su borde:

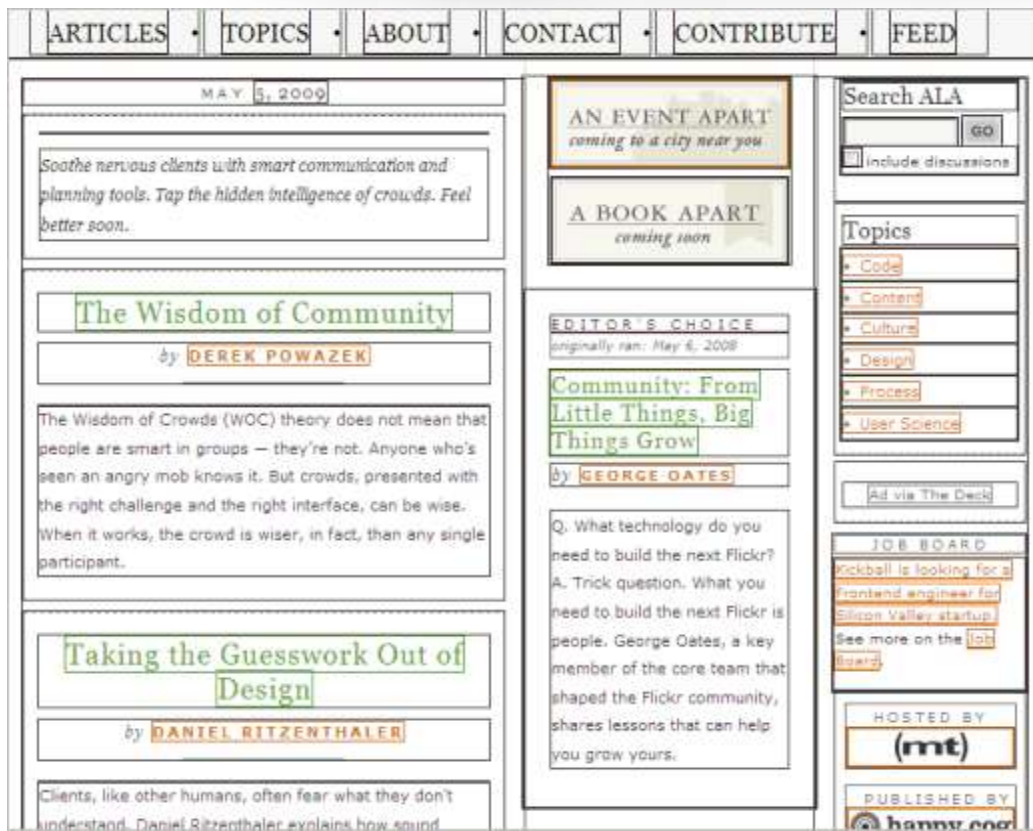


Figura 4.2 Cajas que forman la página alistapart.com

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

THE CSS BOX MODEL HIERARCHY

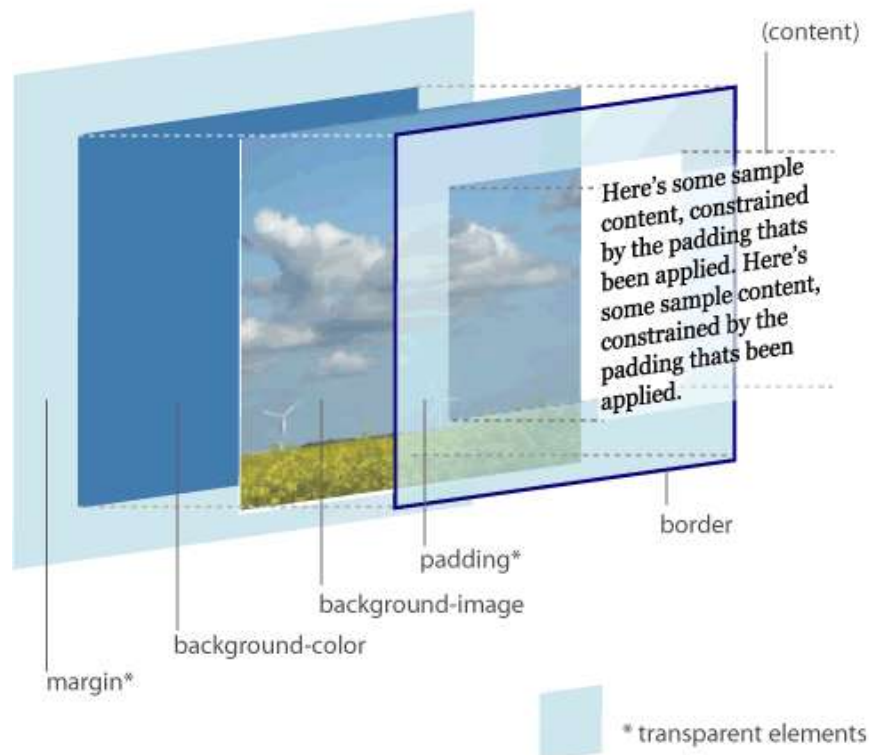


Figura 4.3 Representación tridimensional del box model de CSS

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

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

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



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

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

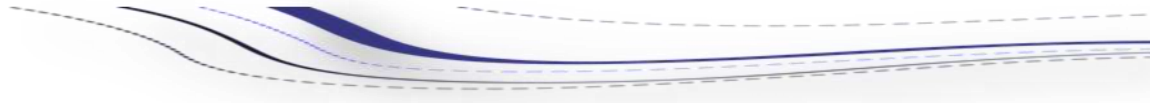
Ancho y Alto

Anchura

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

Propiedad	<code>width</code>
Valores	unidad de medida porcentaje <code>auto</code> 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	<code>auto</code>
Descripción	Establece la anchura de un elemento





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

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

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

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

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

Altura

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

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





Propiedad	height
Descripción	Establece la altura de un elemento

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

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

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

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

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

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

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



Propiedades	margin-top, margin-right, margin-bottom, margin-left
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:

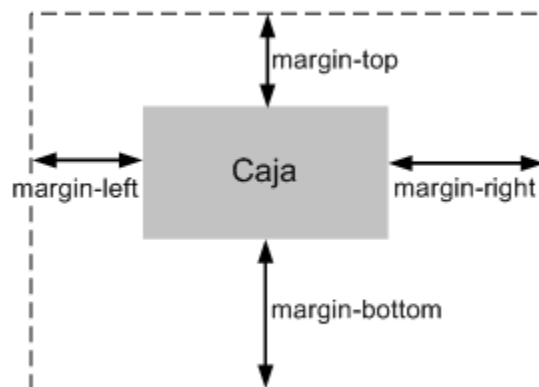


Figura 4.4 Las cuatro propiedades relacionadas con los márgenes

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

El siguiente ejemplo añade un margen izquierdo al segundo párrafo:

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

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

```
<p class="destacado">Vestibulum lectus diam, luctus vel, venenatis 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 consectetur tincidunt, risus turpis laoreet elit, ut tincidunt risus sem et nunc.</p>
```

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

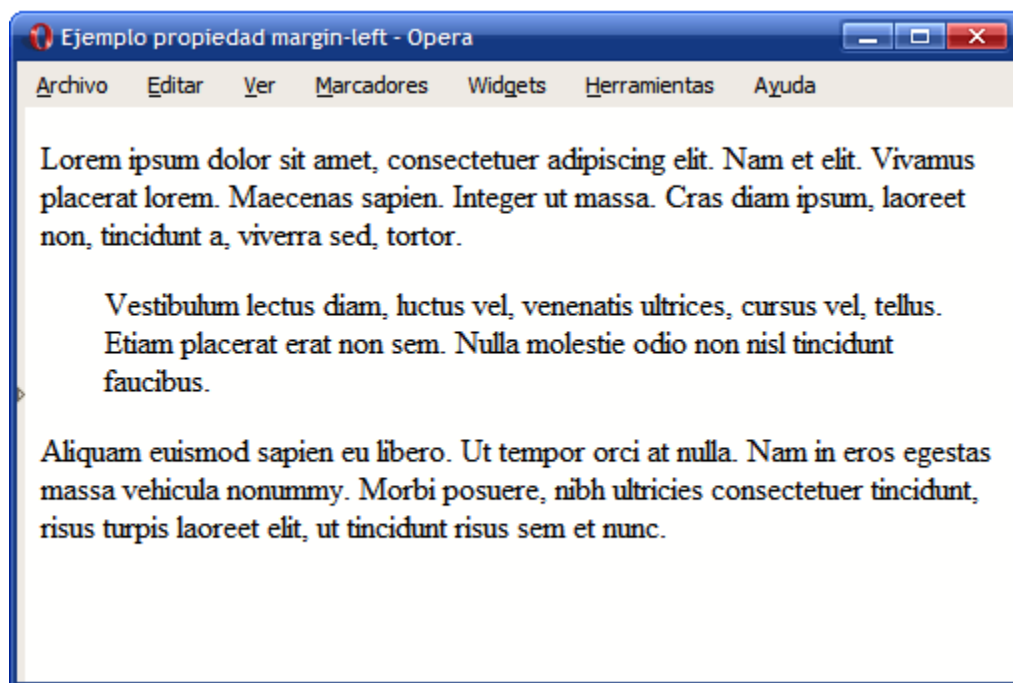


Figura 4.5 Ejemplo de propiedad margin-left

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

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

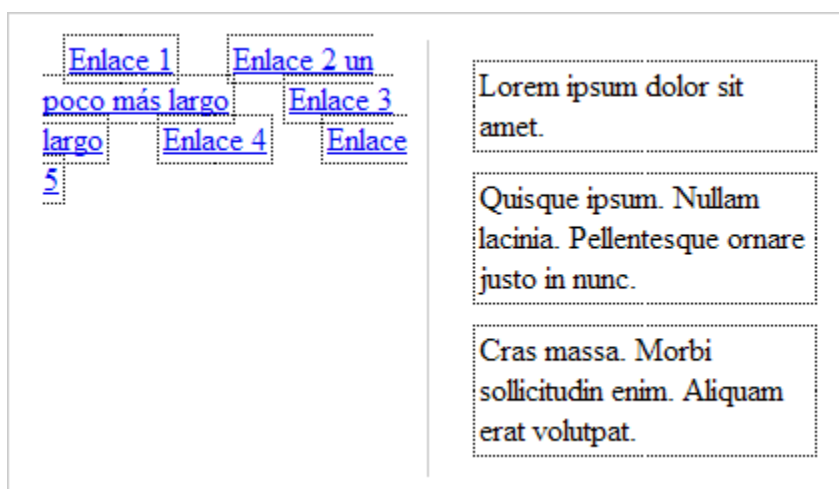


Figura 4.6 Los márgenes verticales sólo se aplican a los elementos de bloque e imágenes

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

Además de las cuatro propiedades que controlan cada uno de los márgenes del elemento, CSS define una propiedad especial que permite establecer los cuatro márgenes de forma

simultánea. Estas propiedades especiales se denominan "*propiedades shorthand*" y CSS define varias propiedades de este tipo, como se verá más adelante.

La propiedad que permite definir de forma 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 .5em 1em;  
}
```

Otra alternativa:

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

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

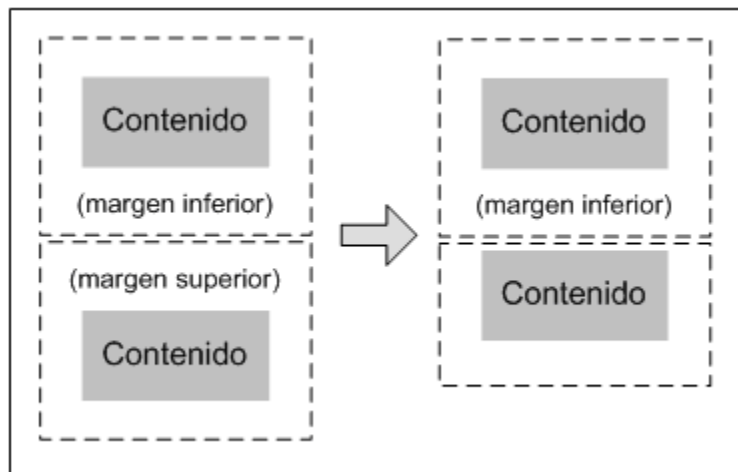


Figura 4.7 Fusión automática de los márgenes verticales

De la misma forma, si un elemento está contenido dentro de otro elemento, sus márgenes verticales se fusionan y resultan en un nuevo margen de la misma altura que el mayor margen de los que se han fusionado:

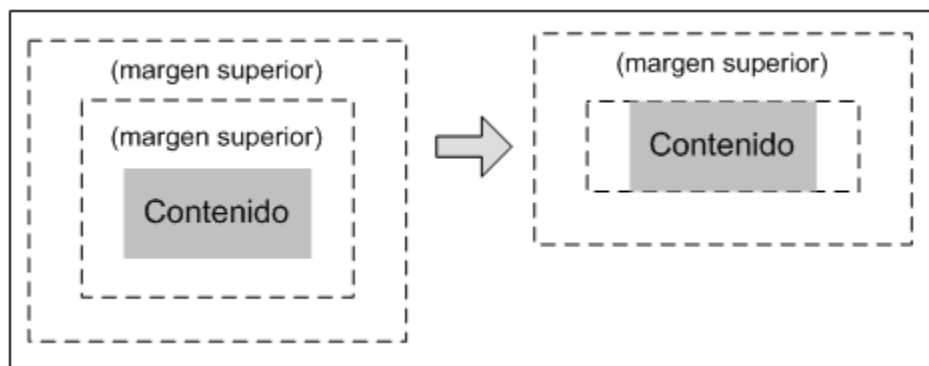


Figura 4.8 Fusión de los márgenes de los elementos interiores

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

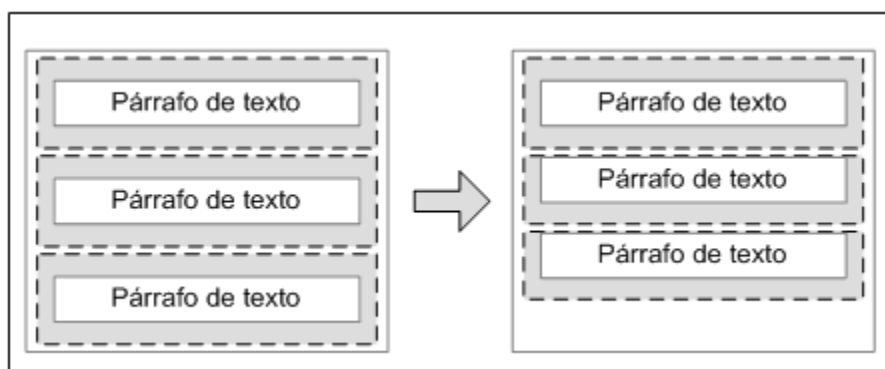


Figura 4.9 Motivo por el que se fusionan automáticamente los márgenes verticales

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

Relleno

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

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

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

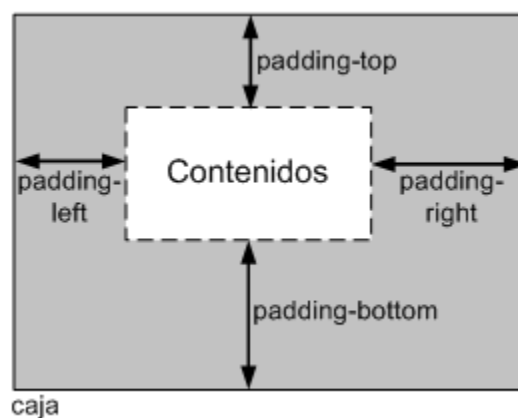


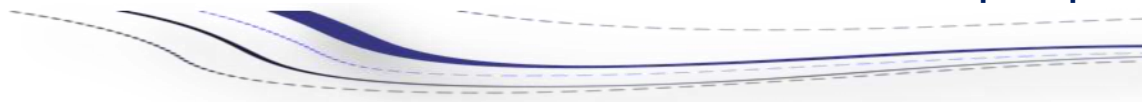
Figura 4.10 Las cuatro propiedades relacionadas con los rellenos

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

Propiedad	<code>padding</code>
Valores	(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 = 2em */
body {padding: 1em 2em 3em 4em}
    /* Superior = 1em, derecho = 2em, inferior = 3em, izquierdo = 4em */
```



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.

Anchura

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

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

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

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



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

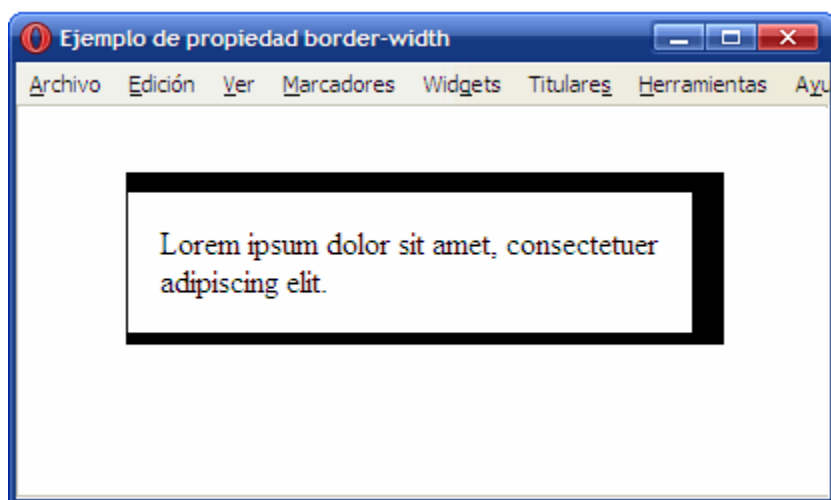


Figura 4.11 Ejemplo de propiedad border-width

Las reglas CSS utilizadas se muestran a continuación:

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

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

Propiedad	border-width
Valores	(unidad de medida thin medium thick) {1, 4} inherit
Se aplica a	Todos los elementos
Valor inicial	Medium

Propiedad	<code>border-width</code>
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.

Color

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

Propiedades	<code>border-top-color</code> , <code>border-right-color</code> , <code>border-bottom-color</code> , <code>border-left-color</code>
Valores	color transparent inherit
Se aplica a	Todos los elementos
Valor inicial	-

Propiedades	border-top-color, border-right-color, border-bottom-color, border-left-color
Descripción	Establece el color de cada uno de los cuatro bordes de los elementos

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

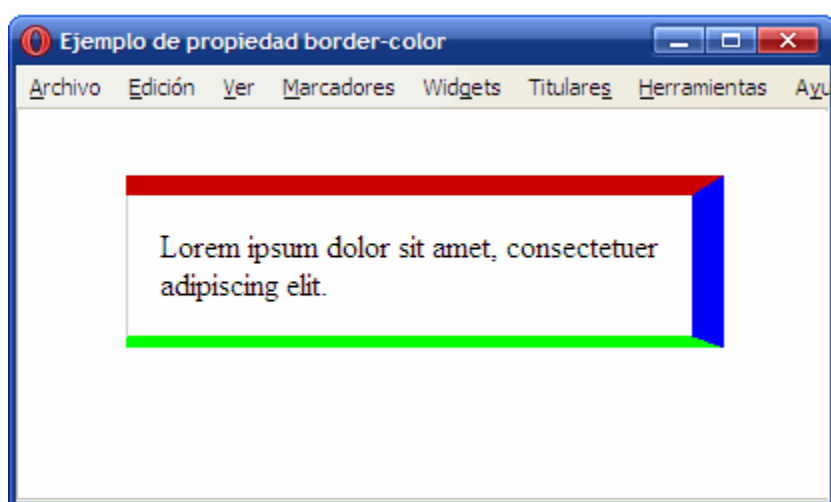
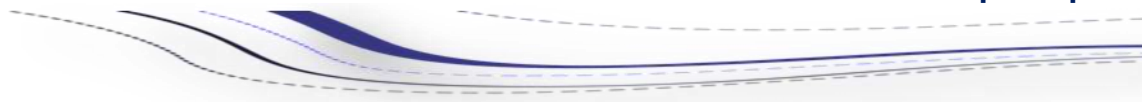


Figura 4.12 Ejemplo de propiedad border-color

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

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



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

Propiedad	<code>border-color</code>
Valores	(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`.

Estilo

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

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



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

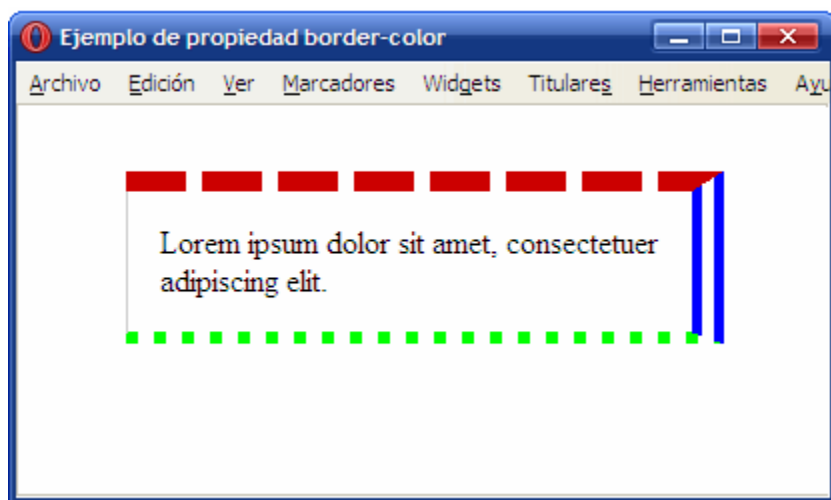


Figura 4.13 Ejemplo de propiedad border-style

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

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

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

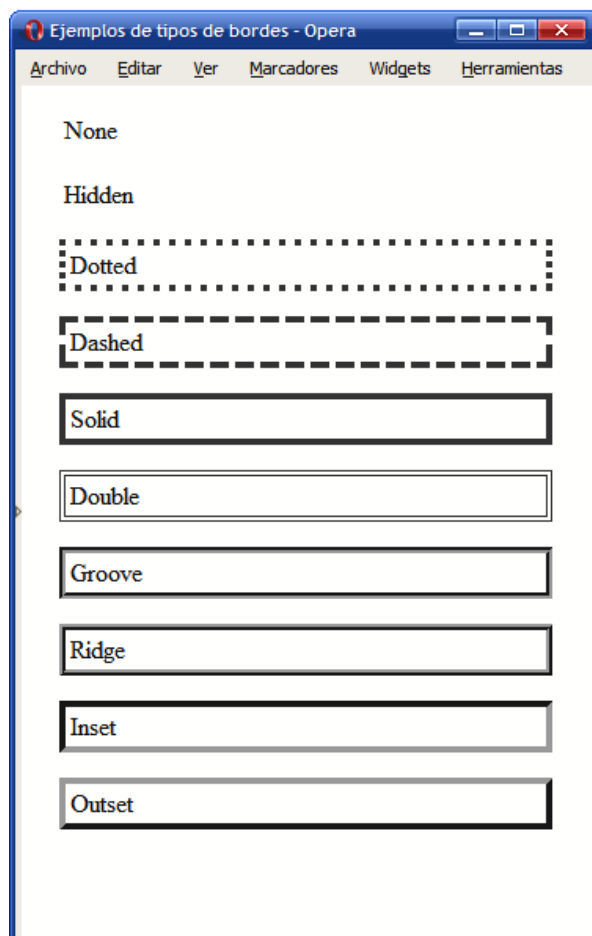


Figura 4.14 Tipos de bordes definidos por CSS

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



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

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

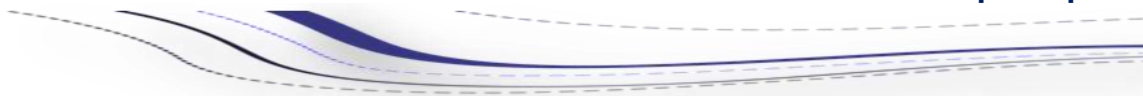
Como es habitual, la propiedad permite indicar de uno a cuatro valores diferentes y las reglas de aplicación son las habituales de las propiedades "*shorthand*".

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





Propiedades	border-top, border-right, border-bottom, border-left
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 último, 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 propiedad `border` para establecer de forma directa los atributos comunes de todos los



bordes y posteriormente especificar para cada uno de los cuatro bordes sus propiedades particulares:

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

Margen, relleno, bordes y modelo de cajas

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

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

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

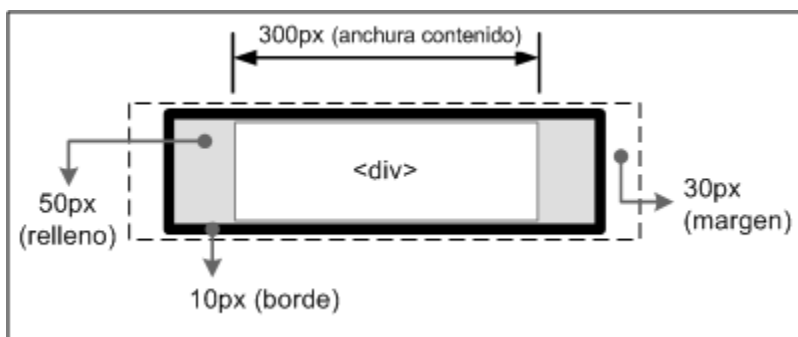


Figura 4.15 La anchura total de un elemento tiene en cuenta los márgenes, rellenos y bordes



De esta forma, la anchura del elemento en pantalla sería igual a la suma de la anchura original, los márgenes, los bordes y los rellenos:

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

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

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

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

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

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

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



```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
```

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

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

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

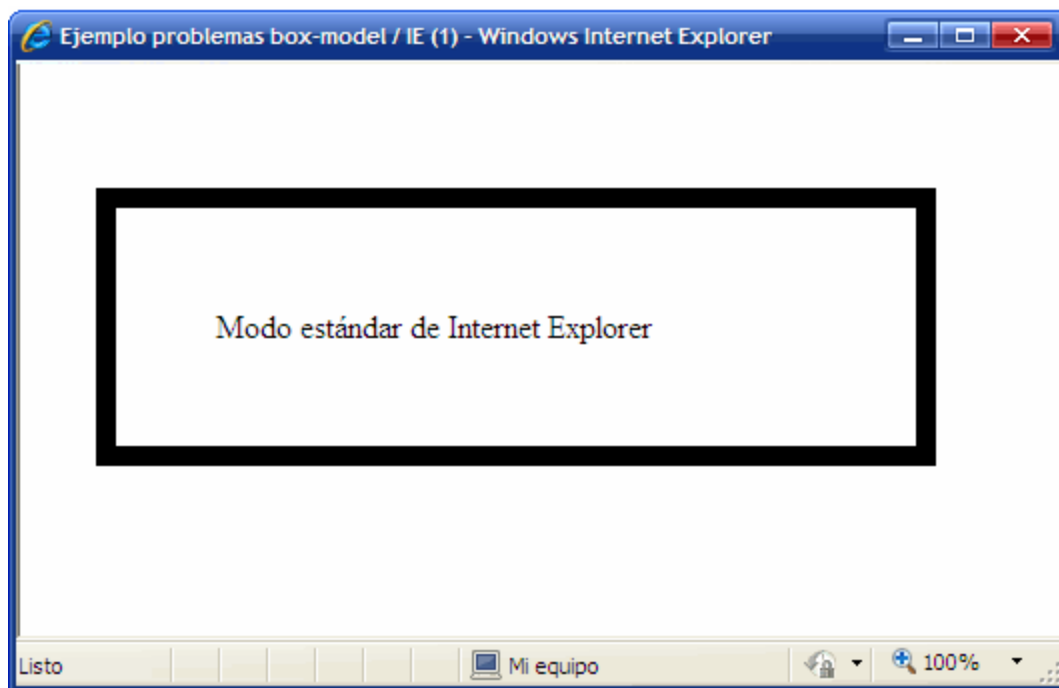


Figura 4.16 Internet Explorer 6 en modo estándar

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

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

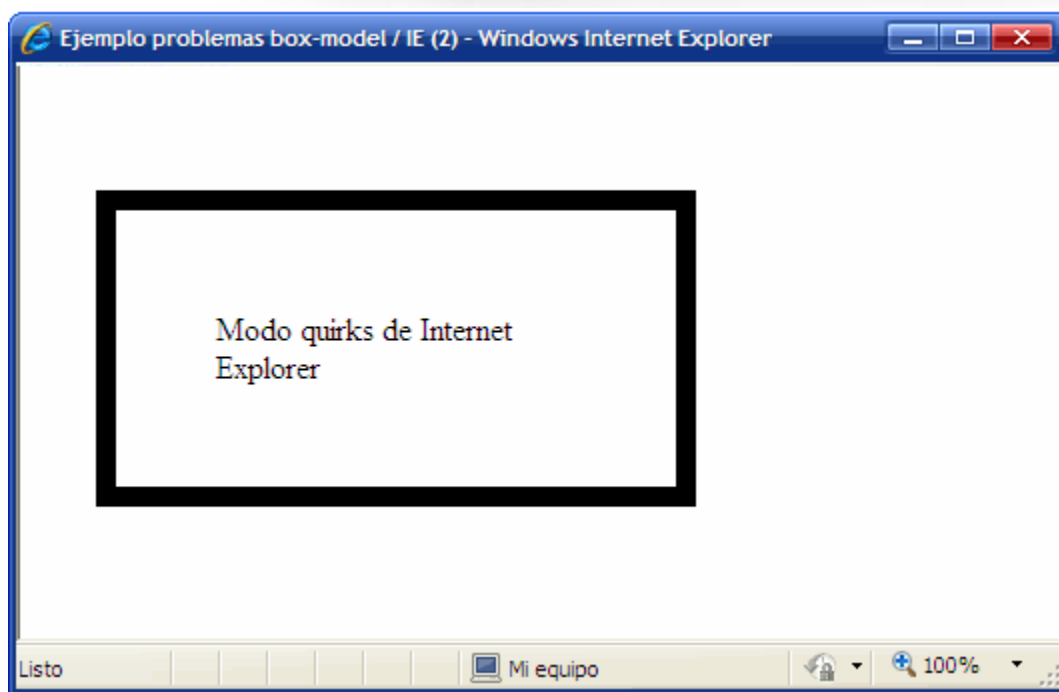


Figura 4.17 Internet Explorer 6 en modo quirks


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

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

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

Los modos de compatibilidad de Internet Explorer 8

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



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

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

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

- **Modo IE5:** la página se muestra según el modo *quirks* de Internet Explorer 7, que es casi idéntico a como se veían las páginas en el navegador Internet Explorer 5.
- **Modo IE7:** la página se muestra en el modo estándar de Internet Explorer 7, sin importar si la página contiene o no la directiva `<!DOCTYPE>`.
- **Modo IE8:** los contenidos se muestran en el modo estándar de Internet Explorer 8, que es el más parecido al del resto de navegadores que soportan los estándares (Firefox, Opera, Safari y Google Chrome).
- **Emular el modo IE7:** el navegador decide cómo mostrar los contenidos a partir de la directiva `<!DOCTYPE>` de la página. Si esa directiva es una de las que activan el modo estándar, la página se muestra en el modo estándar de Internet Explorer 7. En otro caso, se muestra en el modo *quirks* de Internet Explorer 5. Este modo es el más útil para la mayoría de sitios web.
- **Emular el modo IE8:** el navegador decide cómo mostrar los contenidos a partir de la directiva `<!DOCTYPE>` de la página. Si esa directiva es una de las que activan el modo estándar, la página se muestra en el modo estándar de Internet Explorer 8. En otro caso, se muestra en el modo *quirks* de Internet Explorer 5.
- **Modo límite ("edge mode"):** indica a Internet Explorer que los contenidos se deben mostrar en el modo de compatibilidad más avanzado disponible. Actualmente, este modo es equivalente al modo IE8. Si las futuras versiones Internet Explorer 9 y 10 incluyeran mejor compatibilidad, las páginas se visualizarían en ese modo avanzado de compatibilidad.

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

```
<!-- Modo IE5 -->
<head>
  <meta http-equiv="X-UA-Compatible" content="IE=5" />
  ...
</head>

<!-- Modo IE7 -->
<head>
  <meta http-equiv="X-UA-Compatible" content="IE=7" />
  ...
</head>

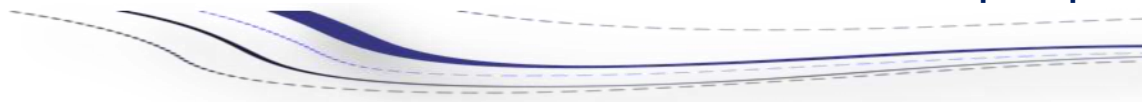
<!-- Modo IE8 -->
<head>
  <meta http-equiv="X-UA-Compatible" content="IE=8" />
  ...
</head>

<!-- Emular el modo IE7 -->
<head>
  <meta http-equiv="X-UA-Compatible" content="IE=EmulateIE7" />
  ...
</head>

<!-- Emular el modo IE8 -->
<head>
  <meta http-equiv="X-UA-Compatible" content="IE=EmulateIE8" />
  ...
</head>

<!-- Modo límite -->
<head>
  <meta http-equiv="X-UA-Compatible" content="IE=edge" />
  ...
</head>
```

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



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	<code>background-color</code>
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

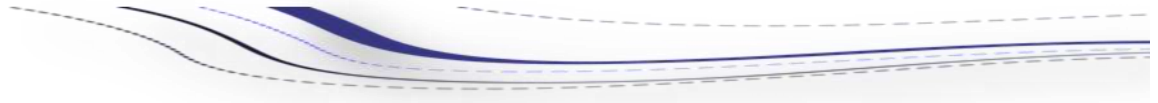
CSS permite establecer de forma simultánea un color y una imagen de fondo. En este caso, la imagen se muestra delante del color, por lo que solamente si la imagen contiene zonas transparentes es posible ver el color de fondo.

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



Figura 4.18 Imagen original utilizada para el fondo de la página

Reglas CSS

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

Resultado



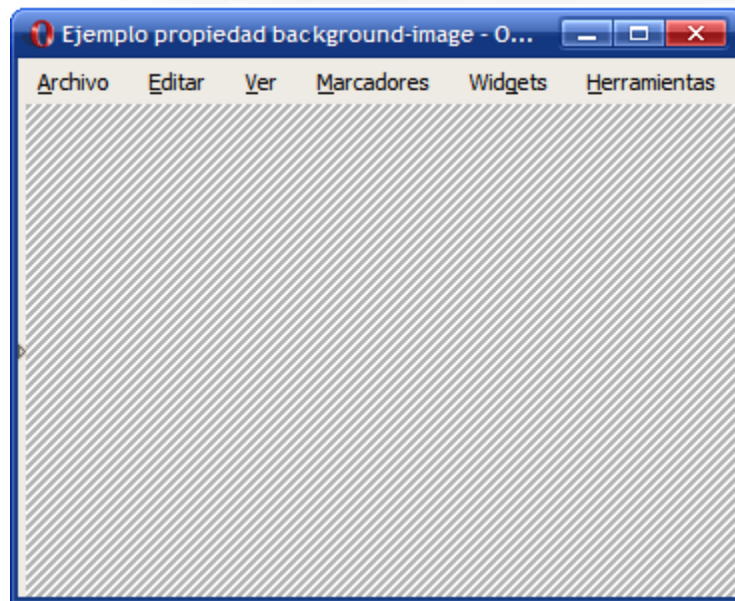


Figura 4.19 Página con una imagen de fondo

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

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

Propiedad	background-repeat
Valores	repeat repeat-x repeat-y no-repeat inherit
Se aplica a	Todos los elementos
Valor inicial	repeat

Propiedad	background-repeat
Descripción	Controla la forma en la que se repiten las imágenes de fondo

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

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

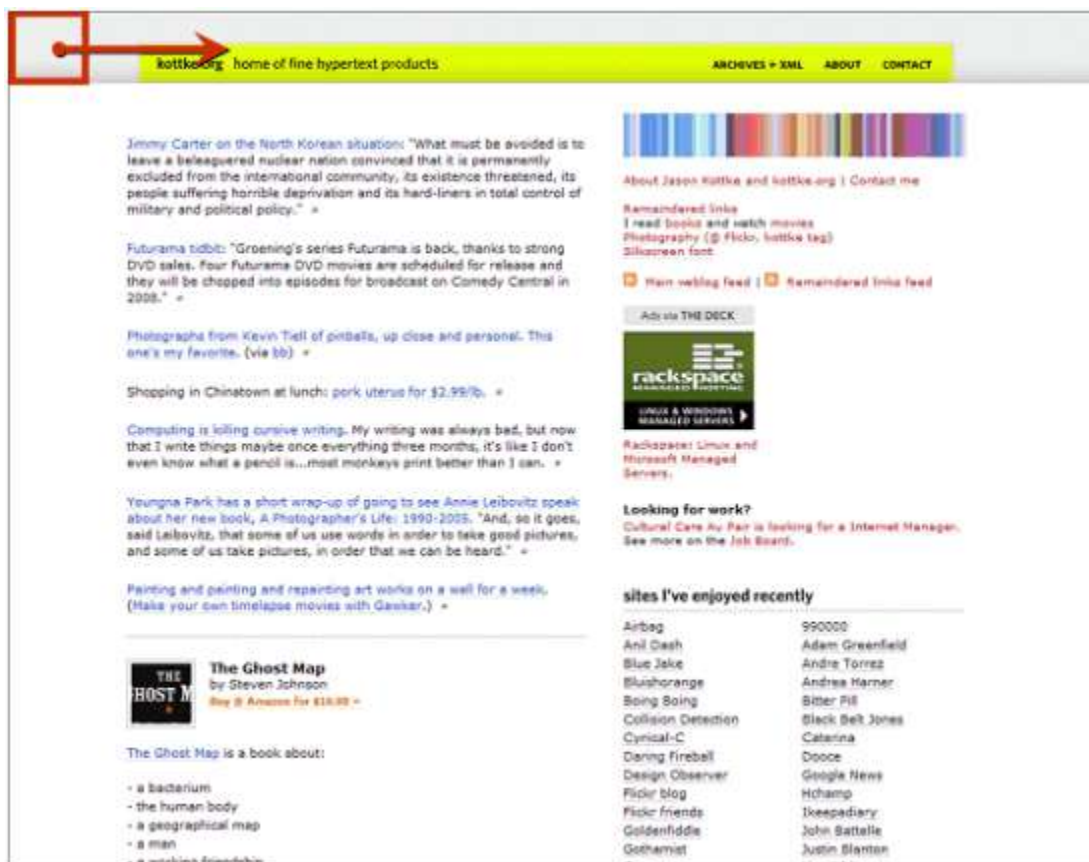


Figura 4.20 Uso de repeat-x en la página de Kottke.org

Las reglas CSS definidas para la cabecera son:

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

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

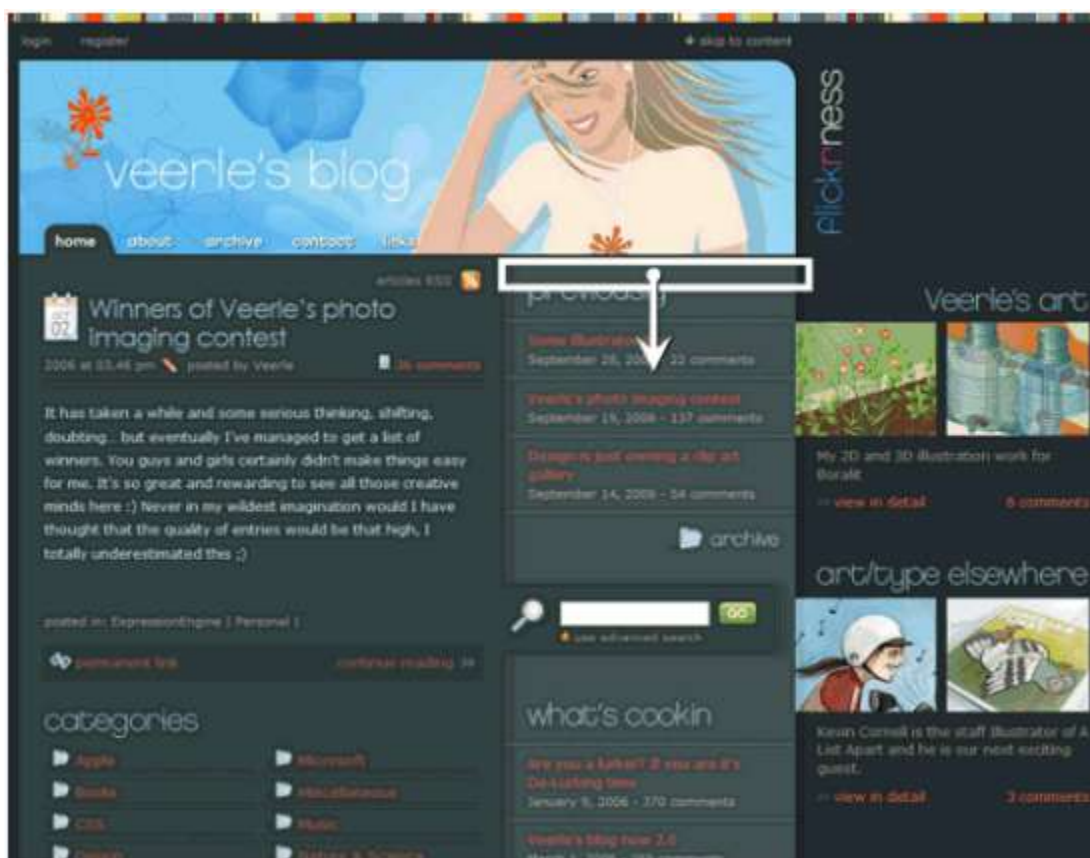


Figura 4.21 Uso de `repeat-y` en la página de veerle.duoh.com

Las reglas CSS definidas para esa columna de contenidos son:

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




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

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

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

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

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

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



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

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

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

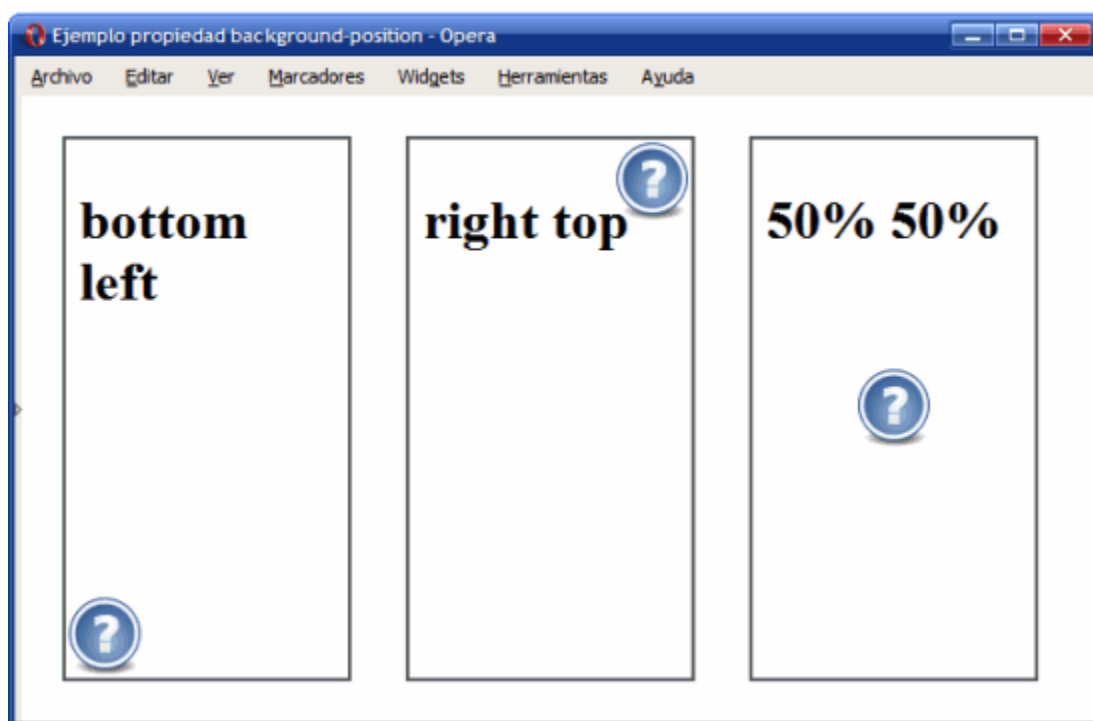



Figura 4.22 Ejemplo de propiedad background-position

Las reglas CSS del ejemplo anterior se muestran a continuación:

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



```
background-position: 50% 50%;
}

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

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

Propiedad	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.

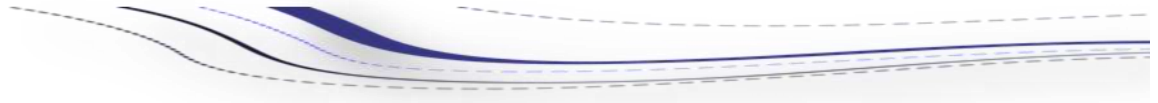
Propiedad	background
Valores	(background-color background-image background-repeat background-attachment background-position) inherit
Se aplica a	Todos los elementos
Valor inicial	-
Descripción	Establece todas las propiedades del fondo de un elemento

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;
```



Unidad 8 CSS 3

La especificación de CSS Level 3 o más comúnmente conocida como CSS3 no es una especificación completa sino que se divide en diferentes **módulos**. Cada uno de ellos añade nuevas capacidades o extiende aquellas ya existentes en especificaciones anteriores. La nueva especificación debe preservar compatibilidad hacia atrás. Según la W3C existen **53 módulos** actualmente en diferentes estados.

Debido a su carácter modular, hay partes de la especificación en **diferentes estados** dentro del proceso de estandarizado. Así el pasado día 7 de junio el módulo *Color Module* de CSS3 era publicado como recomendación por el consorcio [W3C](#).

Introducción al proceso de estandarizado

La **mejor forma** de estar al día sobre los avances de la especificación de CSS3 es seguir la página del estado de sus diferentes módulos. Al lado de cada módulo puede verse el **estado actual** del mismo, y el estado que alcanzara en la siguiente revisión. Cada uno de ellos lleva a un link de un borrador de especificación. Las diferentes abreviaturas son:

WD: Working Draft

Un Borrador de Trabajo o *Working Draft* es un documento que la W3C publica para su **revisión** por parte de la Comunidad, incluyendo a miembros del W3C, el público, y otras organizaciones y empresas.

LC: Last Call

En la Última Llamada o *Last Call* el grupo de trabajo del W3C anuncia una **última llamada** hacia otros grupos del W3C, el público y otras organizaciones donde se especifica:

1. Fecha límite para revisiones y comentarios
2. Una identificación exhaustiva de todas las dependencias y una solicitud de revisión de todos los grupos de trabajo dependientes
3. Una solicitud de revisión pública



Cuando un grupo fija el estado de un borrador a **LC** es una señal de:

- El grupo de trabajo cree que se han satisfecho los requerimientos técnicos (o de cualquier otro tipo) en el borrador de trabajo
- El grupo de trabajo cree que se han satisfecho las dependencias con otros grupos de trabajo
- Otros grupos **deben** revisar el documento para confirmar que dichas dependencias han sido satisfechas

Por regla general, cuando un grupo de trabajo fija el estado de un borrador de trabajo a **LC** es una indicación de que están planeando avanzar hacia estados **más maduros** del borrador.

RC: Candidate Recommendation

Un Candidato de Recomendación o *Candidate Recommendation* es un documento que el W3C opina que ha sido lo **suficientemente revisado** y satisface los requerimientos técnicos del grupo de trabajo. La W3C publica candidatos de recomendación para iniciar un proceso de experiencia en su implementación.

PR: Proposed Recommendation

Una Propuesta de Recomendación o una *Proposed Recommendation* es un reporte técnico **maduro** después de una amplia revisión técnica de su solidez y robustez en la implementación. Cuando un documento llega a este estado, el W3C lo envía a su Comité Asesor para su aprobación final.

REC: W3C Recommendation

Una Recomendación del W3C o *W3C Recommendation* es una especificación o conjunto de directrices que tras un extenso consenso, ha recibido el **respaldo de los miembros del W3C** y su director. El W3C recomienda un amplio despliegue de sus recomendaciones que se convierten en parte del estándar.

Este proceso puede durar años hasta completarse.

Descripción de CSS3

CSS es un lenguaje (no un lenguaje de programación) utilizado para especificar el aspecto de una página web para diferentes dispositivos. Esto entra en contraste con HTML que es un lenguaje que define la estructura de un documento para su distribución por la web. El HTML le dice al navegador **como se estructura** el documento mientras que el CSS le dice **como debe renderizarlo**.

CSS3 añade **muchas capacidades** nuevas a la especificación anterior. Aunque aún está en proceso de estandarizado y le queda un **largo camino por recorrer**, ya se ha hecho un hueco al lado de HTML5 y supone uno de los mayores adelantos en el diseño web actual.

Novedades de CSS3

Como ya he dicho, CSS3 viene con muchas novedades, en este apartado se van a mencionar algunas de las más relevantes y seguramente se dejarán muchos en el tintero:

- **Bordes:**
 - Colores múltiples de borde en un mismo lado
 - Imágenes de borde
 - Bordes redondeados
- **Fondos:**
 - Fondos Múltiples pueden ser añadidos al mismo elemento como capas
 - Posicionamiento del fondo con mayor precisión
 - Pueden ser redimensionados
- **Color:**
 - Opacidad
 - Gradientes
 - Valores de color: [HSL](#)
- **Text:**
 - Sombras
 - Desbordamiento
 - Ajuste de línea



- **Transformaciones:**
 - Escalar
 - Sesgar
 - Mover
 - Rotar en 2D o 3D
- **Transiciones:**
 - Transición sencilla de estilos
- **Cajas:**
 - Sombras
 - Cajas redimensionables
 - Overflow separado en vertical u horizontal
 - Compensación entre contorno y borde
 - Modelos para especificar altura y anchura
- **Contenido:**
 - Los estilos pueden añadir contenido a los elementos
- **Opacidad:**
 - Los elementos pueden ser transparentes
- **Fuentes Web:**
 - Capacidad de añadir fuentes *en vivo* a los documentos mejorada

No todos los navegadores soportan el mismo nivel de CSS3 y no todos los módulos de especificación están aún siquiera implementados.



Conclusiones finales

La estandarización de CSS3 es un paso fundamental para que sus características logren un mayor grado de adopción en la Web actual. Lo que algunos diseñadores no saben es que varias de sus características ya han completado su ruta y son recomendación oficial del W3C. Prueba de ésto es que ya existen documentos que hacen referencia a la próxima versión: CSS4.

También hay que destacar que los navegadores modernos están ofreciendo una buena compatibilidad para las características básicas de CSS3, como el caso de sombras, bordes redondeados y fuentes, entre otras.

Aquí es importante estar atento a algunos aspectos. Debemos utilizar prefijos para algunas características. Muchas herramientas nos ayudan con esta tarea, entre ellas. [CSSPrefixer](#). También encontramos soluciones online para hacernos más fácil la creación de estilos CSS3, un caso claro de esto es [CSS3.0 Maker](#).

Entre las características que aún no tienen un alto soporte en algunos navegadores (especialmente en browsers antiguos) encontramos las que hacen referencias a efectos de giro, transiciones, animaciones y especialmente los que están vinculados al 3D con CSS.

Quizás las características mencionadas en el párrafo anterior son las más complejas para emular en navegadores antiguos. Más allá de esto, cada día existen más opciones para compatibilizar CSS3 con navegadores que no soportan sus características, como el ejemplo de IE8 (o versiones inferiores). Buenos ejemplos los encontramos en [selectivizr](#) y en [CSS3PIE](#), entre otros.

Todos estos datos hacen evidente que ya contamos con muy buenas opciones para comenzar a trabajar con CSS3 en nuestros diseños Web. ¡Adelante, entonces!

Unidades de medida

Las unidades de medida son una de las partes más importantes de CSS, ya que se utilizan para definir la altura, anchura y márgenes de todos los elementos y para establecer el tamaño de letra del texto.

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

Las medidas, absolutas y relativas, se indican como un valor numérico entero o decimal seguido de una unidad de medida (sin ningún espacio en blanco entre el número y la unidad de medida).

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

Unidades relativas

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

em, relativa respecto del tamaño de letra empleado. Aunque no es una definición exacta, el valor de **1 em** se puede aproximar por la anchura de la letra **M** "eme mayúscula") del tipo de letra que se esté utilizando

ex, relativa respecto de la altura de la letra **X** "equis minúscula") del tipo de letra que se esté utilizando

px, (píxel) relativa respecto de la pantalla del usuario



Cuando se establece el tamaño de la letra, la unidad relativa más utilizada es **em**. En el siguiente ejemplo, se define el valor del tamaño de la letra de varios elementos utilizando la unidad **em**:

[1em] Lorem ipsum dolor sit amet, consectetur adipiscing elit.

[1.5em] Lorem ipsum dolor sit amet, consectetur adipiscing elit.

[2em] Lorem ipsum dolor sit amet, consectetur adipiscing elit.

Cuando se utilizan unidades relativas, es imprescindible conocer la referencia que se toma para obtener el valor definitivo. En otras palabras, un tamaño de letra **1em** o **3em** no significan nada a menos que se conozca su valor de referencia.

Si se define el tamaño de letra de un elemento mediante la unidad **em**, su referencia es el tamaño de letra del primer elemento contenedor que tenga establecido un tamaño de letra. Si el elemento no se encuentra dentro de ningún otro elemento, entonces la referencia es el tamaño de letra del **<body>**.

En el siguiente ejemplo, los elementos interiores tienen el mismo tamaño de letra (**1em**). Sin embargo, como sus elementos contenedores tienen tamaños muy diferentes (**0.9em** y **1.5em**) el tamaño de letra resultante es muy diferente:

[0.9em]


[1em] Lorem ipsum dolor sit amet, consectetur adipiscing elit.

[1.5em]

[1em] Lorem ipsum dolor sit amet, consectetur adipiscing elit.

Por otra parte, si se utiliza la unidad **em** para definir la anchura de un elemento, la referencia es el tamaño de letra del propio elemento. En el siguiente ejemplo, los dos elementos tienen la misma anchura (**20em**), pero como su tamaño de letra es muy diferente (**1em** y **2em**) la anchura resultante también es muy diferente:





```
div { width: 20em; font-size: 1em; }  
div { width: 20em; font-size: 2em; }
```

La unidad de medida **ex** se utiliza mucho menos que la unidad **em**, aunque el funcionamiento de ambas es idéntico. Por su parte, la unidad de medida píxel (**px**) sí que se utiliza de forma habitual y toma como referencia el tamaño de la pantalla del usuario.

```
div { width: 250px; }  
div { width: 450px; }
```

Aunque los píxel normalmente sólo se utilizan para definir la anchura y altura de los elementos, también se pueden utilizar para establecer el tamaño de la letra, aunque en este caso se recomienda utilizar la unidad **em**:

[10px] Lorem ipsum dolor sit amet, consectetur adipiscing elit.

[18px] Lorem ipsum dolor sit amet, consectetur adipiscing elit.

[25px] Lorem ipsum dolor sit amet, consectetur adipiscing elit.

Unidades absolutas

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

in, del inglés *"inches"*, pulgadas (1 pulgada son **2.54** centímetros)

cm, centímetros

mm, milímetros

pt, puntos (1 punto equivale a **1 pulgada/72**, es decir, unos **0.35** milímetros)

pc, picas (1 pica equivale a **12** puntos, es decir, unos **4.23** milímetros)

A continuación se muestran ejemplos de utilización de unidades absolutas:

```
[12pt] Lorem ipsum dolor sit amet, consectetur adipiscing elit.
```

```
div { width: 10cm; height: 1in; }  
div { width: 80mm; height: 7pc; }
```

A pesar de que parecen más fáciles de comprender, las unidades absolutas se utilizan mucho menos que las unidades relativas. El motivo es que las unidades absolutas son adecuadas para medios impresos pero no lo son tanto para medios flexibles como una pantalla. De todas las unidades absolutas, los puntos (**pt**) es la única que se emplea ocasionalmente y casi siempre para definir el tamaño de letra para los **medios** impresos (**print**) de CSS.

Porcentajes

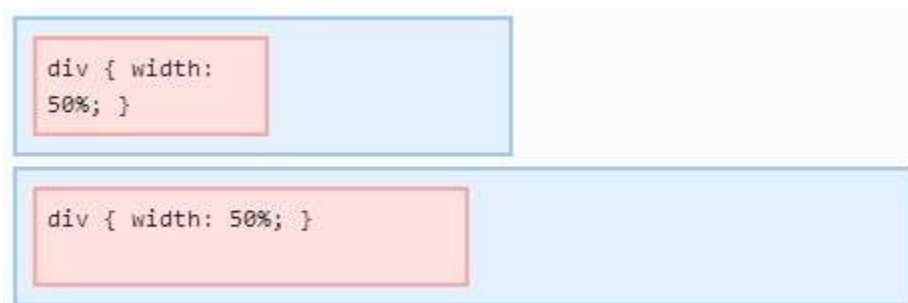
Además de las [unidades de medida](#) absolutas y relativas, CSS define otra unidad de medida especial basada en los porcentajes. Un porcentaje se indica mediante un valor numérico seguido del símbolo “%” y siempre está referenciado a otra medida. Cada una de las propiedades de CSS que permiten indicar como valor un porcentaje, también define el valor al que hace referencia ese porcentaje.

Los porcentajes se utilizan habitualmente para establecer el tamaño de la letra y la anchura de los elementos. Cuando se utiliza para definir el tamaño de letra de los elementos, es equivalente a la unidad `em`, como se muestra en el siguiente ejemplo:

```
[0.8em] Lorem ipsum dolor sit amet, consectetur adipiscing elit.  
[80%] Lorem ipsum dolor sit amet, consectetur adipiscing elit.  
[1.2em] Lorem ipsum dolor sit amet, consectetur adipiscing elit.  
[120%] Lorem ipsum dolor sit amet, consectetur adipiscing elit.
```

Si los porcentajes se utilizan para establecer la anchura de un elemento, su referencia es la anchura de su elemento contenedor. Si el elemento no se encuentra dentro de ningún otro elemento, su referencia es la anchura de la página entera.

En el siguiente ejemplo, los dos elementos `<div>` interiores tienen la misma anchura del 50%, pero como la anchura de su elemento contenedor es diferente, la anchura real de cada `<div>` interno también es diferente:



El valor inherit

Muchas de las propiedades CSS heredan su valor *"de padres a hijos"*, lo que permite no tener que establecer el valor de todas las propiedades CSS para todos los elementos de la página.

En el siguiente ejemplo, se muestra un párrafo que contiene en su interior un elemento `` y otro elemento ``. Si se define el color de la letra del párrafo, los elementos `` y `` se mostrarán del mismo color (salvo que definan otro color de forma explícita) ya que el color de la letra se hereda:

Lorem ipsum dolor sit amet, consectetur adipiscing elit.

Para el resto de propiedades CSS que por defecto no se heredan, se puede utilizar el valor `inherit` para forzar a que se produzca la herencia automática. En el siguiente ejemplo, se establece un estilo discontinuo al borde superior del primer `<div>`. Como la propiedad `border-top-style` no se hereda, el `<div>` interior no mostrará ese tipo de borde:

```
div { border-top-style: dashed; }  
  
El estilo del borde no se hereda
```

Utilizando el valor `inherit`, es posible forzar a que el `<div>` interior muestre el mismo estilo de borde que el `<div>` que lo encierra:

```
div { border-top-style: dashed; }  
  
div { border-top-style: inherit; }
```


El valor `inherit` también se puede utilizar para que los enlaces muestren el mismo color de letra que el párrafo en el que se encuentran:

```

Lorem ipsum dolor sit amet, consectetuer adipiscing elit.
Lorem ipsum dolor sit amet, \[color:inherit\] consectetuer adipiscing elit.

```

Para saber si el valor de una propiedad se hereda o no, puedes consultar la tabla de características que se incluye en la definición de cada propiedad CSS de esta referencia.

Fuentes de consulta

Bibliografía

Autor:	Dan Cederholm
Título:	<i>Bulletproof Web Design. Improving Flexibility and Protecting Against Worst-case Scenarios with XHTML and CSS</i>
País:	
Editorial:	New Riders Press
Año de publicación:	2013

Autor:	Dave Shea y Molly E. Holzschlag
Título:	<i>The Zen of CSS Design. Visual Enlightenment for the Web</i>
País:	U.S.A.
Editorial:	New Riders Press
Año de publicación:	

Autor:	Ethan Marcotte
Título:	<i>Responsive Web Design</i>
País:	U.S.A.
Editorial:	A Book Apart
Año de publicación:	2010

Autor:	Juan Diego Gauchat
Título:	<i>El gran libro de html5 css3 y javascript 2 edicion</i>
País:	España
Editorial:	Ediciones técnicas marcombo
Año de publicación:	2013

Cibergrafía

Autor:	Freddie
Título:	Curso de CSS en CristaLab
Vínculo:	http://www.cristalab.com/blog/curso-de-css-3-online-en-cristalab-c97207/
Editor:	CristaLab Foro de Tutoriales
Año de publicación:	2012

Autor:	Guerrero SanMartín Sandra
Título:	<i>Aprende ya Diseño Web con HTML5 y CSS 3</i>
Vínculo:	http://www.esandra.com/disenoweb/
Editor:	http://www.esandra.com
Año de publicación:	2013