

Introducción a **CSS3**

| | |
|---|------------|
| Capítulo 1. Introducción | 5 |
| 1.1. ¿Qué es CSS?..... | 5 |
| 1.2. Breve historia de CSS | 5 |
| 1.3. Soporte de CSS en los navegadores | 6 |
| 1.4. Especificación oficial | 7 |
| 1.5. Funcionamiento básico de CSS | 7 |
| 1.6. Cómo incluir CSS en un documento XHTML | 9 |
| 1.7. Glosario básico | 12 |
| 1.8. Medios CSS | 12 |
| 1.9. Comentarios | 15 |
| 1.10. Sintaxis de la definición de cada propiedad CSS | 15 |
| Capítulo 2. Selectores | 17 |
| 2.1. Selectores básicos | 17 |
| 2.2. Selectores avanzados | 25 |
| 2.3. Agrupación de reglas | 27 |
| 2.4. Herencia | 28 |
| 2.5. Colisiones de estilos | 29 |
| Capítulo 3. Unidades de medida y colores | 31 |
| 3.1. Unidades de medida | 31 |
| 3.2. Colores | 37 |
| Capítulo 4. Modelo de cajas (box model) | 42 |
| 4.1. Anchura y altura | 44 |
| 4.2. Margen y relleno | 45 |
| 4.3. Bordes | 53 |
| 4.4. Margen, relleno, bordes y modelo de cajas | 60 |
| 4.5. Fondos | 64 |
| Capítulo 5. Posicionamiento y visualización | 72 |
| 5.1. Tipos de elementos | 72 |
| 5.2. Posicionamiento | 74 |
| 5.3. Posicionamiento normal | 75 |
| 5.4. Posicionamiento relativo | 77 |
| 5.5. Posicionamiento absoluto | 79 |
| 5.6. Posicionamiento fijo | 84 |
| 5.7. Posicionamiento flotante | 84 |
| 5.8. Visualización | 91 |
| Capítulo 6. Texto | 98 |
| 6.1. Tipografía | 98 |
| 6.2. Texto | 105 |
| Capítulo 7. Enlaces | 116 |
| 7.1. Estilos básicos | 116 |
| 7.2. Estilos avanzados | 118 |
| Capítulo 8. Imágenes | 122 |

| | |
|--|------------|
| 8.1. Estilos básicos | 122 |
| 8.2. Estilos avanzados | 122 |
| Capítulo 9. Listas | 125 |
| 9.1. Estilos básicos | 125 |
| 9.2. Estilos avanzados | 132 |
| Capítulo 10. Tablas | 138 |
| 10.1. Estilos básicos | 138 |
| 10.2. Estilos avanzados | 141 |
| Capítulo 11. Formularios | 145 |
| 11.1. Estilos básicos | 145 |
| 11.2. Estilos avanzados | 150 |
| Capítulo 12. Layout | 153 |
| 12.1. Centrar una página horizontalmente | 153 |
| 12.2. Centrar una página verticalmente | 157 |
| 12.3. Estructura o layout | 159 |
| 12.4. Alturas/anchuras máximas y mínimas | 164 |
| 12.5. Estilos avanzados | 166 |
| Capítulo 15. Ejercicios | 190 |
| Capítulo 16. Ejercicios resueltos | 215 |

Capítulo 1. Introducción

1.1. ¿Qué es CSS?

CSS es un lenguaje de hojas de estilos creado para controlar el aspecto o presentación de los documentos electrónicos definidos con HTML. 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 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 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.

Las siglas CSS (Cascading Style Sheets) significan «Hojas de estilo en cascada» y parten de un concepto simple pero muy potente: aplicar estilos (colores, formas, márgenes, etc...) a uno o varios documentos (generalmente documentos HTML, páginas webs) de forma masiva.

Se le denomina estilos en cascada porque se aplican de arriba a abajo (siguiendo un patrón denominado herencia que trataremos más adelante) y en el caso de existir ambigüedad, se siguen una serie de normas para resolverla.

La idea de CSS es la de utilizar el concepto de separación de presentación y contenido, intentando que los documentos HTML incluyan sólo información y datos, relativos al significado de la información a transmitir (el contenido), y todos los aspectos relacionados con el estilo (diseño, colores, formas, etc..) se encuentren en un documento CSS independiente (la presentación):



De esta forma, se puede unificar todo lo relativo al diseño visual en un solo documento CSS, y con ello, varias ventajas:

Si necesitamos hacer modificaciones visuales lo hacemos en un sólo lugar. No necesitamos editar todo el HTML en cuestión por separado.

Se reduce la duplicación de estilos en diferentes lugares, por lo que es más fácil de organizar y hacer cambios. Además, al final la información a transmitir es considerablemente menor (las páginas se descargan más rápido).

Es más fácil crear versiones diferentes de presentación para otros tipos de dispositivos: tablets, smartphones o dispositivos móviles, etc...

1.2. Soporte de CSS en los navegadores

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

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

The screenshot shows the homepage of CanIuse.com. At the top, there's a navigation bar with 'Home' (highlighted in orange), 'News', 'January 29, 2022 - New feature: COLR/CPAL(v1) Font Formats', 'Compare browsers', and 'About'. Below the navigation is a search bar with 'Can I use' and a 'Settings' button. The main content area has several sections: 'Index of features', 'Latest features' (listing COLR/CPAL(v1) Font Formats, CSS @when / @else conditional rules, LCH and Lab color values, CSS Cascade Layers, and CSS Nesting), 'Most searched features' (listing CSS Grid, Flexbox, WebP image format, gap property for Flexbox, and ES6), 'Test a feature' (with a note about a partnership with BrowserStack), 'Did you know?' (with a note about Google Analytics integration), 'Third party tools' (listing CanIuse Embed, CanIuse Component, and CanIuse command line tool), and 'Browser scores' (showing Chrome 99: 397, Safari 13: 373, and Firefox 97: 373). There are also 'Current version' and 'Dev version' buttons for activating Windows.

1.3. Especificación oficial

El sitio web del organismo W3C dispone de una sección en la que se detalla el trabajo que el W3C, <https://www.w3.org/> está desarrollando actualmente en relación a CSS3 también dispone de un blog en el que se publican todas las novedades relacionadas con CSS (<http://www.w3.org/blog/CSS>).

1.4. Cómo incluir CSS en un documento HTML

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

1.4.1. Incluir CSS en el mismo documento HTML

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

Ejemplo:

```
<!DOCTYPE>
<html>
<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.

1.4.2. Definir CSS en un archivo externo

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

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

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

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

- 2) Se guarda el archivo de texto con el nombre `estilos.css`. Se debe poner especial atención a que el archivo tenga extensión `.css` y no `.txt`.
- 3) En la página HTML se enlaza el archivo CSS externo mediante la etiqueta `<link>`:

```
<!DOCTYPE>
<html>
<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" />
</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 se enlaza un archivo CSS:

- `rel`: indica el tipo de relación que tiene 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.

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.

1.4.3. Incluir CSS en los elementos HTML

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

Ejemplo:

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

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

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

1.7. Glosario básico

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

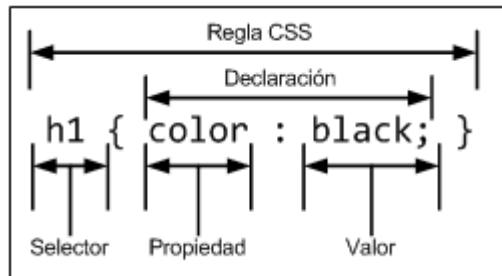
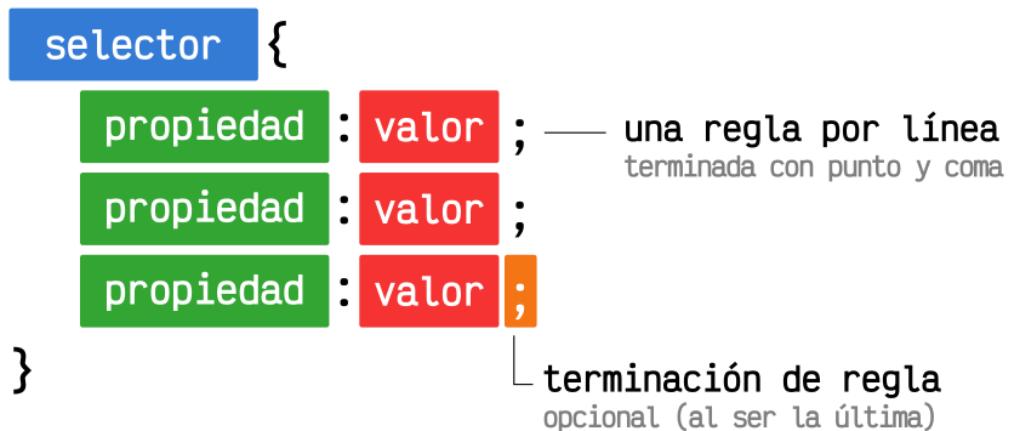


Figura 1.1. Componentes de un estilo CSS básico



Los diferentes términos se definen a continuación:

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

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

1.9. Comentarios

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

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

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

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

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

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

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

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

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

Capítulo 2. Selectores

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

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

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

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

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

2.1. Selectores básicos

2.1.1. Selector universal

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

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

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

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

2.1.2. Selector de tipo o etiqueta

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

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

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

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

```
h1 {  
    color: red;  
}  
  
h2 {  
    color: blue;  
}  
  
p {  
    color: black;  
}
```

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

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

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

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

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

```
h1, h2, h3 {  
    color: #8A8E27;
```

```

    font-weight: normal;
    font-family: Arial, Helvetica, sans-serif;
}

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

```

2.1.3. Selector descendente

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

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

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

Si el código HTML de la página es el siguiente:

```

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

```

El selector `p span` selecciona tanto `texto1` como `texto2`. El motivo es que en el selector descendente, un elemento no tiene que ser "*hijo directo*" de otro. La única condición es que un elemento debe estar dentro de otro elemento, sin importar lo profundo que se encuentre.

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

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

```

| p span { color: red; }
| h1 span { color: blue; }

```

Con las reglas CSS anteriores:

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

La sintaxis formal del selector descendente se muestra a continuación:

```
| elemento1 elemento2 elemento3 ... elementoN
```

Los selectores descendentes siempre están formados por dos o más partes separadas entre sí por espacios en blanco. La última parte indica el elemento sobre el que se aplican los estilos y todas las partes anteriores indican el lugar en el que se tiene que encontrar ese elemento para que los estilos se apliquen realmente.

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

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

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

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

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

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

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

```
p a { color: red; }

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

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

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

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

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

2.1.4. Selector de clase

Si se considera el siguiente código HTML de ejemplo:

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

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

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

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

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

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

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

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

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

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

A continuación, se muestra otro ejemplo de selectores de clase:

```
.aviso {
  padding: 0.5em;
  border: 1px solid #98be10;
  background: #f6fed4;
}

.error {
  color: #930;
  font-weight: bold;
}
```



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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

2.1.5. Selectores de ID

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

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

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

```
#destacado { color: red; }

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

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

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

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

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

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

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

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

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

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

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

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

2.1.6. Combinación de selectores básicos

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

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

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

Si se modifica el anterior selector:

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

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

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

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

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

Ejercicio 1 Ver enunciado en la página 190

2.2. Selectores avanzados

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

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

- Lista completa de los selectores que soporta cada versión de cada navegador (<http://dev.l-c-n.com/CSS3-selectors/browser-support.php>)
- Test online en el que puedes comprobar los selectores que soporta el navegador con el que realizas el test (<http://www.css3.info/selectors-test/>)

2.2.1. Selector de hijos

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

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

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

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

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

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

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

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

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

2.2.2. Selector adyacente

El selector adyacente utiliza el signo + y su sintaxis es:

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

La explicación del comportamiento de este selector no es sencilla, ya que selecciona todos los elementos de tipo elemento2 que cumplan las dos siguientes **condiciones**:

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

En el siguiente ejemplo:

```
h1 + h2 { color: red }

<body>
<h1>Título1</h1>

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

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

- El elemento padre de `<h1>` es `<body>`, el mismo padre que el de los dos elementos `<h2>`. Así, los dos elementos `<h2>` cumplen la primera condición del selector adyacente.
- El primer elemento `<h2>` aparece en el código HTML justo después del elemento `<h1>`, por lo que este elemento `<h2>` también cumple la segunda condición del selector adyacente.
- Por el contrario, el segundo elemento `<h2>` no aparece justo después del elemento `<h1>`, por lo que no cumple la segunda condición del selector adyacente y por tanto no se le aplican los estilos de `h1 + h2`.

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

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

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

2.2.3. Selector de atributos

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

Los cuatro tipos de selectores de atributos son:

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

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

```

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

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

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

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

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

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

```

2.3. Agrupación de reglas

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

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

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

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

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

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

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

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

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

2.4. Herencia

Uno de los conceptos más característicos de CSS es la herencia de los estilos definidos para los elementos. Cuando se establece el valor de alguna propiedad en un elemento, todos sus descendientes heredan inicialmente ese mismo valor.

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

```
<!DOCTYPE >
<html >
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
<title>Ejemplo de herencia de estilos</title>
<style type="text/css">
  body { font-family: Arial; color: black; }
  h1 { font-family: Verdana; }
  p { color: red; }
</style>
</head>

<body>
<h1>Titular de la página</h1>
```

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

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

La segunda regla indica que los elementos <h1> se muestran con otra tipografía diferente a la heredada. La tercera regla indica que los elementos <p> varían su color respecto del color que han heredado.

La herencia de estilos no funciona en todas las propiedades CSS, por lo que se debe estudiar cada propiedad de forma individual.

2.5. Colisiones de estilos

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

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

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

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

En este caso, ¿cuál de las dos reglas prevalece, si tenemos en cuenta que se refieren al mismo elemento y están al mismo nivel? La respuesta es muy fácil: **Prevalece siempre la última regla definida**, la cuál mezcla y sobreescribe las propiedades anteriores.

Sin embargo, puede ocurrir que en determinados casos no esté tan claro cuál es el estilo que debería sobreescibir a los anteriores. Ahí es cuando entra en juego el concepto de cascada en CSS, que es el que se encarga de eliminar la ambigüedad y determinar el que tiene prioridad.

Supongamos el siguiente caso, donde tenemos un mismo elemento <div> con un id y una clase:

```
<div id="nombre" class="clase">Texto del elemento</div>

<style>
    div { color: red; }
    #nombre { color: blue; }
    .clase { color: green; }
</style>
```

Tenemos un elemento HTML <div id="nombre" class="clase"> que encaja con los tres bloques del ejemplo anterior. ¿Cómo sabe CSS qué estilo aplicar? ¿Cuál tiene prioridad sobre los demás? Aquí es donde entra en acción el concepto de cascada en CSS.

Cascada CSS

El navegador, para saber qué bloque de estilos tiene prioridad sobre los demás, analiza (por orden) tres conceptos clave del código CSS: su importancia, su especificidad y su orden. Veamos en qué se basa cada uno de ellos.

Importancia

La importancia de un código CSS se determina dependiendo de las hojas de estilo donde está colocado. Generalmente, no necesitaremos preocuparnos de este factor, pero siempre es una buena idea conocer como funciona. Existen varios tipos de hojas de estilo, de menor a mayor importancia:

| Tipo de CSS | Descripción |
|-------------------|---|
| Agente de usuario | Son los estilos CSS que aplica el navegador por defecto. |
| CSS de usuario | Son los estilos CSS que añade el usuario, por razones de personalización. |
| CSS de autor | Son los estilos CSS que coloca el autor de la página. |

Aunque no es recomendable utilizarlo frecuentemente (puede convertirse en una mala práctica), se puede añadir al final de cada regla el texto !important, consiguiendo que la regla en cuestión tenga prioridad sobre las demás, independientemente del nivel o la altura a la que estén:

```
<div>Texto del elemento</div>

<style>
    div {
        color: red !important;
        padding: 8px
    }

    div {
        color: blue;
        background-color: grey
    }
</style>
```

Especificidad

En segundo caso, y si la importancia no elimina la ambigüedad, se pasa a determinar la especificidad de los selectores CSS, que es uno de los criterios más importantes de la cascada de CSS (y también más desconocido).

Para determinar la especificidad de un selector, se sigue un cálculo matemático basado en 4 componentes: A, B, C, D:

| Componente | |
|--------------|---|
| Componente 1 | Estilos aplicados mediante un atributo style . |
| Componente 2 | Número de veces que aparece un id en el selector. |
| Componente 3 | Número de veces que aparece una clase , pseudoclase o atributo en el selector. |
| Componente 4 | Número de veces que aparece un elemento o un pseudoelementos en el selector. |

Para saber si un bloque de CSS es más específico que otro (y por lo tanto, tiene mayor prioridad) sólo hay que calcular sus componentes. Se ordenan teniendo en cuenta los valores de cada componente, de izquierda a derecha. Veamos algunos ejemplos:

```

div { ... }                      /* Especificidad: 0,0,0,1 */
div div { ... }                   /* Especificidad: 0,0,0,2 */
#pagina div { ... }              /* Especificidad: 0,1,0,1 */
#pagina div:hover { ... }        /* Especificidad: 0,1,1,1 */
#pagina div:hover a { ... }      /* Especificidad: 0,1,1,2 */
#pagina .sel:hover>a { ... }    /* Especificidad: 0,1,2,1 */

```

En <https://specificity.keegan.st/> tienes una excelente calculadora de especificidad CSS donde podrás calcular la especificidad de un selector CSS rápida y cómodamente.

Orden

En CSS, es posible crear múltiples reglas CSS para definir un mismo concepto. En este caso, la que prevalece ante todas las demás depende de ciertos factores, como es la «altura» a la que está colocada la regla:

- El CSS en linea en un elemento HTML es el que tiene mayor precedencia, por lo que siempre será el que tenga prioridad sobre otras reglas CSS. Recuerda que son los estilos incluidos en una etiqueta HTML a través del atributo style.
- En segundo lugar, el CSS interno definido a través de bloques <style> en el propio documento HTML será el siguiente a tener en cuenta en orden de prioridad.

- Por último, los documentos CSS externos son la tercera opción de prioridad a la hora de tomar en cuenta las reglas CSS. Son aquellos que se relacionan en un documento HTML a través de la etiqueta <link>.

Teniendo esto en cuenta, hay que recordar que las propiedades que prevalecerán serán las que estén en último lugar, siempre respetando la prioridad de la lista anterior.

Ejercicio 2 Ver enunciado en la página 191

Capítulo 3. Unidades de medida y colores

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

3.1. Unidades de medida

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

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

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

3.1.1. Unidades relativas

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 se toma para determinar su valor real:

Unidades relativas



- `em`, (no confundir con la etiqueta `` de HTML) relativa respecto del tamaño de letra empleado. Aunque no es una definición exacta, el valor de `1em` se puede aproximar por la anchura de la letra M ("eme mayúscula") del tipo de letra que se esté utilizando.
- `rem`, es parecida a la anterior pero sin aplicarle la anidación.

- **ex**, relativa respecto de la altura de la letra x ("equis minúscula") del tipo de letra que se esté utilizando
- **px** (píxel) relativa respecto de la pantalla del usuario

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

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

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

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

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

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

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

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

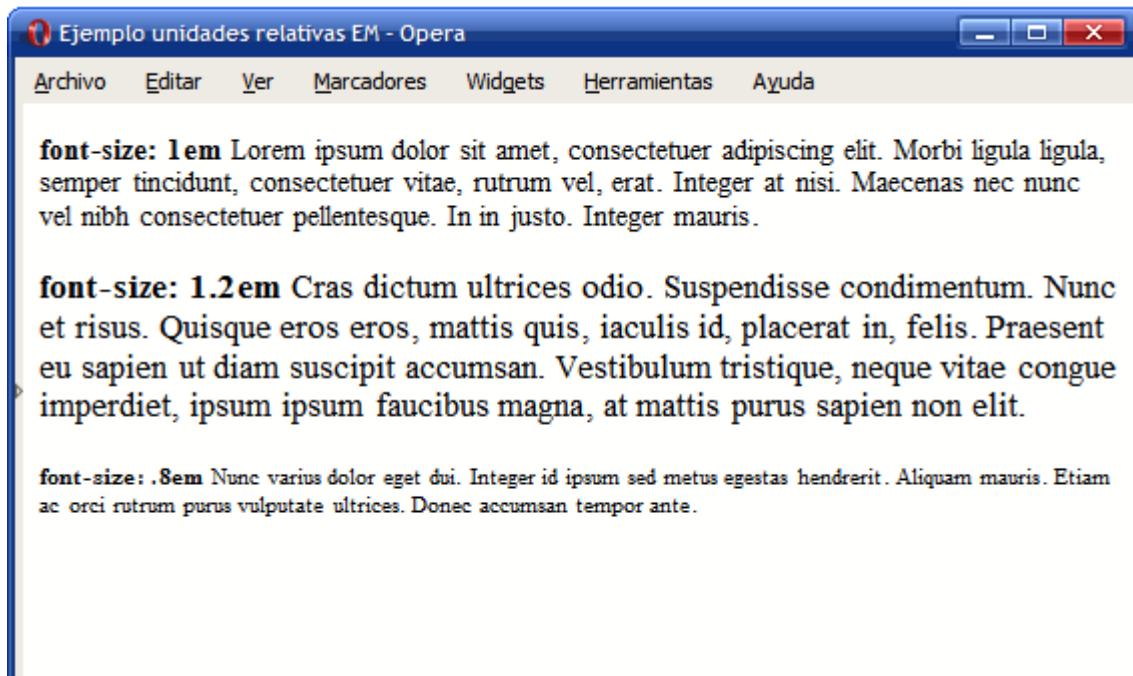


Figura 3.1. Ejemplo de tamaño de letra definido con la unidad relativa **em**

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

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

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

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

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

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

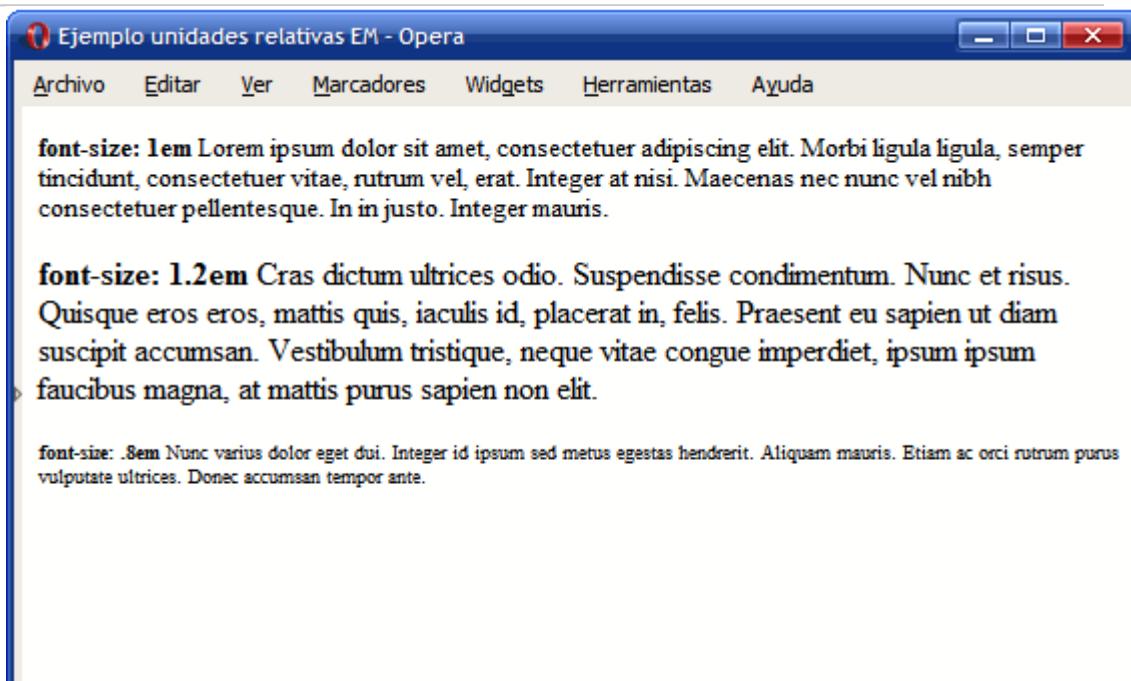


Figura 3.2. Ejemplo de tamaño de letra definido con la unidad relativa em

Al haber reducido el tamaño de letra que era la referencia del tamaño de letra de los tres párrafos, su texto se ve con una letra más pequeña, aunque manteniendo las proporciones: el

primer párrafo se ve con un tamaño de letra normal, el segundo se ve con un tamaño grande y el tercero se visualiza con un tamaño de letra más pequeño de lo normal.

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

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

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

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

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

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

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

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

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

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

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

3.1.2. Unidades absolutas

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

Unidades absolutas



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

A continuación se muestran ejemplos de utilización de unidades absolutas:

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

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

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

3.1.3. Porcentajes

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

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

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

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

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

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

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

En el ejemplo anterior, la referencia del valor `80%` es la anchura de su elemento padre. Por tanto, el elemento `<div>` cuyo atributo `class` vale `principal` tiene una anchura de $80\% \times 600\text{px} = 480\text{px}$.

3.1.4. Unidades flexibles (viewport)

Existen unas unidades de "nueva generación" que resultan muy útiles, porque dependen del **viewport** (región visible de la página web en el navegador). Con estas unidades podemos hacer referencia a un porcentaje concreto del tamaño específico que tengamos en la ventana del navegador, independientemente de si es redimensionado o no.

Estas unidades son las siguientes:

| Unidad | Significado | Medida aproximada |
|-------------|-----------------------|---|
| vw | viewport width | 1vw = 1% ancho de navegador |
| vh | viewport height | $1vh = 1\% \text{ alto de navegador}$ |
| vmin | viewport minimum | $1vmin = 1\% \text{ de alto o ancho (el mínimo)}$ |
| vmax | viewport maximum | $1vmax = 1\% \text{ de alto o ancho (el máximo)}$ |

La unidad **vw** hace referencia al ancho del **viewport**, mientras que **vh** hace referencia al **alto**. Por ejemplo, si utilizamos `100vw` estaremos haciendo referencia al **100%** del ancho del navegador, o sea, todo lo que se está viendo de ancho en pantalla, mientras que si indicamos `50vh` estaremos haciendo referencia a la mitad del ancho del navegador.

Por último, tenemos **vmin** y **vmax**, que simplemente se utilizan para hacer referencia al **porcentaje de ancho o alto del viewport, dependiendo cual sea más pequeño o más grande de los dos**, lo que puede ser útil en algunas situaciones donde quieras flexibilidad con diseños adaptables.

3.1.5. Recomendaciones

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

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

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

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

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

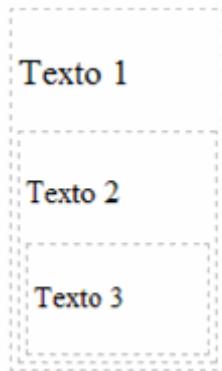


Figura 3.3. El texto cada vez se ve más pequeño

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

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

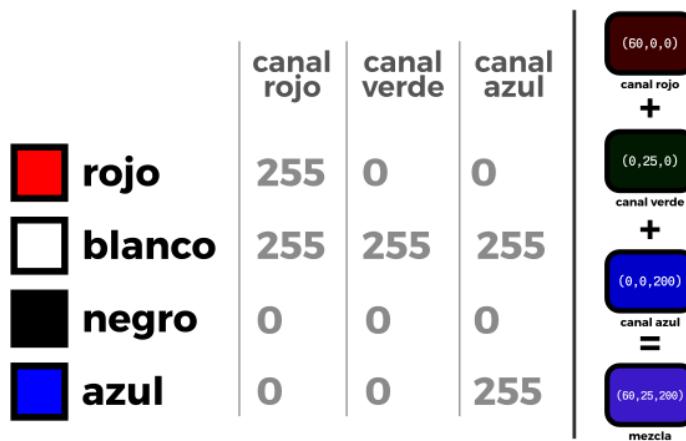
Por último, el tamaño de letra del tercer `<div>` será igual a `0.9em` respecto al tamaño de la letra

del segundo elemento `<div>`, es decir, $0.9\text{em} \times 0.9\text{em} \times 0.9\text{em} = 0.729\text{em}$ respecto del tamaño de letra por defecto. De esta forma, el tamaño de letra de este tercer `<div>` es mucho más pequeño que el del primer `<div>`. Si se anidan varios elementos `<div>`, la letra se hará tan pequeña que no será posible leerla.

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

3.2. Colores

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



3.2.1. Palabras clave

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

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



Figura 3.4. Colores definidos mediante las palabras clave de CSS

Pero hay muchas más: https://developer.mozilla.org/es/docs/Web/CSS/color_value

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 <http://en.wikipedia.org/wiki/Websafe>.

3.2.2. RGB decimal

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

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

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

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

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

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

3.2.3. RGB porcentual

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

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

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

3.2.4. RGB hexadecimal

Aunque es el método más complicado 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.

Por tanto, para definir un color en CSS con RGB hexadecimal se realizan los siguientes pasos:

1. Se determinan las componentes RGB del color original, por ejemplo: R = 71, G = 98, B = 176.
2. El valor numérico de cada componente se transforma 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.
3. Para obtener el color completo en formato RGB hexadecimal, se concatenan los valores 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 ejemplo de las secciones anteriores, el color del párrafo se indica de la siguiente forma utilizando el color en 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:

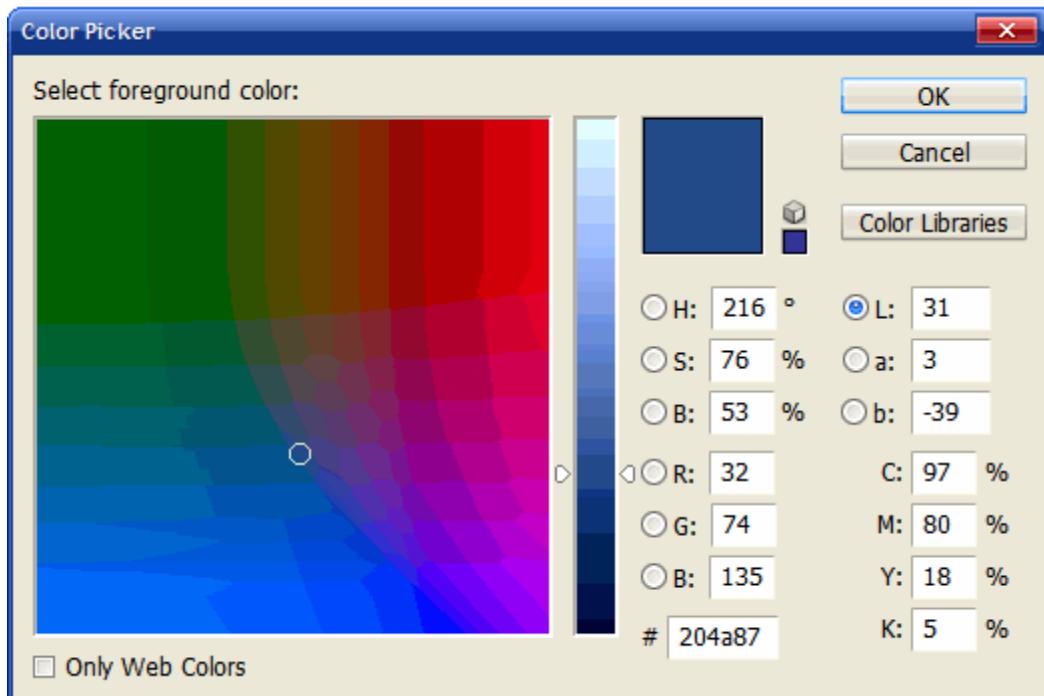


Figura 3.5. Herramienta de color de Photoshop para definir los colores según los modelos RGB, CMYK, Lab, HSB y RGB hexadecimal

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

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

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

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

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

3.2.5. Colores 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.

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

3.2.6. Canales Alfa

En algunos casos, es muy posible que deseemos indicar un color que tenga cierto grado de transparencia, y de esta forma, refleje el contenido, color o imágenes que se encuentren detrás parcial o completamente.

Existe la posibilidad de utilizar los denominados canales alfa, que permiten establecer un porcentaje de transparencia parcial sobre un color. Estos se pueden establecer en los diferentes formatos vistos hasta ahora, salvo en los colores con palabras clave.

`rgba(0, 0, 0, 0.5);`

Capítulo 4. Modelo de cajas (box model)

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 "box model" es el comportamiento de CSS que hace que todos los elementos incluidos en una página HTML se representen mediante cajas rectangulares. CSS permite controlar el aspecto de todas las cajas.

El diseño de cualquier página HTML está compuesto por cajas rectangulares. CSS permite definir la altura y anchura de cada caja, el margen existente entre cajas y el espacio de relleno interior que muestra cada caja. Además, CSS permite controlar la forma en la que se visualizan las cajas: se pueden ocultar, desplazar respecto de su posición original y fijarlas en una posición específica dentro del documento.



Las cajas de una página se crean automáticamente. Cada vez que se inserta una etiqueta o elemento en la página, se crea una nueva caja rectangular que encierra los contenidos del

elemento. El siguiente esquema muestra la creación automática de cajas por parte de HTML para cada elemento definido en el código HTML de la página:

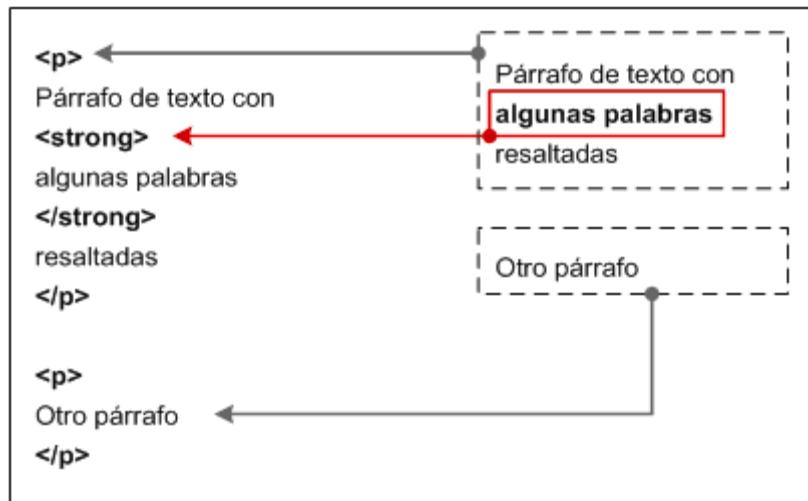


Figura 4.2. Las cajas se crean automáticamente al definir cada elemento HTML

Cada una de las cajas está formada por seis partes, tal y como se muestra en la siguiente imagen tridimensional:

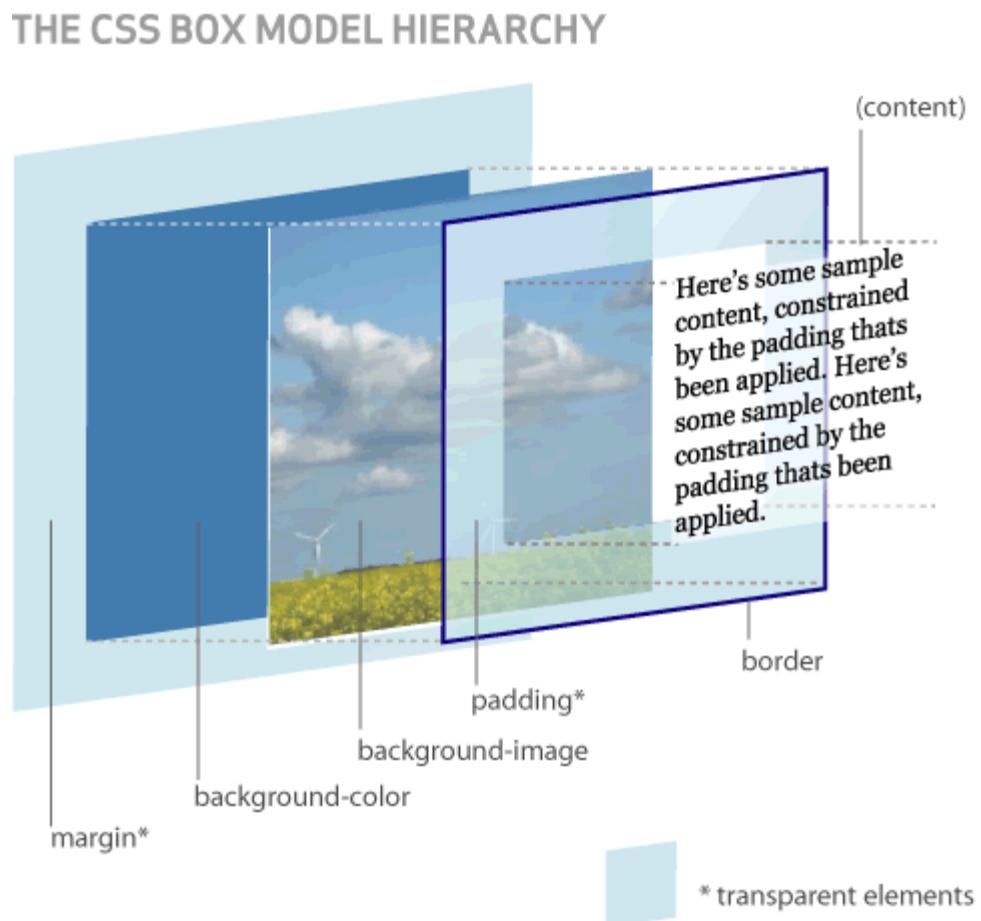


Figura 4.3. Representación tridimensional del box model de CSS

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 entre el contenido y el borde que lo encierra.
- **Borde (border):** línea que encierra completamente el contenido y su relleno.
- **Imagen de fondo (background image):** imagen que se muestra por 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):** espacio libre entre la caja y las posibles cajas adyacentes.

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

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

4.1. Anchura y altura

4.1.1. Anchura

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

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

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

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

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

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

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

4.1.2. Altura

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

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

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

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

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

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

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

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

4.2. Margen y relleno

4.2.1. Margen

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

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

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

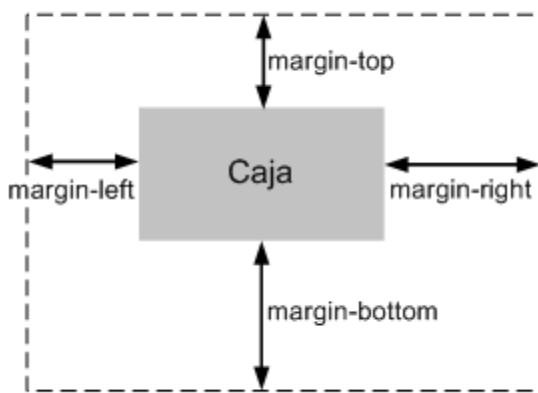


Figura 4.4. Las cuatro propiedades relacionadas con los márgenes

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

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

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

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

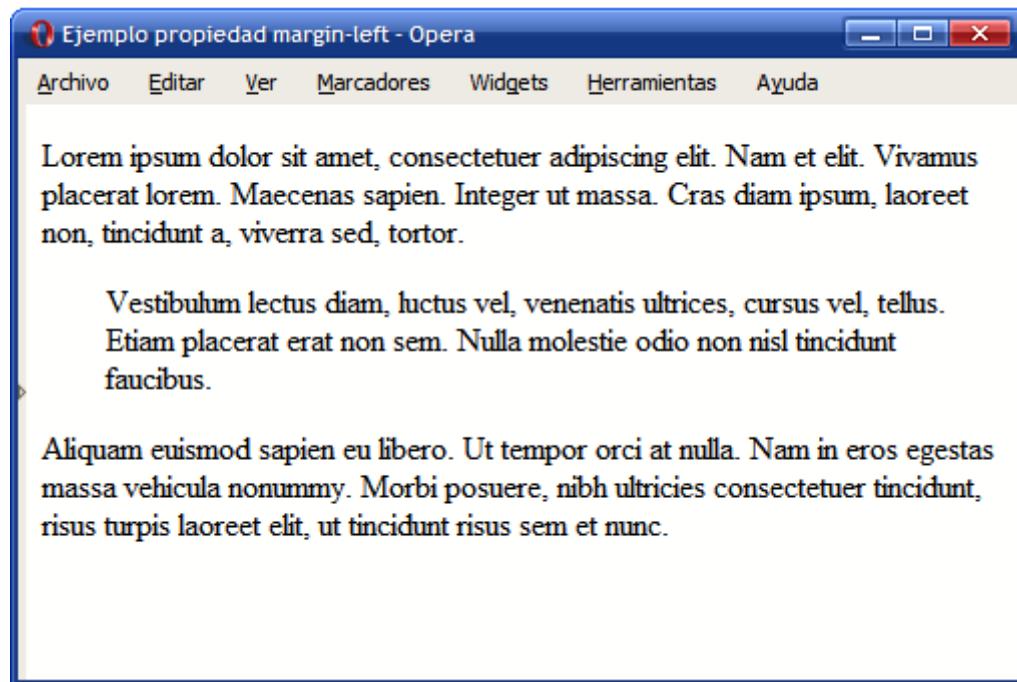


Figura 4.5. Ejemplo de propiedad margin-left

Algunos diseñadores web utilizan la etiqueta <blockquote> para encerrar los contenidos de un párrafo que se quiere mostrar tabulado respecto al resto de contenidos, como en el ejemplo anterior. Se trata de un error grave porque utiliza código XHTML erróneo para modificar el aspecto de los contenidos. Como ya se sabe, CSS es el único responsable del aspecto de los contenidos y dispone de propiedades como margin-left que permite conseguir los mismos resultados de forma correcta.

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:

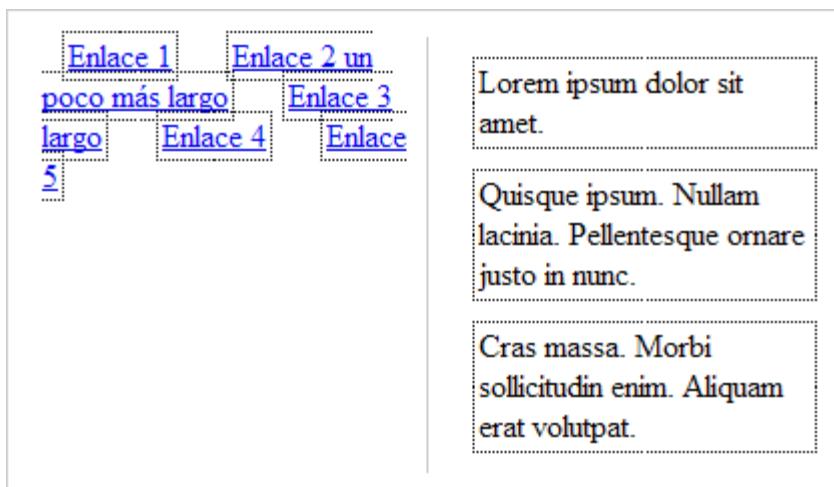


Figura 4.6. Los márgenes verticales sólo se aplican a los elementos de bloque e imágenes

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.

El siguiente ejemplo utiliza el mismo valor en los cuatro márgenes de cada imagen para facilitar su identificación y mejorar el diseño general de la página:

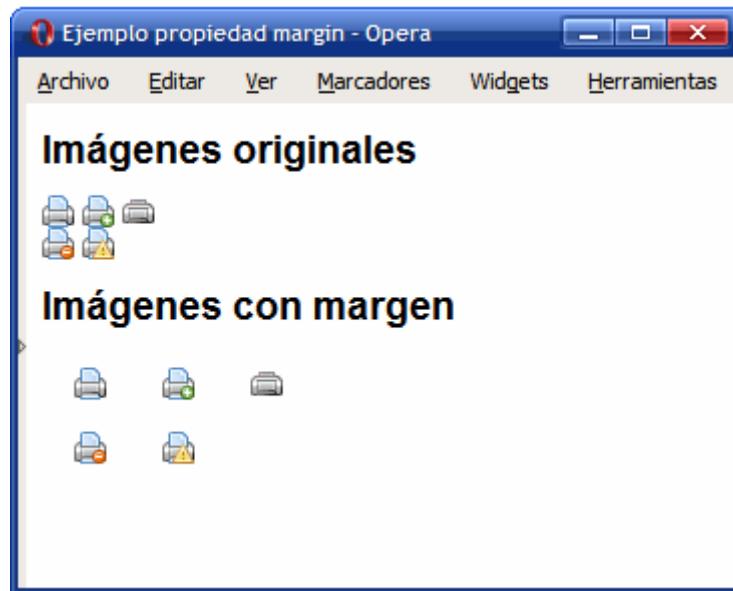


Figura 4.7. Ejemplo de propiedad margin

El código CSS del ejemplo anterior se muestra a continuación:

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

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

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

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

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.

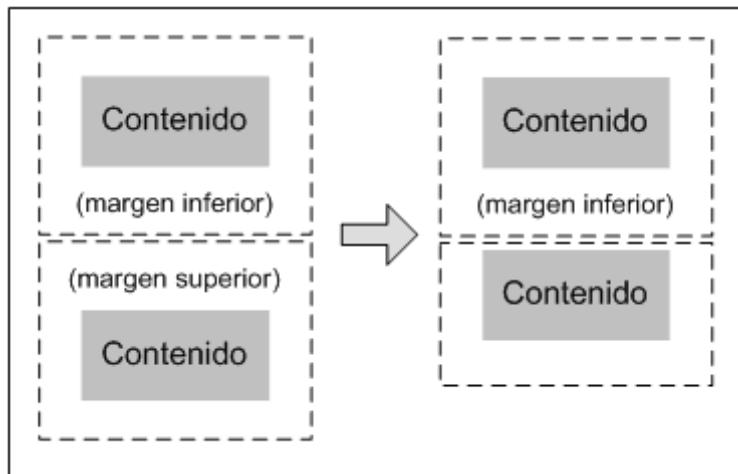


Figura 4.8. Fusión automática de los márgenes verticales

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:

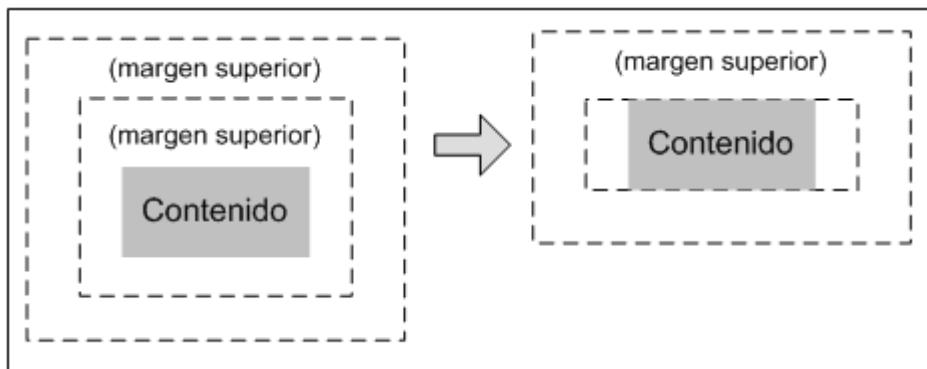


Figura 4.9. Fusión de los márgenes de los elementos interiores

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.

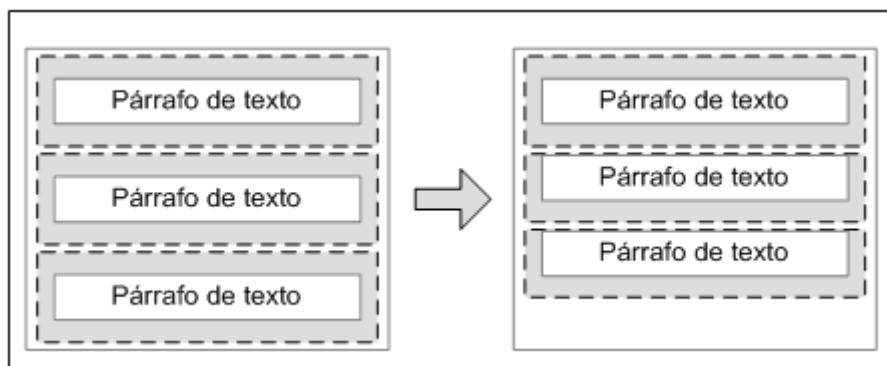


Figura 4.10. Motivo por el que se fusionan automáticamente los márgenes verticales

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

4.2.2. Relleno

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

| | |
|-----------------------|--|
| padding-top | Relleno superior |
| padding-right | Relleno derecho |
| padding-bottom | Relleno inferior |
| padding-left | Relleno izquierdo |
| Valores | <code><medida> <porcentaje> inherit</code> |
| Se aplica a | Todos los elementos excepto algunos elementos de tablas como grupos de cabeceras y grupos de pies de tabla |
| Valor inicial | 0 |
| Descripción | Establece cada uno de los rellenos horizontales y verticales de un elemento |

Cada una de las propiedades establece la separación entre el lateral de los contenidos y el borde lateral de la caja:

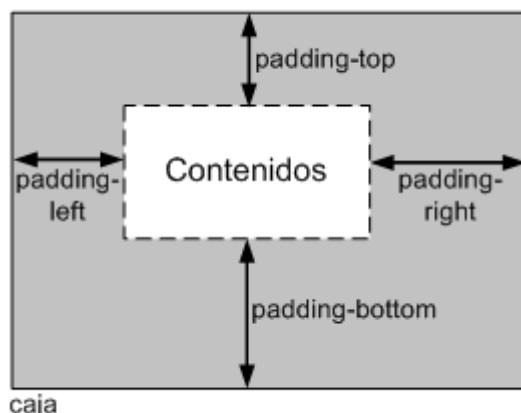


Figura 4.11. Las cuatro propiedades relacionadas con los rellenos

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

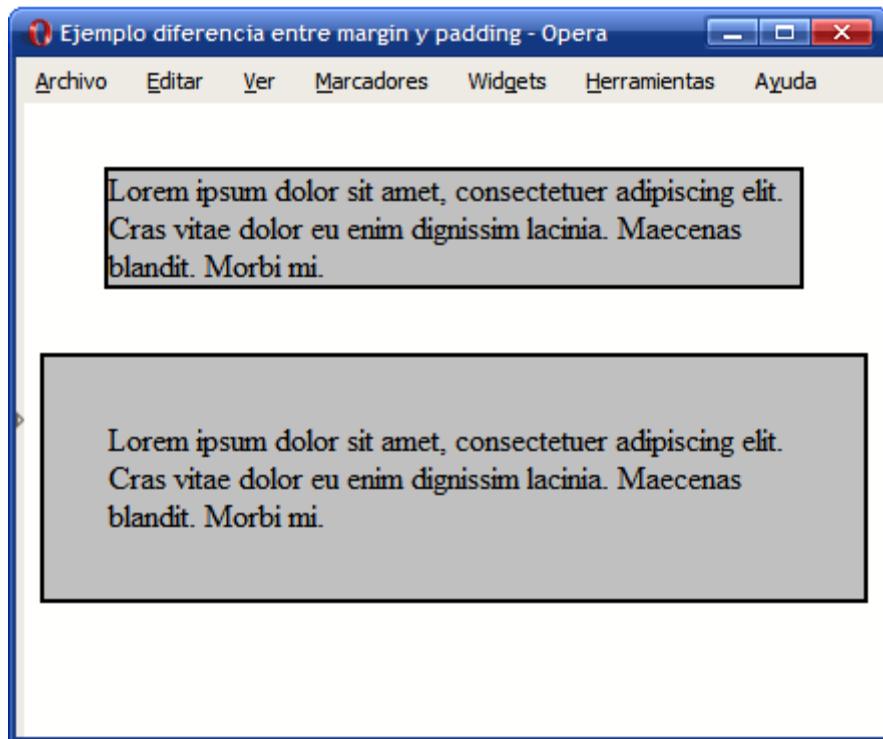


Figura 4.12. Diferencia entre relleno (padding) y margen (margin)

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

```
.margen {
    margin-top: 2em; margin-right: 2em; margin-bottom: 2em; margin-left: 2em;
}
.relleno {
    padding-top: 2em; padding-right: 2em; padding-bottom: 2em; padding-left: 2em;
}

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

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

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

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

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

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

Ejercicio 3 Ver enunciado en la página 193

4.3. Bordes

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

4.3.1. Anchura

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

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

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

La medida más habitual para indicar la anchura de los bordes es el píxel, ya que permite un control preciso del grosor. Las palabras clave apenas se utilizan, ya que impiden mostrar bordes iguales en diferentes navegadores. El motivo es que el estándar CSS no indica explícitamente el grosor al que equivale cada palabra clave. Así por ejemplo, el grosor **medium** equivale a 4px en Internet Explorer y 3px en el resto de navegadores.

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



Figura 4.13. Ejemplo de propiedad border-width

Las reglas CSS utilizadas se muestran a continuación:

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

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

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

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

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

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

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

4.3.2. Color

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

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

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

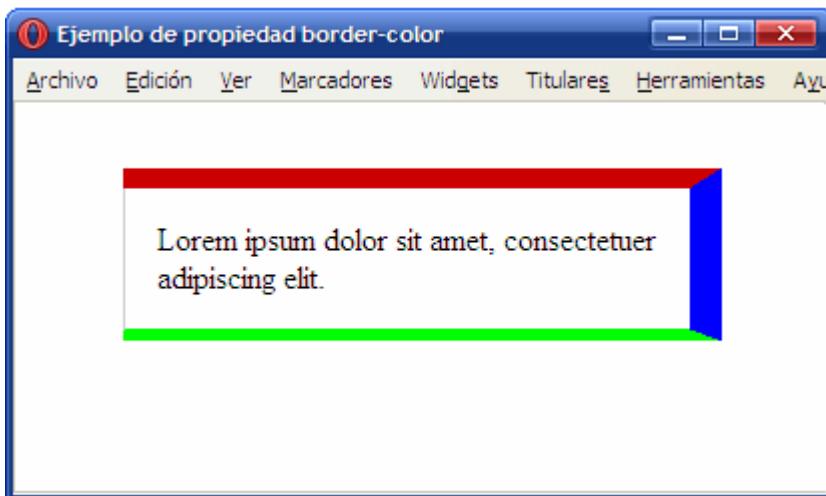


Figura 4.14. Ejemplo de propiedad border-color

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

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

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

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

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

4.3.3. Estilo

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

| | |
|----------------------------|--|
| border-top-style | Estilo del borde superior |
| border-right-style | Estilo del borde derecho |
| border-bottom-style | Estilo del borde inferior |
| border-left-style | Estilo del borde izquierdo |
| Valores | none hidden dotted dashed solid double groove ridge inset outset inherit |
| Se aplica a | Todos los elementos |
| Valor inicial | none |
| Descripción | Establece el estilo de cada uno de los cuatro bordes de los elementos |

El estilo de los bordes sólo se puede indicar mediante alguna de las palabras reservadas definidas por CSS. 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:

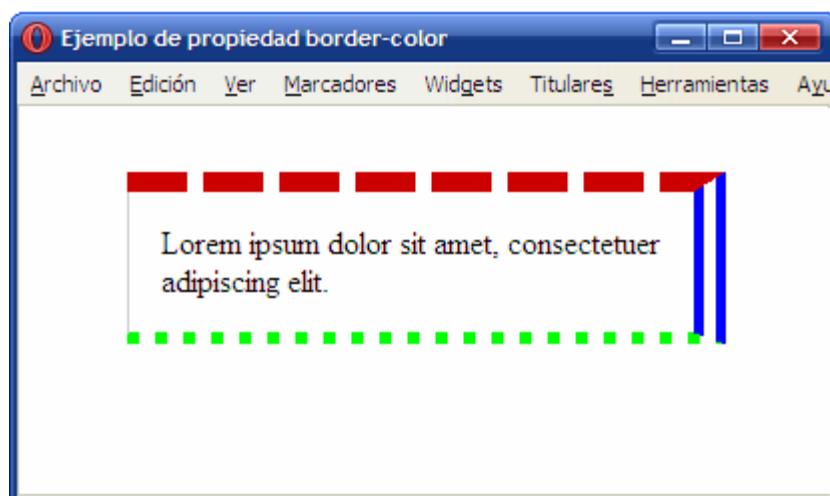


Figura 4.15. Ejemplo de propiedad border-style

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:

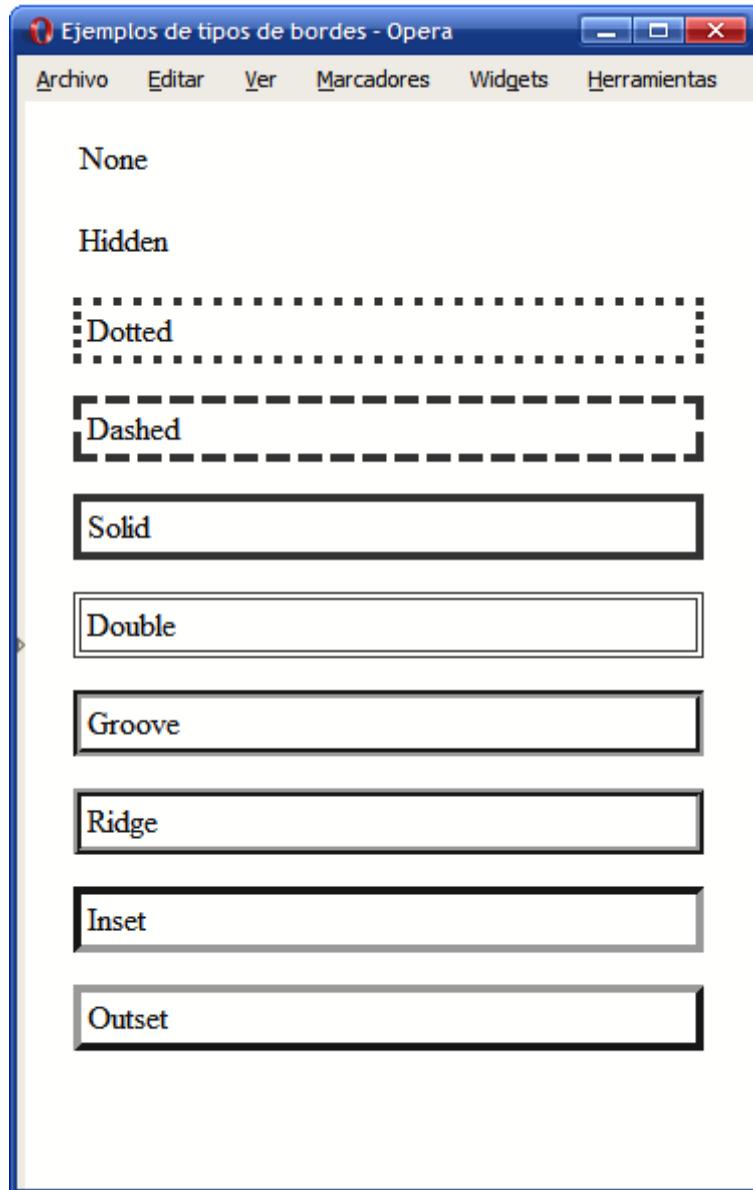


Figura 4.16. Tipos de bordes definidos por CSS

Los bordes más utilizados en los diseños habituales 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.

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

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

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

4.3.4. Propiedades shorthand

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

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

```

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

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

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

```

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

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

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

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

| | |
|----------------------|---|
| border | Estilo completo de todos los bordes |
| Valores | (<code><medida_borde></code> <code><color_borde></code> <code><estilo_borde></code>) <code>inherit</code> |
| 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;
}
```

Ejercicio 4 Ver enunciado en la página 195

4.4. Margen, relleno, bordes y modelo de cajas

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

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

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

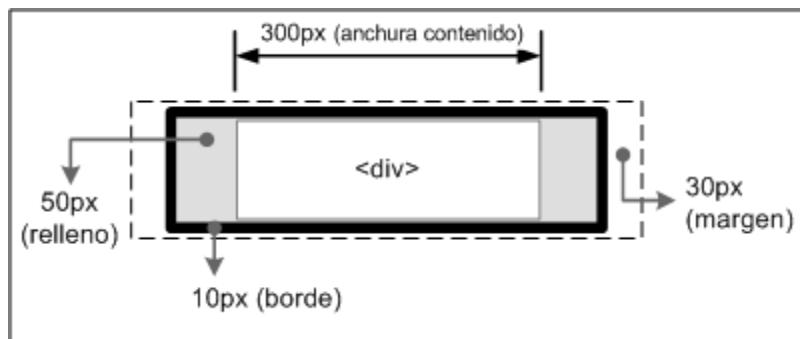


Figura 4.17. La anchura total de un elemento tiene en cuenta los márgenes, rellenos y bordes

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*.

4.5. BOX-SIZING

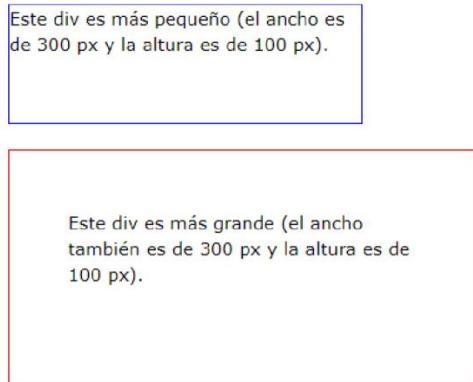
La box-sizing propiedad CSS nos permite incluir el relleno y el borde en el ancho y alto total de un elemento.

Por defecto, el ancho y el alto de un elemento se calcula así:

- ancho + relleno + borde = ancho real de un elemento
- alto + relleno + borde = altura real de un elemento

Esto significa: cuando establece el ancho / alto de un elemento, el elemento a menudo parece más grande de lo que ha establecido (porque el borde y el relleno del elemento se agregan al ancho / alto especificado del elemento).

La siguiente ilustración muestra dos elementos con el mismo ancho y alto especificados:



Los dos elementos <div> anteriores terminan con diferentes tamaños en el resultado (porque div2 tiene un relleno especificado):

```
.div1 {  
    width: 300px;  
    height: 100px;  
    border: 1px solid blue;  
}  
.div2 {  
    width: 300px;  
    height: 100px;  
    padding: 50px;  
    border: 1px solid red;  
}
```

La box-sizing propiedad nos permite incluir el relleno y el borde en el ancho y alto total de un elemento.

Si establece box-sizing: border-box; en un elemento, el relleno y el borde se incluyen en el ancho y la altura: ¡Ambos divs son del mismo tamaño ahora!

```
.div1 {  
    width: 300px;  
    height: 100px;  
    border: 1px solid blue;  
    box-sizing: border-box; }  
.div2 {  
    width: 300px;  
    height: 100px;  
    padding: 50px;  
    border: 1px solid red;  
    box-sizing: border-box; }
```

4.5. Fondos

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

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

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

| background-color Color de fondo | |
|--|--|
| Valores | <code><color></code> <code>transparent</code> <code>inherit</code> |
| Se aplica a | Todos los elementos |
| Valor inicial | <code>transparent</code> |
| Descripción | Establece un color de fondo para los elementos |

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

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

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

| background-image Imagen de fondo | |
|---|---|
| Valores | <code><url></code> <code>none</code> <code>inherit</code> |
| Se aplica a | Todos los elementos |
| Valor inicial | <code>none</code> |
| 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/imagen.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.

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 4.20. Imagen original utilizada para el fondo de la página

Reglas CSS

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

Resultado

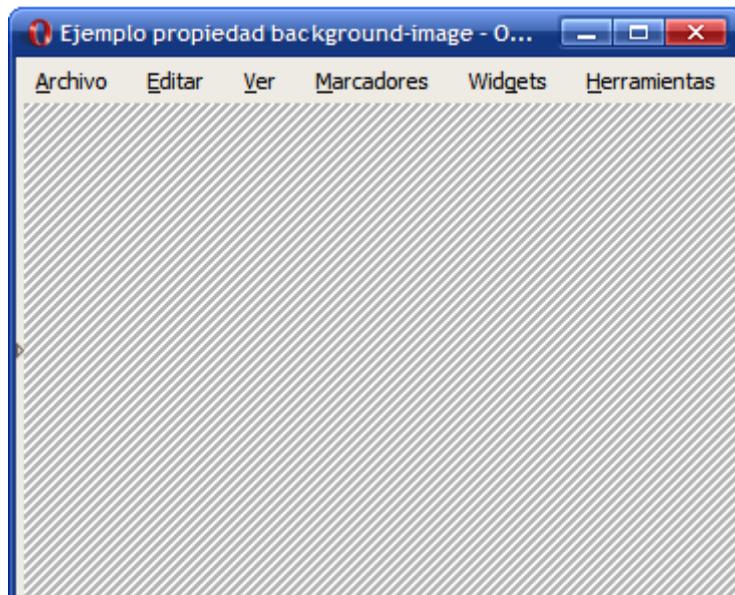


Figura 4.21. Página con una imagen de fondo

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.

background-repeat Repetición de la imagen de fondo

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

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

Las reglas CSS definidas para la cabecera son:

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

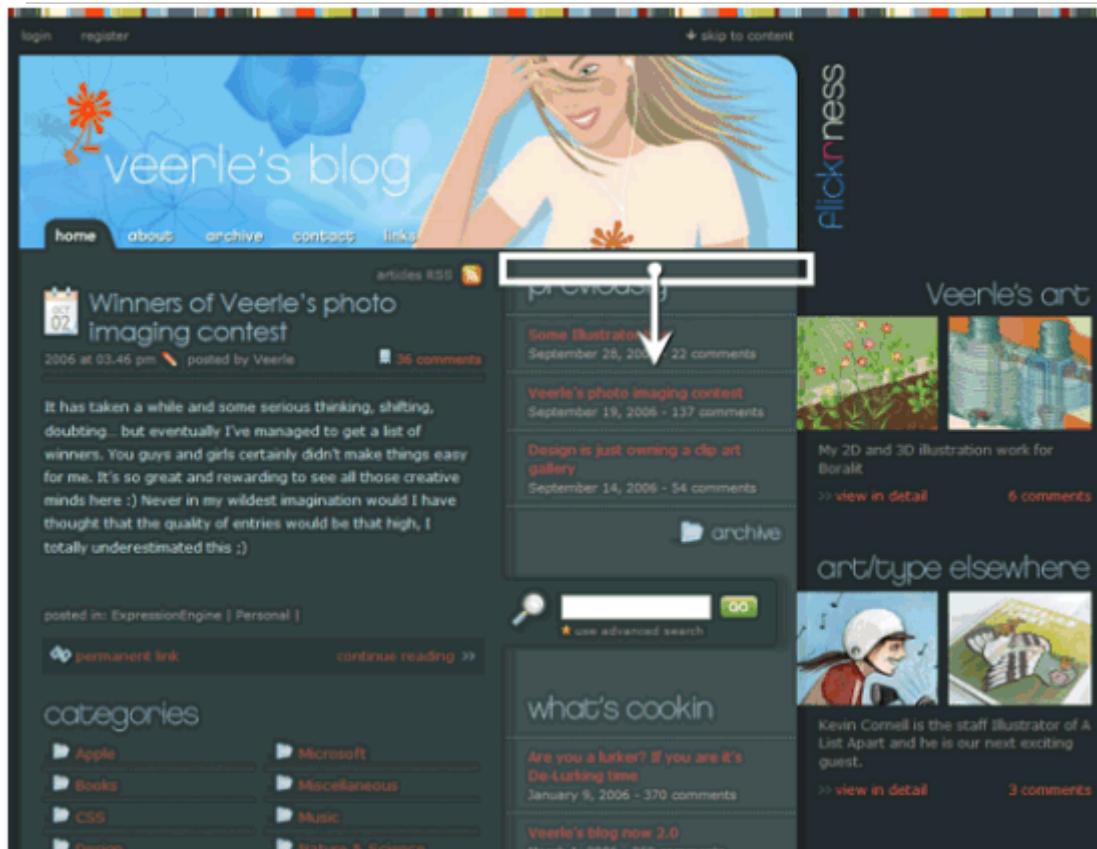


Figura 4.23. Uso de repeat-y en la página de Veerle.duoh.com

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`.

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

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

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

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

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

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

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

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

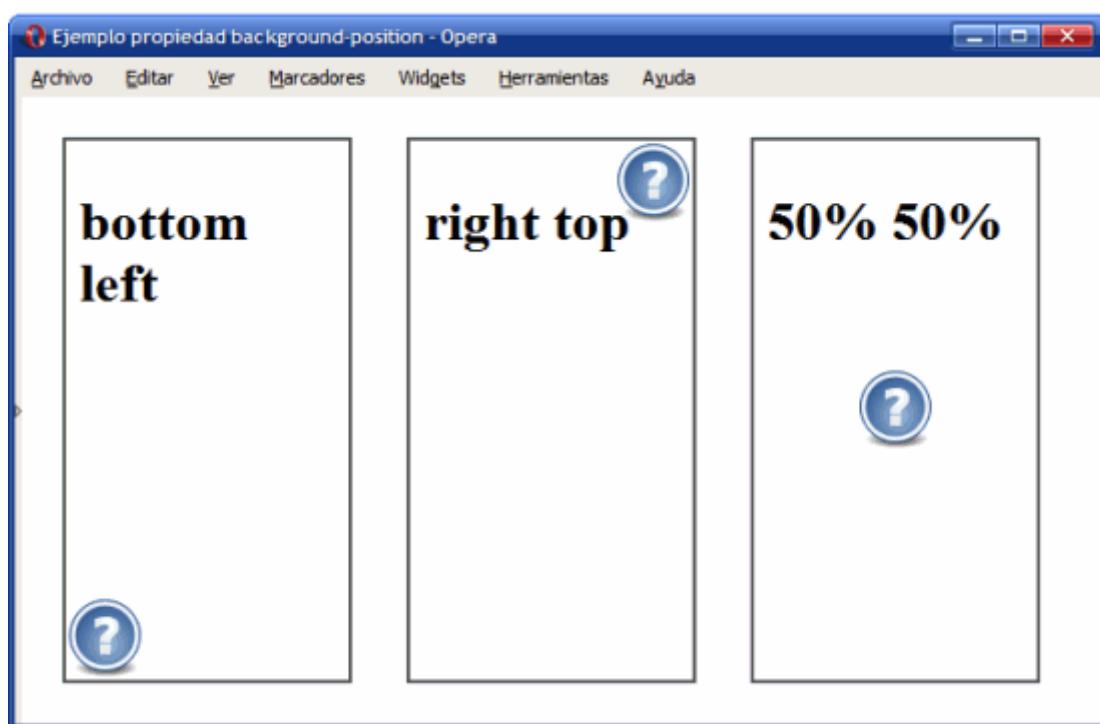


Figura 4.24. Ejemplo de propiedad `background-position`

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>
```

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.

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

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

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

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

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

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

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

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

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

background: none;

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

Ejercicio 5 Ver enunciado en la página 197

Capítulo 5. Posicionamiento y visualización

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

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

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

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

5.1. Tipos de elementos

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

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

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

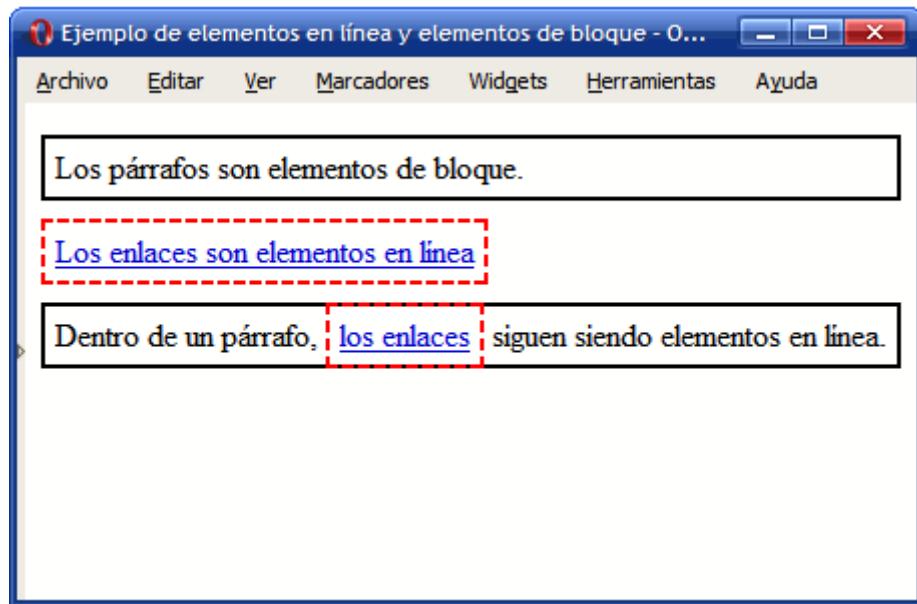


Figura 5.1. Cajas creadas por los elementos de línea y los elementos de bloque

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

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

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

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

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

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

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

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

5.2. Posicionamiento

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

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

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

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

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

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

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

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

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

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

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

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

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

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

5.3. Posicionamiento normal

El posicionamiento normal o estático es el modelo que utilizan por defecto los navegadores para mostrar los elementos de las páginas. En este modelo, ninguna caja se desplaza respecto de su posición original, por lo que sólo se tiene en cuenta si el elemento es de bloque o en línea.

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

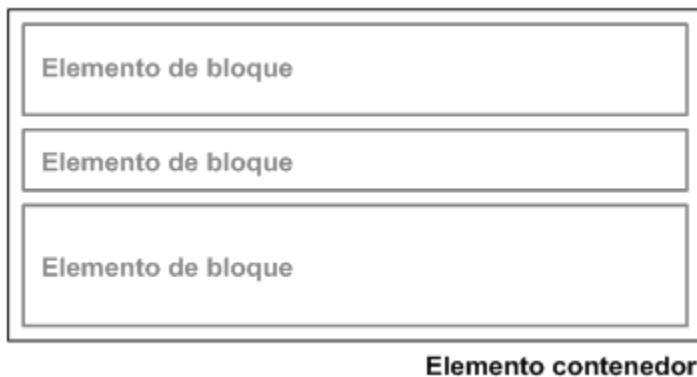


Figura 5.2. Posicionamiento normal de los elementos de bloque

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

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

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

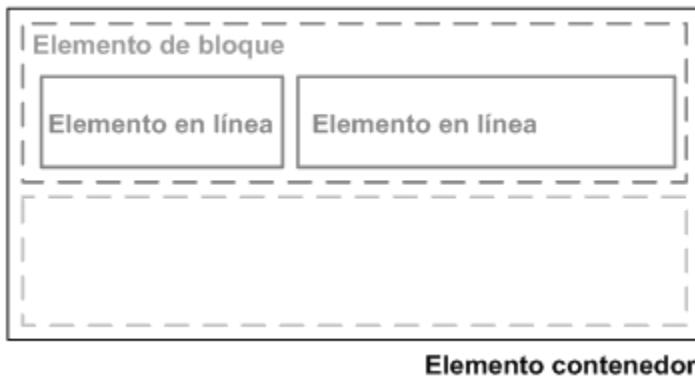


Figura 5.3. Posicionamiento normal de los elementos en línea

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

5.4. Posicionamiento relativo

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

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

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

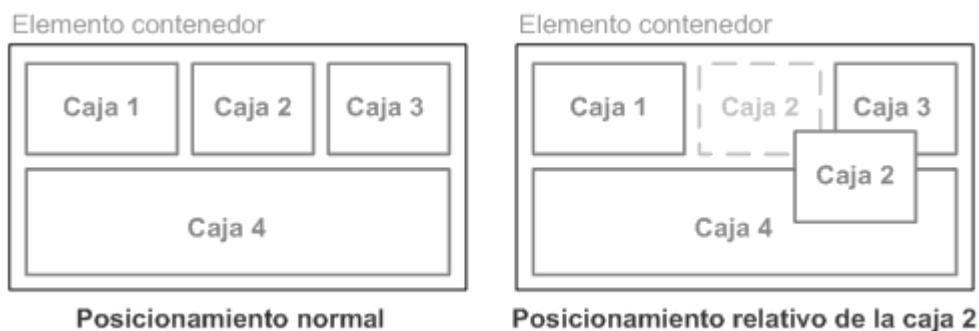


Figura 5.4. Ejemplo de posicionamiento relativo de un elemento

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

La propiedad `left` desplaza la caja hacia su derecha, la propiedad `right` la desplaza hacia su izquierda, la posición `top` desplaza la caja de forma descendente y la propiedad `bottom` desplaza la caja de forma ascendente. **Si se utilizan valores negativos en estas propiedades, su efecto es justamente el inverso.**

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

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

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

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

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

El siguiente ejemplo muestra tres imágenes posicionadas de forma normal:

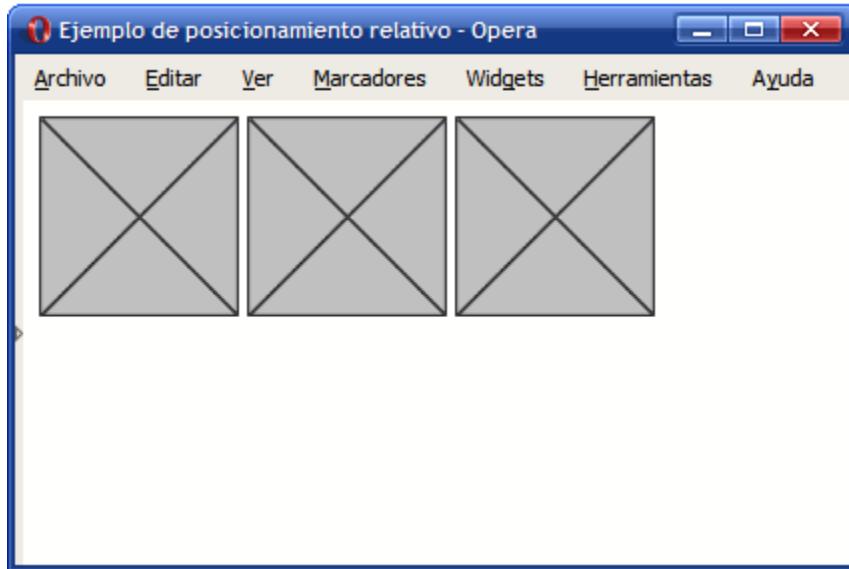


Figura 5.5. Elementos posicionados de forma normal

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

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

```

El aspecto que muestran ahora las imágenes es el siguiente:



Figura 5.6. Elemento posicionado de forma relativa

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

5.5. Posicionamiento absoluto

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

Cuando una caja se posiciona de forma absoluta, el resto de elementos de la página la ignoran y ocupan el lugar original ocupado por la caja posicionada. Al igual que en el posicionamiento relativo, cuando se posiciona de forma absoluta una caja es probable que se produzcan solapamientos con otras cajas.

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



Figura 5.7. Ejemplo de posicionamiento absoluto de un elemento

La caja 2 está posicionada de forma absoluta, lo que implica que el resto de elementos ignoran que esa caja exista. Por este motivo, la caja 3 deja su lugar original y pasa a ocupar el hueco dejado por la caja 2.

En el estándar de CSS, esta característica de las cajas posicionadas de forma absoluta se explica como que la caja *sale por completo del flujo normal del documento*. De hecho, las cajas posicionadas de forma absoluta parecen que están en un nivel diferente al resto de elementos de la página.

Por otra parte, el desplazamiento de una caja posicionada de forma absoluta se indica mediante las propiedades `top`, `right`, `bottom` y `left`. A diferencia de posicionamiento relativo, en este caso *la referencia de los valores de esas propiedades es el origen de coordenadas de su primer elemento contenedor posicionado*.

Determinar el origen de coordenadas a partir del cual se desplaza una caja posicionada de forma absoluta es un proceso complejo que se compone de los siguientes pasos:

- Se buscan todos los elementos contenedores de la caja hasta llegar al elemento `<body>` de la página.
- Se recorren todos los elementos contenedores empezando por el más cercano a la caja y llegando hasta el `<body>`
- De todos ellos, el navegador se queda con el primer elemento contenedor que esté posicionado de cualquier forma diferente a `position: static`
- La esquina superior izquierda de ese elemento contenedor posicionado es el origen de coordenadas.

Una vez obtenido el origen de coordenadas, se interpretan los valores de las propiedades `top`, `right`, `bottom` y `left` respecto a ese origen y se desplaza la caja hasta su nueva posición.

En los siguientes ejemplos, se va a utilizar la página HTML que muestra la siguiente imagen:

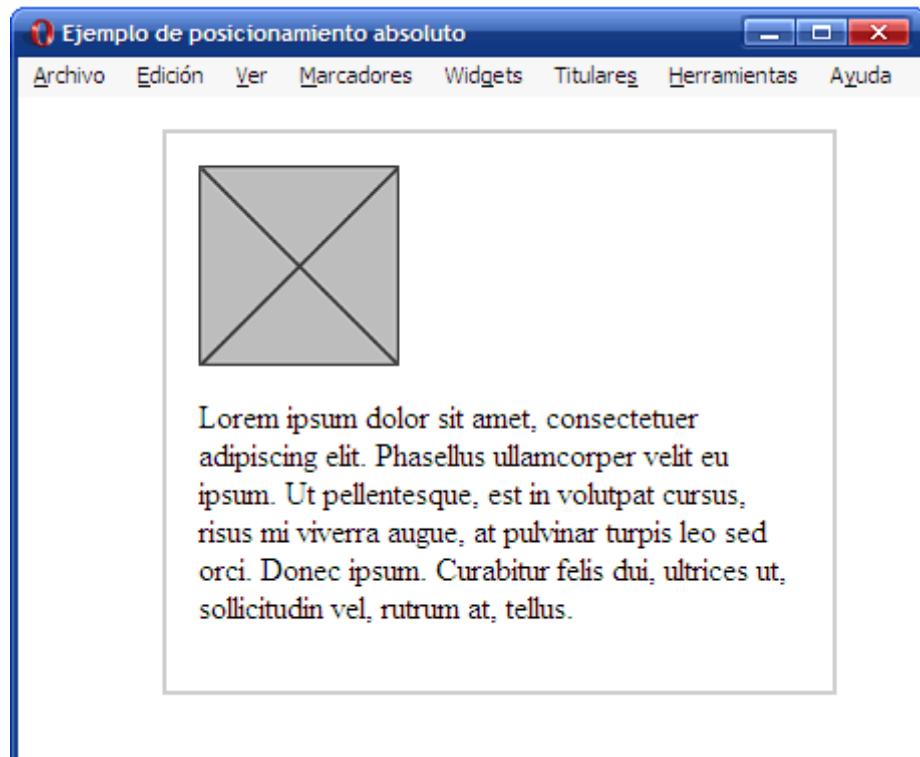


Figura 5.8. Situación original antes de modificar el posicionamiento

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

```
div {  
    border: 2px solid #CCC;  
    padding: 1em;  
    margin: 1em 0 1em 4em;  
    width: 300px;  
}  
  
<div>  
      
    <p>Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Phasellus ullamcorper velit eu ipsum. Ut pellentesque, est in volutpat cursus, risus mi viverra augue, at pulvinar turpis leo sed orci. Donec ipsum. Curabitur felis dui, ultrices ut, sollicitudin vel, rutrum at, tellus.</p>  
</div>
```

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

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

El resultado visual se muestra en la siguiente imagen:

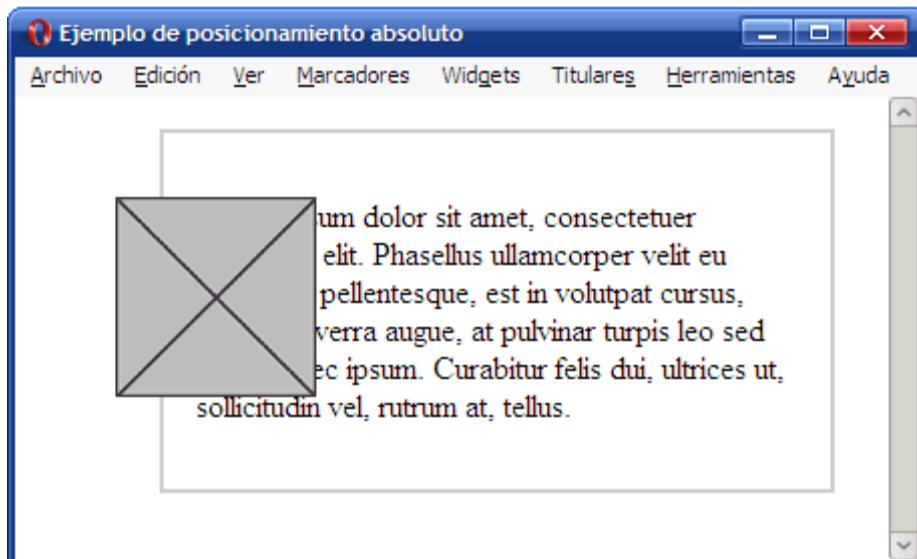


Figura 5.9. Imagen posicionada de forma absoluta

La imagen posicionada de forma absoluta no toma como origen de coordenadas la esquina superior izquierda de su elemento contenedor `<div>`, sino que su referencia es la esquina superior izquierda de la página:

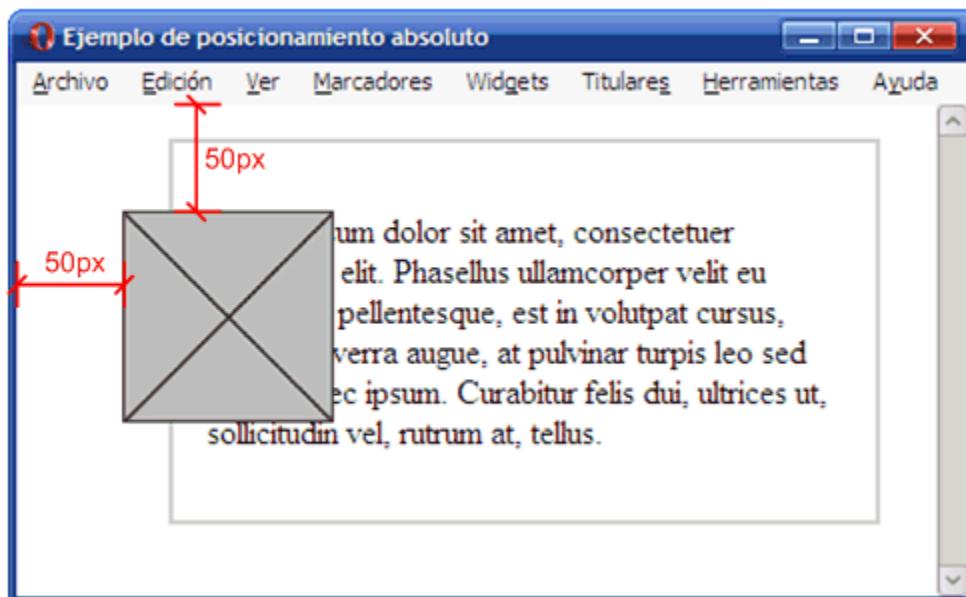


Figura 5.10. La referencia del posicionamiento absoluto es la página entera

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

1. Obtiene la lista de elementos contenedores de la imagen: `<div>` y `<body>`.
2. Recorre la lista de elementos desde el más cercano a la imagen (el `<div>`) hasta terminar en el `<body>` buscando el primer elemento contenedor que esté posicionado.
3. El primer elemento contenedor es el `<div>`, pero su posicionamiento es el normal o estático, ya que ni siquiera tiene establecida la propiedad `position`.
4. Como el siguiente elemento contenedor es el `<body>`, el navegador establece directamente el origen de coordenadas en la esquina superior izquierda de la página.

5. A partir de ese origen de coordenadas, la caja se desplaza 50px hacia la derecha (left: 50px) y otros 50px de forma descendente (top: 50px).

Además, el párrafo que se mostraba debajo de la imagen sube y ocupa el lugar dejado por la imagen. El resultado es que el elemento `<div>` ahora sólo contiene el párrafo y la imagen se muestra en un nivel superior y cubre parcialmente los contenidos del párrafo.

A continuación, se posiciona de forma relativa el elemento `<div>` que contiene la imagen y el resto de reglas CSS no se modifican. La única propiedad añadida al `<div>` es `position: relative` por lo que el elemento contenedor se posiciona pero no se desplaza respecto de su posición original:

```
div {  
    border: 2px solid #CCC;  
    padding: 1em;  
    margin: 1em 0 1em 4em;  
    width: 300px;  
    position: relative;  
}  
  
div img {  
    position: absolute;  
    top: 50px;  
    left: 50px;  
}
```

La siguiente imagen muestra el resultado obtenido:

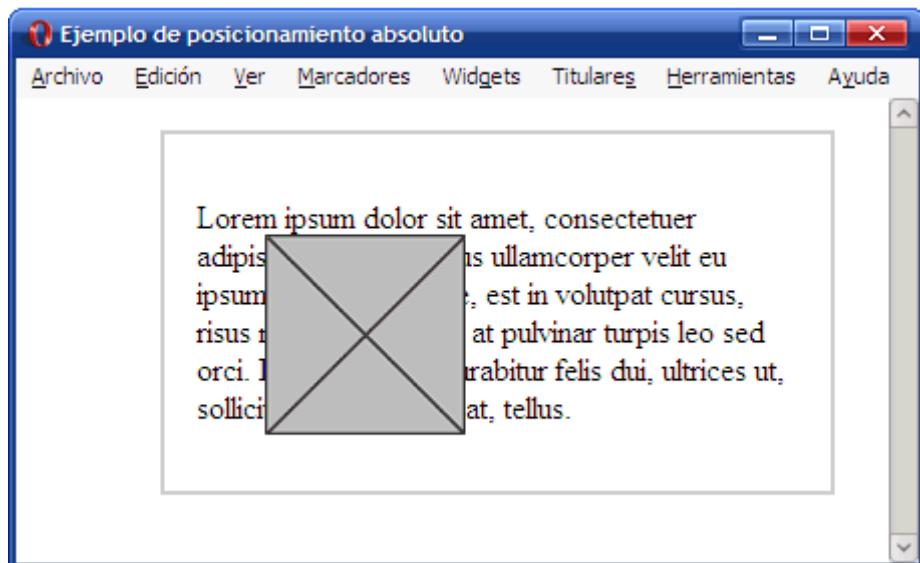


Figura 5.11. Imagen posicionada de forma absoluta

En este caso, el origen de coordenadas para determinar la nueva posición de la imagen corresponde a la esquina superior izquierda del elemento `<div>`:

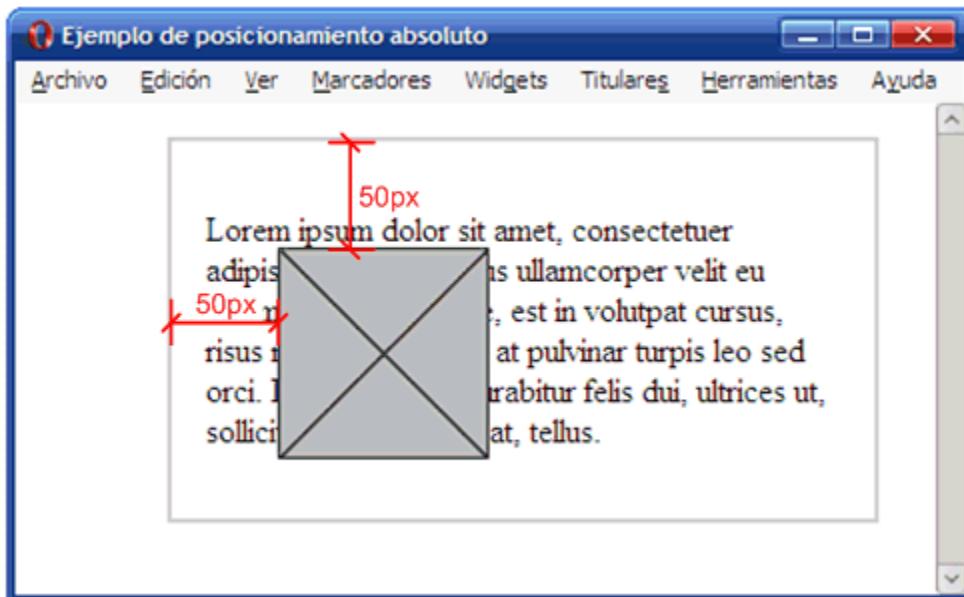


Figura 5.12. La referencia del posicionamiento absoluto es el elemento contenedor de la imagen

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

5.6. Posicionamiento fijo

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

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

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

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

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

5.7. position: sticky

Un elemento con **position: sticky;** se posiciona en función de la posición de desplazamiento del usuario.

Un elemento fijo alterna entre relativey fixed, dependiendo de la posición de desplazamiento. Se coloca en relación hasta que se alcanza una posición de desplazamiento determinada en la ventana gráfica; luego, se "pega" en su lugar (como posición: fija).

```
div.sticky {
    position: sticky;
    top: 0;
    background-color: green;
    border: 2px solid #4CAF50;
}
```

5.8. Posicionamiento flotante

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

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

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

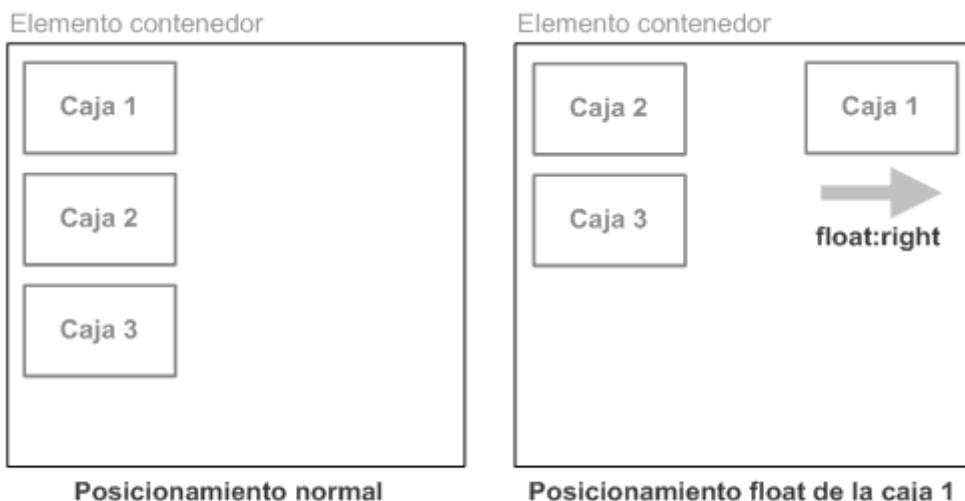


Figura 5.13. Ejemplo de posicionamiento float de una caja

Cuando se posiciona una caja de forma flotante:

- La caja deja de pertenecer al flujo normal de la página, lo que significa que el resto de cajas

ocupan el lugar dejado por la caja flotante.

- La caja flotante se posiciona lo más a la izquierda o lo más a la derecha posible de la posición en la que se encontraba originalmente.

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

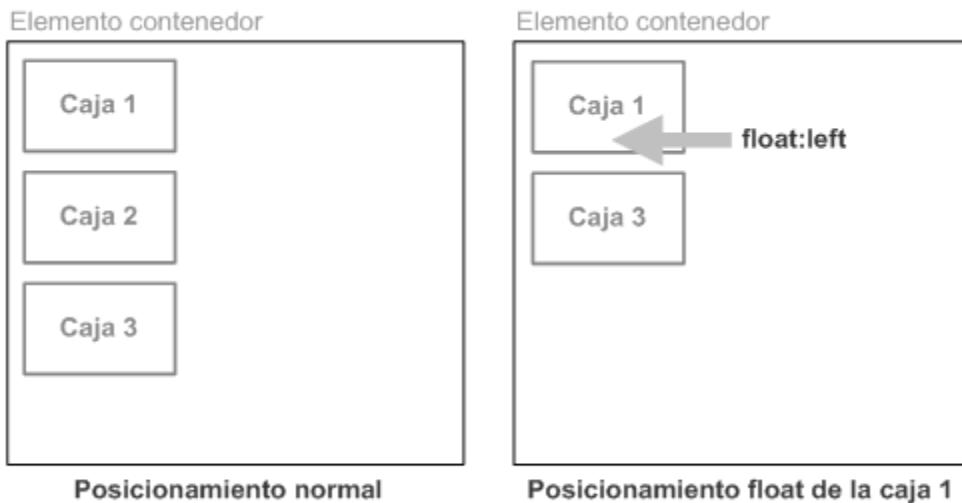


Figura 5.14. Ejemplo de posicionamiento float de una caja

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

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

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

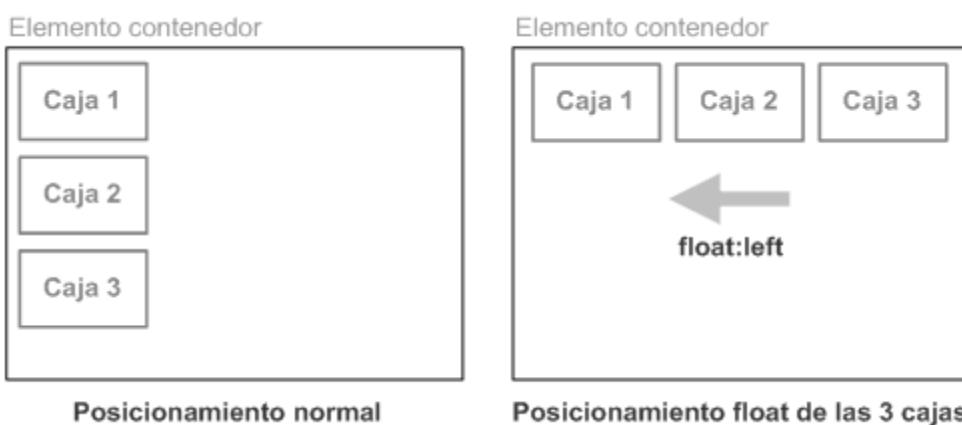


Figura 5.15. Ejemplo de posicionamiento float de varias cajas

En el ejemplo anterior, las cajas no se superponen entre sí porque las cajas flotantes tienen en cuenta las otras cajas flotantes existentes. Como la caja 1 ya estaba posicionada lo más a la

izquierda posible, la caja 2 sólo puede colocarse al lado del borde derecho de la caja 1, que es el sitio más a la izquierda posible respecto de la zona en la que se encontraba.

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

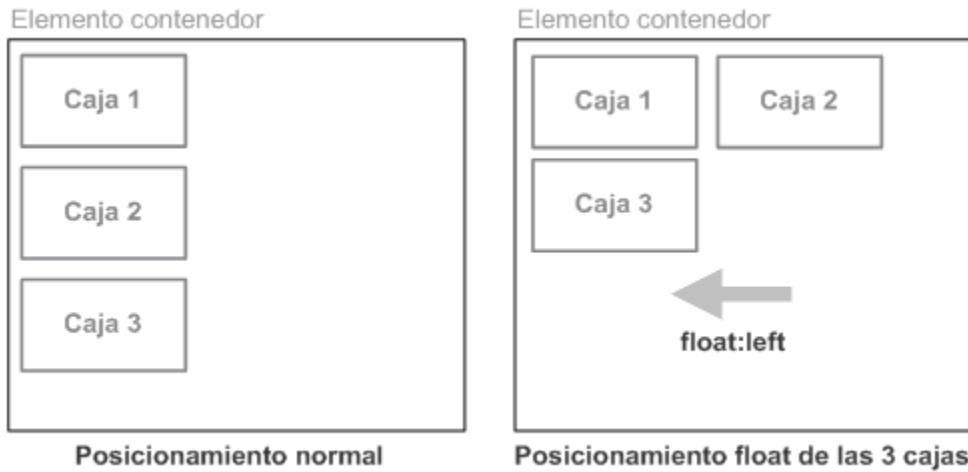


Figura 5.16. Ejemplo de posicionamiento float cuando no existe sitio suficiente

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

La propiedad CSS que permite posicionar de forma flotante una caja se denomina **float**:

| | |
|----------------------|--|
| float | Posicionamiento float |
| Valores | <code>left right none inherit</code> |
| Se aplica a | Todos los elementos |
| Valor inicial | <code>none</code> |
| Descripción | Establece el tipo de posicionamiento flotante del elemento |

Si se indica un valor `left`, la caja se desplaza hasta el punto más a la izquierda posible en esa misma línea (si no existe sitio en esa línea, la caja baja una línea y se muestra lo más a la izquierda posible en esa nueva línea). El resto de elementos adyacentes se adaptan y *fluyen* alrededor de la caja flotante.

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

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

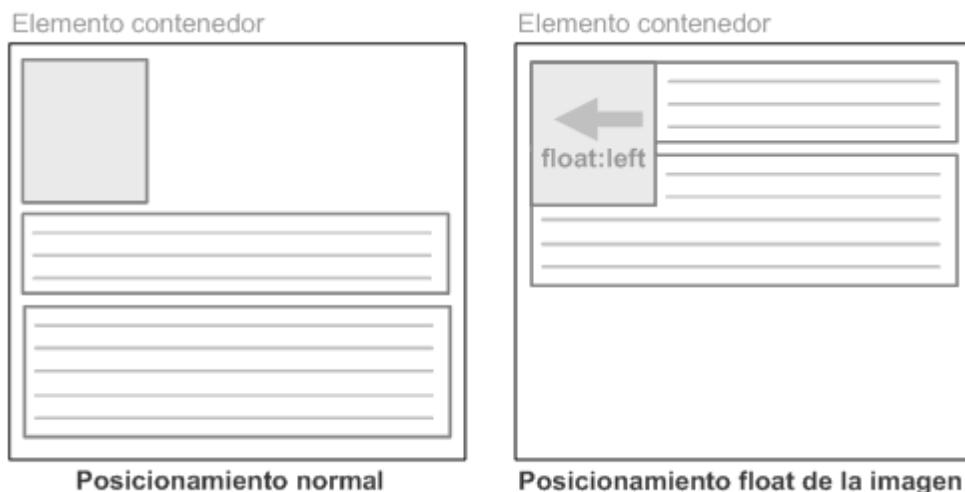


Figura 5.17. Elementos que fluyen alrededor de un elemento posicionado mediante `float`

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

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


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

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

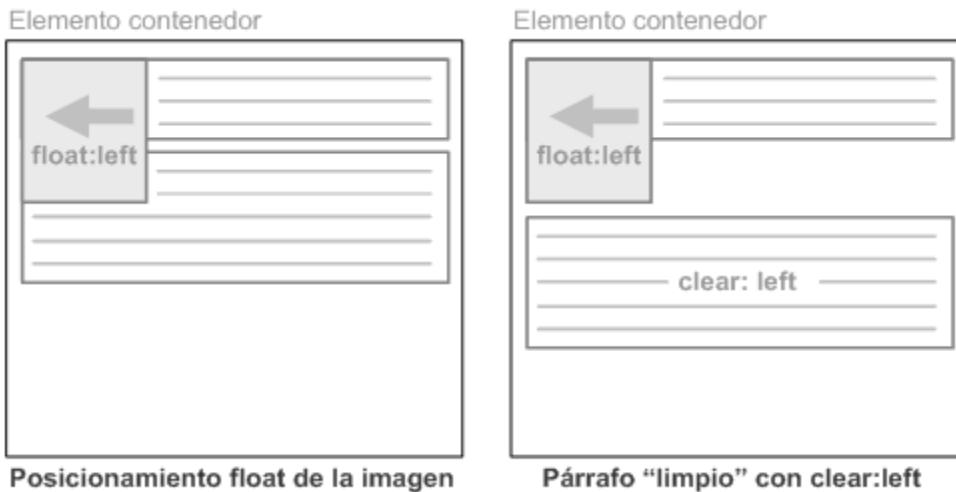


Figura 5.18. Forzando a que un elemento no fluya alrededor de otro elemento posicionado mediante `float`

5.9. Visualización

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

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

5.8.1. Propiedades `display` y `visibility`

Las propiedades `display` y `visibility` controlan la visualización de los elementos. Las dos propiedades permiten ocultar cualquier elemento de la página. Habitualmente se utilizan junto con JavaScript para crear efectos dinámicos como mostrar y ocultar determinados textos o imágenes cuando el usuario pincha sobre ellos.

La propiedad `display` permite ocultar completamente un elemento haciendo que desaparezca de la página. Como el elemento oculto no se muestra, el resto de elementos de la página se mueven para ocupar su lugar.

Por otra parte, la propiedad `visibility` permite hacer invisible un elemento, lo que significa que el navegador crea la caja del elemento pero no la muestra. En este caso, el resto de elementos de la página no modifican su posición, ya que aunque la caja no se ve, sigue ocupando sitio.

La siguiente imagen muestra la diferencia entre ocultar la caja número 5 mediante la propiedad `display` o hacerla invisible mediante la propiedad `visibility`:

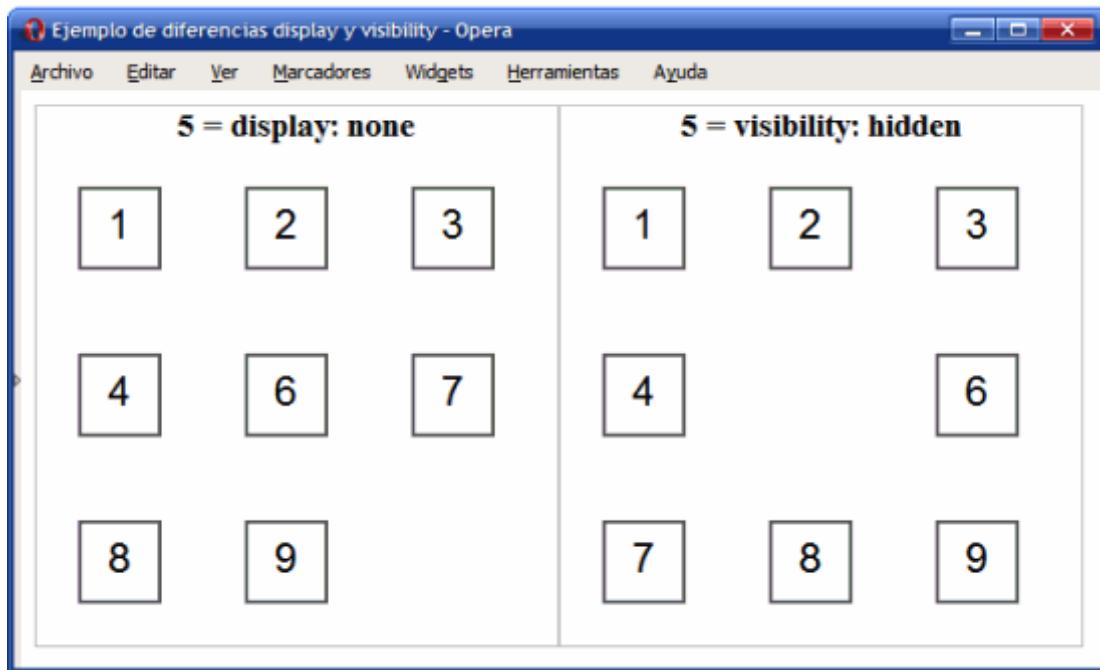


Figura 5.22. Diferencias visuales entre las propiedades display y visibility

En general, cuando se oculta un elemento no es deseable que siga ocupando sitio en la página, por lo que la propiedad `display` se utiliza mucho más que la propiedad `visibility`.

A continuación se muestra la definición completa de la propiedad `display`:

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

Las posibilidades de la propiedad `display` son mucho más avanzadas que simplemente ocultar elementos. En realidad, la propiedad `display` modifica la forma en la que se visualiza un elemento.

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

El valor `none` oculta un elemento y hace que desaparezca de la página. El resto de elementos de la página se visualizan como si no existiera el elemento oculto, es decir, pueden ocupar el espacio en el que se debería visualizar el elemento.

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

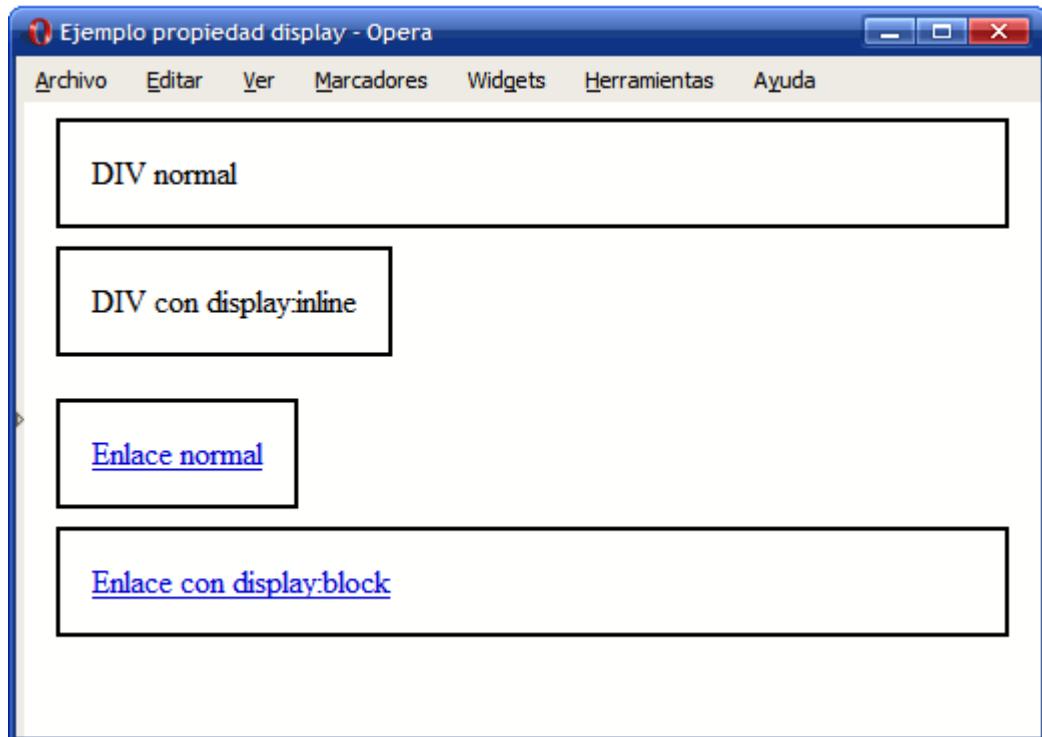


Figura 5.23. Ejemplo de propiedad display

Las reglas CSS del ejemplo anterior son las siguientes:

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

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

Como se verá más adelante, la propiedad `display: inline` se puede utilizar en las listas (``, ``) que se quieren mostrar horizontalmente y la propiedad `display: block` se emplea frecuentemente para los enlaces que forman el menú de navegación.

Por su parte, la definición completa de la propiedad `visibility` es mucho más sencilla:

| | |
|----------------------|---|
| visibility | Visibilidad de un elemento |
| Valores | <code>visible</code> <code>hidden</code> <code>collapse</code> <code>inherit</code> |
| Se aplica a | Todos los elementos |
| Valor inicial | <code>visible</code> |
| Descripción | Permite hacer visibles e invisibles a los elementos |

Las posibilidades de la propiedad `visibility` son mucho más limitadas que las de la propiedad `display`, ya que sólo permite hacer visibles o invisibles a los elementos de la página.

Inicialmente todas las cajas que componen la página son visibles. Empleando el valor `hidden` es posible convertir una caja en invisible para que no muestre sus contenidos. El resto de elementos de la página se muestran como si la caja todavía fuera visible, por lo que en el lugar donde originalmente se mostraba la caja invisible, ahora se muestra un hueco vacío.

Por último, el valor `collapse` de la propiedad `visibility` sólo se puede utilizar en las filas, grupos de filas, columnas y grupos de columnas de una tabla. Su efecto es similar al de la propiedad `display`, ya que oculta completamente la fila y/o columna y se pueden mostrar otros contenidos en ese lugar. Si se utiliza el valor `collapse` sobre cualquier otro tipo de elemento, su efecto es idéntico al valor `hidden`.

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

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

1. Si `display` vale `none`, se ignoran las propiedades `float` y `position` y la caja no se muestra en la página.
2. Si `position` vale `absolute` o `fixed`, la caja se posiciona de forma absoluta, se considera que `float` vale `none` y la propiedad `display` vale `block` tanto para los elementos en línea como para los elementos de bloque. La posición de la caja se determina mediante el valor de las propiedades `top`, `right`, `bottom` y `left`.
3. En cualquier otro caso, si `float` tiene un valor distinto de `none`, la caja se posiciona de forma flotante y la propiedad `display` vale `block` tanto para los elementos en línea como para los elementos de bloque.

5.8.3. Propiedad `overflow`

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

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

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

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

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

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

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

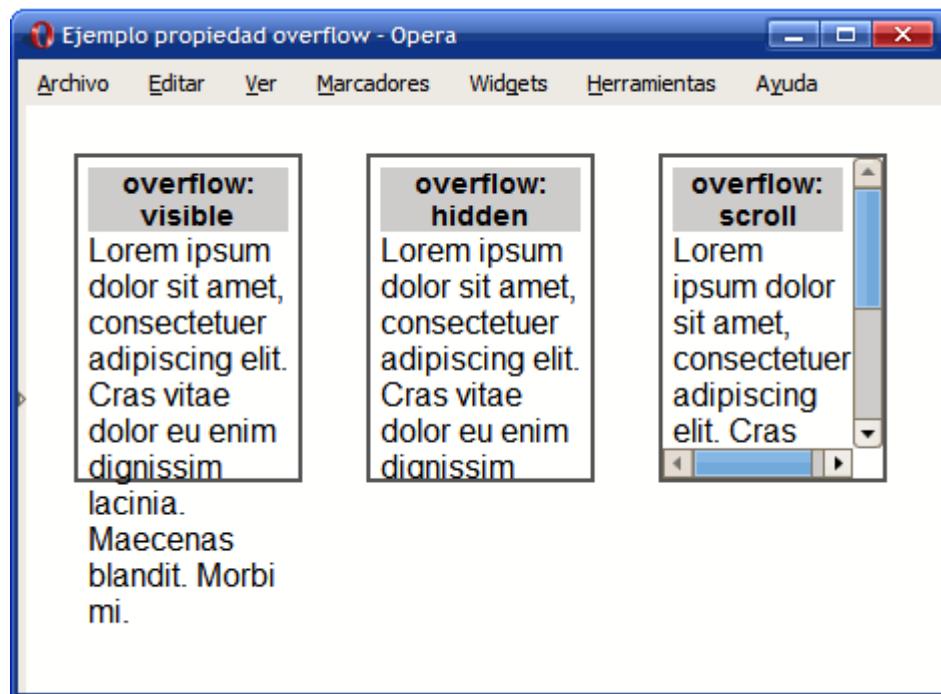


Figura 5.24. Ejemplo de propiedad `overflow`

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

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

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

```
blandit. Morbi mi.</div>

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

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

5.8.4. Propiedad z-index

Además de posicionar una caja de forma horizontal y vertical, CSS **permite controlar la posición tridimensional de las cajas posicionadas.** De esta forma, es posible indicar las cajas que se muestran delante o detrás de otras cajas cuando se producen solapamientos.

La posición tridimensional de un elemento **se establece sobre un tercer eje llamado Z** y se controla mediante la propiedad z-index. Utilizando esta propiedad es posible crear páginas complejas con varios niveles o capas.

A continuación se muestra la definición formal de la propiedad z-index:

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

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

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

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

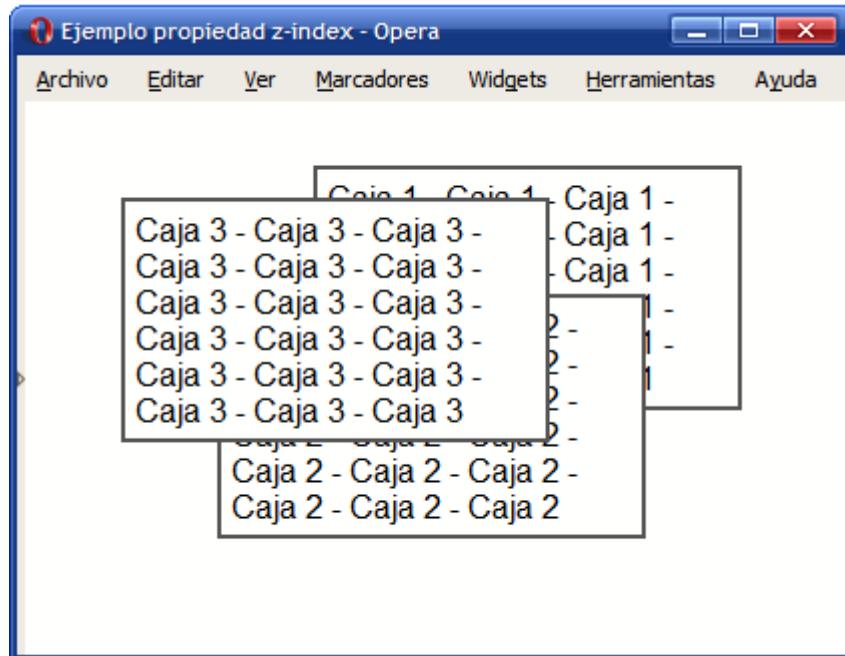


Figura 5.25. Ejemplo de propiedad z-index

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

```
div { position: absolute; }

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

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

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

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

La propiedad `z-index` sólo tiene efecto en los elementos posicionados, por lo que es obligatorio que la propiedad `z-index` vaya acompañada de la propiedad `position`. Si debes posicionar un elemento pero no quieras moverlo de su posición original ni afectar al resto de elementos de la página, puedes utilizar el posicionamiento relativo (`position: relative`).

Capítulo 6. Texto

6.1. Tipografía

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



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

| | |
|----------------------|---|
| color | Color del texto |
| 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.

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

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

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

Los navegadores muestran el texto de las páginas web utilizando los tipos de letra instalados en el ordenador 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`.

6.1.2 Sustitución de texto

La limitación más frustrante para los diseñadores web que cuidan especialmente la tipografía de sus páginas es la imposibilidad de utilizar cualquier tipo de letra para mostrar los contenidos de texto. Como se sabe, las fuentes que utiliza una página deben estar instaladas en el ordenador o dispositivo del usuario para que el navegador pueda mostrarlas.

Por lo tanto, si el diseñador utiliza en la página una fuente especial poco común entre los usuarios normales, el navegador no encuentra esa fuente y la sustituye por una fuente por defecto.

El resultado es que la página que ve el usuario y la que ve el diseñador se diferencian completamente en su tipografía.

Con la directiva `@font-face` podemos incluir fuentes que no estén en nuestro sistema operativo.

6.1.2.1 La directiva `@font-face`

[Web Fonts](http://www.w3.org/TR/css3webfonts/) (<http://www.w3.org/TR/css3webfonts/>).

La directiva `@font-face` permite describir las fuentes que utiliza una página web. Como parte de la descripción se puede indicar la dirección o URL desde la que el navegador se puede descargar la fuente utilizada si el usuario no dispone de ella. A continuación se muestra un ejemplo de uso de la directiva `@font-face`:

```
@font-face {
    font-family: "Fuente muy rara";
    src: url("http://www.ejemplo.com/fuentes/fuente_muy_rara.ttf");
}

h1 { font-family: "Fuente muy rara", sans-serif; }
```

La directiva `@font-face` también permite definir otras propiedades de la fuente, como su formato, grosor y estilo. A continuación, se muestran otros ejemplos:

```
@font-face {
    font-family: Fontinsans;
    src: url("fonts/Fontin_Sans_SC_45b.otf") format("opentype");
    fontweight: bold; font-style: italic;
}

@font-face {
    font-family: Tagesschrift;
    src: url("fonts/YanoneTagesschrift.ttf") format("truetype");
}
```

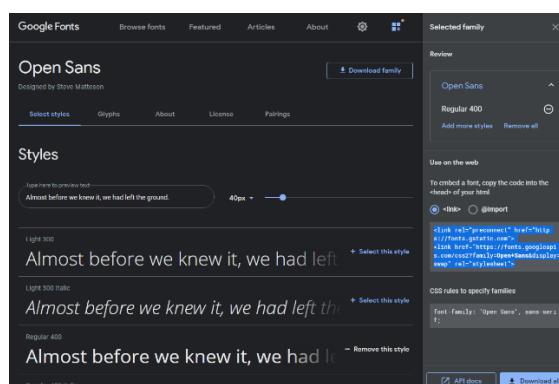
6.1.2.2 Google Fonts

En la actualidad, es muy común utilizar Google Fonts como repositorio proveedor de tipografías para utilizar en nuestros sitios web por varias razones:

- **Gratis:** Disponen de un amplio catálogo de fuentes y tipografías libres y/o gratuitas.
- **Cómodo:** Resulta muy sencillo su uso: Google nos proporciona un código y el resto lo hace él.
- **Rápido:** El servicio está muy extendido y utiliza un CDN, que brinda ventajas de velocidad

En la propia página de Google Fonts podemos seleccionar las fuentes con las características deseadas y generar un código HTML con la tipografía (o colección de tipografías) que vamos a utilizar.

Seleccionamos la tipografía que nos interesa, y de la lista de estilos (generalmente aparecen varios), pulsamos en «Select this style». Esto hará que aparezca una barra lateral, donde nos mostrará un fragmento de código en la zona «Use on the web»:



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

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

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

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

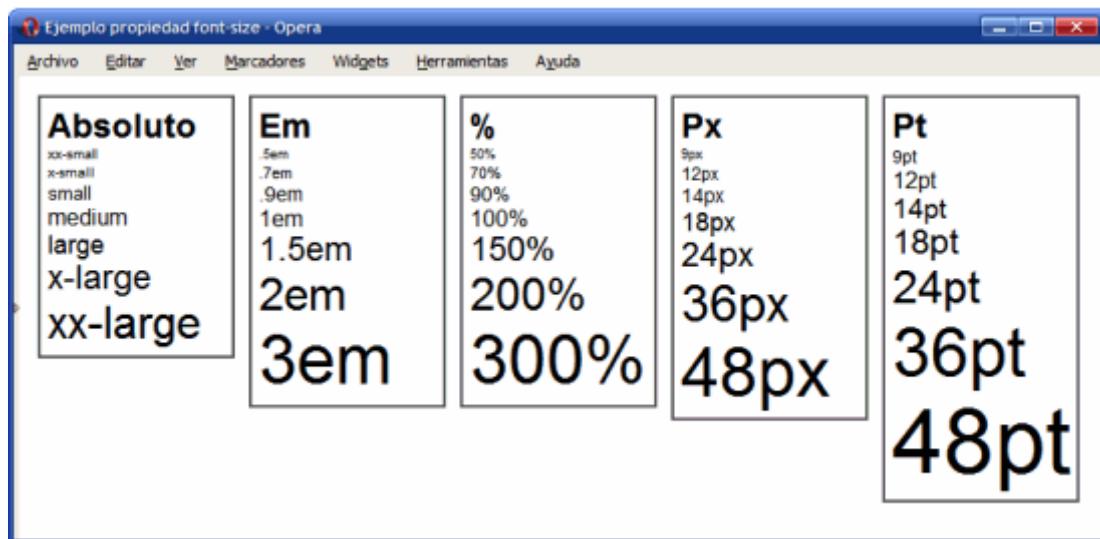


Figura 6.1. Comparación visual de las distintas unidades para indicar el tamaño del texto

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

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

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

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

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



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

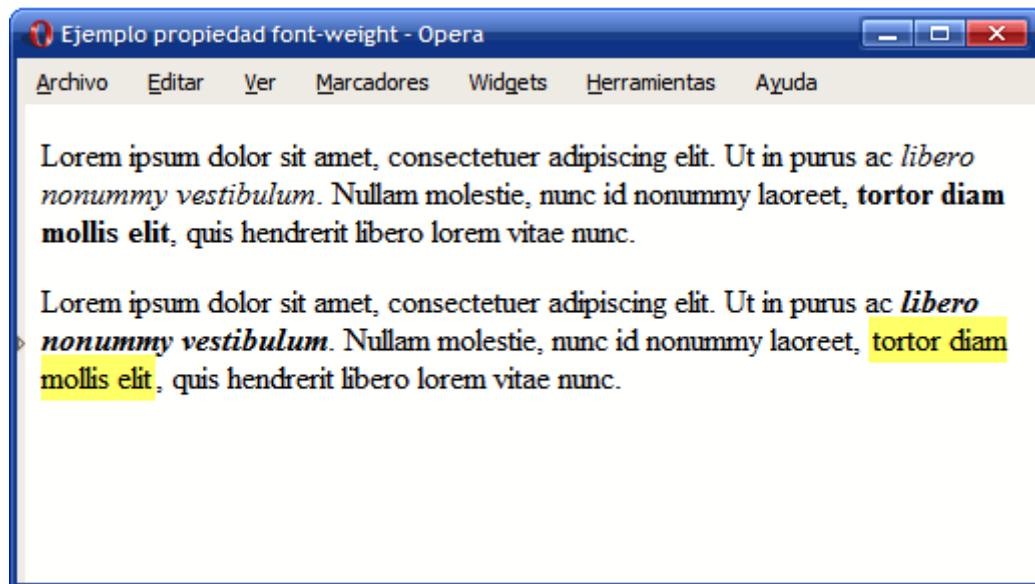


Figura 6.2. Ejemplo de propiedad `font-weight`

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

Las reglas CSS del ejemplo anterior se muestran a continuación:

```
#especial em {  
    font-weight: bold;  
}  
  
#especial strong {  
    font-weight: normal;  
    background-color: #FFFF66;  
    padding: 2px;  
}  
  
<p>Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut in  
purus ac <em>libero nonummy vestibulum</em>. Nullam molestie, nunc id  
nonummy laoreet, <strong>tortor diam mollis elit</strong>, quis hendrerit  
libero lorem vitae nunc.</p>  
  
<p id="especial">Lorem ipsum dolor sit amet, consectetuer adipiscing elit.  
Ut in purus ac <em>libero nonummy vestibulum</em>. Nullam molestie, nunc id  
nonummy laoreet, <strong>tortor diam mollis elit</strong>, quis hendrerit  
libero lorem vitae nunc.</p>
```

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

font-style Estilo de la letra

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

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

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

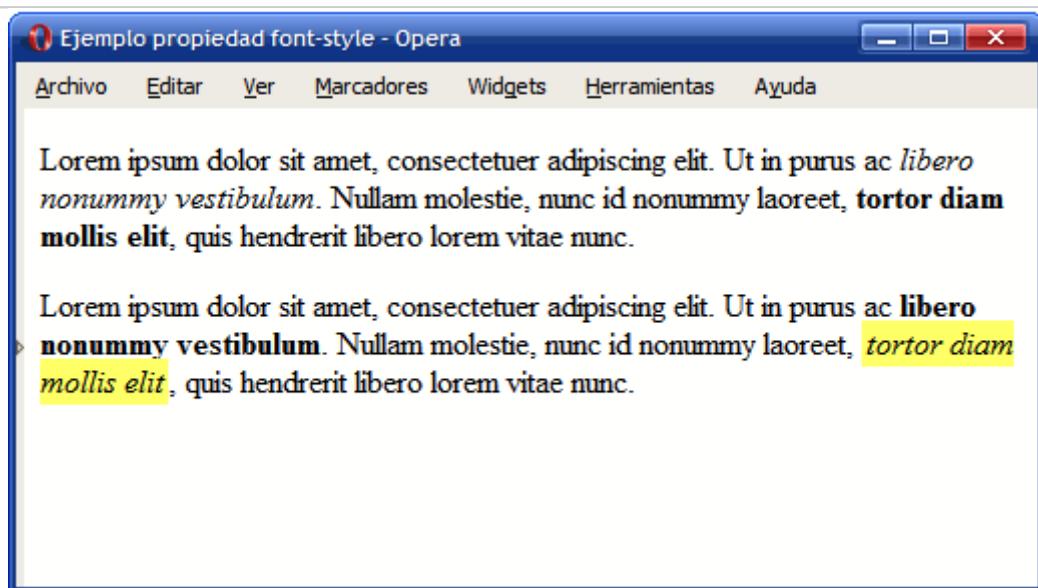


Figura 6.3. Ejemplo de propiedad font-style

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

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

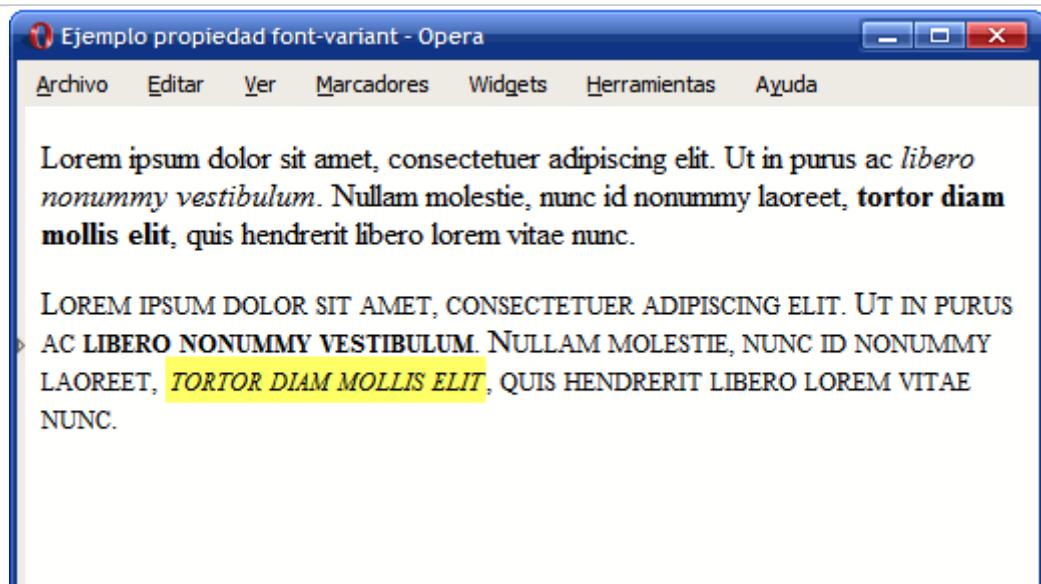
```
#especial em {
    font-weight: bold;
    font-style: normal;
}
#especial strong { font-
    weight: normal; font-
    style: italic;
    background-
    color:#FFFF66; padding:
    2px;
}
```

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

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

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

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



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.

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

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

- En primer lugar y de forma opcional se indican el `font-style`, `font-variant` y `font-weight` en cualquier orden.
- A continuación, se indica 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.

Ejercicio 7 Ver enunciado

6.2. Texto

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

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

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

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

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



Figura 6.5. Ejemplo de propiedad text-align

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

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

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

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

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

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

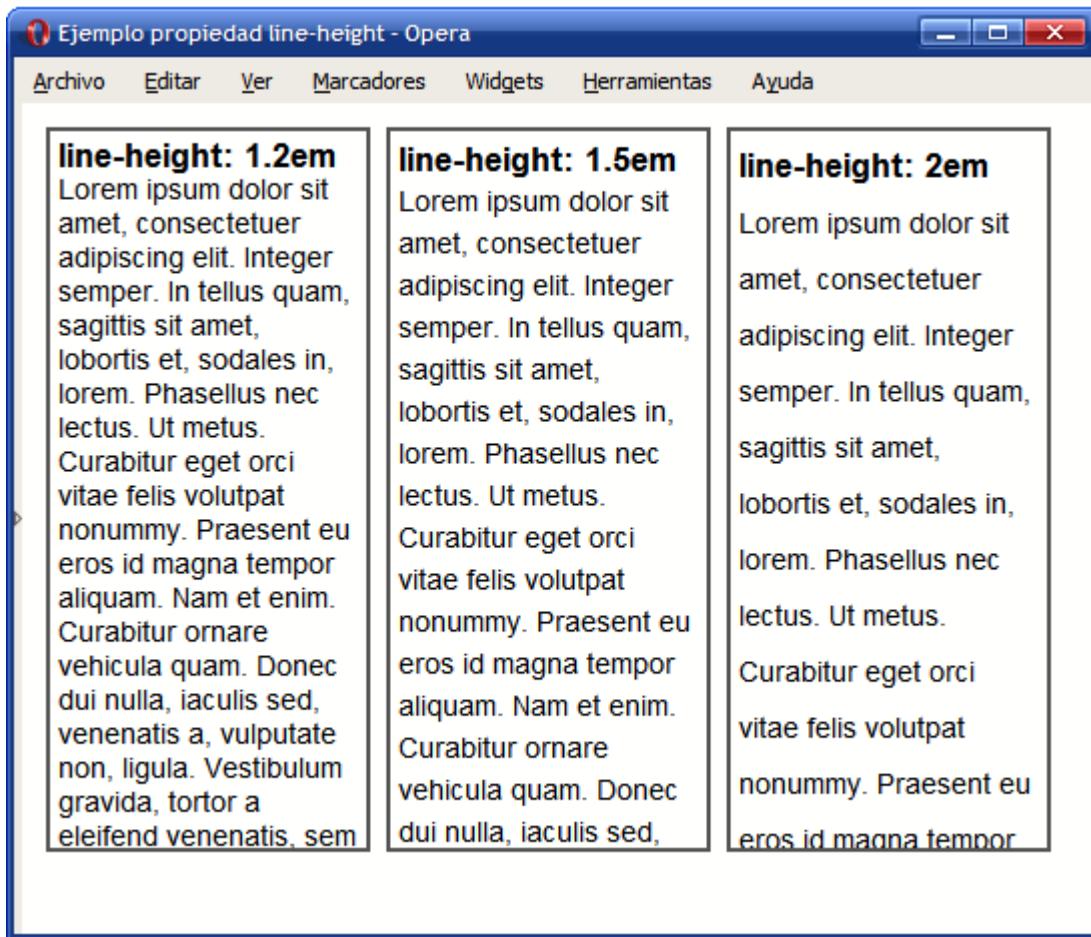


Figura 6.6. Ejemplo de propiedad line-height

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

| | |
|---|---|
| text-decoration Decoración del texto | |
| 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.

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

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

La siguiente imagen muestra cada uno de los posibles valores:

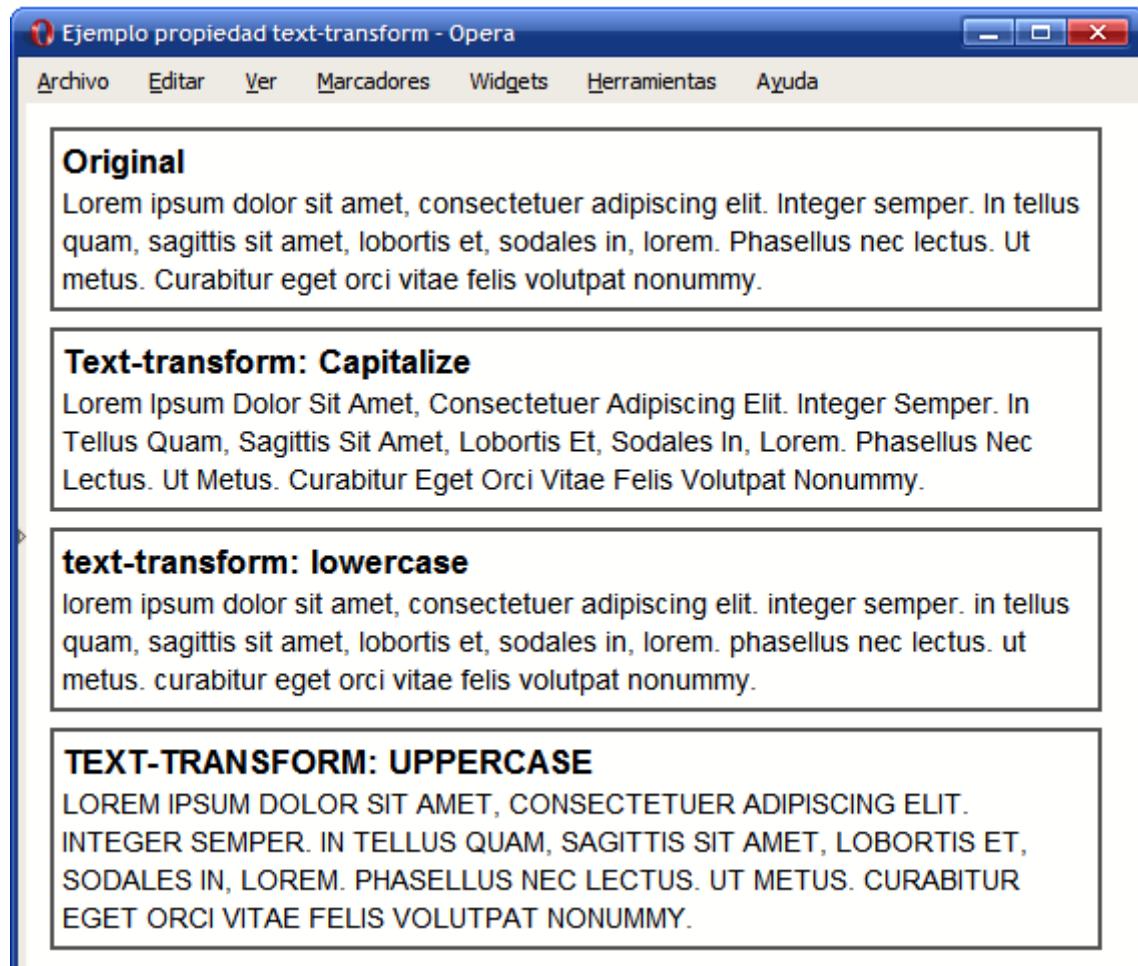


Figura 6.7. Ejemplo de propiedad `text-transform`

Las reglas CSS del ejemplo anterior se muestran a continuación:

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

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

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

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

vertical-align Alineación vertical

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

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

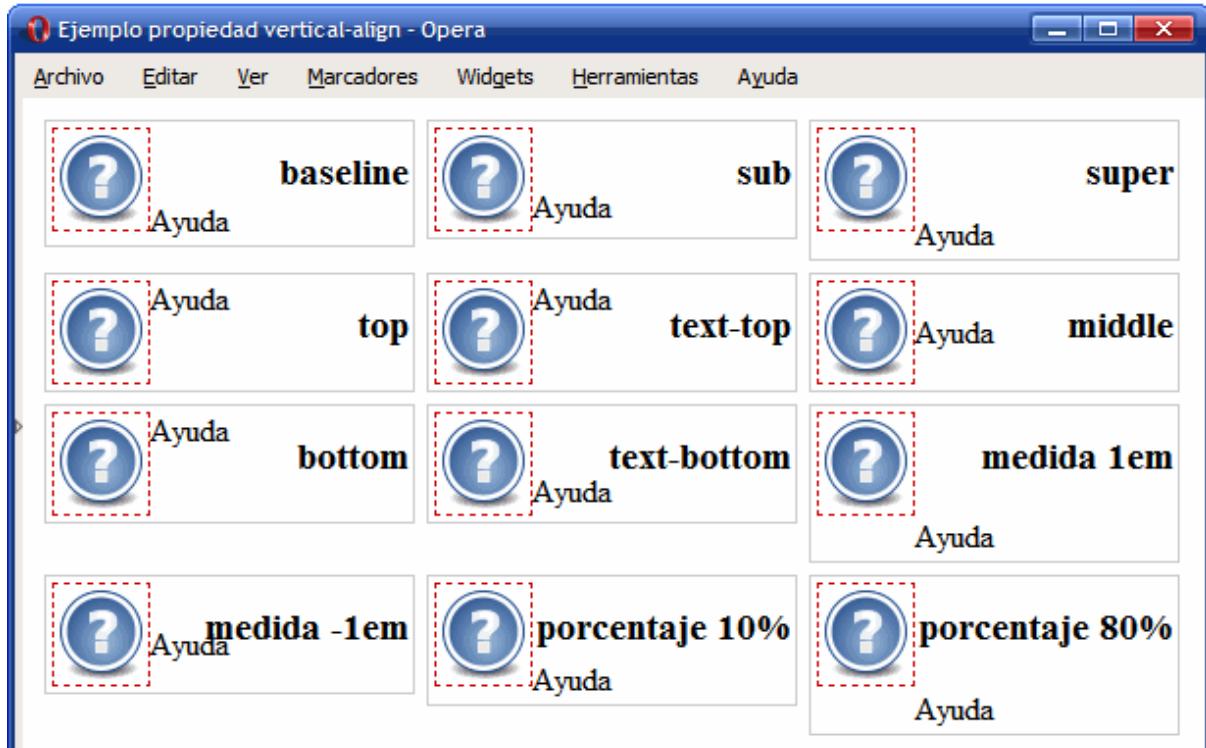


Figura 6.8. Ejemplo de propiedad `vertical-align`

El 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`.

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



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:

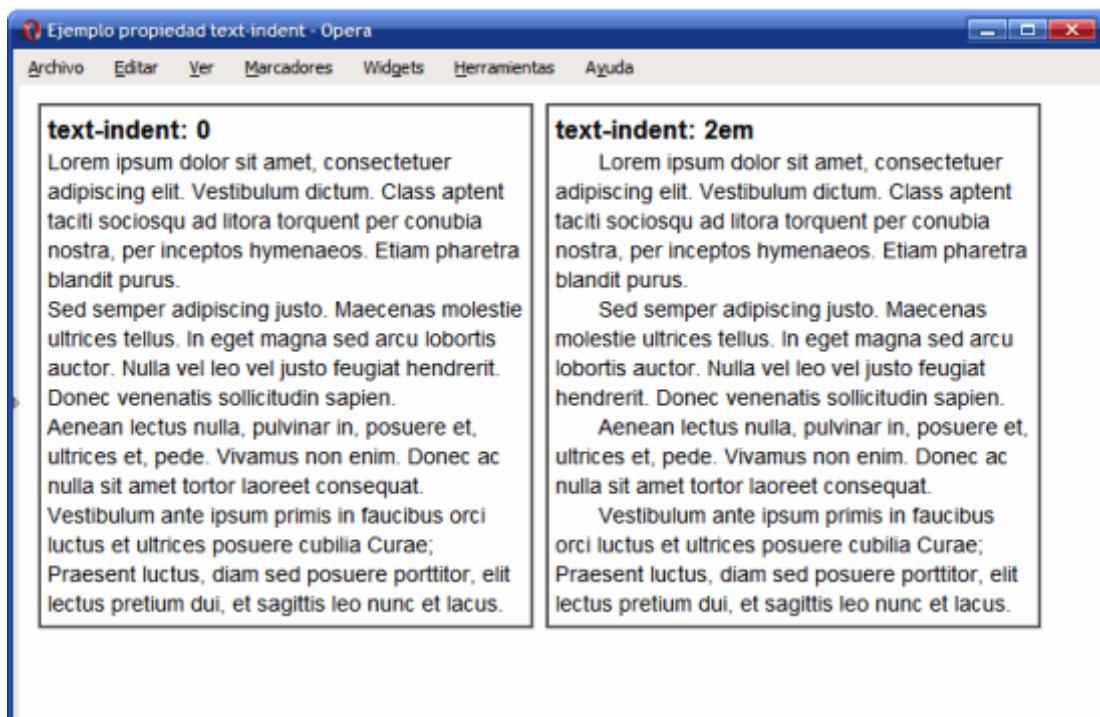


Figura 6.9. Ejemplo de propiedad `text-indent`

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.

letter-spacing Espaciado entre letras

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

word-spacing Espaciado entre palabras

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

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

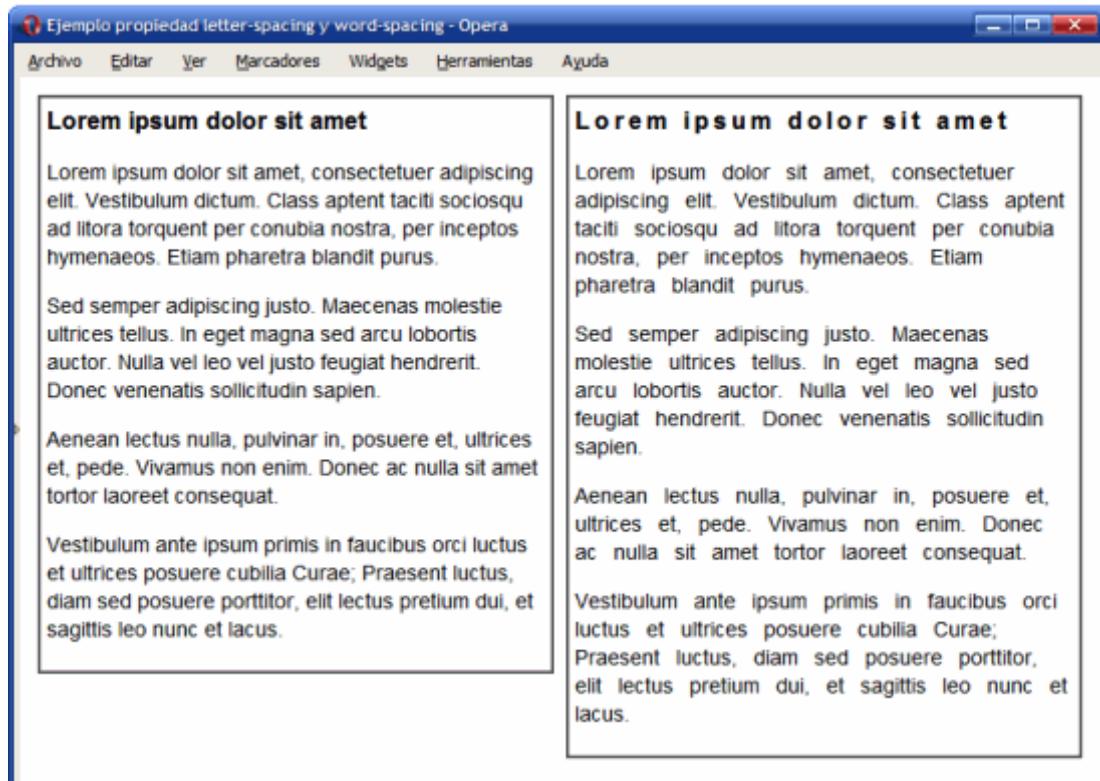


Figura 6.10. Ejemplo de propiedades letter-spacing y word-spacing

Las reglas CSS del ejemplo anterior se muestran a continuación:

```
.dspecial 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 `&nbsp` 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`.

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

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

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

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

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

| Valor | Respetá espacios en blanco | Respetá 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:

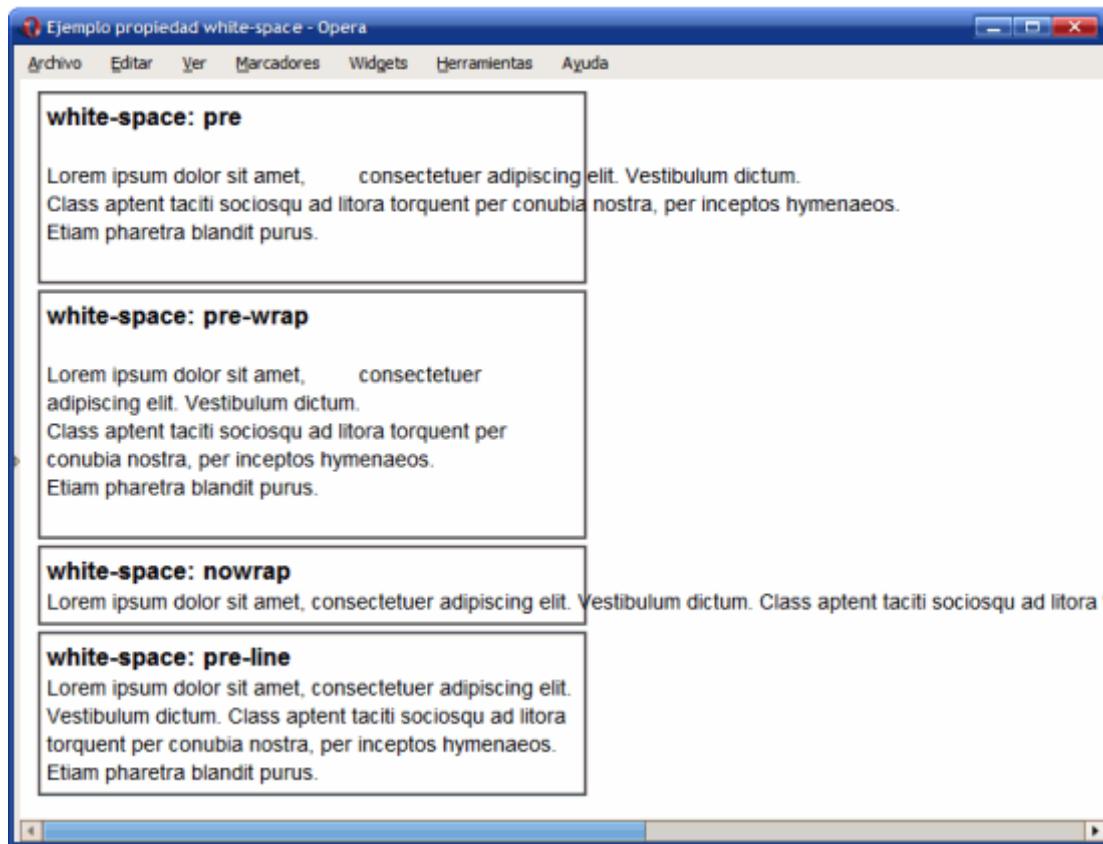


Figura 6.11. Ejemplo de propiedad `white-space`

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:

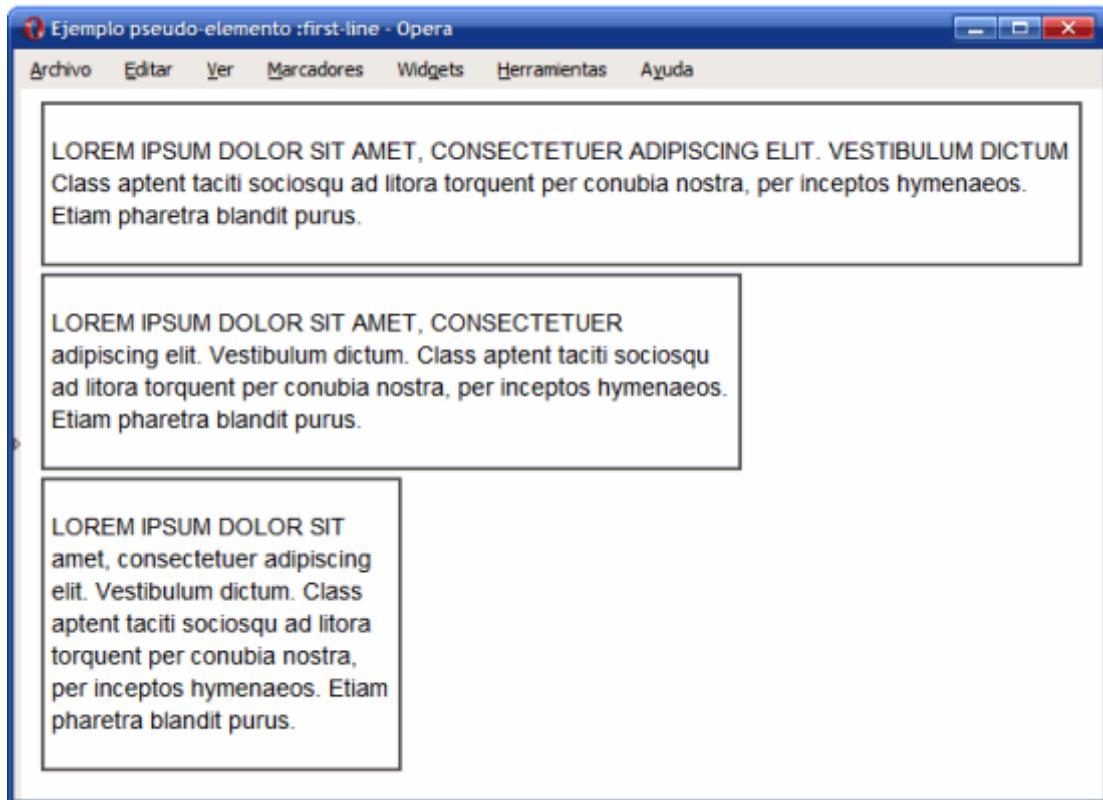


Figura 6.12. Ejemplo de pseudo-elemento first-line

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:

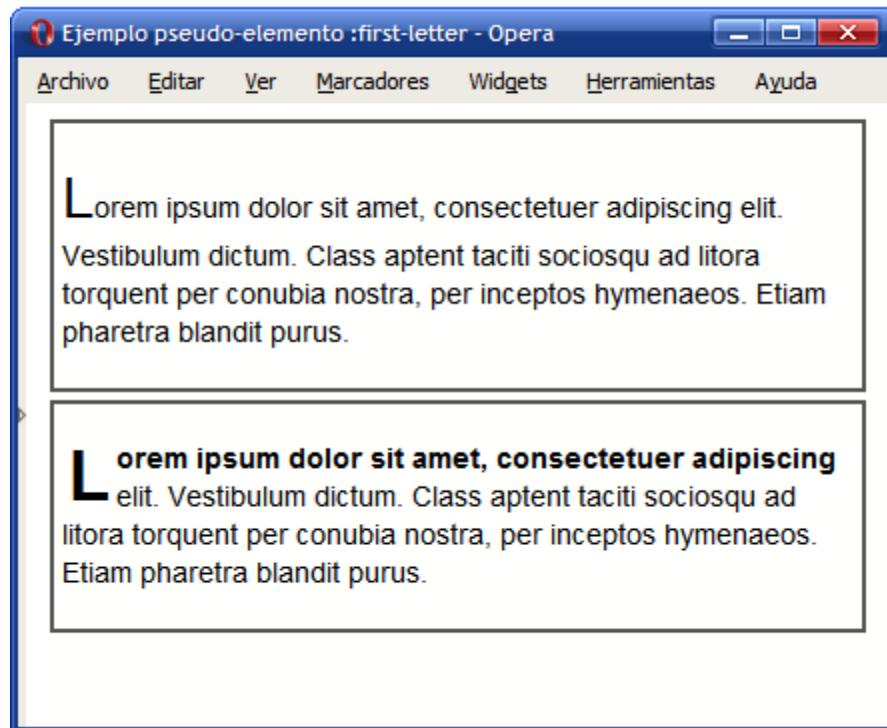


Figura 6.13. Ejemplo de pseudo-elemento first-letter

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

Capítulo 7. Enlaces

7.1. Estilos básicos

7.1.1. Tamaño, color y decoración

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

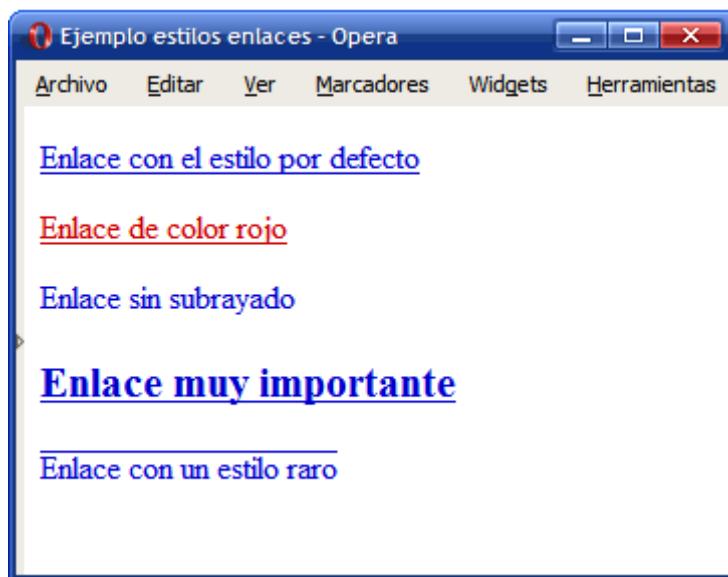


Figura 7.1. Ejemplo de enlaces con estilos diferentes

A continuación se muestran las reglas CSS del ejemplo anterior:

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

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

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

7.1.2. Pseudo-clases

CSS 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.primero:active { ... }
```

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

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

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

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

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

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

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

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

Ejercicio 8 Ver enunciado en la página 202

7.2. Estilos avanzados

7.2.1. Decoración personalizada

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

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

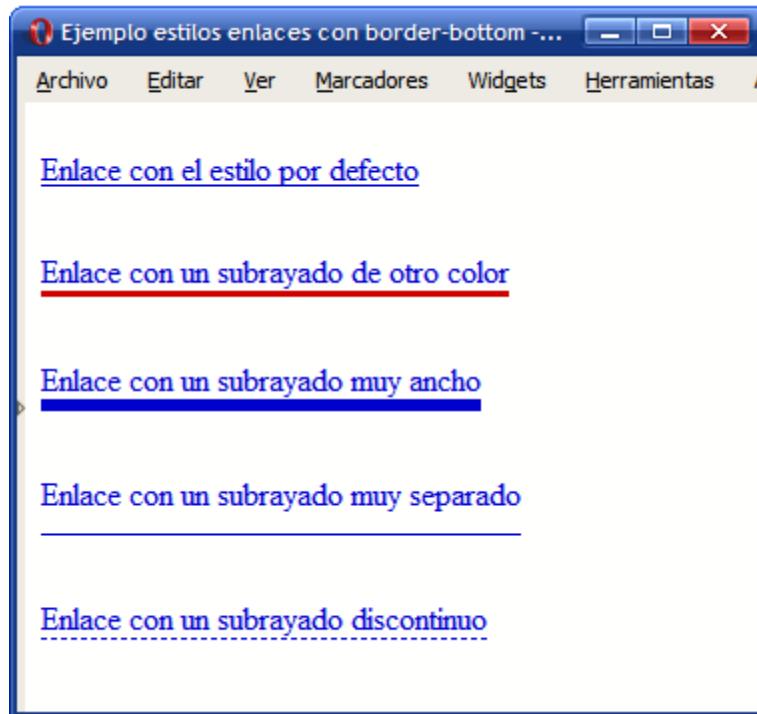


Figura 7.3. Enlaces con subrayado personalizado mediante la propiedad border

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

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

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

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

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

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

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

7.2.2. Imágenes según el tipo de enlace

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

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

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

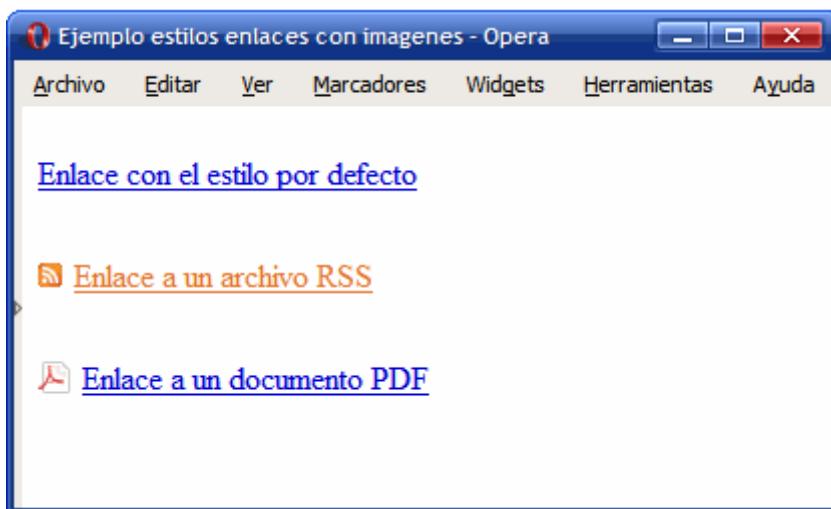


Figura 7.4. Personalizando el aspecto de los enlaces en función de su tipo

Las reglas CSS necesarias se muestran a continuación:

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

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

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

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

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

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

7.2.3. Mostrar los enlaces como si fueran botones

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

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

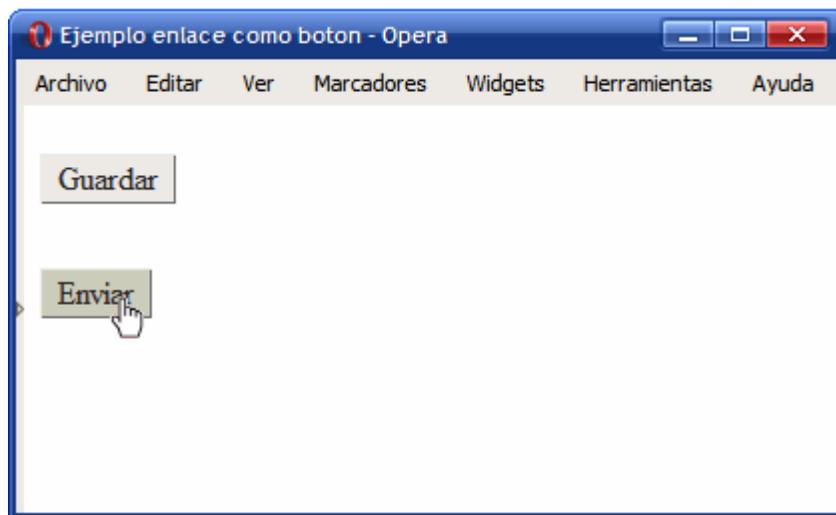


Figura 7.5. Mostrando un enlace como si fuera un botón

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

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

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

Capítulo 8. Imágenes

8.1. Estilos básicos

8.1.1. Establecer la anchura y altura de las imágenes

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

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

```

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

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

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

```

```

8.1.2. Eliminar el borde de las imágenes con enlaces

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

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

Ejercicio 9 Ver enunciado en la página 202

8.1.3. Estilos avanzados. Filtros

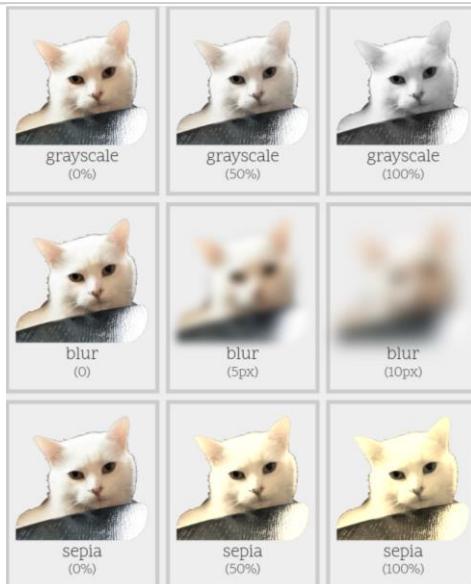
Los filtros CSS son una característica muy atractiva de CSS que permite aplicar ciertos efectos de imagen propios de aplicaciones de retoque fotográfico, como sepia, variaciones de brillo o contraste (u otros) al vuelo en el propio navegador, sin hacer cambios permanentes sobre una imagen.

Dichos filtros funcionan a través de la propiedad filter, a la cuál hay que especificarle una función concreta de las existentes, como por ejemplo la función de blanco y negro (grayscale):

```
img {
    filter: grayscale(75%);
}
```

Nota: Aunque es la que se ha utilizado en el ejemplo, esta propiedad funciona con otros elementos, es decir, no debe aplicarse necesariamente sobre una imagen.

| Función | Significado | Valor | Mínimo | Máximo | >100% |
|--------------------|---------------------------|-------|------------------------------|-----------------------------|-------|
| grayscale | Escala de blanco y negro | | 0% ¹ | 100% = B&N | |
| blur | Grado de difuminado | | Tamaño de radio (desenfoque) | | |
| sepia | Grado de color sepia | | 0% ¹ | 100% = sepia | |
| saturate | Grado de saturación | | 0% = B&N | 100% ¹ | Sí |
| opacity | Grado de transparencia | | 0% = invisible | 100% = visible ¹ | |
| brightness | Brillo | | 0% = negro | 100% ¹ | Sí |
| contrast | Contraste | | 0% = gris | 100% ¹ | Sí |
| hue-rotate | Rotación de color (matiz) | | 0/360deg ¹ | | |
| invert | Invertir | | 0% ¹ | 100% = invertido | |
| drop-shadow | Sombra idéntica | | (2px 2px 5px gold) | | |



Capítulo 9. Listas

9.1. Estilos básicos

9.1.1. Viñetas personalizadas

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

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

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

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

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

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

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

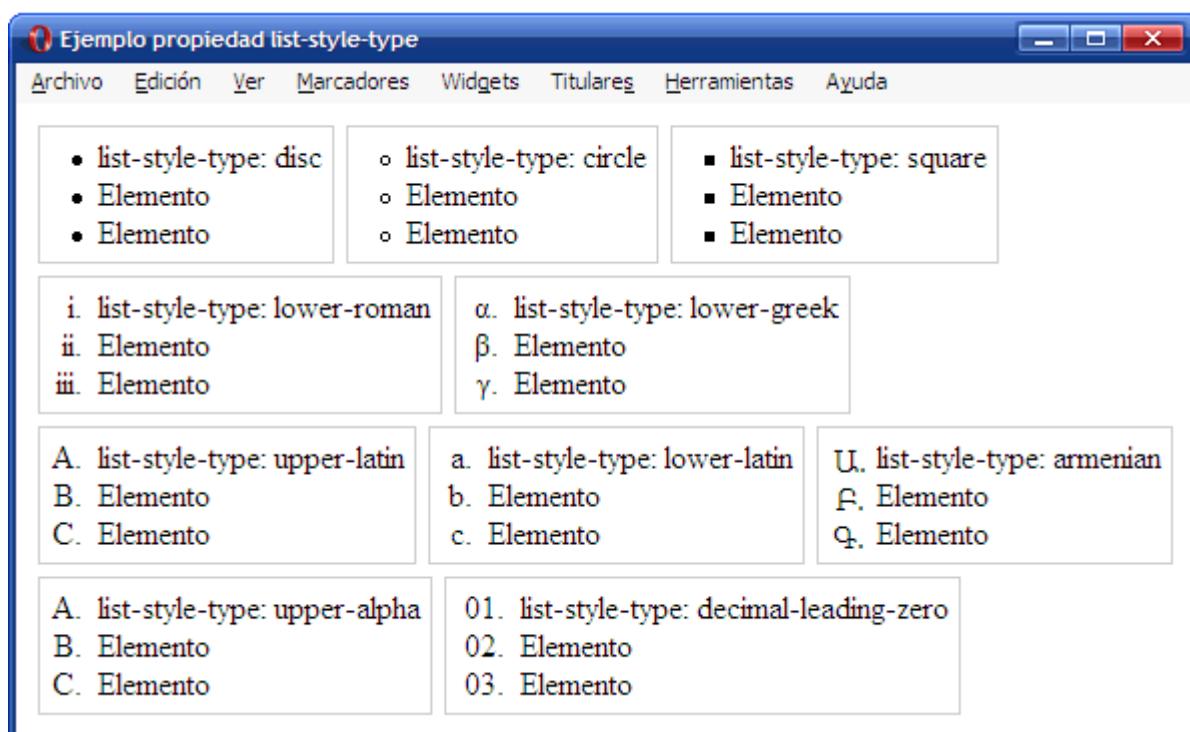


Figura 9.1. Ejemplo de propiedad `list-style-type`

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

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

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

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

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

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

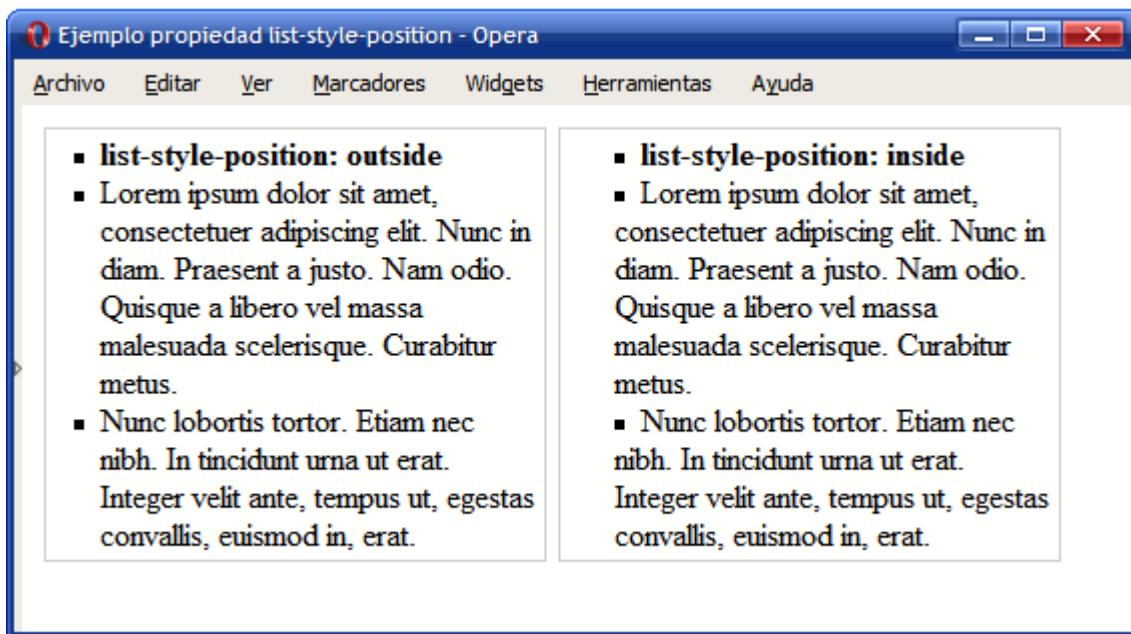


Figura 9.2. Ejemplo de propiedad `list-style-position`

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

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

| list-style-image Imagen de la viñeta | |
|---|------------------------|
| Valores | <url> none inherit |
| Se aplica a | Elementos de una lista |

| | |
|----------------------|---|
| Valor inicial | none |
| Descripción | Permite reemplazar las viñetas automáticas por una imagen personalizada |

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

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

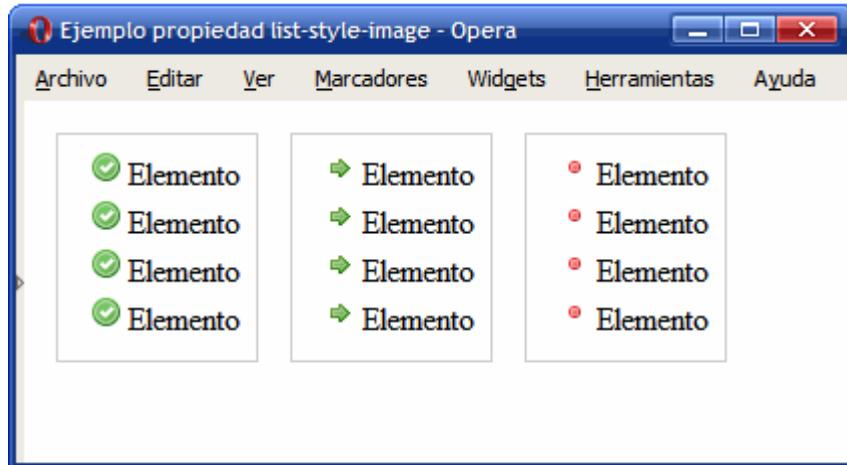


Figura 9.3. Ejemplo de propiedad `list-style-image`

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

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

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

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

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

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

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

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

9.1.2. Menú vertical sencillo

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

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

Lista de enlaces original:

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

Aspecto final del menú vertical:



Figura 9.4. Menú vertical sencillo creado con CSS

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

1) Definir la anchura del menú:

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



Figura 9.5. Menú vertical: definiendo su anchura

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

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



Figura 9.6. Menú vertical: eliminar viñetas por defecto

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

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

```
}

ul.menu li {
    border-bottom: 1px solid #7C7C7C;
    border-top: 1px solid #FFF;
    background: #F4F4F4;
}
```



Figura 9.7. Menú vertical: añadiendo bordes

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

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

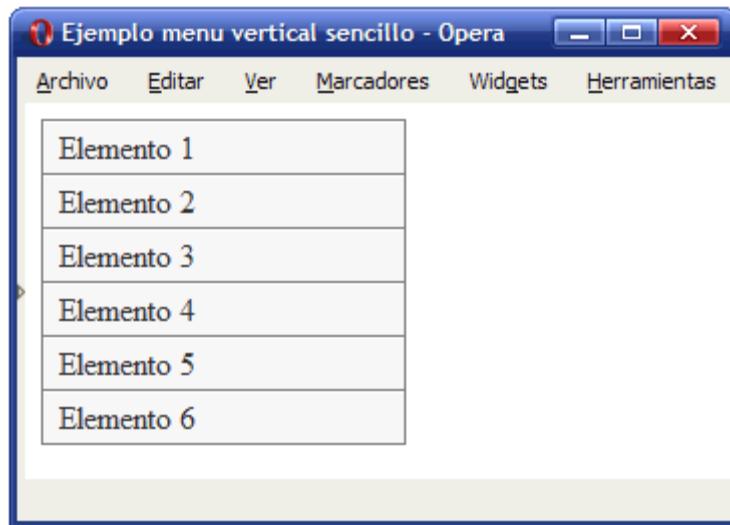


Figura 9.8. Aspecto final del menú vertical sencillo creado con CSS

Ejercicio 10 [Ver enunciado](#)

9.2. Estilos avanzados

9.2.1. Menú horizontal básico

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

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

Código HTML del menú horizontal:

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

Código CSS del menú vertical anterior:

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

Aspecto final del menú horizontal:

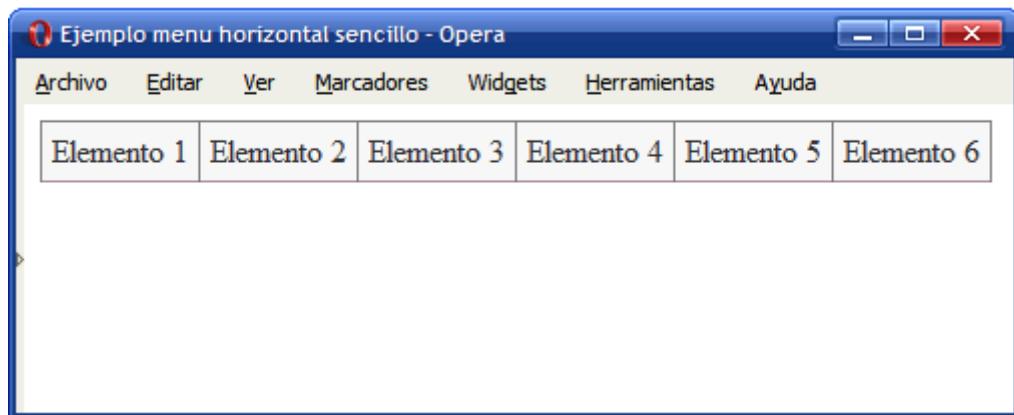


Figura 9.9. Menú horizontal sencillo creado con CSS

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

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

```
ul.menu {
    clear: both;
    float: left;
    width: 100%;
    list-style: none;
    margin: 0;
    padding: 0;
}
```



Figura 9.10. Menú horizontal: definiendo anchura y bordes

- 2) La clave de la transformación reside en modificar la propiedad `float` de los elementos `` del menú:

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

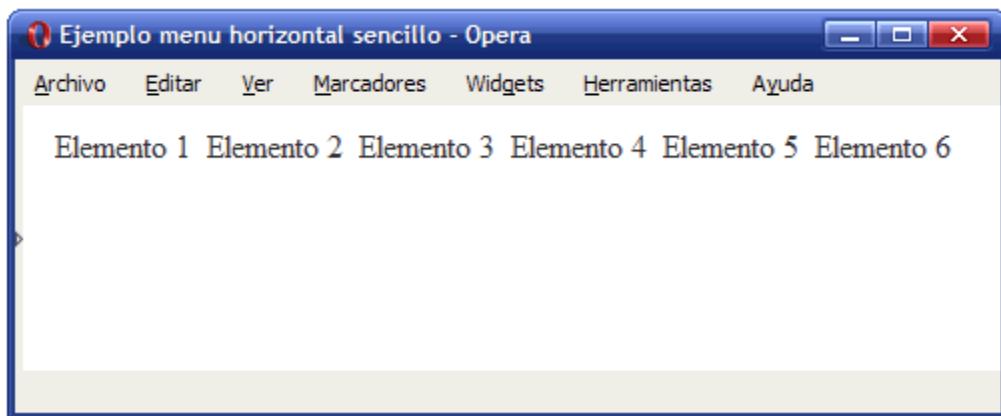


Figura 9.11. Menú horizontal: propiedades float y display

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

```
ul.menu li a {
    padding: .3em;
    display: block;
    text-decoration: none;
    color: #333;
    background: #F4F4F4;
    border-top: 1px solid #7C7C7C; border-
    right: 1px solid #7C7C7C; border-bottom:
    1px solid #9C9C9C;
}
```

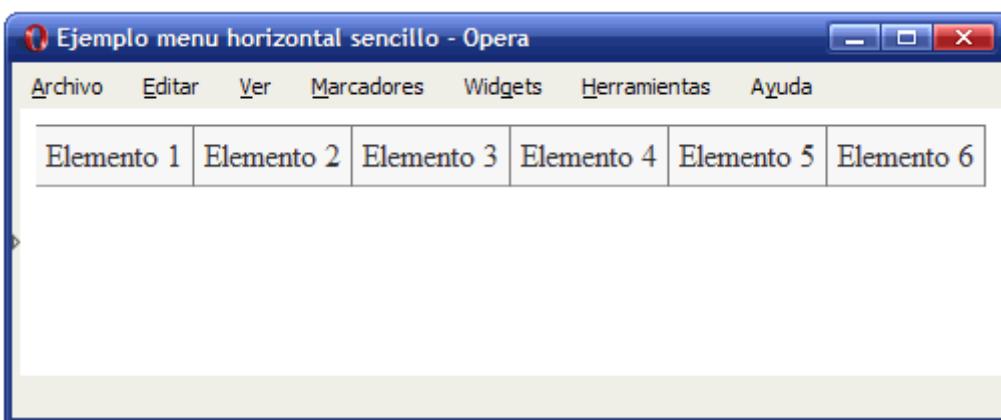


Figura 9.12. Menú horizontal: relleno y borde de los elementos

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

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

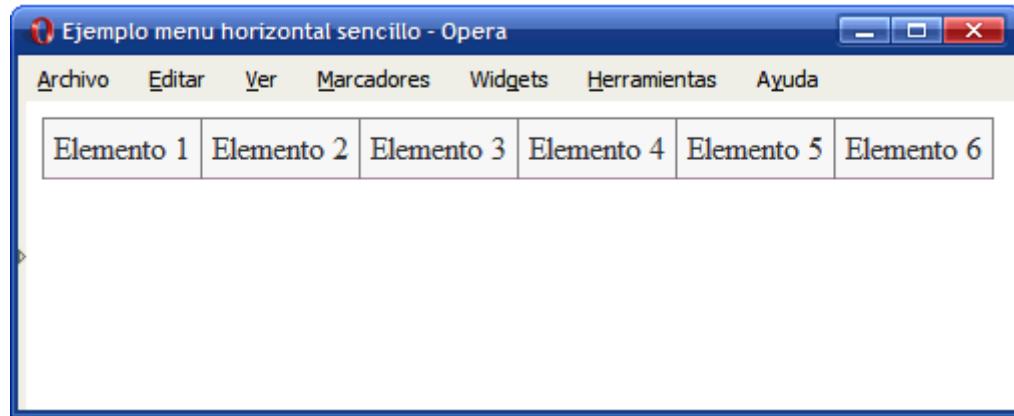


Figura 9.13. Aspecto final del menú horizontal sencillo creado con CSS

El código CSS final se muestra a continuación:

```
ul.menu {
    clear: both;
    float: left;
    width: 100%;
    list-style: none;
    margin: 0;
    padding: 0;
    border-left: 1px solid #7C7C7C;
}
ul.menu li {
    float: left;
}
ul.menu li a {
    padding: .3em;
    display: block;
    text-decoration: none;
    color: #333;
    background: #F4F4F4;
    border-top: 1px solid #7C7C7C;
    border-right: 1px solid #7C7C7C;
    border-bottom: 1px solid #9C9C9C;
}
```

9.1.3. Ejemplo de Menú horizontal con iconos.



```
<!DOCTYPE html>
<html>
<meta name="viewport" content="width=device-width, initial-scale=1">

<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-awesome.min.css">
```

```
<style>
    body {margin:0;}

    .icon-bar {
        width: 100%;
        background-color: #555;
        overflow: auto;
    }

    .icon-bar a {
        float: left;
        width: 20%;
        text-align: center;
        padding: 12px 0;
        transition: all 0.3s ease;
        color: white;
        font-size: 36px;
    }

    .icon-bar a:hover {
        background-color: #000;
    }

    .active {
        background-color: #04AA6D;
    }
</style>
<body>

<div class="icon-bar">
    <a class="active" href="#"><i class="fa fa-home"></i></a>
    <a href="#"><i class="fa fa-search"></i></a>
    <a href="#"><i class="fa fa-envelope"></i></a>
    <a href="#"><i class="fa fa-globe"></i></a>
    <a href="#"><i class="fa fa-trash"></i></a>
</div>

</body>
</html>
```

Capítulo 10. Tablas

10.1. Estilos básicos

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

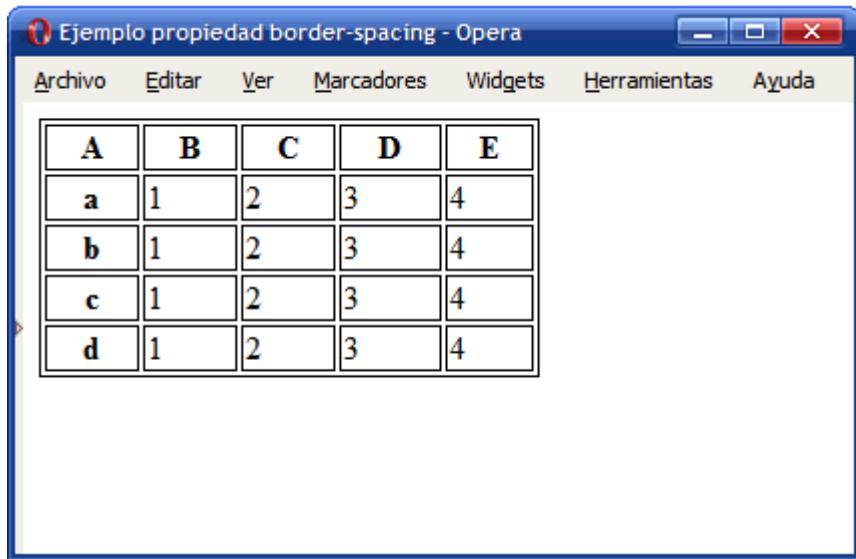


Figura 10.1. Aspecto por defecto de los bordes de una tabla

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

```
.normal {  
    width: 250px;  
    border: 1px solid #000;  
}  
.normal th, .normal td {  
    border: 1px solid #000;  
}  
  
<table class="normal" summary="Tabla genérica">  
    <tr>  
        <th scope="col">A</th>  
        <th scope="col">B</th>  
        <th scope="col">C</th>  
        <th scope="col">D</th>  
        <th scope="col">E</th>  
    </tr>  
    ...  
</table>
```

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

El 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:

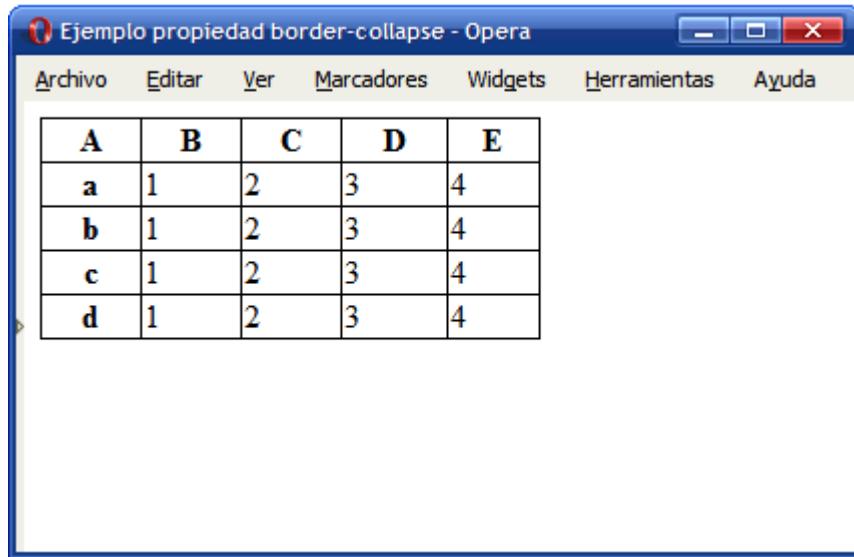


Figura 10.2. Ejemplo de propiedad border-collapse

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 modelo por defecto si no se indica lo contrario) se puede utilizar la propiedad `border-spacing` para controlar la separación entre los bordes de cada celda.

border-spacing Espaciado entre bordes

| | |
|----------------|---|
| Valores | <code><medida> <medida>? inherit</code> |
|----------------|---|

| | |
|--------------------|------------------|
| Se aplica a | Todas las tablas |
|--------------------|------------------|

| | |
|----------------------|---|
| Valor inicial | 0 |
|----------------------|---|

| | |
|--------------------|--|
| Descripción | Establece la separación entre los bordes de las celdas adyacentes de una tabla |
|--------------------|--|

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

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



Figura 10.3. Ejemplo de propiedad border-spacing

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

ha establecido de forma explícita ningún modelo de bordes, el navegador utiliza el modelo `separate`.

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

Ejercicio 11 [Ver enunciado](#)

Ejemplo de tabla con colores en diferentes filas y background resaltado con la pseudoclase hover.

Colored Table Header

| Firstname | Lastname | Savings |
|-----------|----------|---------|
| Peter | Griffin | \$100 |
| Lois | Griffin | \$150 |
| Joe | Swanson | \$300 |
| Cleveland | Brown | \$250 |

```
<!DOCTYPE html>
<html>
<head>
<style>
table {
    border-collapse: collapse;
    width: 100%;
}

th, td {
    text-align: left;
    padding: 8px;
}

tr:nth-child(even){background-color: #f2f2f2}

tr:hover {background-color: coral;}

th {
    background-color: #04AA6D;
    color: white;
}
</style>
</head>
<body>

<h2>Colored Table Header</h2>

<table>
<tr>
<th>Firstname</th>
<th>Lastname</th>
```

```
<th>Savings</th>
</tr>
<tr>
  <td>Peter</td>
  <td>Griffin</td>
  <td>$100</td>
</tr>
<tr>
  <td>Lois</td>
  <td>Griffin</td>
  <td>$150</td>
</tr>
<tr>
  <td>Joe</td>
  <td>Swanson</td>
  <td>$300</td>
</tr>
<tr>
  <td>Cleveland</td>
  <td>Brown</td>
  <td>$250</td>
</tr>
</table>

</body>
</html>
```

Capítulo 11. Formularios

11.1. Estilos básicos

11.1.1. Mostrar un botón como un enlace

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

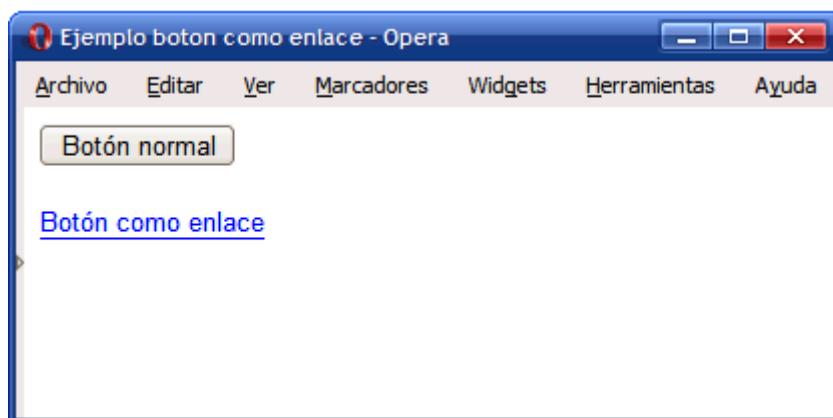


Figura 11.1. Mostrando un botón como si fuera un enlace

Las reglas CSS del ejemplo anterior son las siguientes:

```
.enlace {  
    border: 0;  
    padding: 0;  
    background-color: transparent;  
    color: blue;  
    border-bottom: 1px solid blue;  
}  
  
<input type="button" value="Botón normal" />  
  
<input class="enlace" type="button" value="Botón como enlace" />
```

11.1.2. Inputs submit, reset y button.

Se podrá cambiar el estilo de los inputs relacionados con los botones (Submit, reset y button)

```
<style>  
input[type=button], input[type=submit], input[type=reset] {  
    background-color: #04AA6D;  
    border: none;  
    color: white;  
    padding: 16px 32px;  
  
    margin: 4px 2px;  
    cursor: pointer;  
}  
  
<input type="button" value="Button">  
<input type="reset" value="Reset">  
<input type="submit" value="Submit">
```



11.1.3. 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:

Two input fields are shown vertically. The top field is labeled 'First Name' and the bottom field is labeled 'Last Name'. Both fields have a light gray border and a white interior. There is a small amount of padding inside each field, creating a visual separation between the text and the field's boundaries.

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

```
input[type=text] {  
    width: 100%;  
    padding: 12px 20px;  
    margin: 8px 0;  
    box-sizing: border-box;  
}
```

11.1.4. Labels alineadas y formateadas

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

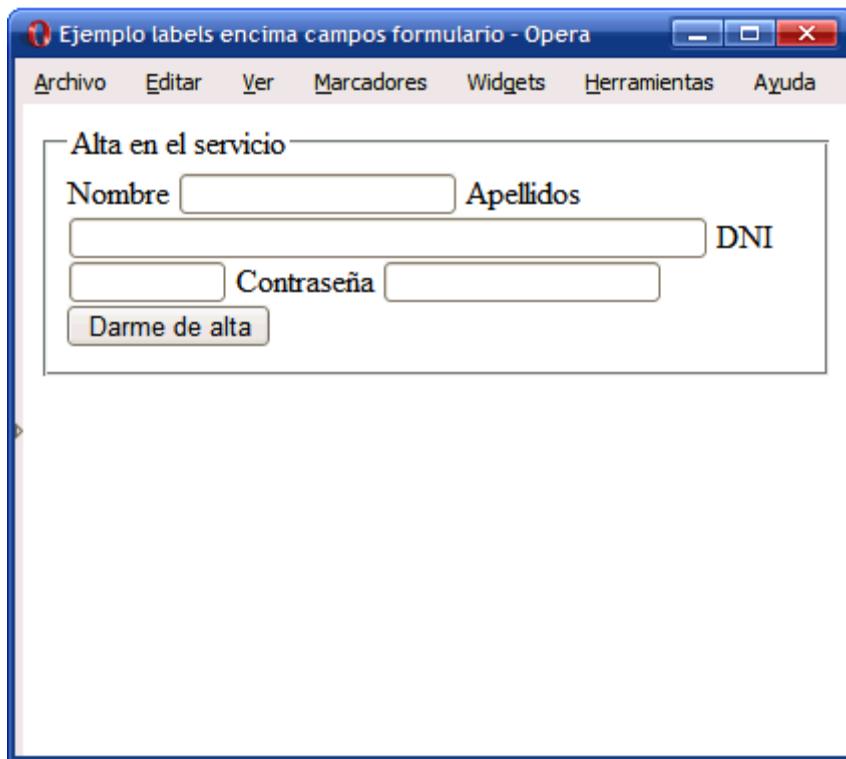


Figura 11.3. Aspecto por defecto que muestran los formularios

El código HTML del ejemplo anterior es el siguiente:

```
<form>  
<fieldset>  
    <legend>Alta en el servicio</legend>  
  
    <label for="nombre">Nombre</label>  
    <input type="text" id="nombre" />  
  
    <label for="apellidos">Apellidos</label>  
    <input type="text" id="apellidos" size="50" />  
  
    <label for="dni">DNI</label>
```

```
<input type="text" id="dni" size="10" maxlength="9" />  
  
<label for="contrasena">Contraseña</label>  
<input type="password" id="contrasena" />  
  
<input class="btn" type="submit" value="Darme de alta" />  
</fieldset>  
</form>
```

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

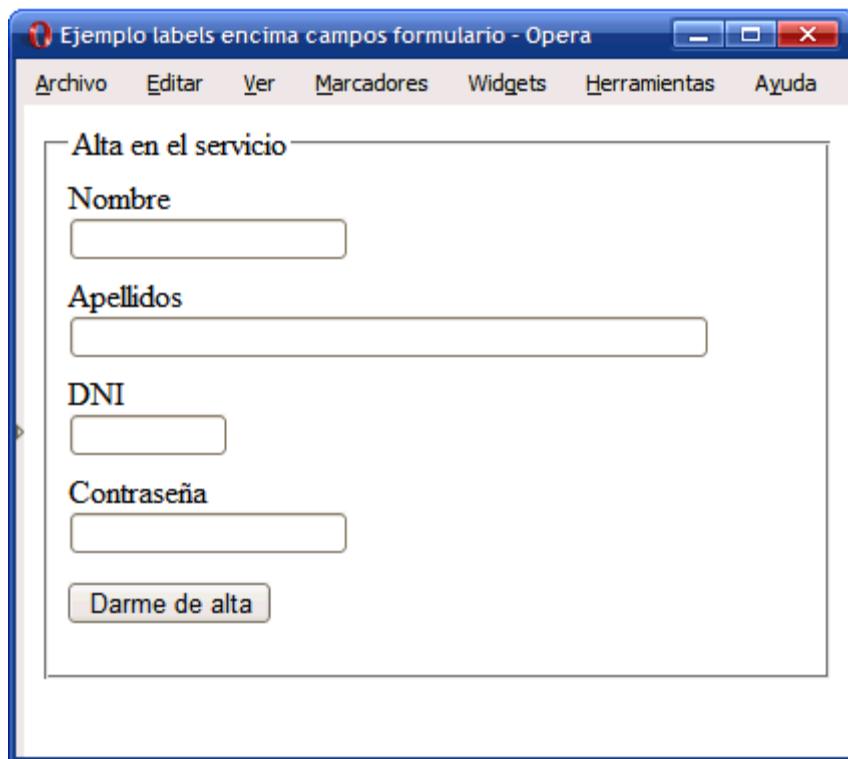


Figura 11.4. Mostrando las etiquetas label encima de los campos del formulario

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

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

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

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

Otra buena práctica sería incluir un background que destaque los campos y utilizar toda la anchura del formulario:

The form consists of three text input fields and one dropdown select field, all contained within a light gray div. The first two inputs have placeholder text "Your name.." and "Your last name..". The third input has the value "Australia". Below the inputs is a large green rectangular button with the word "Submit" in white.

```
input[type=text], select {
    width: 100%;
    padding: 12px 20px;
    margin: 8px 0;
    display: inline-block;
    border: 1px solid #ccc;
    border-radius: 4px;
    box-sizing: border-box;
}

input[type=submit] {
    width: 100%;
    background-color: #4CAF50;
    color: white;
    padding: 14px 20px;
    margin: 8px 0;
    border: none;
    border-radius: 4px;
    cursor: pointer;
}

input[type=submit]:hover {
    background-color: #45a049;
}

div {
    border-radius: 5px;
    background-color: #f2f2f2;
    padding: 20px;
}

<div>
    <form action="/action_page.php">
        <label for="fname">First Name</label>
        <input type="text" id="fname" name="firstname" placeholder="Your name..">

        <label for="lname">Last Name</label>
        <input type="text" id="lname" name="lastname" placeholder="Your last name..">

        <label for="country">Country</label>
        <select id="country" name="country">
            <option value="australia">Australia</option>
            <option value="canada">Canada</option>
            <option value="usa">USA</option>
        </select>

        <input type="submit" value="Submit">
    </form>
</div>
```

11.2. Estilos avanzados

11.2.1. 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:

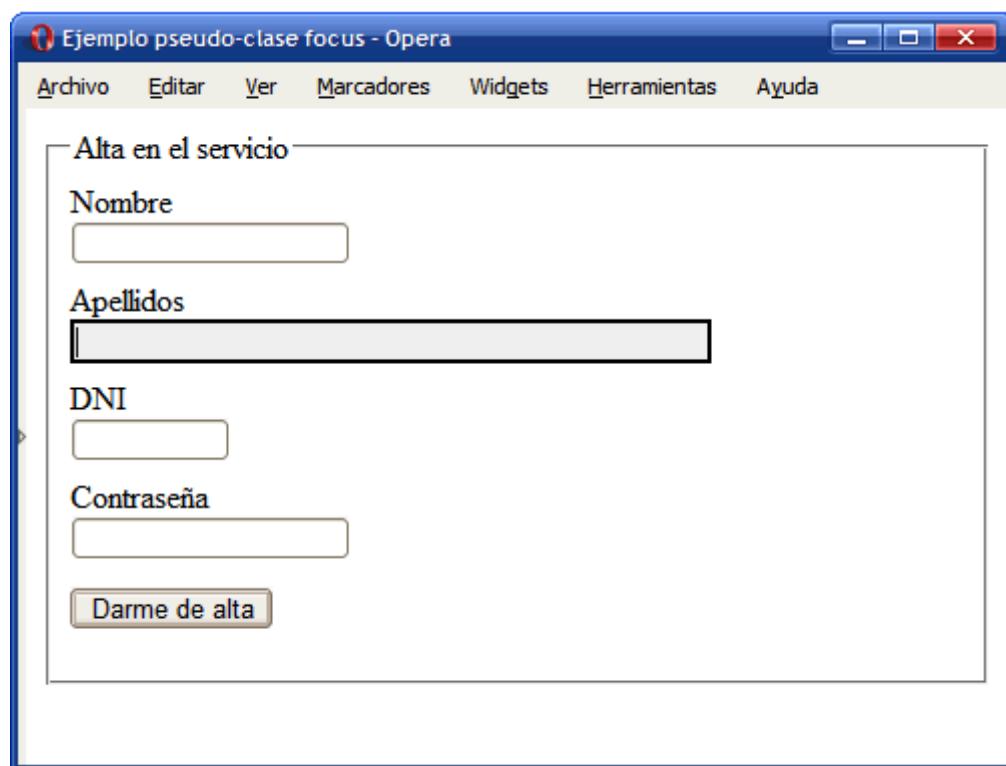


Figura 11.8. Ejemplo de pseudo-clase `:focus`

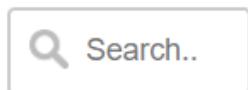
Añadiendo la pseudo-clase `:focus` después del selector normal, el navegador se encarga de aplicar esos estilos cuando el usuario activa el elemento:

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

Se podrá cambiar también el border:

```
input[type=text]:focus {  
    background-color: lightblue;  
    border: 3px solid #555;  
}
```

Incluir una imagen en el background del input junto con el placeholder, puede servir de gran ayuda:



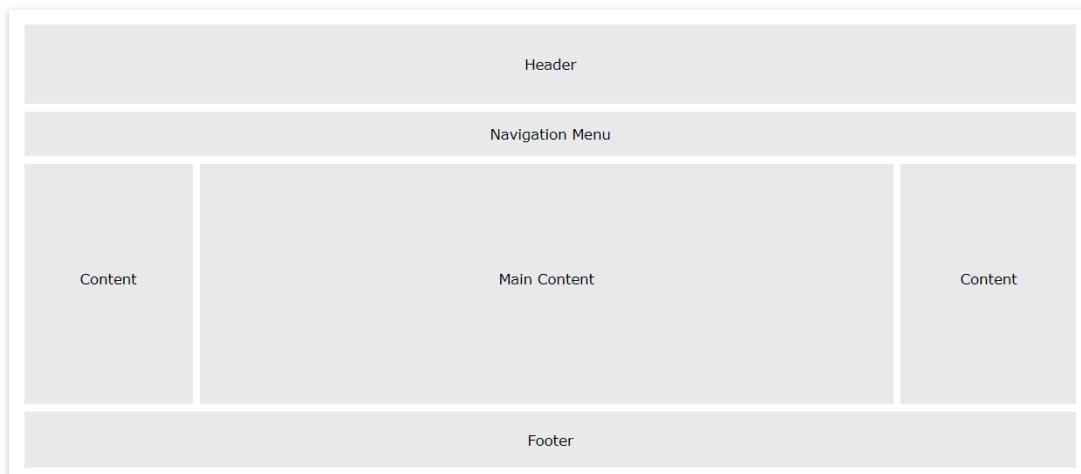
```
input[type=text] {  
    width: 100%;  
    box-sizing: border-box;  
    border: 2px solid #ccc;  
    border-radius: 4px;  
    font-size: 16px;  
    background-color: white;  
    background-image: url('searchicon.png');  
    background-position: 10px 10px;  
    background-repeat: no-repeat;  
    padding: 12px 20px 12px 40px;  
}  
  
<form>  
    <input type="text" name="search" placeholder="Search..">  
</form>
```

Ejercicio 12 Ver enunciado.

Capítulo 12. Layout

El diseño de las páginas web habituales se divide en bloques: cabecera, menú, contenidos y pie de página. Visualmente, los bloques se disponen en varias filas y columnas.

Una de las estructuras más comunes es la siguiente:



12.1. Centrar una página horizontalmente

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

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

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



Figura 12.1. Página de anchura fija centrada mediante CSS



Figura 12.2. Página de anchura fija centrada mediante CSS



Figura 12.3. Página de anchura fija centrada mediante CSS

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

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

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

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

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

Estableciendo la anchura del elemento contenedor mediante un porcentaje, su anchura se adapta de forma continua a la anchura de la ventana del navegador. De esta forma, si se reduce la anchura de la ventana del navegador, la página se verá más estrecha y si se maximiza la ventana del navegador, la página se verá más ancha.

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



Figura 12.4. Página de anchura variable centrada mediante CSS



Figura 12.5. Página de anchura variable centrada mediante CSS



Figura 12.6. Página de anchura variable centrada mediante CSS

12.2. Centrar una página verticalmente

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

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

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

```
#contenedor {  
    width: 300px;  
    height: 250px;  
    margin: auto;  
}  
  
<body>  
    <div id="contenedor">  
        <h1>Lorem ipsum dolor sit amet</h1>  
        ...  
    </div>  
</body>
```

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

12.3. Estructuras o layouts

12.3.1. Header

Un encabezado generalmente se encuentra en la parte superior del sitio web (o justo debajo del menú de navegación superior). A menudo contiene un logotipo o el nombre del sitio web:



```
header {  
    background-color: #F1F1F1;  
    text-align: center;  
    padding: 20px;  
}
```

12.3.2. Navigation Bar

Una barra de navegación contiene una lista de enlaces para ayudar a los visitantes a navegar por su sitio web:

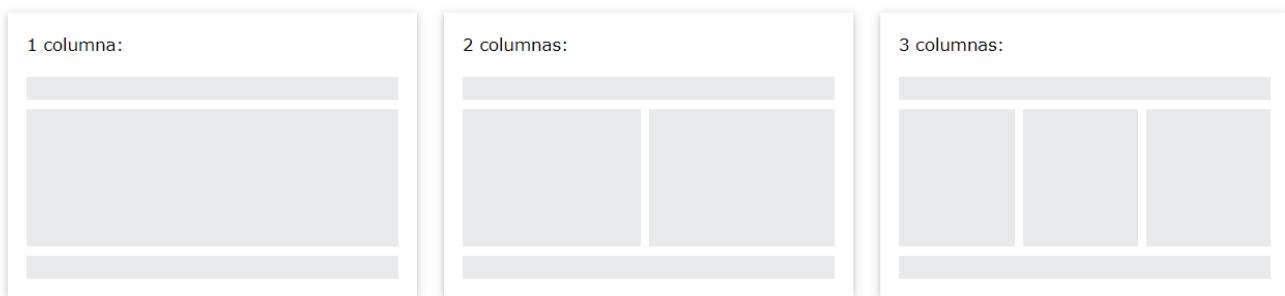


```
nav {  
    overflow: hidden;  
    background-color: #333;  
}  
nav a {  
    float: left;  
    display: block;  
    color: #f2f2f2;  
    text-align: center;  
    padding: 14px 16px;  
    text-decoration: none;  
}  
nav a:hover {  
    background-color: #ddd;  
    color: black;  
}
```

12.3.3. Content

El diseño de esta sección a menudo depende de los usuarios objetivo. El diseño más común es uno (o combinarlos) de los siguientes:

- 1 columna (a menudo se usa para navegadores móviles).
- columnas (a menudo se usa para tabletas y computadoras portátiles).
- Diseño de 3 columnas (solo se usa para escritorios).



Tener en cuenta las siguientes etiquetas semánticas de HTML5:

El elemento `<main>` especifica el contenido principal de la página, o sea, de mayor relevancia para el usuario. Por eso es una buena práctica utilizarlos sólo una vez dentro del código.

El elemento `<section>` representa una parte o una sección de la página. Dentro de la sección, en general, contiene el título seguido del contenido relacionado.

El elemento `<article>` declara un contenido que es independiente de cualquier otras secciones. Podemos utilizarlos en los contenidos de blog, fórum, artículos de un periódico, comentarios de los usuarios etc. Es una buena práctica atribuir un título para cada etiqueta de `<article>`.

El elemento `<aside>` es utilizado para crear un contenido de apoyo o adicional al contenido principal. Es una manera de enfatizar o posicionar un contenido aislado del restante de la página, como una sidebar.

```
section {  
    float: left;  
    width: 33.33%;  
    padding: 15px;  
}
```

12.3.4. Footer

El pie de página se coloca en la parte inferior de su página. A menudo contiene información como derechos de autor e información de contacto:



```
footer {
    background-color: #F1F1F1;
    text-align: center;
    padding: 10px;
}
```

12.4. Alturas/anchuras máximas y mínimas

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

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

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

max-width Anchura máxima

Valores `<medida>` | `<porcentaje>` | `none` | `inherit`

Se aplica a Todos los elementos salvo filas y grupos de filas de tablas

Valor inicial `none`

Descripción Permite definir la anchura máxima de un elemento

min-width Anchura mínima

Valores `<medida>` | `<porcentaje>` | `inherit`

Se aplica a Todos los elementos salvo filas y grupos de filas de tablas

Valor inicial `0`

Descripción Permite definir la anchura mínima de un elemento

max-height Altura máxima**Valores** <medida> | <porcentaje> | none | inherit**Se aplica a** Todos los elementos salvo columnas y grupos de columnas de tablas**Valor inicial** none**Descripción** Permite definir la altura máxima de un elemento**min-height** Altura mínima**Valores** <medida> | <porcentaje> | inherit**Se aplica a** Todos los elementos salvo columnas y grupos de columnas de tablas**Valor inicial** 0**Descripción** Permite definir la altura mínima de un elemento

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

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

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

Capítulo 15. Ejercicios

15.1. Ejercicio 1

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

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

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

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

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

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

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

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

</style>
</head>

<body>

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

<div class="normal">
<p>Phasellus eu velit sed lorem sodales egestas. Ut feugiat. <span><a href="#">Donec
```

```

porttitor</a>, magna eu varius luctus,</span> metus massa tristique massa, in imperdiet
est velit vel magna. Phasellus erat. Duis risus. <a href="#">Maecenas dictum</a>, nibh
vitae pellentesque auctor, tellus velit consectetur tellus, tempor pretium felis
tellus at metus.</p>

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

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

</body>
</html>

```

15.2. Ejercicio 2

A partir del código HTML proporcionado, añadir las reglas CSS necesarias para que la página resultante tenga el mismo aspecto que el de la siguiente imagen:

Lorem ipsum dolor sit amet

Nulla pretium. Sed tempus nunc vitae neque. **Suspendisse gravida**, metus a scelerisque sollicitudin,
lacus velit ultricies nisl, nonummy tempus neque diam quis felis. **Etiam sagittis tortor** sed arcu sagittis
tristique.

Aliquam tincidunt, sem eget volutpat porta

Vivamus velit dui, placerat vel, feugiat in, ornare et, urna. [Aenean turpis metus, aliquam non, tristique](#)
[in,](#) pretium varius, sapien. Proin vitae nisi. Suspendisse porttitor purus ac elit. Suspendisse eleifend odio
at dui. In in elit sed metus pretium elementum.

Título de la tabla

| | Título columna 1 | Título columna 2 |
|---------------|-------------------------|--------------------------|
| Título fila 1 | Donec purus ipsum | Curabitur <i>blandit</i> |
| Título fila 2 | Donec purus ipsum | Curabitur <i>blandit</i> |
| | Título columna 1 | Título columna 2 |

Donec purus ipsum, posuere id, venenatis at, placerat ac, lorem. Curabitur *blandit*, eros sed gravida
aliquet, risus justo porta lorem, ut mollis lectus tortor in orci. Pellentesque nec augue.

Fusce nec felis eu diam pretium adipiscing. [Nunc elit elit, vehicula vulputate](#), venenatis in, posuere id,
lorem. Etiam sagittis, tellus in ultrices accumsan, diam nisi feugiat ante, eu congue magna mi non nisl.

Vivamus ultrices aliquet augue. [Donec arcu pede, pretium vitae, rutrum aliquet, tincidunt blandit, pede.](#)
[Aliquam in nisi.](#) Suspendisse volutpat. Nulla facilisi. Ut ullamcorper nisi quis mi.

Figura 15.1. Aspecto final de la página

A continuación se muestra el código HTML de la página sin estilos:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>Ejercicio de selectores</title>
</head>

<body>
<h1 id="titulo">Lorem ipsum dolor sit amet</h1>

<p>Nulla pretium. Sed tempus nunc vitae neque. <strong>Suspendisse gravida</strong>, metus a scelerisque sollicitudin, lacus velit ultricies nisl, nonummy tempus neque diam quis felis. <span class="destacado">Etiam sagittis tortor</span> sed arcu sagittis tristique.</p>

<h2 id="subtitulo">Aliquam tincidunt, sem eget volutpat porta</h2>

<p>Vivamus velit dui, placerat vel, feugiat in, ornare et, urna. <a href="#">Aenean turpis metus, <em>aliquam non</em>, tristique in</a>, pretium varius, sapien. Proin vitae nisi. Suspendisse <span class="especial">porttitor purus ac elit</span>. Suspendisse eleifend odio at dui. In in elit sed metus pretium elementum.</p>

<table summary="Descripción de la tabla y su contenido">
<caption>Título de la tabla</caption>
<thead>
<tr>
<th scope="col"></th>
<th scope="col" class="especial">Título columna 1</th>
<th scope="col" class="especial">Título columna 2</th>
</tr>
</thead>

<tfoot>
<tr>
<th scope="col"></th>
<th scope="col">Título columna 1</th>
<th scope="col">Título columna 2</th>
</tr>
</tfoot>

<tbody>
<tr>
<th scope="row" class="especial">Título fila 1</th>
<td>Donec purus ipsum</td>
<td>Curabitur <em>blandit</em></td>
</tr>
<tr>
<th scope="row">Título fila 2</th>
<td>Donec <strong>purus ipsum</strong></td>
<td>Curabitur blandit</td>
</tr>
</tbody>
</table>
```

```
</table>

<div id="adicional">


Donec purus ipsum, posuere id, venenatis at, <span>placerat ac, lorem</span>.
Curabitur blandit, eros sed gravida aliquet, risus justo
porta lorem, ut mollis lectus tortor in orci. Pellentesque nec augue.</p>



Fusce nec felis eu diam pretium adipiscing. <span id="especial">Nunc elit elit,
vehicula vulputate</span>, venenatis in,
posuere id, lorem. Etiam sagittis, tellus in ultrices accumsan, diam nisi feugiat ante,
eu congue magna mi non nisl.</p>



Vivamus ultrices aliquet augue. <a href="#">Donec arcu pede, pretium vitae</a>,
rutrum aliquet, tincidunt blandit, pede.
Aliquam in nisi. Suspendisse volutpat. Nulla facilisi. Ut ullamcorper nisi quis mi.</p>
</div>

</body>
</html>


```

Aunque la propiedad que modifica el color del texto se explica detalladamente en los próximos capítulos, en este ejercicio solamente es preciso conocer que la propiedad se llama `color` y que como valor se puede indicar directamente el nombre del color.

Los nombres de los colores también están estandarizados y se corresponden con el nombre en inglés de cada color. En este ejercicio, se deben utilizar los colores: `teal`, `red`, `blue`, `orange`, `purple`, `olive`, `fuchsia` y `green`.

15.3. Ejercicio 3

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

LOGOTIPO

Buscar

Noticias

dd/mm/aaaa [Lorem ipsum dolor sit amet](#)
 dd/mm/aaaa [Consectetuer adipiscing elit](#)
 dd/mm/aaaa [Donec molestie nunc eu sapien](#)
 dd/mm/aaaa [Maecenas aliquam dolor eget metus](#)
 dd/mm/aaaa [Fusce tristique lorem id metus](#)

Enlaces relacionados

[Proin placerat](#)
[Nulla in felis](#)
[Nam luctus](#)
Publicidad

Etiam fermentum, nisl tincidunt blandit interdum, massa velit posuere dolor, sed euismod sem odio at mi. Duis porta placerat arcu. Nullam felis pede, commodo vel, suscipit a, molestie vel, felis. Maecenas mattis est vel est.

[Seguir leyendo...](#)

Lorem ipsum dolor sit amet, consectetuer adipiscing elit



Nullam est lacus, suscipit ut, dapibus quis, condimentum ac, risus. Vivamus vestibulum, ipsum sollicitudin faucibus pharetra, dolor metus fringilla dui, vel aliquet pede diam tempor tortor. Vestibulum pulvinar urna et quam. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Nullam vel turpis vitae dui imperdiet laoreet. Quisque eget ipsum. Donec commodo, turpis vel venenatis sollicitudin, quam ante convallis justo, sed eleifend justo lectus quis sapien. Ut consequat libero eget est.

[Seguir leyendo...](#)

Vivamus lobortis turpis ac ante fringilla faucibus



Quisque eget ipsum. Donec commodo, turpis vel venenatis sollicitudin, quam ante convallis justo, sed eleifend justo lectus quis sapien. Ut consequat libero eget est.

Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Nullam dictum hendrerit neque. Mauris id ligula non elit mattis semper. Fusce arcu ipsum, tempus eget, tincidunt at, imperdiet in, mi. Sed fermentum cursus dolor. Aenean a diam. Phasellus feugiat. Donec tempor dignissim sem.

[Seguir leyendo...](#)

Phasellus blandit

Praesent sodales imperdiet augue. Mauris lorem felis, semper eu, tincidunt eu, sollicitudin eget, sem. Nulla facilisi. Morbi ut enim ut enim ultricies dapibus.

[Seguir leyendo...](#)

Nullam vel turpis

Donec commodo, turpis vel venenatis sollicitudin, quam ante convallis justo, sed eleifend justo lectus quis sapien. Ut consequat libero eget est.

[Seguir leyendo...](#)

[Nulla](#) | [Pharetra](#) | [Luctus](#) | [Ipsum](#) | [Proin](#) | [Placerat](#)

© Copyright Lorem ipsum

Figura 15.2. Página original

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

LOGOTIPO

[Lorem Ipsum Dolor Sit Amet](#)

Noticias

[dd/mm/aaaa Lorem ipsum dolor sit amet](#)

[dd/mm/aaaa Consectetuer adipiscing elit](#)

[dd/mm/aaaa Donec molestie nunc eu sapien](#)

[dd/mm/aaaa Maecenas aliquam dolor eget metus](#)

[dd/mm/aaaa Fusce tristique lorem id metus](#)

Enlaces relacionados

[Proin placerat](#)

[Nulla in felis](#)

[Nam luctus](#)

Publicidad

[Etiam fermentum, nisl tincidunt blandit interdum, massa velit posuere dolor, sed euismod sem odio at mi.](#)

[Duis porta placerat arcu. Nullam felis pede, commodo vel, suscipit a, molestie vel, felis.](#)

[Maecenas mattis est vel est.](#)

[Seguir leyendo...](#)

Lorem ipsum dolor sit amet, consectetuer adipiscing elit



Nulum est lacus, suscipit ut, dapibus quis, condimentum ac, risus. Vivamus vestibulum, ipsum sollicitudin faucibus pharetra, dolor metus fringilla dui, vel aliquet pede diam tempore tortor. Vestibulum pulvinar urna et quam. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Nulum vel turpis vitae dui imperdiet laoreet. Quisque eget ipsum.

Donec commodo, turpis vel venenatis sollicitudin, quam ante convallis justo, sed eleifend justo lectus quis sapien. Ut consequat libero eget est.

[Seguir leyendo...](#)

Vivamus lobortis turpis ac ante fringilla faucibus



Quisque eget ipsum. Donec commodo, turpis vel venenatis sollicitudin, quam ante convallis justo, sed eleifend justo lectus quis sapien. Ut consequat libero eget est.

Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Nullam dictum hendrerit neque. Mauris id ligula non elit mattis semper. Fusce arcu ipsum, tempus eget, tincidunt at, imperdiet in, mi.

Sed fermentum cursus dolor. Aenean a diam. Phasellus feugiat. Donec tempor dignissim sem.

[Seguir leyendo...](#)

Phasellus blandit

Praesent sodales imperdiet augue. Mauris lorem felis, semper eu, tincidunt eu, sollicitudin eget, sem. Nulla facilisi. Morbi ut enim ut enim ultricies dapibus.

[Seguir leyendo...](#)

Nullam vel turpis

Donec commodo, turpis vel venenatis sollicitudin, quam ante convallis justo, sed eleifend justo lectus quis sapien. Ut consequat libero eget est.

[Seguir leyendo...](#)

[Nulla](#) | [Pharetra](#) | [Luctus](#) | [Ipsum](#) | [Proin](#) | [Placerat](#)

© Copyright Lorem ipsum

Figura 15.3. Página con márgenes y rellenos

15.4. Ejercicio 4

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

LOGOTIPO

[Lorem Ipsum Dolor Sit Amet](#)
Buscar

Noticias

dd/mm/aaaa [Lorem ipsum dolor sit amet](#)
 dd/mm/aaaa [Consectetuer adipiscing elit](#)
 dd/mm/aaaa [Donec molestie nunc eu sapien](#)
 dd/mm/aaaa [Maecenas aliquam dolor eget metus](#)
 dd/mm/aaaa [Fusce tristique lorem id metus](#)

Enlaces relacionados

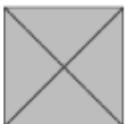
[Proin placerat](#)
[Nulla in felis](#)
[Nam luctus](#)

Publicidad

Etiam fermentum, nisl tincidunt blandit interdum, massa vel posuere dolor, sed euismod sem odio at mi.
 Duis porta placerat arcu. Nullam felis pede, commodo vel, suscipit a, molestie vel, felis. Maecenas mattis est vel est.

[Seguir leyendo...](#)

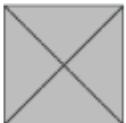
Lorem ipsum dolor sit amet, consectetuer adipiscing elit



Nullam est lacus, suscipit ut, dapibus quis, condimentum ac, risus. Vivamus vestibulum, ipsum sollicitudin faucibus pharetra, dolor metus fringilla dui, vel aliquet pede diam tempor tortor. Vestibulum pulvinar urna et quam. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Nullam vel turpis vitae dui imperdiet laoreet. Quisque eget ipsum. Donec commodo, turpis vel venenatis sollicitudin, quam ante convallis justo, sed eleifend justo lectus quis sapien. Ut consequat libero eget est.

[Seguir leyendo...](#)

Vivamus lobortis turpis ac ante fringilla faucibus



Quisque eget ipsum. Donec commodo, turpis vel venenatis sollicitudin, quam ante convallis justo, sed eleifend justo lectus quis sapien. Ut consequat libero eget est. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Nullam dictum hendrerit neque. Mauris id ligula non elit mattis semper. Fusce arcu ipsum, tempus eget, tincidunt at, imperdiet in, mi. Sed fermentum cursus dolor. Aenean a diam. Phasellus feugiat. Donec tempor dignissim sem.

[Seguir leyendo...](#)

Phasellus blandit

Praesent sodales imperdiet augue. Mauris lorem felis, semper eu, tincidunt eu, sollicitudin eget, sem. Nulla facilisi. Morbi ut enim ut enim ultricies dapibus.

[Seguir leyendo...](#)

Nullam vel turpis

Donec commodo, turpis vel venenatis sollicitudin, quam ante convallis justo, sed eleifend justo lectus quis sapien. Ut consequat libero eget est.

[Seguir leyendo...](#)

Figura 15.4. Página original

1. Eliminar el borde gris que muestran por defecto todos los elementos.
2. El elemento `#menu` debe tener un borde inferior de 1 píxel y azul (#004C99).
3. El elemento `#noticias` muestra un borde de 1 píxel y gris claro (#C5C5C5).
4. El elemento `#publicidad` debe mostrar un borde discontinuo de 1 píxel y de color #CC6600.
5. El lateral formado por el elemento `#secundario` muestra un borde de 1 píxel y de color #CC6600.
6. El elemento `#pie` debe mostrar un borde superior y otro inferior de 1 píxel y color gris claro #C5C5C5.

LOGOTIPO

Buscar [Lorem ipsum dolor sit amet](#)

Noticias

dd/mm/aaaa [Lorem ipsum dolor sit amet](#)
 dd/mm/aaaa [Consectetuer adipiscing elit](#)
 dd/mm/aaaa [Donec molestie nunc eu sapien](#)
 dd/mm/aaaa [Maecenas aliquam dolor eget metus](#)
 dd/mm/aaaa [Fusce tristique lorem id metus](#)

Enlaces relacionados

[Proin placerat](#)
[Nulla in felis](#)
[Nam luctus](#)

Publicidad

Etiam fermentum, nisl tincidunt blandit interdum, massa velit posuere dolor, sed euismod sem odio at mi.
 Duis porta placerat arcu. Nullam felis pede, commodo vel, suscipit a, molestie vel, felis.
 Maecenas mattis est vel est.

[Seguir leyendo...](#)

Lorem ipsum dolor sit amet, consectetuer adipiscing elit



Nulum est lacus, suscipit ut, dapibus quis, condimentum ac, risus. Vivamus vestibulum, ipsum sollicitudin faucibus pharetra, dolor metus fringilla dui, vel aliquet pede diam tempore tortor. Vestibulum pulvinar urna et quam. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Nulum vel turpis vitae dui imperdiet laoreet. Quisque eget ipsum.

Donec commodo, turpis vel venenatis sollicitudin, quam ante convallis justo, sed eleifend justo lectus quis sapien. Ut consequat libero eget est.

[Seguir leyendo...](#)

Vivamus lobortis turpis ac ante fringilla faucibus



Quisque eget ipsum. Donec commodo, turpis vel venenatis sollicitudin, quam ante convallis justo, sed eleifend justo lectus quis sapien. Ut consequat libero eget est.

Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Nullam dictum hendrerit neque. Mauris id ligula non elit mattis semper. Fusce arcu ipsum, tempus eget, tincidunt at, imperdiet in, mi.

Sed fermentum cursus dolor. Aenean a diam. Phasellus feugiat. Donec tempor dignissin sem.

[Seguir leyendo...](#)

Phasellus blandit

Praesent sodales imperdiet augue. Mauris lorem felis, semper eu, tincidunt eu, sollicitudin eget, sem. Nulla facilisi. Morbi ut enim ut enim ultricies dapibus.

[Seguir leyendo...](#)

Nullam vel turpis

Donec commodo, turpis vel venenatis sollicitudin, quam ante convallis justo, sed eleifend justo lectus quis sapien. Ut consequat libero eget est.

[Seguir leyendo...](#)

[Nulla](#) | [Pharetra](#) | [Luctus](#) | [Ipsum](#) | [Proin](#) | [Placerat](#)

© Copyright Lorem ipsum

Figura 15.5. Página con bordes

15.5. Ejercicio 5

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

LOGOTIPO

Buscar [LoremIpsumDolorSitAmet](#)

[Noticias](#)
dd/mm/aaaa [Lorem ipsum dolor sit amet](#)

dd/mm/aaaa [Consectetuer adipiscing elit](#)

dd/mm/aaaa [Donec molestie nunc eu sapien](#)

dd/mm/aaaa [Maecenas aliquam dolor eget metus](#)

dd/mm/aaaa [Fusce tristique lorem id metus](#)

[Enlaces relacionados](#)

[Proin placerat](#)

[Nulla in felis](#)

[Nam luctus](#)

[Publicidad](#)

Etiam fermentum, nisl tincidunt blandit interdum, massa velit posuere dolor, sed euismod sem odio at mi.

Duis porta placerat arcu. Nullam felis pede, commodo vel, suscipit a, molestie vel, felis.

Maecenas mattis est vel est.

[Seguir leyendo...](#)

Lore ipsum dolor sit amet, consectetur adipiscing elit



Nullam est lacus, suscipit ut, dapibus quis, condimentum ac, risus. Vivamus vestibulum, ipsum sollicitudin faucibus pharetra, dolor metus fringilla dui, vel aliquet pede diam tempor tortor. Vestibulum pulvinar urna et quam. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Nullam vel turpis vitae dui imperdiet laoreet. Quisque eget ipsum.

Donec commodo, turpis vel venenatis sollicitudin, quam ante convallis justo, sed eleifend justo lectus quis sapien. Ut consequat libero eget est.

[Seguir leyendo...](#)

Vivamus lobortis turpis ac ante fringilla faucibus



Quisque eget ipsum. Donec commodo, turpis vel venenatis sollicitudin, quam ante convallis justo, sed eleifend justo lectus quis sapien. Ut consequat libero eget est.

Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Nullam dictum hendrerit neque. Mauris id ligula non elit mattis semper. Fusce arcu ipsum, tempus eget, tincidunt at, imperdiet in, mi.

Sed fermentum cursus dolor. Aenean a diam. Phasellus feugiat. Donec tempor dignissim sem.

[Seguir leyendo...](#)

Phasellus blandit

Praesent sodales imperdiet augue. Mauris lorem felis, semper eu, tincidunt eu, sollicitudin eget, sem. Nulla facilisi. Morbi ut enim ut enim ultricies dapibus.

[Seguir leyendo...](#)

Nullam vel turpis

Donec commodo, turpis vel venenatis sollicitudin, quam ante convallis justo, sed eleifend justo lectus quis sapien. Ut consequat libero eget est.

[Seguir leyendo...](#)

[Nulla](#) | [Pharetra](#) | [Luctus](#) | [Ipsum](#) | [Proin](#) | [Placerat](#)

© Copyright Lorem ipsum

Figura 15.6. Página original

1. Los elementos `#noticias` y `#pie` tiene un color de fondo gris claro (#F8F8F8).
2. El elemento `#publicidad` muestra un color de fondo amarillo claro (#FFF6CD).
3. Los elementos `<h2>` de `#secundario` muestran un color de fondo #DB905C y un pequeño padding de 0.2em.
4. El fondo del elemento `#menu` se construye con un degradado: <https://cssgradient.io/>.
5. El logotipo del sitio se muestra mediante una imagen de fondo del elemento `<h1>` contenido en el elemento `#cabecera` (la imagen se llama `logo.gif`).

Logotipo

Buscar

[Lorem ipsum dolor sit amet](#)

Noticias

- [dd/mm/aaaa Lorem ipsum dolor sit amet](#)
- [dd/mm/aaaa Consectetuer adipiscing elit](#)
- [dd/mm/aaaa Donec molestie nunc eu sapien](#)
- [dd/mm/aaaa Maecenas aliquam dolor eget metus](#)
- [dd/mm/aaaa Fusce tristique lorem id metus](#)

Enlaces relacionados

- [Proin placerat](#)
- [Nulla in felis](#)
- [Nam luctus](#)

Publicidad

Etiam fermentum, nisl tincidunt blandit interdum, massa velit posuere dolor, sed euismod sem odio at mi.

Duis porta placerat arcu. Nullam felis pede, commodo vel, suscipit a, molestie vel, felis. Maecenas mattis est vel est.

[Seguir leyendo...](#)

Lorem ipsum dolor sit amet, consectetuer adipiscing elit



Nulum est lacus, suscipit ut, dapibus quis, condimentum ac, risus. Vivamus vestibulum, ipsum sollicitudin faucibus pharetra, dolor metus fringilla dui, vel aliquet pede diam tempore tortor. Vestibulum pulvinar urna et quam. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Nulum vel turpis vitae dui imperdiet laoreet. Quisque eget ipsum.

Donec commodo, turpis vel venenatis sollicitudin, quam ante convallis justo, sed eleifend justo lectus quis sapien. Ut consequat libero eget est.

[Seguir leyendo...](#)

Vivamus lobortis turpis ac ante fringilla faucibus



Quisque eget ipsum. Donec commodo, turpis vel venenatis sollicitudin, quam ante convallis justo, sed eleifend justo lectus quis sapien. Ut consequat libero eget est.

Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Nullam dictum hendrerit neque. Mauris id ligula non elit mattis semper. Fusce arcu ipsum, tempus eget, tincidunt at, imperdiet in, mi.

Sed fermentum cursus dolor. Aenean a diam. Phasellus feugiat. Donec tempor dignissim sem.

[Seguir leyendo...](#)

Phasellus blandit

Praesent sodales imperdiet augue. Mauris lorem felis, semper eu, tincidunt eu, sollicitudin eget, sem. Nulla facilisi. Morbi ut enim ut enim ultricies dapibus.

[Seguir leyendo...](#)

Nullam vel turpis

Donec commodo, turpis vel venenatis sollicitudin, quam ante convallis justo, sed eleifend justo lectus quis sapien. Ut consequat libero eget est.

[Seguir leyendo...](#)

[Nulla](#) | [Pharetra](#) | [Luctus](#) | [Ipsum](#) | [Proin](#) | [Placerat](#)

© Copyright Lorem ipsum

Figura 15.7. Página con colores e imágenes de fondo

15.6. Ejercicio 6

A partir del código HTML proporcionado:

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

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

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

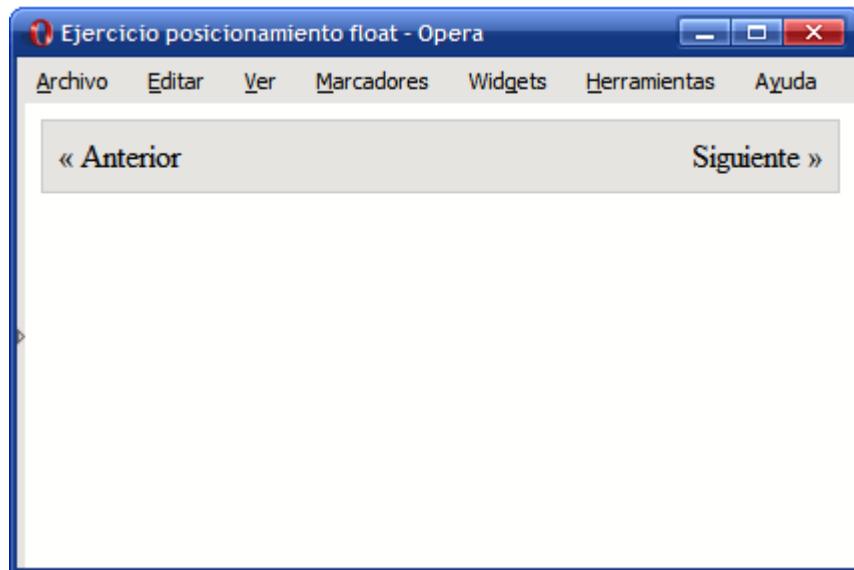


Figura 15.8. Elementos posicionados mediante float

15.7. Ejercicio 7

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

Logotipo

Buscar

[Lorem ipsum dolor sit amet](#)

Noticias

[dd/mm/aaaa Lorem ipsum dolor sit amet](#)

[dd/mm/aaaa Consectetuer adipiscing elit](#)

[dd/mm/aaaa Donec molestie nunc eu sapien](#)

[dd/mm/aaaa Maecenas aliquam dolor eget metus](#)

[dd/mm/aaaa Fusce tristique lorem id metus](#)

Enlaces

[Proin placerat](#)

[Nulla felis](#)

[Nam luctus](#)

relacionados

[Seguir leyendo...](#)

Lore ipsum dolor sit amet, consectetuer adipiscing elit



Nullam est lacus, suscipit ut, dapibus quis, condimentum ac, risus. Vivamus vestibulum, ipsum sollicitudin faucibus pharetra, dolor metus fringilla dui, vel aliquet pede diam tempor tortor. Vestibulum pulvinar urna et quam. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Nullam vel turpis vitae dui imperdiet laoreet. Quisque eget ipsum.

Donec commodo, turpis vel venenatis sollicitudin, quam ante convallis justo, sed eleifend justo lectus quis sapien. Ut consequat libero eget est.

[Seguir leyendo...](#)

Vivamus lobortis turpis ac ante fringilla faucibus



Quisque eget ipsum. Donec commodo, turpis vel venenatis sollicitudin, quam ante convallis justo, sed eleifend justo lectus quis sapien. Ut consequat libero eget est.

Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Nullam dictum hendrerit neque. Mauris id ligula non elit mattis semper. Fusce arcu ipsum, tempus eget, tincidunt at, imperdiet in, mi.

Sed fermentum cursus dolor. Aenean a diam. Phasellus feugiat. Donec tempor dignissim sem.

[Seguir leyendo...](#)

Phasellus blandit

Praesent sodales imperdiet augue. Mauris lorem felis, semper eu, tincidunt eu, sollicitudin eget, sem. Nulla facilisi. Morbi ut enim ut enim ultricies dapibus.

[Seguir leyendo...](#)

Nullam vel turpis

Donec commodo, turpis vel venenatis sollicitudin, quam ante convallis justo, sed eleifend justo lectus quis sapien. Ut consequat libero eget est.

[Seguir leyendo...](#)

[Nulla](#) | [Pharetra](#) | [Luctus](#) | [Ipsum](#) | [Proin](#) | [Placerat](#)

© Copyright Lorem ipsum

Figura 15.9. Página original

1. La fuente base de la página debe ser: color negro, tipo Arial, tamaño 0.9em , interlineado 1.4
 2. Los elementos `<h2>` de `.articulo` se muestran en color `#CC6600`, con un tamaño de letra de 1.6em , un interlineado de 1.2 y un margen inferior de 0.3em .
 3. Los elementos del `#menu` deben mostrar un margen a su derecha de 1em y sus enlaces deben ser de color blanco y tamaño de letra 1.3em .
 4. El tamaño del texto de todos los contenidos de `#lateral` debe ser de 0.9em . La fecha de cada noticia debe ocupar el espacio de toda su línea y mostrarse en color gris claro `#999`. El elemento `<h3>` de `#noticias` debe mostrarse de color `#003366`.
 5. El texto del elemento `#publicidad` es de color gris oscuro `#555` y todos los enlaces de color `#CC6600`.
 6. Los enlaces contenidos dentro de `.articulo` son de color `#CC6600` y todos los párrafos muestran un margen superior e inferior de 0.3em .
 7. Añadir las reglas necesarias para que el contenido de `#secundario` se vea como en la imagen que se muestra.
 8. Añadir las reglas necesarias para que el contenido de `#pie` se vea como en la imagen que se muestra.

Logotipo

Buscar

Lorem Ipsum Dolor Sit Amet

Noticias

dd/mm/aaaa
[Lorem ipsum dolor sit amet](#)
dd/mm/aaaa
[Consectetuer adipiscing elit](#)
dd/mm/aaaa
[Donec molestie nunc eu sapien](#)
dd/mm/aaaa
[Maecenas aliquam dolor eget metus](#)
dd/mm/aaaa
[Fusce tristique lorem id metus](#)
[Enlaces relacionados](#)

[Proin placerat](#)
[Nulla in felis](#)
[Nam luctus](#)

Publicidad

Etiam fermentum, nisl tincidunt blandit interdum, massa velit posuere dolor, sed euismod sem odio at mi.
Duis porta placerat arcu. Nullam felis pede, commodo vel, suscipit a, molestie vel, felis. Maecenas mattis est vel est.

[Seguir leyendo...](#)

Phasellus blandit

Praesent sodales imperdiet augue. Mauris lorem felis, semper eu, tincidunt eu, sollicitudin eget, sem. Nulla facilisi. Morbi ut enim ut enim ultricies dapibus.

[Seguir leyendo...](#)

Lorem ipsum dolor sit amet, consectetuer adipiscing elit



Nullam est lacus, suscipit ut, dapibus quis, condimentum ac, risus. Vivamus vestibulum, ipsum sollicitudin faucibus pharetra, dolor metus fringilla dui, vel aliquet pede diam tempor tortor. Vestibulum pulvinar urna et quam. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Nullam vel turpis vitae dui imperdiet laoreet. Quisque eget ipsum.

Donec commodo, turpis vel venenatis sollicitudin, quam ante convallis justo, sed eleifend justo lectus quis sapien. Ut consequat libero eget est.

[Seguir leyendo...](#)

Vivamus lobortis turpis ac ante fringilla faucibus



Quisque eget ipsum. Donec commodo, turpis vel venenatis sollicitudin, quam ante convallis justo, sed eleifend justo lectus quis sapien. Ut consequat libero eget est.

Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Nullam dictum hendrerit neque. Mauris id ligula non elit mattis semper. Fusce arcu ipsum, tempus eget, tincidunt at, Imperdiet in, mi.

Sed fermentum cursus dolor. Aenean a diam. Phasellus feugiat. Donec tempor dignissim sem.

[Seguir leyendo...](#)

[Nulla vel turpis](#)

Donec commodo, turpis vel venenatis sollicitudin, quam ante convallis justo, sed eleifend justo lectus quis sapien. Ut consequat libero eget est.

[Seguir leyendo...](#)

Figura 15.10. Página con propiedades tipográficas

15.8. Ejercicio 8

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

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



Figura 15.11. Enlaces con estilos aplicados mediante CSS

15.9. Ejercicio 9

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

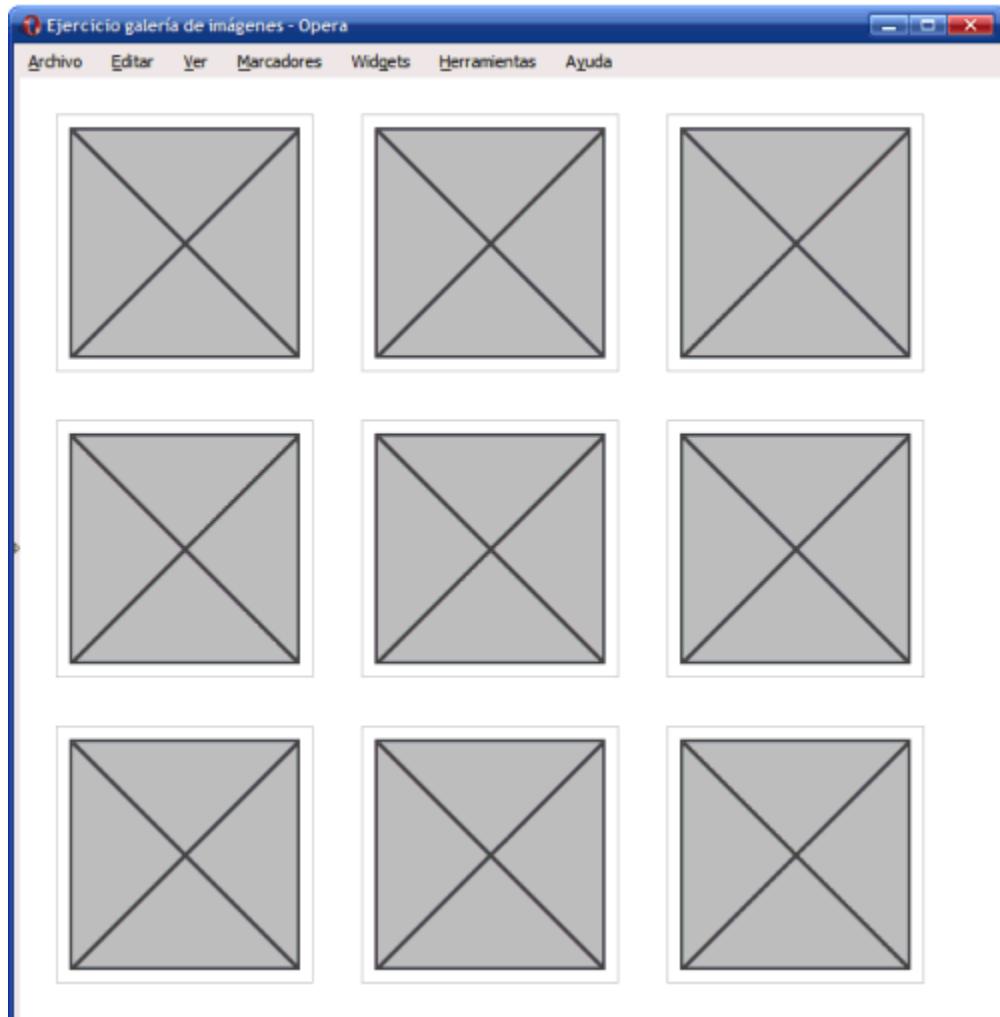


Figura 15.12. Galería de imágenes construida con CSS

15.10. Ejercicio 10

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

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

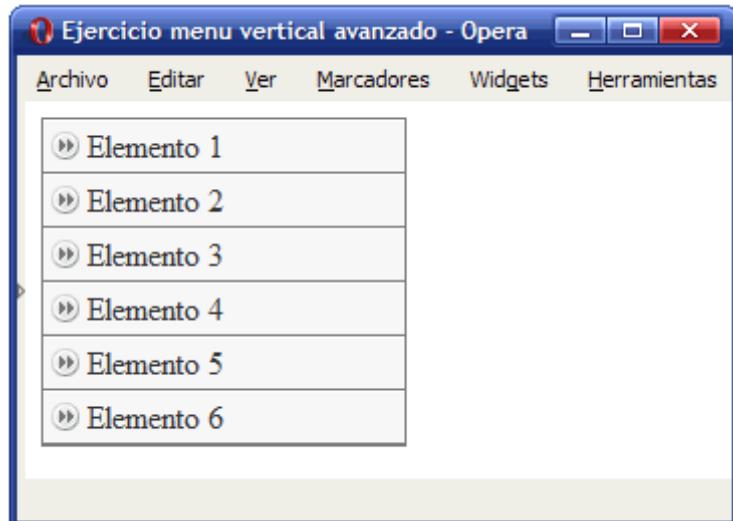


Figura 15.13. Menú vertical con imagen de fondo

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

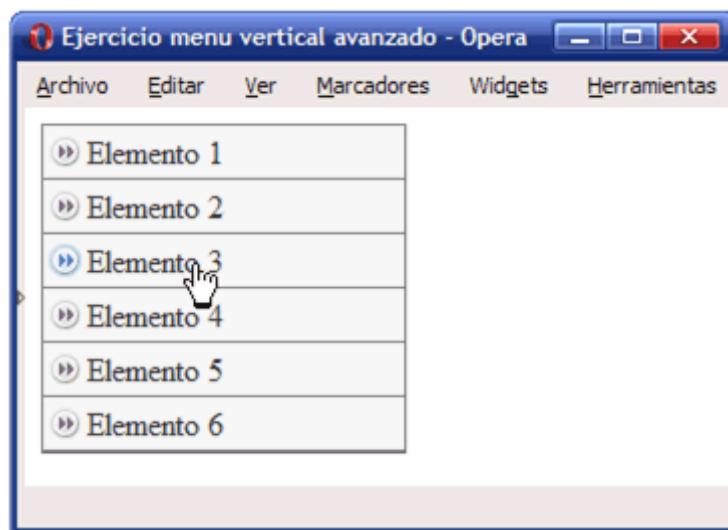


Figura 15.14. Menú vertical con imagen de fondo alternativa

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

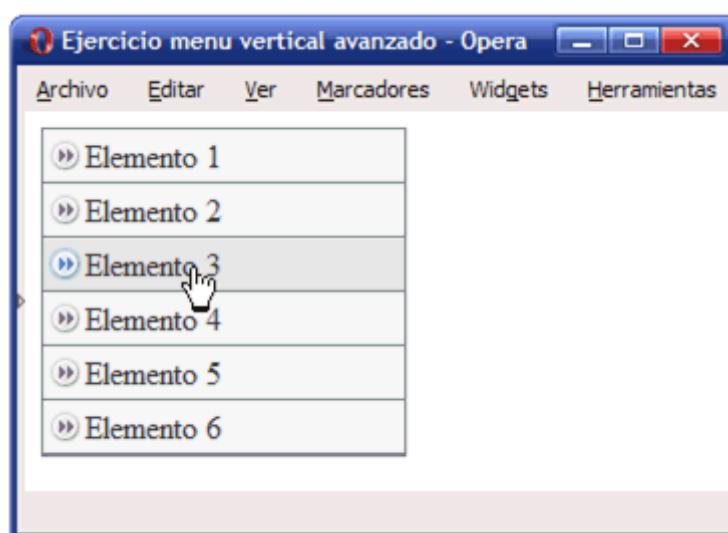


Figura 15.15. Menú vertical con imagen de fondo y color alternativos

4. El comportamiento anterior se debe producir cuando el usuario pasa el ratón por encima de cualquier zona del elemento del menú, no solo cuando se pasa el ratón por encima del texto del elemento:

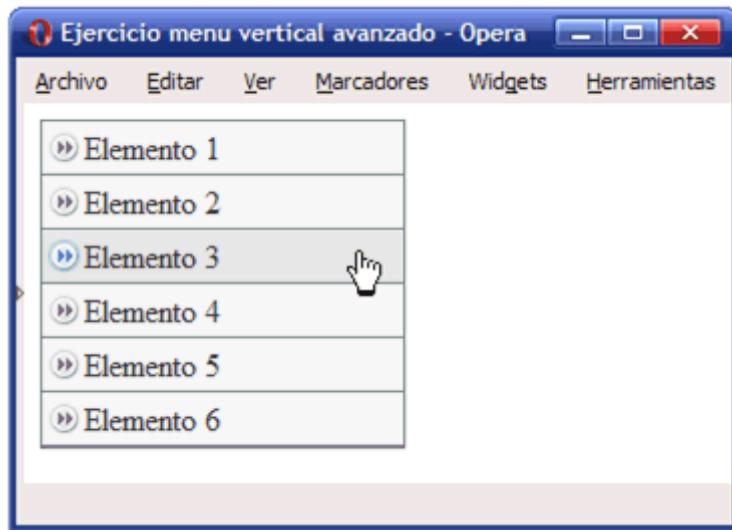


Figura 15.16. Aspecto final del menú vertical avanzado creado con CSS

15.11. Ejercicio 11

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

Tabla
original:

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

Figura 15.17. Aspecto original de la tabla

Tabla final:

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

Figura 15.18. Aspecto definitivo de la tabla después de aplicar estilos CSS

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

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

Figura 15.19. Tabla con texto alineado y bordes

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

The screenshot shows a table with five rows and six columns. The columns are labeled: Cambio, Compra, Venta, Máximo, and Mínimo. The rows contain currency symbols and exchange rates. The first row has a light gray background. The second row has a white background with a green dollar sign icon in the 'Cambio' column. The third row has a white background with a blue pound sign icon in the 'Cambio' column. The fourth row has a white background with a green yen symbol icon in the 'Cambio' column.

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

Figura 15.20. Tabla con colores e imágenes de fondo

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

The screenshot shows the same table as Figure 15.20, but with alternating colors for each row. The first row has a light gray background. The second row has a yellow background. The third row has a light gray background. The fourth row has a yellow background. The fifth row has a light gray background.

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

Figura 15.21. Tabla con colores de fila alternos

15.12. Ejercicio 12

A partir del código HTML proporcionado:

- 1) Aplicar las reglas CSS necesarias para que el formulario muestre el siguiente aspecto:

Formulario de alta

| | |
|-----------------------------|-----------------------------|
| Nombre y apellidos * | Dirección * |
| <input type="text"/> | <input type="text"/> |
| Nombre | Calle, número, piso, puerta |
| <input type="text"/> | <input type="text"/> |
| Primer apellido | Código postal |
| <input type="text"/> | <input type="text"/> |
| Segundo apellido | Municipio |
| <input type="text"/> | <input type="text"/> |
| Provincia | País |
| <hr/> | |
| Email | Teléfono * |
| <input type="text"/> | <input type="text"/> |
| Fijo | Móvil |
| Darme de alta → | |

Figura 15.22. Formulario estructurado a dos columnas

2) Cuando el usuario pasa el ratón por encima de cada grupo de elementos de formulario (es decir, por encima de cada ``) se debe modificar su color de fondo (sugerencia: color amarillo claro #FF9). Además, cuando el usuario se posiciona en un cuadro de texto, se debe modificar su borde para resaltar el campo que está activo cada momento (sugerencia: color amarillo #E6B700):

Formulario de alta

Nombre y apellidos *

*

Nombre

Primer apellido

Segundo apellido

Figura 15.23. Mejoras en los campos de formulario

3) Utilizando el menor número de reglas CSS, cambiar el aspecto del formulario para que se muestre como la siguiente imagen:

Formulario de alta

Nombre y apellidos *

Nombre
 Primer apellido
 Segundo apellido

Dirección *

Calle, número, piso, puerta
 Código postal Municipio
 Provincia País

Email

Teléfono *

Fijo Móvil

Darme de alta →

Figura 15.24. Formulario estructurado a una columna

- 4) Cuando el usuario pasa el ratón por encima de un grupo de elementos de formulario (es decir, por encima de cada ``) se debe mostrar el mensaje de ayuda asociado. Añadir las reglas CSS necesarias para que el formulario tenga el aspecto definitivo mostrado en la siguiente imagen:

Formulario de alta

Nombre y apellidos *

Nombre

 Primer apellido

 Segundo apellido

Dirección *

Calle, número, piso, puerta
 Calle, número, piso, puerta

Código postal

Municipio

Provincia País

El código postal es imprescindible para poder recibir los pedidos

Email

Teléfono *

Fijo Móvil

Darme de alta →

Figura 15.25. Aspecto final del formulario

Código HTML:

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

<html xmlns="http://www.w3.org/1999/xhtml" lang="es" xml:lang="es">
<head>
<title>Ejercicio 12 - Formulario de alta</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />

<style type="text/css">
</style>
</head>

<body>
<div id="contenedor">

<h2>Formulario de alta</h2>

<form method="post" action="#">
<ul>
<li>
```

```
<label class="titulo" for="nombre">Nombre y apellidos <span>
  <span>*</span></label>
  <div>
    <span>
      <input id="nombre" name="nombre" value="" />
      <label for="nombre">Nombre</label>
    </span>

    <span>
      <input id="apellido1" name="apellido1" value="" />
      <label for="apellido1">Primer apellido</label>
    </span>

    <span>
      <input id="apellido2" name="apellido2" value="" />
      <label for="apellido2">Segundo apellido</label>
    </span>
  </div>

  <p class="ayuda">No te olvides de escribir también tu segundo apellido</p>
</li>

<li>
  <label class="titulo" for="direccion">Dirección <span>
    <span>*</span></label>

  <div>
    <span>
      <input id="direccion" name="direccion" value="" />
      <label for="direccion">Calle, número, piso, puerta</label>
    </span>

    <span>
      <input id="codigopostal" name="codigopostal" value="" />
      <label for="codigopostal">Código postal</label>
    </span>

    <span>
      <input id="municipio" name="municipio" value="" />
      <label for="municipio">Municipio</label>
    </span>

    <span>
      <select id="provincia" name="provincia">
        <option value=""></option>
        <option value="provincial">Provincia 1</option>
        <option value="provincia2">Provincia 2</option>
        <option value="provincia3">Provincia 3</option>
      </select>
      <label for="provincia">Provincia</label>
    </span>

    <span>
      <select id="pais" name="pais">
        <option value=""></option>
```

```
<option value="pais1">País 1</option>
<option value="pais2">País 2</option>
<option value="pais3">País 3</option>
</select>
<label for="pais">País</label>
</span>
</div>

<p class="ayuda">El código postal es imprescindible para poder recibir los pedidos</p>
</li>

<li>
    <label class="titulo" for="email">Email</label>

    <div>
        <span>
            <input id="email" name="email" value="" />
        </span>
    </div>

    <p class="ayuda">Asegúrate de que sea válido</p>
</li>

<li>
    <label class="titulo" for="telefonofijo">Teléfono <span
    class="requerido">*</span></label>

    <div>
        <span>
            <input id="telefonofijo" name="telefonofijo" value="" />
            <label for="telefonofijo">Fijo</label>
        </span>

        <span>
            <input id="telefonomovil" name="telefonomovil" value="" />
            <label for="telefonomovil">Móvil</label>
        </span>
    </div>

    <p class="ayuda">Sin prefijo de país y sin espacios en blanco</p>
</li>

<li>
    <input id="alta" type="submit" value="Darme de alta &rarr;" />
</li>

</ul>
</form>

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

15.13. Ejercicio 13

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



Figura 15.26. Aspecto final que debe mostrar la página HTML proporcionada

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

- Añadir los estilos básicos de la página (tipo de letra Verdana, color de letra #192666, imagen de fondo llamada `fondo.gif`, color de fondo #F2F5FE).
- Definir la estructura básica de la página: anchura fija de 770 píxel, centrada en la ventana del navegador, cabecera y pie, columna central de contenidos de anchura 530 píxel y columna secundaria de contenidos de 200 píxel de anchura.
- La cabecera tiene una altura de 100 píxel y una imagen de fondo llamada `cabecera.jpg`.
- Los elementos del menú de navegación tienen un color de fondo #253575, un color de letra #B5C4E3. Cuando el ratón pasa por encima de cada elemento, su color de fondo cambia a #31479B. Los elementos seleccionados se muestran con un color de fondo blanco y un color de letra #FF9000:

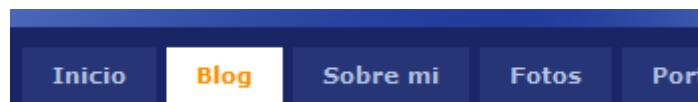


Figura 15.27. Imagen detallada del aspecto que muestran los elementos del menú de navegación

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

The screenshot shows a blog post titled "Lorem ipsum dolor sit amet". Below the title is a timestamp "dd-mm-aaaa 00:00" and author information "diseño Nombre Apellido". There is also a link "Añadir comentario". The main content area contains placeholder text: "Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Aliquam pellentesque enim blandit enim bibendum blandit. Integer eu leo ac est aliquet imperdiet. Quisque nec justo id augue posuere malesuada. Nullam ac metus. Cras non leo ut est placerat condimentum." At the bottom, there is a link "Sequir leyendo...".

Figura 15.28. Aspecto de un artículo de la sección principal de contenidos

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

The screenshot shows the sidebar content sections. The first section is titled "Sobre mí" and contains fields for "Nombre apellidos", "Edad", and "Ciudad de residencia", along with a link "Mi perfil público". The second section is titled "Categorías" and lists five categories: "Categoría 1", "Categoría 2", "Categoría 3", "Categoría 4", and "Categoría 5".

Figura 15.29. Aspecto de las secciones de la columna secundaria de contenidos

Capítulo 16. Ejercicios resueltos

Este último capítulo muestra las soluciones completas de todos los ejercicios planteados en el libro. Además de visualizarlas, todas las soluciones se pueden descargar.

16.1. Solución ejercicio 1

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

/* Todos Los párrafos de La pagina */
p { color: #555; }

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

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

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

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

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

16.2. Solución ejercicio 2

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/
xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>Ejercicio de selectores</title>
<style type="text/css">
/* No borrar la siguiente regla CSS porque es necesaria para ver los bordes de la tabla
*/
table, tr, th, td {border:1px solid #000; border-collapse:collapse; padding:5px;}

h1#titulo { color: teal; }

strong { color: red; }
span.destacado { color: orange; }

h2#subtitulo { color: blue; }

span.especial { color: purple; }

a { color: red; }
a em { color: blue; }
```

```
div#adicional p { color: olive; }
div#adicional span#especial { color: fuchsia; }
div#adicional a { color: green; }

caption { color: blue; }
td { color: green; }
td strong { color: orange; }
th { color: red; }
th.especial { color: orange; }
</style>
</head>

<body>
<h1 id="titulo">Lorem ipsum dolor sit amet</h1>

<p>Nulla pretium. Sed tempus nunc vitae neque. <strong>Suspendisse gravida</strong>, metus a scelerisque sollicitudin, lacus velit ultricies nisl, nonummy tempus neque diam quis felis. <span class="destacado">Etiam sagittis tortor</span> sed arcu sagittis tristique.</p>

<h2 id="subtitulo">Aliquam tincidunt, sem eget volutpat porta</h2>

<p>Vivamus velit dui, placerat vel, feugiat in, ornare et, urna. <a href="#">Aenean turpis metus, <em>aliquam non</em>, tristique in</a>, pretium varius, sapien. Proin vitae nisi. Suspendisse <span class="especial">porttitor purus ac elit</span>. Suspendisse eleifend odio at dui. In in elit sed metus pretium elementum.</p>

<table summary="Descripción de la tabla y su contenido">
<caption>Título de la tabla</caption>
<thead>
<tr>
<th scope="col"></th>
<th scope="col" class="especial">Título columna 1</th>
<th scope="col" class="especial">Título columna 2</th>
</tr>
</thead>

<tfoot>
<tr>
<th scope="col"></th>
<th scope="col">Título columna 1</th>
<th scope="col">Título columna 2</th>
</tr>
</tfoot>

<tbody>
<tr>
<th scope="row" class="especial">Título fila 1</th>
<td>Donec purus ipsum</td>
<td>Curabitur <em>blandit</em></td>
</tr>
<tr>
<th scope="row">Título fila 2</th>
<td>Donec <strong>purus ipsum</strong></td>
```

```

        <td>Curabitur blandit</td>
    </tr>
</tbody>
</table>

<div id="adicional">
<p>Donec purus ipsum, posuere id, venenatis at, <span>placerat ac, lorem</span>.
Curabitur blandit, eros sed gravida aliquet, risus justo
porta lorem, ut mollis lectus tortor in orci. Pellentesque nec augue.</p>

<p>Fusce nec felis eu diam pretium adipiscing. <span id="especial">Nunc elit elit,
vehicula vulputate</span>, venenatis in,
posuere id, lorem. Etiam sagittis, tellus in ultrices accumsan, diam nisi feugiat ante,
eu congue magna mi non nisl.</p>

<p>Vivamus ultrices aliquet augue. <a href="#">Donec arcu pede, pretium vitae</a>,
rutrum aliquet, tincidunt blandit, pede.
Aliquam in nisi. Suspendisse volutpat. Nulla facilisi. Ut ullamcorper nisi quis mi.</p>
</div>

</body>
</html>

```

16.3. Solución ejercicio 3

```

/* === IMPORTANTE =====
No modificar estos estilos, ya que son imprescindibles para
que la página se vea correctamente.
===== */

/*-- Básico -----*/
ul, ul li { margin: 0; padding: 0; list-style: none; }
h1, h2, h3, p, form { margin: 0; padding: 0; }
.clear { clear: both; }
img { border: none; }

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

#cabeza, #menu, #lateral, #contenido, #contenido #principal, #contenido #secundario,
#pie {
    border: 2px solid #777;
}

#cabeza { clear: both; }
#menu { clear: both; }
#lateral { float: left; width: 20%; }
#contenido { float: right; width: 78%; }
#contenido #principal { float: left; width: 78%; }
#contenido #secundario { float: right; width: 20%; }

```

```
#pie { clear: both; }

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

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

/*-- Sección Principal -----
#contenido #principal .articulo img { width: 100px; float: left; }

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

/* === IMPORTANTE =====
   A partir de aquí, se pueden añadir todos los estilos propios que
   sean necesarios.
===== */



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

#lateral {
    padding: 0;
}

#cabeza {
    padding: 1em;
}

#menu {
    margin-bottom: .5em;
}

#contenido {
    width: 77%;
    padding: 0;
}

#contenido #principal {
    width: 73%;
}
```

```

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

#contenido #principal .articulo {
    margin-bottom: 1em;
}

#contenido #principal .articulo img {
    margin: .5em;
}

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

```

16.4. Solución ejercicio 4

```

/* === IMPORTANTE =====
   No modificar estos estilos, ya que son imprescindibles para
   que la página se vea correctamente.
===== */

/*-- Básico -----*/
ul, ul li { margin: 0; padding: 0; list-style: none; }
h1, h2, h3, p, form { margin: 0; padding: 0; }
.clear { clear: both; }
img { border: none; }

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

#cabeza, #menu, #lateral, #contenido, #contenido #principal, #contenido #secundario,
#pie {
    border: 2px solid #777;
}

#cabeza { clear: both; }
#menu { clear: both; }
#lateral { float: left; width: 20%; }
#contenido { float: right; width: 78%; }
#contenido #principal { float: left; width: 78%; }
#contenido #secundario { float: right; width: 20%; }
#pie { clear: both; }

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

```

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

/*-- Sección Principal -----*/
#contenido #principal .articulo img { width: 100px; float: left; }

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

/* === IMPORTANTE =====
   A partir de aquí, se pueden añadir todos los estilos propios que
   sean necesarios.
===== */



#cabecera,
#menu,
#lateral,
#lateral #noticias,
#lateral #publicidad,
#contenido,
#contenido #principal,
#contenido #secundario,
#pie {
    padding: .5em;
    border: none;
}

#lateral {
    padding: 0;
}

#cabecera {
    padding: 1em;
}

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

#contenido {
    width: 77%;
    padding: 0;
}

#contenido #principal {
    width: 73%;
}

#contenido #secundario {
    border: 1px solid #C60;
}
```

```

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

#contenido #principal .articulo {
    margin-bottom: 1em;
}

#contenido #principal .articulo img {
    margin: .5em;
}

#lateral #noticias {
    border: 1px solid #C5C5C5;
}

#lateral #publicidad {
    margin-top: 1em;
    border: 1px dashed #C60;
}

```

16.5. Solución ejercicio 5

```

/* === IMPORTANTE =====
   No modificar estos estilos, ya que son imprescindibles para
   que la página se vea correctamente.
===== */

/*-- Básico -----*/
ul, ul li { margin: 0; padding: 0; list-style: none; }
h1, h2, h3, p, form { margin: 0; padding: 0; }
.clear { clear: both; }
img { border: none; }

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

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

#cabecera { clear: both; }
#menu { clear: both; }
#lateral { float: left; width: 20%; }
#contenido { float: right; width: 78%; }
#contenido #principal { float: left; width: 78%; }

```

```
#contenido #secundario { float: right; width: 20%; }
#pie { clear: both; }

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

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

/*-- Sección Principal -----*/
#contenido #principal .articulo img { width: 100px; float: left; }

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

/* === IMPORTANTE =====
   A partir de aquí, se pueden añadir todos los estilos propios que
   sean necesarios.
===== */
```



```
#cabeza,
#menu,
#lateral,
#lateral #noticias,
#lateral #publicidad,
#contenido,
#contenido #principal,
#contenido #secundario,
#pie {
    padding: .5em;
    border: none;
}

#lateral {
    padding: 0;
}

#cabeza {
    padding: 1em;
}

#cabeza h1 {
    background: url(..//comun/imagenes/logo.gif) no-repeat -5px -10px;
    width: 180px;
}

#cabeza h1 span {
    visibility: hidden;
}

#menu {
    margin-bottom: .5em;
```

```
border-bottom: 1px solid #004C99;
background: url(..../comun/imagenes/fondo_menu.gif) repeat-x;
}

#contenido {
width: 77%;
padding: 0;
}

#contenido #principal {
width: 73%;
}

#contenido #secundario {
border: 1px solid #C60;
}

#contenido #secundario h2 {
background: #DB905C;
padding: .2em;
}

#pie {
padding: .5em 0;
margin-top: 1em;
border-top: 1px solid #C5C5C5;
border-bottom: 1px solid #C5C5C5;
background: #F8F8F8;
}

#contenido #principal .articulo {
margin-bottom: 1em;
}

#contenido #principal .articulo img {
margin: .5em;
}

#lateral #noticias {
border: 1px solid #C5C5C5;
background: #F8F8F8;
}

#lateral #publicidad {
margin-top: 1em;
border: 1px dashed #C60;
background: #FFF6CD;
}
```

16.6. Solución ejercicio 6

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

```

<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
<title>Ejercicio posicionamiento float</title>
<style type="text/css">
div#paginacion {
    border: 1px solid #CCC;
    background-color: #E0E0E0;
    padding: .5em;
}
.derecha {
    float: right;
}
.izquierda {
    float: left;
}
div.clear {
    clear: both;
}
</style>
</head>

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

```

16.7. Solución ejercicio 7

```

/* === IMPORTANTE =====
   No modificar estos estilos, ya que son imprescindibles para
   que la página se vea correctamente.
===== */

/*-- Básico -----*/
ul, ul li { margin: 0; padding: 0; list-style: none; }
h1, h2, h3, p, form { margin: 0; padding: 0; }
.clear { clear: both; }
img { border: none; }

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

#cabeza, #menu, #lateral, #contenido, #contenido #principal, #contenido #secundario,
#pie {
    border: 2px solid #777;
}

```

```
#cabecera { clear: both; }
#menu { clear: both; }
#lateral { float: left; width: 20%; }
#contenido { float: right; width: 78%; }
#contenido #principal { float: left; width: 78%; }
#contenido #secundario { float: right; width: 20%; }
#pie { clear: both; }

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

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

/*-- Sección Principal -----*/
#contenido #principal .articulo img { width: 100px; float: left; }

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

/* === IMPORTANTE =====
   A partir de aquí, se pueden añadir todos los estilos propios que
   sean necesarios.
===== */
```

```
body {
    font: .9em/1.4 Arial, Helvetica, sans-serif;
    color: #000;
}

#cabeza,
#menu,
#lateral,
#lateral #noticias,
#lateral #publicidad,
#contenido,
#contenido #principal,
#contenido #secundario,
#pie {
    padding: .5em;
    border: none;
}

#lateral {
    padding: 0;
    font-size: .9em;
}

#cabeza {
    padding: 1em;
}

#cabeza h1 {
```

```
background: url(..../comun/imagenes/logo.gif) no-repeat -5px -5px;
width: 180px;
}

#cabeza h1 span {
  visibility: hidden;
}

#menu {
  margin-bottom: .5em;
  border-bottom: 1px solid #004C99;
  background: url(..../comun/imagenes/fondo_menu.gif) repeat-x;
}

#menu li {
  margin-right: 1em; font-
  size: 1.3em;
}

#menu li a {
  color: #FFF;
}

#contenido {
  width: 77%;
  padding: 0;
}

#contenido #principal {
  width: 73%;
}

#contenido #secundario {
  border: 1px solid #C60;
}

#contenido #secundario h2 {
  background: #DB905C;
  padding: .2em;
  font-size: 1em;
  color: #FFF;
}

#contenido #secundario p {
  margin: .5em 0;
}

#pie {
  padding: .5em 0; margin-
  top: 1em;
  border-top: 1px solid #C5C5C5; border-
  bottom: 1px solid #C5C5C5; background:
  #F8F8F8;
  color: #555;
  font-size: .75em;
```

```
}

#contenido #principal .articulo {
    margin-bottom: 1em;
}

#contenido #principal .articulo p {
    margin: .3em 0;
}

#contenido #principal .articulo a {
    color: #C60;
}

#contenido #principal .articulo h2 {
    color: #C60;
    font-size: 1.6em; line-
    height: 1.2; margin-bottom:
    .3em;
}

#contenido #principal .articulo img {
    margin: .5em;
}

#lateral #noticias {
    border: 1px solid #C5C5C5;
    background: #F8F8F8;
}

#lateral #noticias h3 {
    color: #036;
}

#lateral #noticias span.fecha {
    display: block;
    color: #999;
}

#lateral #publicidad {
    margin-top: 1em;
    border: 1px dashed #C60;
    background: #FFF6CD;
    color: #555;
}

#lateral #publicidad a {
    color: #C60;
}
```

16.8. Solución ejercicio 8

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

```

<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
<title>Ejercicio de pseudo clases de enlaces</title>
<style type="text/css">
a {
  margin: 1em 0;
  color: #CC0000;
  float: left;
  clear: left;
  padding: 2px;
}
a:hover {
  text-decoration: none;
  background-color: #CC0000;
  color: #FFF;
}
a:visited {
  color: #CCC;
}
</style>
</head>

<body>
<a href="#">Enlace número 1</a>

<a href="#">Enlace número 2</a>

<a href="#">Enlace número 3</a>

<a href="#">Enlace número 4</a>

<a href="#">Enlace número 5</a>
</body>
</html>

```

16.9. Solución ejercicio 9

```

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
<title>Ejercicio galería de imágenes</title>
<style type="text/css">
#galeria {
  /* En el navegador Internet Explorer versión 6 y anteriores no funciona la propiedad
  "max-width" */
  max-width: 650px;
}
#galeria img {
  float: left;
  margin: 1em;
  padding: .5em;
  border: 1px solid #CCC;
}

```

```
</style>
</head>

<body>
<div id="galeria">










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

16.10. Solución ejercicio 10

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
<title>Ejercicio menu vertical avanzado</title>
<style type="text/css">
ul.menu {
    width: 180px;
    list-style: none;
    margin: 0;
    padding: 0;
    border: 1px solid #7C7C7C;
}
ul.menu li {
    border-bottom: 1px solid #7C7C7C;
    border-top: 1px solid #FFF;
    background: #F4F4F4;
}
ul.menu li a {
    padding: .2em 0 .2em 1.5em;
    display: block;
    text-decoration: none;
    color: #333;
    background: #F4F4F4 url("../comun/imagenes/flecha_inactiva.png") no-repeat 3px;
}
ul.menu li a:hover, ul.menu li a:active {
    background: #E4E4E4 url("../comun/imagenes/flecha_activa.png") no-repeat 3px;
}
</style>
</head>

<body>
<ul class="menu">
    <li><a href="#" title="Enlace genérico">Elemento 1</a></li>
```

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

16.11. Solución ejercicio 11

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/
xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
<title>Ejercicio formatear tabla</title>
<style type="text/css">
table {
    font: .9em Arial, Helvetica, sans-serif;
    border: 1px solid #333;
    border-collapse: collapse; text-
    align: center;
}
table th {
    background: #F5F5F5 url(../../../comun/imagenes/fondo_gris.gif) repeat-x;
    padding: 0 .3em;
    text-align: left;
}
table thead th {
    text-align: center;
}
table th.euro {
    background: #E6F3FF url(../../../comun/imagenes/euro.png) no-repeat left center;
    padding: 0 .3em 0 1.2em;
}
table th.dolar {
    background: #E6F3FF url(../../../comun/imagenes/dolar.png) no-repeat left center;
    padding: 0 .3em 0 1.2em;
}
table th.libra {
    background: #E6F3FF url(../../../comun/imagenes/libra.png) no-repeat left center;
    padding: 0 .3em 0 1.2em;
}
table th.yen {
    background: #E6F3FF url(../../../comun/imagenes/yen.png) no-repeat left center;
    padding: 0 .3em 0 1.2em;
}
table th, table, td {
    border: 1px solid #333;
    line-height: 2em;
}
.par {
    background-color:#FFFFCC;
}
```

```
table tr:hover {
    background: #FFFF66 !important;
}
</style>
</head>

<body>
<table summary="Tipos de cambio">
    <thead>
        <tr class="cabecera">
            <th scope="col">Cambio</th>
            <th scope="col">Compra</th>
            <th scope="col">Venta</th>
            <th scope="col">M&aacute;ximo</th>
            <th scope="col">M&iacute;nimo</th>
        </tr>
    </thead>
    <tbody>
        <tr class="par">
            <th scope="row" class="euro">Euro/Dolar</th>
            <td>1.2524</td>
            <td>1.2527</td>
            <td>1.2539</td>
            <td>1.2488</td>
        </tr>
        <tr>
            <th scope="row" class="dolar">Dolar/Yen</th>
            <td>119.01</td>
            <td>119.05</td>
            <td>119.82</td>
            <td>119.82</td>
        </tr>
        <tr class="par">
            <th scope="row" class="libra">Libra/Dolar</th>
            <td>1.8606</td>
            <td>1.8611</td>
            <td>1.8651</td>
            <td>1.8522</td>
        </tr>
        <tr>
            <th scope="row" class="yen">Yen/Euro</th>
            <td>0.6711</td>
            <td>0.6705</td>
            <td>0.6676</td>
            <td>0.6713</td>
        </tr>
    </tbody>
</table>

</body>
</html>
```

16.12. Solución ejercicio 12

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

<html xmlns="http://www.w3.org/1999/xhtml" lang="es" xml:lang="es">
<head>
<title>Ejercicio 12 - Formulario de alta</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />

<style type="text/css">
/* Formatear el formulario a dos columnas */
body {
    font: 13px/1.6 Tahoma, sans-serif;
    background: #F5F5F5;
}

.izquierda {
    float: left;
    clear: left;
}

.derecha {
    float: right;
    clear: right;
}

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

#contenedor {
    background: #FFF;
    border: 1px solid silver;
    margin: 1em auto;
    padding: 1em;
    width: 768px;
}

span.requerido { font-
    size: 1.3em; font-
    weight: bold; color:
    #C00;
}

h2 {
    font: normal 2em arial, sans-serif;
    margin: 0;
}

ul li.botones {
    border-top: 2px solid #CCC;
    clear: both;
```

```
float: none;
padding: 1em 0;
margin-top: 1em;
text-align: right;
width: 100%;
}

ul li.botones input {
  font-size: 1.3em;
}

ul li {
  margin: 0.5em 0;
  padding: 0.5em;
  width: 46%;
}

ul li label.titulo { font-
  weight: bold;
}

ul li div span {
  float: left;
  padding: 0.3em 0;
}

ul li div span.completo {
  width: 100%;
}

ul li div span.mitad {
  width: 50%;
}

ul li div span.tercio {
  width: 33%;
}

ul li div span.dostercios {
  width: 66%;
}

ul li div span label {
  display: block;
  font: normal 0.8em arial, sans-serif;
  color: #333;
}

ul li p.ayuda {
  display: none;
}

ul li input {
  padding: 0.2em;
}
```

```
input#apellido1, input#apellido2, input#direccion, input#email {  
    width: 260px;  
}  
  
input#codigopostal {  
    width: 80px;  
}  
  
select#provincia {  
    width: 90px;  
}  
  
select#pais {  
    width: 150px;  
}  
  
input#telefonofijo, input#telefonomovil {  
    width: 135px;  
}  
  
/* Cambiar el color en el :hover y resaltar Los campos en el :focus */  
ul li:hover {  
    background-color: #FF9;  
}  
  
ul li.botones:hover {  
    background-color: transparent;  
}  
  
ul li input:focus {  
    border: 2px solid #E6B700;  
}  
  
/* Formatear el formulario a una columna */  
ul li.izquierda, ul li.derecha {  
    float: none;  
    width: auto;  
}  
  
ul li {  
    overflow: hidden;  
}  
  
ul li label.titulo {  
    float: left;  
    width: 150px;  
}  
  
ul li div {  
    margin-left: 160px;  
}  
  
/* Aspecto final del formulario con Los mensajes de ayuda */  
h2 {
```

```
        margin-bottom: 0.3em;
    }

    ul li {
        border-top: 1px solid #CCC;
        margin: 0;
        padding: 1em;
    }

    ul li.botones {
        margin: 0;
    }

    ul li label.titulo {
        text-align: right;
        width: 100px;
    }

    ul li div {
        margin-left: 110px;
        overflow: hidden;
    }

    ul li {
        position: relative;
    }

    ul li:hover p.ayuda {
        display: block;
        margin: 0.3em;
        position: absolute;
        top: 0;
        right: 0;
        width: 150px;
    }
</style>
</head>

<body>
<div id="contenedor">

<h2>Formulario de alta</h2>

<form method="post" action="#">
<ul>
<li class="izquierda">
    <label class="titulo" for="nombre">Nombre y apellidos <span
    class="requerido">*</span></label>
    <div>
        <span class="completo">
            <input id="nombre" name="nombre" value="" />
            <label for="nombre">Nombre</label>
        </span>
    <span class="completo">
```

```
<input id="apellido1" name="apellido1" value="" />
<label for="apellido1">Primer apellido</label>
</span>

<span class="completo">
    <input id="apellido2" name="apellido2" value="" />
    <label for="apellido2">Segundo apellido</label>
</span>
</div>

<p class="ayuda">No te olvides de escribir también tu segundo apellido</p>
</li>

<li class="derecha">
    <label class="titulo" for="direccion">Dirección <span
    class="requerido">*</span></label>

    <div>
        <span class="completo">
            <input id="direccion" name="direccion" value="" />
            <label for="direccion">Calle, número, piso, puerta</label>
        </span>

        <span class="tercio">
            <input id="codigopostal" name="codigopostal" value="" />
            <label for="codigopostal">Código postal</label>
        </span>

        <span class="dostercios">
            <input id="municipio" name="municipio" value="" />
            <label for="municipio">Municipio</label>
        </span>

        <span class="tercio">
            <select id="provincia" name="provincia">
                <option value=""></option>
                <option value="provincia1">Provincia 1</option>
                <option value="provincia2">Provincia 2</option>
                <option value="provincia3">Provincia 3</option>
            </select>
            <label for="provincia">Provincia</label>
        </span>

        <span class="dostercios">
            <select id="pais" name="pais">
                <option value=""></option>
                <option value="pais1">País 1</option>
                <option value="pais2">País 2</option>
                <option value="pais3">País 3</option>
            </select>
            <label for="pais">País</label>
        </span>
    </div>

    <p class="ayuda">El código postal es imprescindible para poder recibir los pedidos</p>
```

```
</li>

<li class="izquierda">
    <label class="titulo" for="email">Email</label>

    <div>
        <span class="completo">
            <input id="email" name="email" value="" />
        </span>
    </div>

    <p class="ayuda">Asegúrate de que sea válido</p>
</li>

<li class="derecha">
    <label class="titulo" for="telefonofijo">Teléfono <span
class="requerido">*</span></label>

    <div>
        <span class="mitad">
            <input id="telefonofijo" name="telefonofijo" value="" />
            <label for="telefonofijo">Fijo</label>
        </span>

        <span class="mitad">
            <input id="telefonomovil" name="telefonomovil" value="" />
            <label for="telefonomovil">Móvil</label>
        </span>
    </div>

    <p class="ayuda">Sin prefijo de país y sin espacios en blanco</p>
</li>

<li class="botones">
    <input id="alta" type="submit" value="Darme de alta &rarr;" />
</li>

</ul>
</form>

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

16.13. Solución ejercicio 13

```
/* ----- Estilos básicos ----- */

body {
    margin: 0;
    padding: 0;
    background: #F2F5FE url("../imagenes/fondo.gif") 0 0 repeat-x;
    font: 70%/160% "verdana", sans-serif;
    color: #192666;
}
```

```
a {  
    color:#192666;  
}  
a:hover {  
    color:#4F6AD7;  
}  
  
p {  
    margin: 1em 0;  
    padding: 0;  
}  
  
.clear {  
    clear: both;  
}  
  
h1, h2, h3, h4, h5 {  
    margin: 1em 0;  
    padding:0;  
}  
  
h1 {  
    font-size: 260%;  
    font-family: "georgia", serif; font-  
    weight: normal;  
}  
  
h2 {  
    font-size:180%;  
    font-family: "georgia", serif; font-  
    weight: normal;  
}  
  
h3 {  
    font-size:120%; font-  
    weight:bold;  
}  
  
ul, ol {  
    margin: 1em 0 1em 2em;  
    padding:0;  
}  
  
/* ---- Layout ----- */  
  
#contenedor {  
    width:770px;  
    margin: 50px auto 0 auto;  
}  
  
#cabecera {  
    width: 770px;  
    position: relative;  
    height: 100px;
```

```
margin: 0;
padding: 0;
background: #233C9B url("../imagenes/cabecera.jpg") no-repeat;
color: #FFF;
}

#contenido {
    width: 760px;
    margin: 0 5px;
    background: #FFF;
}

#contenido #principal {
    float: left;
    width: 530px;
    margin-top: 15px;
    padding: 0 0 0 20px;
    background: #FFF;
}

#contenido #secundario {
    float: left;
    width: 200px;
    margin: 15px 0 0 0;
    padding: 0;
    background: #CEDBF9 url("../imagenes/fondo_columna.gif") no-repeat;
}

#pie {
    clear: both;
    height: 60px;
    margin-bottom: 50px;
    background: url("../imagenes/fondo_pie.jpg") no-repeat;
    color: #6685CC;
    position: relative;
}

/* ----- Cabecera ----- */
#cabeza #logo {
    position: absolute;
    top: 35px;
    left: 35px;
    margin: 0;
}

#cabeza #logo a {
    color: #FFF;
    display: block;
    line-height: 35px;
}

#cabeza #logo a:hover {
    color: #B5C4E3;
    text-decoration: underline;
}
```

```
#cabecera #buscador {  
    position: absolute;  
    top: 40px;  
    right: 20px;  
}  
  
#cabecera #buscador legend {  
    display: none;  
}  
  
#cabecera #buscador fieldset {  
    border: none;  
}  
  
/* ---- Menú ----- */  
#menu {  
    background: #192666;  
    margin: 0 5px;  
    padding: 10px 0 0 0;  
}  
  
#menu ul {  
    margin: 0 10px;  
    padding: 0;  
    list-style:none;  
}  
  
#menu ul li {  
    margin: 0 5px 0 0;  
    padding: 0;  
    float:left;  
}  
  
#menu ul li a {  
    display: block;  
    position: relative;  
    padding: 5px 15px;  
    border: 0;  
    background: #253575;  
    color: #B5C4E3;  
    font-weight: bold;  
    text-decoration: none;  
    cursor: pointer;  
}  
  
#menu ul li a:hover {  
    background: #31479B;  
}  
  
#menu ul li.seleccionado a {  
    background: #FFF;  
    color: #FF9000;  
}
```

```
/* ----- Contenidos ----- */
#contenido .articulo {
    clear: both;
    margin: 0;
    padding: 20px;
    background: url("../imagenes/fondo_articulo.jpg") no-repeat;
}

#contenido .articulo h2 {
    margin: 0 -20px;
    padding: 10px;
    background: #DEE5FD;
    color: #192666;
}

#contenido .info {
    margin: 10px 0;
    padding-bottom: 8px;
    border-bottom: 1px solid #DEE5FD;
    color:#6685CC;
}

#contenido .info a { color:#6685CC; }
#contenido .info a:hover { color:#FF9000; }

#contenido .info span.fecha, #contenido .info span.categoría, #contenido .info
span.autor, #contenido .info span.comentarios {
    padding-left:15px;
}

#contenido .info span.fecha { background: url("../imagenes/icono_fecha.gif") 0 50% no-
repeat; }
#contenido .info span.categoría { background: url("../imagenes/icono_categoria.gif") 0
50% no-repeat; margin-left: 8px; }
#contenido .info span.autor { background: url("../imagenes/icono_autor.gif") 0 50% no-
repeat; margin-left: 8px; }
#contenido .info span.comentarios { background: url("../imagenes/
icono_comentarios.gif") 0 50% no-repeat; margin-left: 8px; }

#contenido #secundario h3 {
    padding: 10px 0 10px 10px;
    margin-top: 20px;
    background: #A0B9F3;
    color: #192666;
}

#contenido #secundario #sobre mí img {
    float: left;
    margin: 4px 5px 0 0;
}

#contenido #secundario #sobre mí p {
```

```
#contenido #secundario div {  
    padding: 0 10px;  
}  
  
#contenido #secundario ul {  
    margin: 15px 0;  
    padding: 0;  
    list-style:none;  
}  
  
#contenido #secundario li {  
    margin: 0;  
    padding: 0;  
    border-bottom: 1px solid #E0E8FA;  
}  
  
#contenido #secundario li a {  
    display: block;  
    padding: 3px 0 3px 22px;  
    background: url("../imagenes/icono_elemento.gif") 8px no-repeat;  
    text-decoration: none;  
}  
  
#contenido #secundario li a:hover { background-  
    color: #E0E8FA;  
    color: #192666;  
}  
  
#contenido #secundario li.seleccionado a {  
    background: #E0E8FA url("../imagenes/icono_seleccionado.gif") 8px no-repeat;  
    font-weight: bold;  
}  
  
/* ----- Pie ----- */  
#pie p {  
    margin: 0;  
    padding: 0;  
    position: absolute;  
    top: 10px;  
    left: 40px;  
}
```