

## 7. TEXTO

### 7.1. 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.

<b>color</b>	Color del texto
<b>Valores</b>	<code>&lt;color&gt;</code>   <code>inherit</code>
<b>Se aplica a</b>	Todos los elementos
<b>Valor inicial</b>	Depende del navegador
<b>Descripción</b>	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.

<b>font-family</b>	Tipo de letra
<b>Valores</b>	(( <nombre_familia>   <familia_generica> ) (,<nombre_familia>   <familia_generica>)* )   inherit
<b>Se aplica a</b>	Todos los elementos
<b>Valor inicial</b>	Depende del navegador
<b>Descripción</b>	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`.

<b>font-size</b>	Tamaño de letra
<b>Valores</b>	<tamaño_absoluto>   <tamaño_relativo>   <medida>   <porcentaje>   inherit
<b>Se aplica a</b>	Todos los elementos
<b>Valor inicial</b>	medium
<b>Descripción</b>	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:

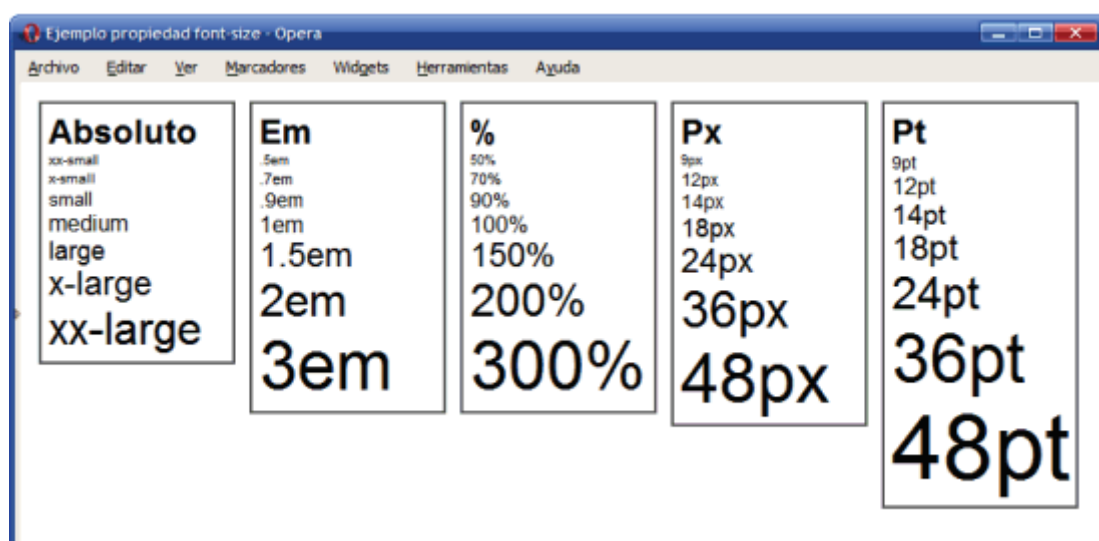


Figura 3.77. Comparación visual de las distintas unidades para indicar el tamaño del texto

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`.

<b>font-weight</b>	Anchura de la letra
<b>Valores</b>	<code>normal</code>   <code>bold</code>   <code>bolder</code>   <code>lighter</code>   <code>100</code>   <code>200</code>   <code>300</code>   <code>400</code>   <code>500</code>   <code>600</code>   <code>700</code>   <code>800</code>   <code>900</code>   <code>inherit</code>
<b>Se aplica a</b>	Todos los elementos
<b>Valor inicial</b>	<code>normal</code>
<b>Descripción</b>	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`:

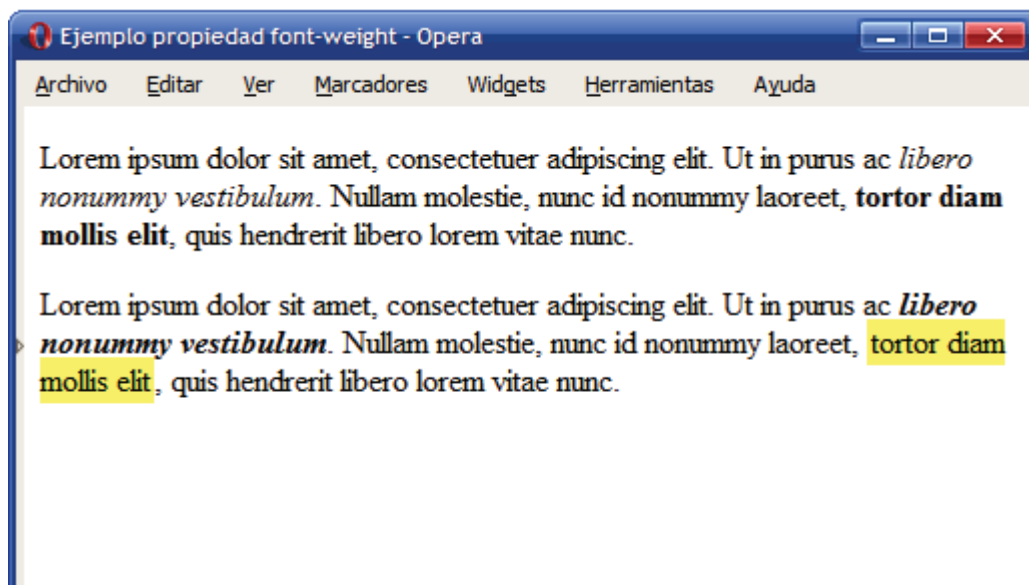


Figura 3.78. Ejemplo de propiedad `font-weight`

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

Las reglas CSS del ejemplo anterior se muestran a continuación:

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

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

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

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

<b>font-style</b>	Estilo de la letra
<b>Valores</b>	<code>normal</code>   <code>italic</code>   <code>oblique</code>   <code>inherit</code>
<b>Se aplica a</b>	Todos los elementos
<b>Valor inicial</b>	normal
<b>Descripción</b>	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 `<em>` y `<strong>`:

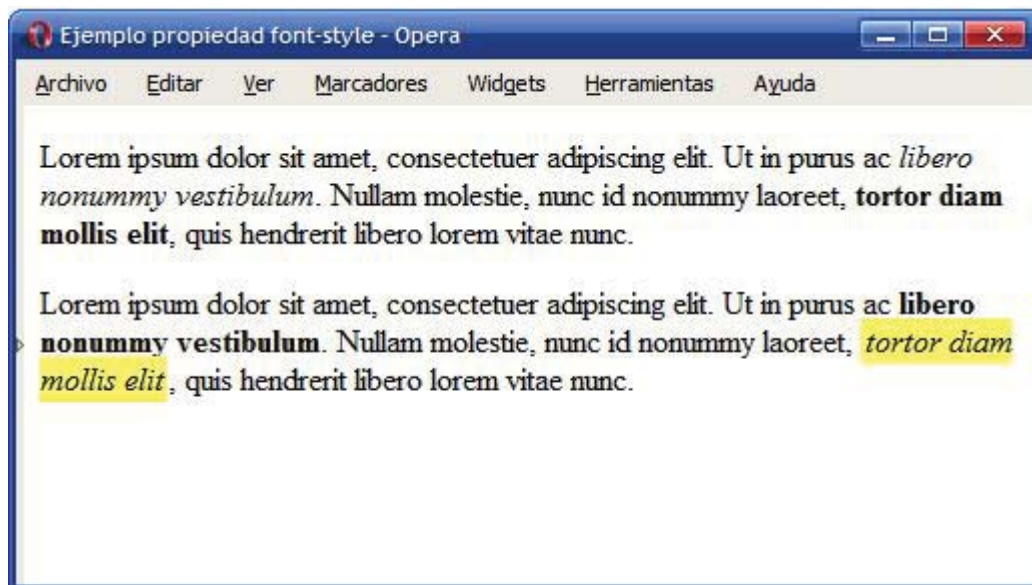


Figura 3.79. Ejemplo de propiedad font-style

Ahora, el texto del elemento `<em>` se muestra como un texto en cursiva y el texto del elemento `<strong>` se muestra como un texto en negrita 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`.

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



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

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

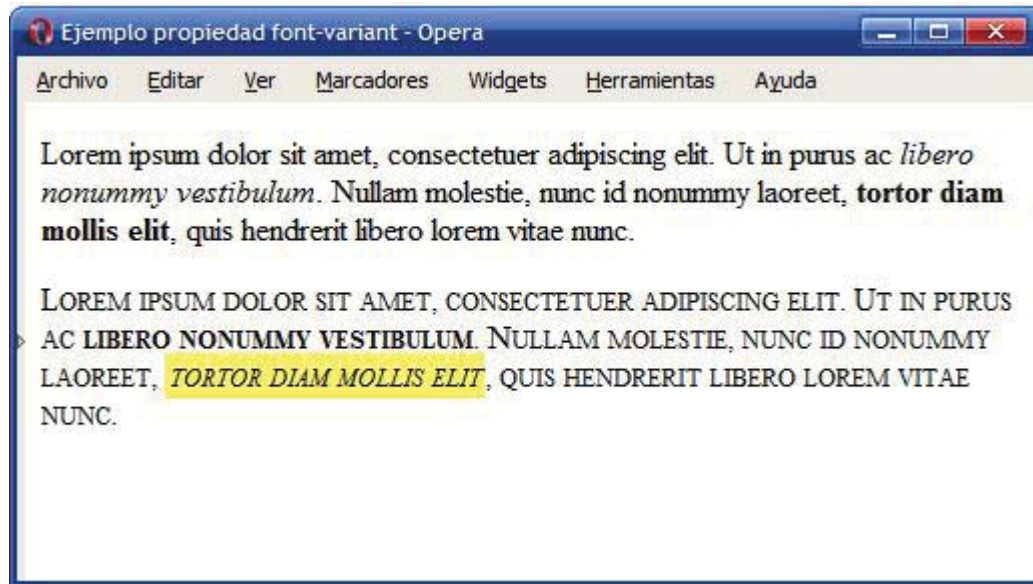


Figura 3.80. Ejemplo de propiedad font-variant

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

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

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

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.

## 7.2. 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`.

<b>text-align</b>	Alineación del texto
<b>Valores</b>	<code>left</code>   <code>right</code>   <code>center</code>   <code>justify</code>   <code>inherit</code>
<b>Se aplica a</b>	Elementos de bloque y celdas de tabla
<b>Valor inicial</b>	left
<b>Descripción</b>	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 3.81. 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:

<b>line-height</b>	Interlineado
<b>Valores</b>	<code>normal</code>   <code>&lt;numero&gt;</code>   <code>&lt;medida&gt;</code>   <code>&lt;porcentaje&gt;</code>   <code>inherit</code>
<b>Se aplica a</b>	Todos los elementos
<b>Valor inicial</b>	normal
<b>Descripción</b>	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:

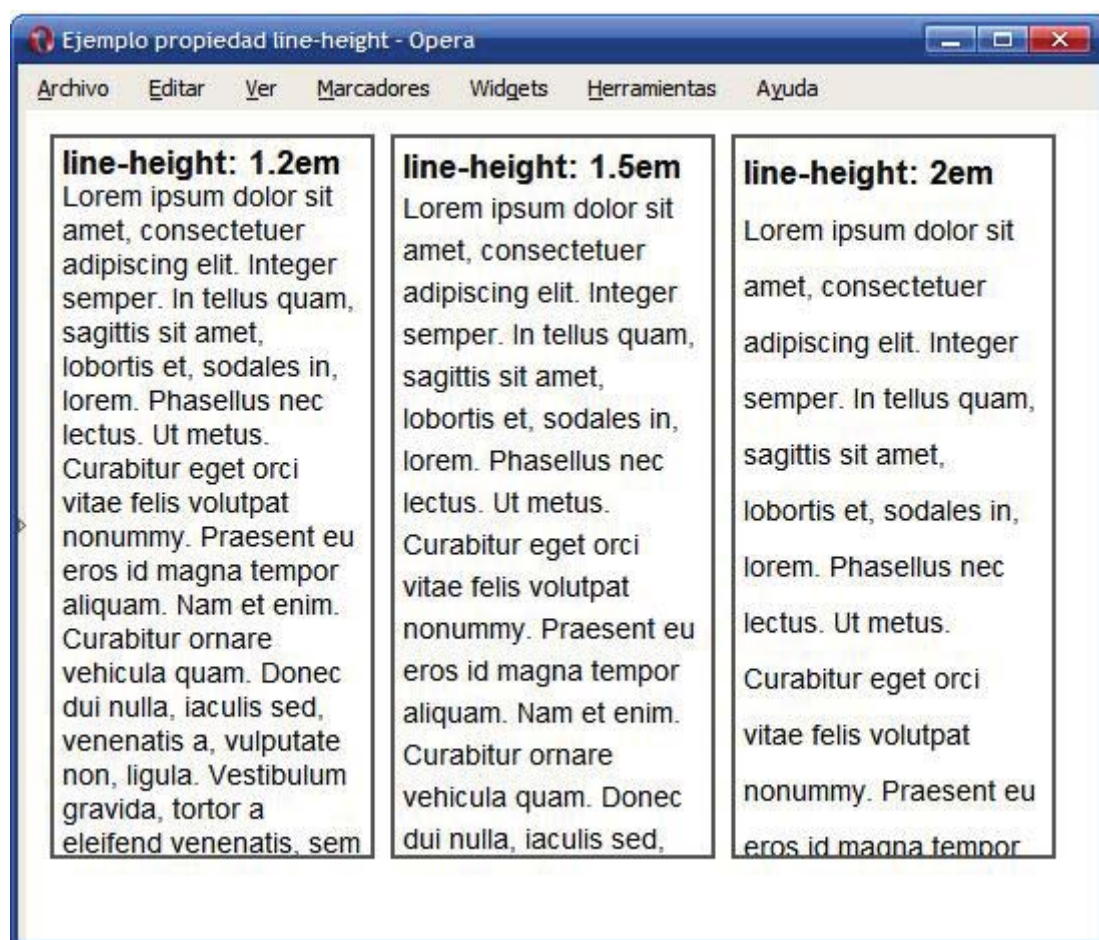


Figura 3.82. Ejemplo de propiedad `line-height`

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`.

<b>text-decoration</b>	Decoración del texto
<b>Valores</b>	<code>none</code>   ( <code>underline</code>    <code>overline</code>    <code>line-through</code>    <code>blink</code> )   <code>inherit</code>
<b>Se aplica a</b>	Todos los elementos
<b>Valor inicial</b>	<code>none</code>
<b>Descripción</b>	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.

<b>text-transform</b>	Transformación del texto
<b>Valores</b>	<code>capitalize</code>   <code>uppercase</code>   <code>lowercase</code>   <code>none</code>   <code>inherit</code>
<b>Se aplica a</b>	Todos los elementos
<b>Valor inicial</b>	<code>none</code>
<b>Descripción</b>	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:

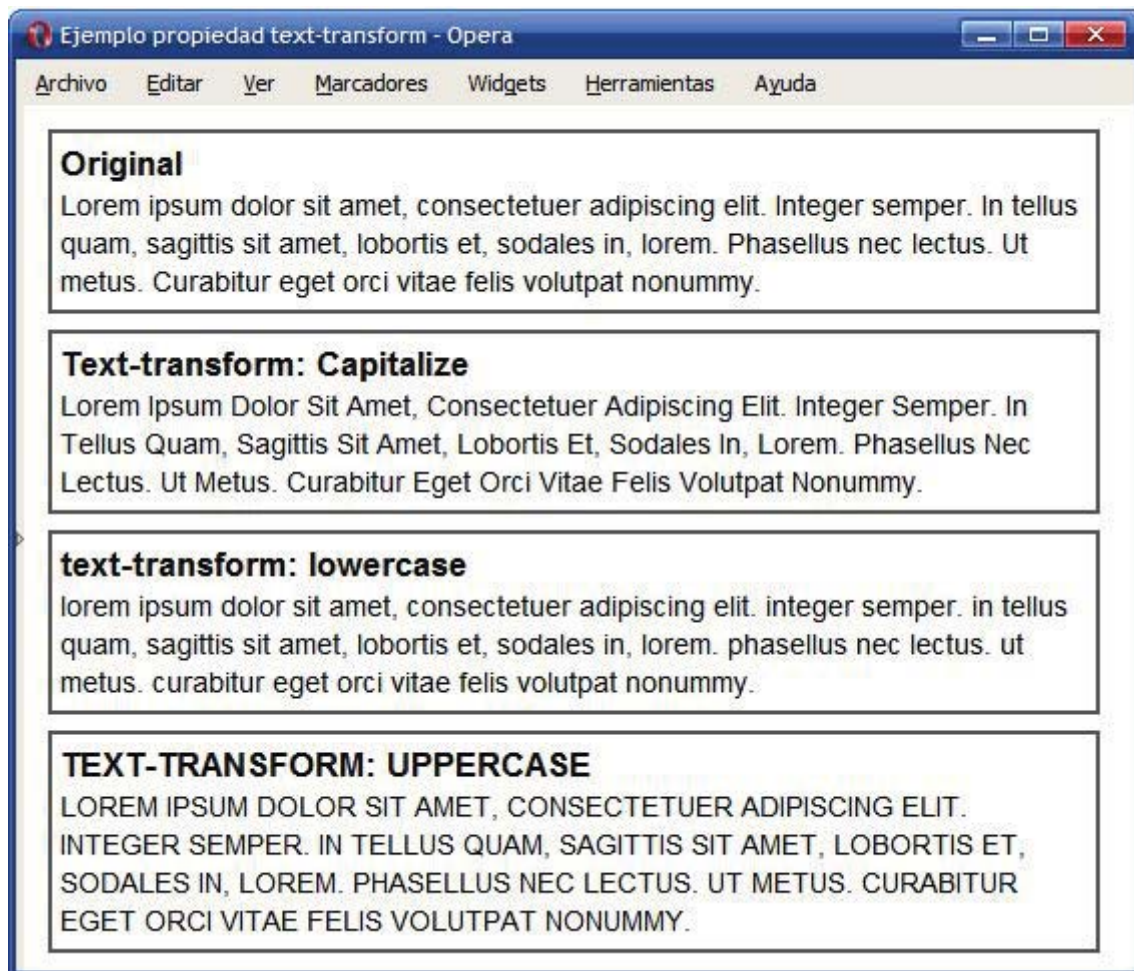


Figura 3.83. Ejemplo de propiedad text-transform

Las reglas CSS del ejemplo anterior se muestran a continuación:

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

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



<b>vertical-align</b>	Alineación vertical
<b>Valores</b>	<code>baseline</code>   <code>sub</code>   <code>super</code>   <code>top</code>   <code>text-top</code>   <code>middle</code>   <code>bottom</code>   <code>text-bottom</code>   <code>&lt;porcentaje&gt;</code>   <code>&lt;medida&gt;</code>   <code>inherit</code>
<b>Se aplica a</b>	Elementos en línea y celdas de tabla
<b>Valor inicial</b>	<code>baseline</code>
<b>Descripción</b>	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`:

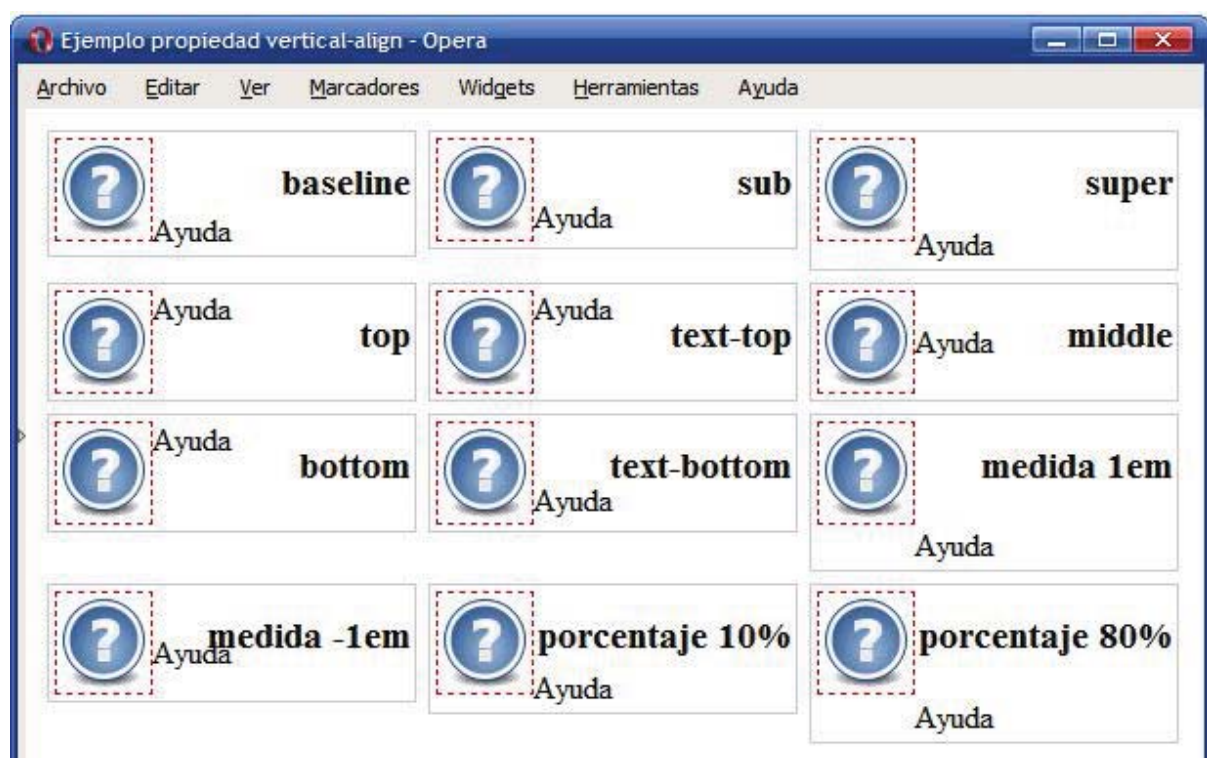


Figura 3.84. Ejemplo de propiedad vertical-align

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`.

<b>text-indent</b>	Tabulación del texto
<b>Valores</b>	<medida>   <porcentaje>   inherit
<b>Se aplica a</b>	Los elementos de bloque y las celdas de tabla
<b>Valor inicial</b>	0
<b>Descripción</b>	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:

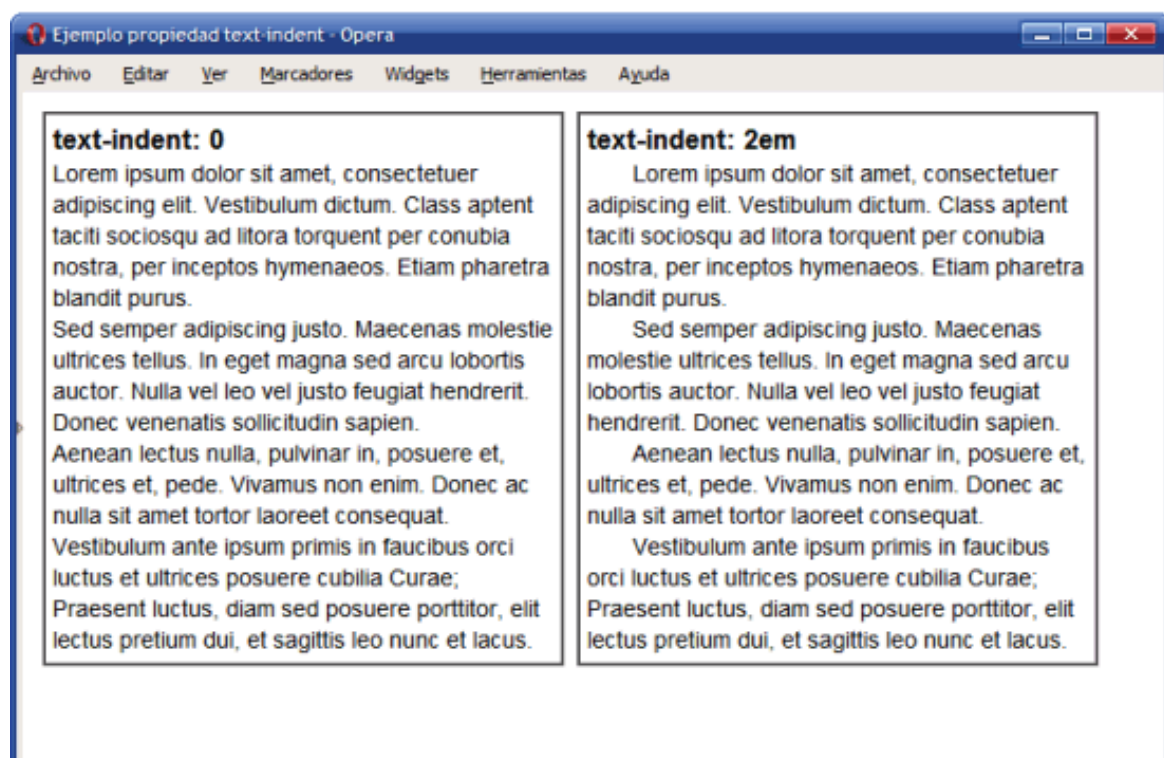


Figura 3.85. Ejemplo de propiedad text-indent

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`.



<b>letter-spacing</b>	Espaciado entre letras
<b>Valores</b>	<code>normal</code>   <code>&lt;medida&gt;</code>   <code>inherit</code>
<b>Se aplica a</b>	Todos los elementos
<b>Valor inicial</b>	normal
<b>Descripción</b>	Permite establecer el espacio entre las letras que forman las palabras del texto

<b>word-spacing</b>	Espaciado entre palabras
<b>Valores</b>	<code>normal</code>   <code>&lt;medida&gt;</code>   <code>inherit</code>
<b>Se aplica a</b>	Todos los elementos
<b>Valor inicial</b>	normal
<b>Descripción</b>	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:

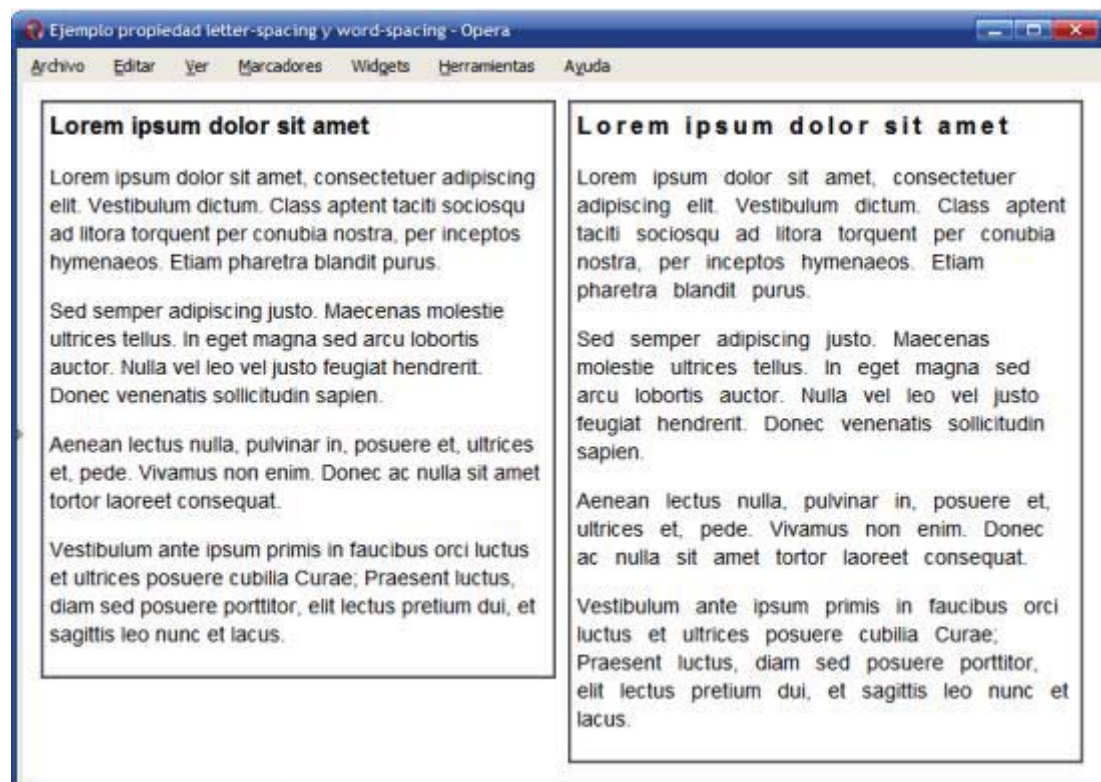


Figura 3.86. Ejemplo de propiedades `letter-spacing` y `word-spacing`

Las reglas CSS del ejemplo anterior se muestran a continuación:

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

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

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

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

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

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

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

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

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

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

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

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

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

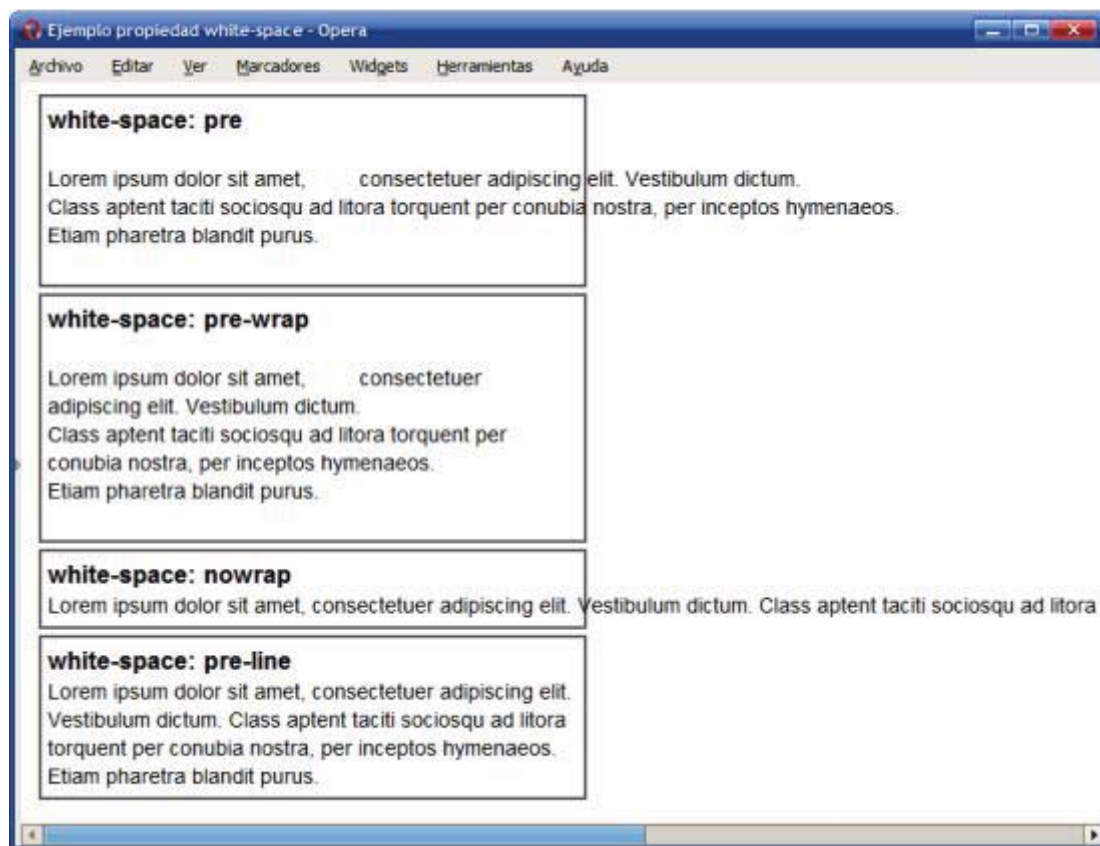


Figura 3.87. Ejemplo de propiedad white-space

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:

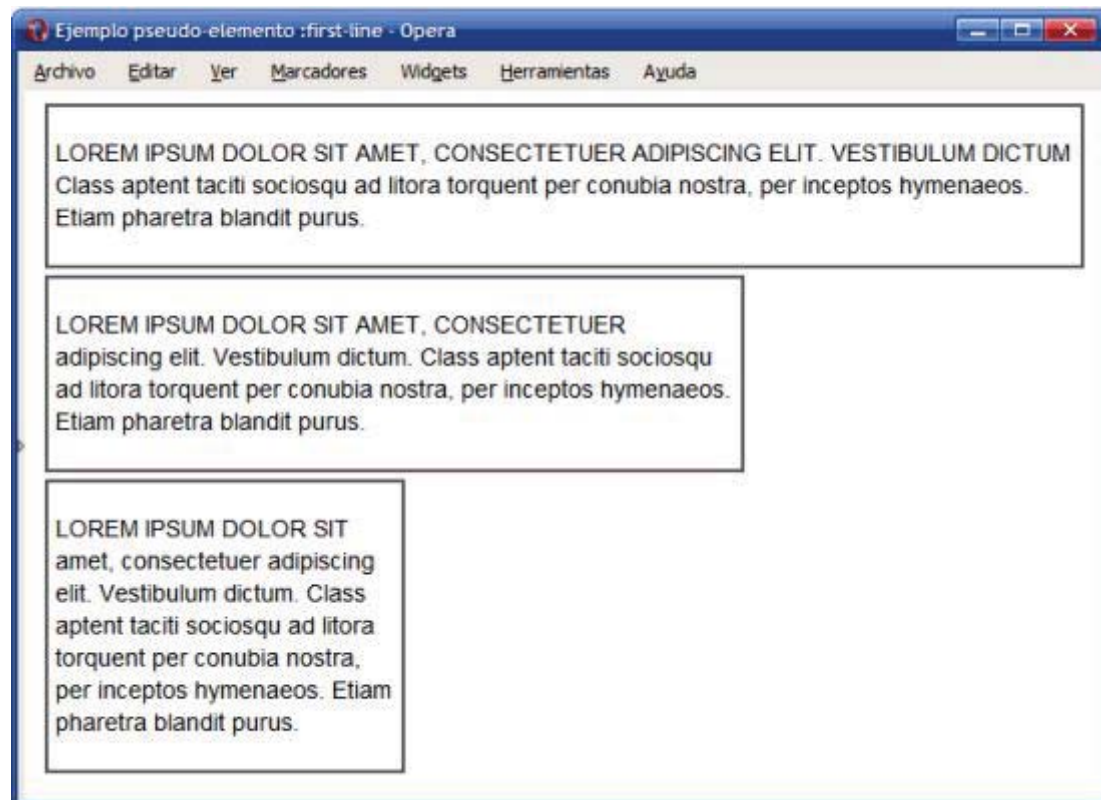


Figura 3.88. Ejemplo de pseudo-elemento first-line

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:

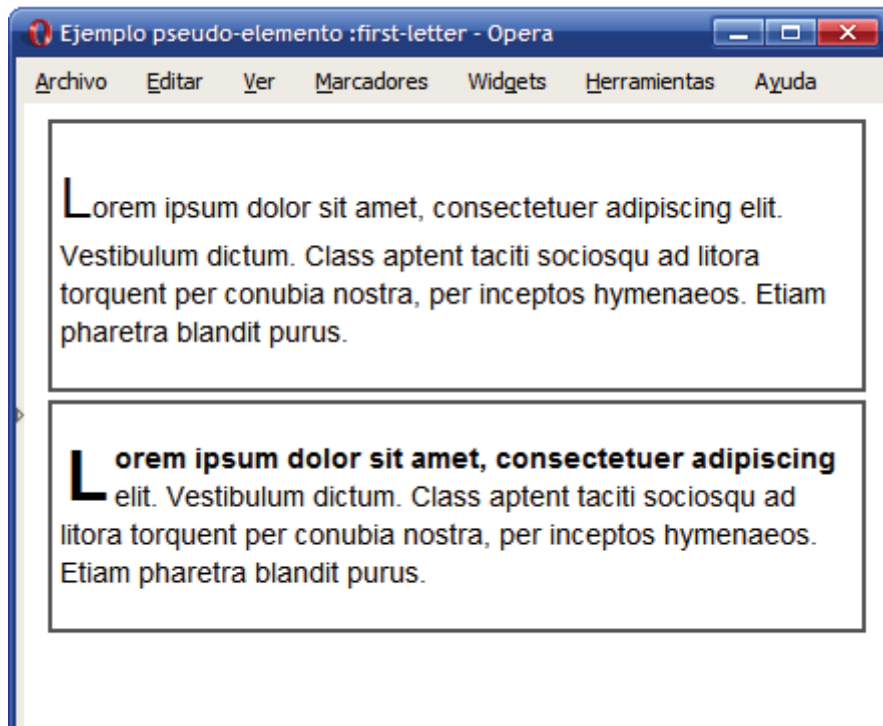


Figura 3.89. Ejemplo de pseudo-elemento first-letter

El código HTML y CSS se muestra a continuación:

```
#normal p:first-letter {  
    font-size: 2em;  
}  
#avanzado p:first-letter {  
    font-size: 2.5em;  
    font-weight: bold;  
    line-height: .9em;  
    float: left;  
    margin: .1em;  
}  
#avanzado p:first-line {  
    font-weight: bold;  
}  
  
<div id="normal">  
<p>Lorem ipsum dolor sit amet, consectetur adipiscing elit.  
Vestibulum dictum. Class aptent taciti sociosqu ad litora torquent per  
conubia nostra, per inceptos hymenaeos. Etiam pharetra blandit  
purus.</p>  
</div>  
<div id="avanzado">  
<p>Lorem ipsum dolor sit amet, consectetur adipiscing elit.  
Vestibulum dictum. Class aptent taciti sociosqu ad litora torquent per  
conubia nostra, per inceptos hymenaeos. Etiam pharetra blandit  
purus.</p>  
</div>
```

## 7.3. Nuevos formatos para el texto con CSS3

El documento de trabajo (*Working Draft*) del W3C para el texto (<http://www.w3.org/TR/css3-text/>) nos “promete” muchos formatos nuevos para el texto: gestión de los espacios, de los cortes con guiones, de los elementos decorativos... Ahora es el turno de los navegadores, quienes deberán trabajar para reconocer todas esas nuevas propiedades.

### 1. Las columnas

Esta novedad bastante útil del CSS3 nos ofrece la posibilidad de maquetar un texto en columnas. Existe un módulo específico del CSS3 para esta funcionalidad: CSS Multi-column Layout Module (<http://www.w3.org/TR/css3-multicol/>) que se encuentra en la fase de *Candidate Recommendation* del 12 de abril de 2011. Por el momento, todavía estamos lejos de alcanzar la compatibilidad con los navegadores, solamente Firefox, Safari y Chrome reconocen parcialmente estas propiedades.

La primera propiedad que podemos aplicar es, simplemente, el ancho de las columnas y su nombre.

- La propiedad `column-count` permite indicar cuántas columnas queremos crear.
- La propiedad `column-width` permite definir el ancho de cada columna.
- La propiedad `column-gap` permite indicar el espacio entre cada columna (la maqueta, en publicación impresa).

Tendremos que usar los prefijos propietarios de los navegadores para usar estas propiedades.

En el siguiente ejemplo se ha dividido en columnas una caja `<div>`.

El estilo CSS:

```
#columnas3 {  
  width: 470px;  
  -moz-column-count: 3;  
  -moz-column-width: 150px;  
  -moz-column-gap: 10px;  
  -webkit-column-count: 3;  
  -webkit-column-width: 150px;  
  -webkit-column-gap: 10px;  
  column-count: 3;  
  column-width: 150px;  
  column-gap: 10px;  
}
```



Este es el código HTML de la caja `<div>`:

```
<div id="columnas3">
<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>
```

El resultado visual:

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Cras auctor tellus ac felis accumsan sit amet pharetra leo dignissim. Phasellus vitae odio arcu, sit amet iaculis dolor. Vivamus ante sem, molestie eget facilisis eget, tincidunt in ipsum. Sed varius, quam in sodales ornare, erat enim suscipit odio, non	pharetra neque tellus pretium ante. Aenean eget risus odio, eu mollis lacus. Sed vestibulum ultrices dolor quis consectetur. Pellentesque eu ante in orci viverra sodales. Sed malesuada pulvinar ligula, quis sagittis sapien congue ut. Etiam sollicitudin turpis ornare quam suscipit accumsan nec vel elit. Sed viverra,	sapien non commodo egestas, ligula lacus elementum lorem, pretium tincidunt augue mi eget mi. Maecenas vitae sagittis eros. Nullam aliquam dolor id nulla egestas varius. Nunc convallis nisl sed odio vulputate gravida. Praesent volutpat dui eu enim elementum sit amet lacinia libero varius. Phasellus non erat elit.
---	---	---

También se puede utilizar la propiedad de tipo "shorthand" `columns`, que permite tener una sintaxis más corta, ya que reúne, en este orden, las propiedades `column-width` y `column-count`.

**Ejemplo:** `columns: 150px 3;`

Podemos definir una línea vertical (de separación) entre las columnas, que se insertará en medio el espacio entre dichas columnas. Disponemos de las propiedades:

- `column-rule-width`: define el grosor de la línea de separación.
- `column-rule-style`: define el tipo o el estilo de la línea de separación.
- `column-rule-color`: define el color de la línea de separación.
- `column-rule`: es la sintaxis abreviada para definir, en este orden, el grosor, el estilo y el color.

Retomemos el ejemplo anterior, con la sintaxis abreviada:

```
#columnas3 {  
  width: 470px;  
  -moz-column-count: 3;  
  -moz-column-width: 150px;  
  -moz-column-gap: 10px;  
  -moz-column-rule: 1px solid grey;  
  -webkit-column-count: 3;  
  -webkit-column-width: 150px;  
  -webkit-column-gap: 10px;  
  -webkit-column-rule: 1px solid grey;  
  column-count: 3;  
  column-width: 150px;  
  column-gap: 10px;  
  column-rule: 1px solid grey;  
}
```

Se visualizará así:

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Cras auctor tellus ac felis accumsan sit amet pharetra leo dignissim. Phasellus vitae odio arcu, sit amet iaculis dolor. Vivamus ante sem, molestie eget facilisis eget, tincidunt in ipsum. Sed varius, quam in sodales ornare, erat enim suscipit odio, non	pharetra neque tellus pretium ante. Aenean eget risus odio, eu mollis lacus. Sed vestibulum ultrices dolor quis consectetur. Pellentesque eu ante in orci viverra sodales. Sed malesuada pulvinar ligula, quis sagittis sapien congue ut. Etiam sollicitudin turpis ornare quam suscipit accumsan nec vel elit. Sed viverra, sapien non commodo	egestas, ligula lacus elementum lorem, pretium tincidunt augue mi eget mi. Maecenas vitae sagittis eros. Nullam aliquam dolor id nulla egestas varius. Nunc convallis nisl sed odio vulputate gravida. Praesent volutpat dui eu enim elementum sit amet lacinia libero varius. Phasellus non erat elit.
---	---	---

Otras propiedades muy interesantes, como la gestión del salto de columna ([break-before](#), [break-after](#) y [break-inside](#)) aún no son gestionadas por los navegadores, pero ofrecen interesantes perspectivas para la maquetación del tipo “revista”.

En Internet encontrarás multitud de herramientas en línea con las que se puede generar todo el código necesario para crear los estilos CSS3, que permiten presentar el texto en columnas.

Es muy recomendable CSS3 Generator (<http://css3generator.com/>). En el menú desplegable **Choose Something** selecciona **Multiple Columns**.

En el campo **# of Columns**, indica el número de columnas que deseas.

En el campo **Column Gap**, indica el espacio entre las columnas que desees aplicar.

En la zona **Your Code**, aparecerá el código compatible con **Firefox**, **Chrome** y **Safari**.



Figura 3.90. Sitio web CSS3 Generator

El sitio web **Dji** (del programador francés Jérôme Debray) nos propone multitud de herramientas en línea, incluido un generador de columnas (<http://debray.jerome.free.fr/index.php?outils/Generateur-de-multi-colonnes-en-css3>).

La interfaz está en francés pero es muy eficaz y fácil de usar. El código obtenido es compatible con **Chrome**, **Safari**, **Firefox** y **Opera**.

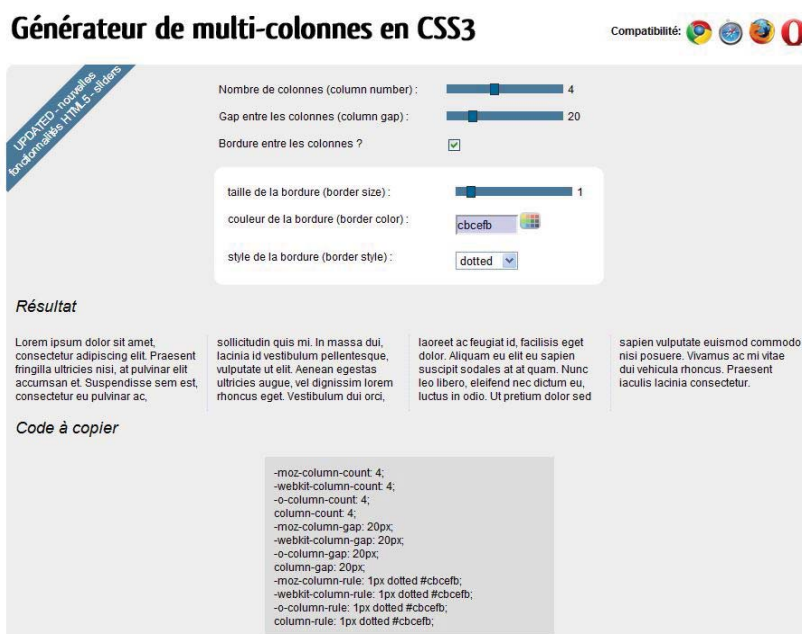


Figura 3.91. Sitio web Dji

## 2. Las palabras cortadas

Los estilos CSS3 relacionados con el texto (<http://www.w3.org/TR/css3-text/>) nos proponen algunas novedades interesantes a la hora de dividir las palabras.

La propiedad `overflow-wrap` (antigua `word-wrap`) permite forzar que se seccione una palabra muy larga, cuando su longitud sea superior al ancho del elemento que la contenga. Si no se usa esta propiedad, la palabra demasiado larga "se saldrá" de su contenedor.

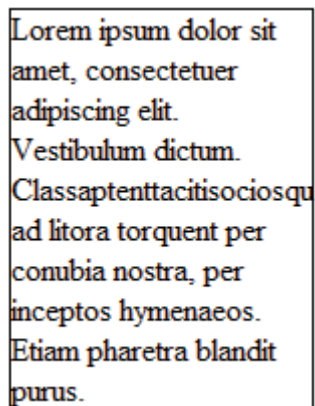
El estilo CSS:

```
.peque {  
  width: 150px;  
  border: 1px solid black;  
  padding: 0;  
}
```

Este sería el código HTML:

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

El resultado visual:



Visual representation of the text wrapped within the defined width (150px). The text is displayed in a single column, wrapping onto multiple lines to fit the container. The text is: Lorem ipsum dolor sit amet, consectetur adipiscing elit. Vestibulum dictum. Class aptent tacit sociosqu ad litora torquent per conubia nostra, per inceptos hymenaeos. Etiam pharetra blandit purus.

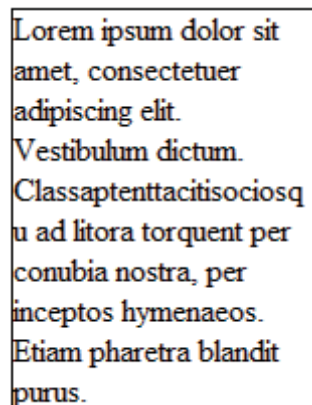
Apliquemos ahora la propiedad `overflow-wrap`, con su antigua sintaxis `word-wrap`, ya que los navegadores aún no reconocen la nueva. Los valores posibles son:

- `normal`: el corte se realiza en aquellos lugares en los que se haya autorizado la división.
- `hyphenate`: el corte se realiza en aquellos lugares en los que sea posible aplicar la división con guión.

- **break-word**: el corte se realiza de manera arbitraria, en función del ancho del contenido. Este es por el momento el único valor que reconocen los navegadores.

```
.peque {  
  width: 150px;  
  border: 1px solid black;  
  padding: 0;  
  word-wrap: break-word;  
}
```

Este es el resultado obtenido:



Lorem ipsum dolor sit  
amet, consectetur  
adipiscing elit.  
Vestibulum dictum.  
Class aptent tacit sociosq  
u ad litora torquent per  
conubia nostra, per  
inceptos hymenaeos.  
Etiam pharetra blandit  
purus.

## 7.4. Importar las fuentes tipográficas

Con los estilos CSS3 y la regla **@font-face** se pueden “importar” las fuentes tipográficas en nuestra página web. De este modo ya no estaremos limitados al uso de las fuentes genéricas.

Pero no hay que olvidar estos tres hechos fundamentales:

- La mayoría de las fuentes “profesionales” están sometidas a derechos de uso y de difusión.
- Cuando se “importa” una fuente tipográfica, se incorporan a los archivos la totalidad de los caracteres de dicha fuente. Esto podría sobrecargar en exceso sus páginas web.
- Por lo general, el *antialiasing* (que evita que los caracteres tengan un aspecto dentado) no se aplica a las páginas web.

En cuanto a la compatibilidad con las diferentes versiones de los navegadores, estos son los formatos de las fuentes tipográficas reconocidas por los navegadores:

- **TrueType**: extensión **.ttf**.
- **OpenType**: extensión **.otf**
- **Web Open Font**: extensión **.woff**.
- **SVG Font**: extensión **.svg** y **.svgz**.
- **Embed Open Type**: extensión **.eot**. Atención, se trata de un formato propietario Microsoft.

Veamos la sintaxis de la regla `@font-face`:

```
@font-face{  
  font-family: "Skia";  
  src: url('Skia.ttf');  
}
```

Una vez que hayamos declarado la regla `@font-face`, con la propiedad `font-family` indicaremos el nombre de la fuente tipográfica que queramos importar. Luego, con la propiedad `src` indicaremos la ruta de acceso al archivo de dicha fuente. Para aplicarla, simplemente tendremos que indicar en el selector en cuestión el nombre de la fuente que deberá usarse con la propiedad `font-family`:

```
h1, h2 {  
  font-family: Skia;  
}
```

Este es el código HTML:

```
<h1>Este es el título de mi página</h1>  
<h2>El subtítulo de mi página</h2>  
<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>
```

Y el resultado visual:

# Este es el título de mi página

## El subtítulo de mi página

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.

En cuanto al **nombre de las tipografías**, el nombre que se indica en font-family es totalmente arbitrario, no tiene por qué corresponder al nombre del archivo de la fuente tipográfica o al nombre que le haya dado el tipógrafo. Se trata el nombre que se le haya asignado a dicha fuente. Lo que importa, es el nombre del archivo que contiene la fuente tipográfica.

Ésta es una sintaxis correcta y que funciona bien:

```
@font-face{
  font-family: "Nombre de la tipografía";
  src: url('Skia.ttf');
}

h1, h2 {
  font-family: 'Nombre de la tipografía', Arial, Helvetica, sans-serif;
}
```

De manera predeterminada con la sintaxis precedente, las fuentes tipográficas se descargarán del servidor en el ordenador del usuario que visite nuestro sitio web. Sin embargo, podemos suponer que algunos visitantes disponen de esa tipografía localmente en su ordenador. Indicaremos entonces que deberá usarse la tipografía local, cuando ésta esté disponible, con local en src.

```
@font-face{
  font-family: "Skia";
  src: local("Skia"), url('Skia.ttf');
}
```

El nombre de la tipografía que se indique en "local" deberá corresponder al nombre del archivo del sistema.

Esta sintaxis se puede leer así: se debe usar una tipografía a la que el diseñador web ha llamado **Skia**, si está disponible en local se deberá usar la tipografía que se llame **Skia** en el sistema del usuario, de lo contrario, se deberá descargar la fuente que se encuentra en el archivo llamado 'Skia.ttf'.

Podemos indicar además varios formatos de fuente (si existen) para mejorar la compatibilidad:

```
@font-face{
  font-family: "Skia";
  src: url('Skia.ttf') format ("truetype"),
  url('Skia.woff') format ("woff");
}
```



Claro está, también podemos indicar otras fuentes tipográficas para aumentar la compatibilidad con los navegadores antiguos.

```
h1, h2 {  
  font-family: Skia, Arial, Helvetica, sans-serif;  
}
```

En la declaración de la regla `@font-face`, puede indicar determinados estilos tipográficos con la propiedad `font-weight`, `font-style` y `font-variant`. Sin embargo, los navegadores aún no los reconocen del todo bien.

```
@font-face{  
  font-family: "Nombre de la tipografía";  
  src: url('Skia.ttf');  
  font-weight: bold;  
}
```

El servicio en línea **CSS 3.0 Maker** (<http://www.css3maker.com/index.html>) propone varios generadores CSS3, entre ellos, un generador de fuentes tipográficas con la regla `@font-face`: <http://www.css3maker.com/font-face.html>

En el cuadro **CSS3 Styles**, selecciona una tipografía, el tamaño, el estilo y el ancho. En el recuadro **CSS3 Preview Area** podrás visualizar el resultado de tu selección. En el recuadro **CSS3 Codeview** podrás copiar el código generado, o bien, puedes hacer clic en el botón **Download**.



Figura 3.92. Sitio web CSS 3.0 Maker

Diferentes servicios en línea ofrecen la posibilidad de usar tipografías para la Web. El servicio de Google, **Google web fonts**, es uno de los más populares. Esta es su URL: <http://www.google.com/webfonts>

Dispone de más de 250 tipografías.

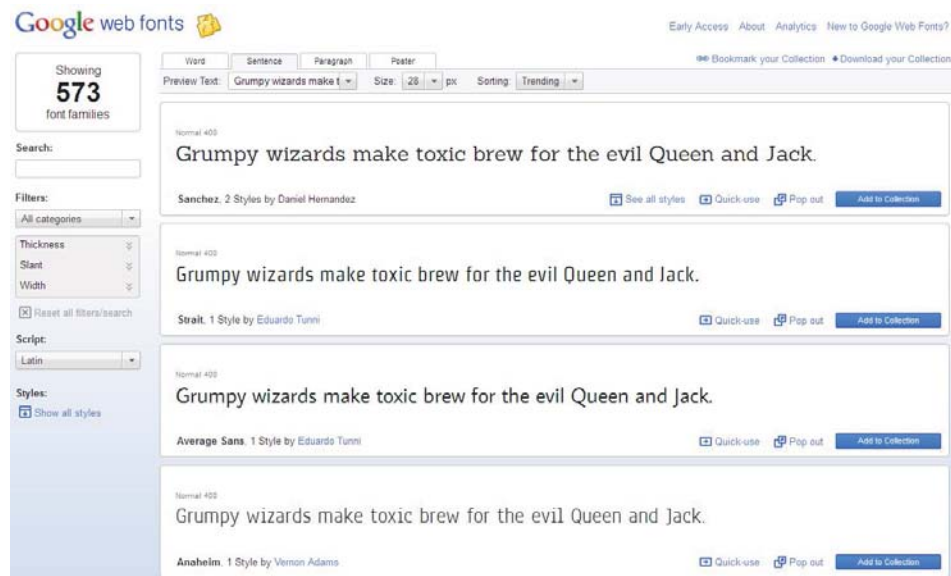


Figura 3.93. Sitio web Google web fonts

En la lista de tipografías, selecciona la que prefieras.

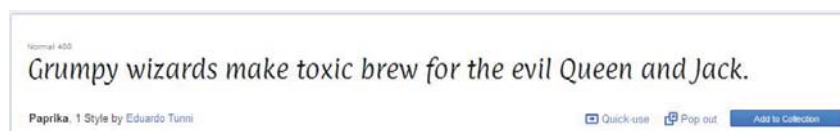


Figura 3.94. Fuente Paprika de Google web fonts

En el recuadro, puedes hacer clic en el vínculo **Pop out** para obtener más información sobre esa tipografía.



Figura 3.95. Sitio web Google web fonts

En el recuadro, haz clic en el vínculo **Quick-use**. En los recuadros **3** y **4** encontrarás las instrucciones para usar esa tipografía en tus hojas de estilo CSS.



The screenshot shows the Google Web Fonts interface. At the top, there are tabs for 'Standard', '@import', and 'Javascript'. Step 3, 'Add this code to your website:', shows a code box with the following HTML link: `<link href='http://fonts.googleapis.com/css?family=Paprika' rel='stylesheet' type='text/css'>`. To the right, instructions state: 'Instructions: To embed your Collection into your web page, copy the code as the first element in the <head> of your HTML document.' and a link '» See an example'. Step 4, 'Integrate the fonts into your CSS:', shows a code box with the CSS rule: `font-family: 'Paprika', cursive;`. To the right, instructions state: 'Instructions: Add the font name to your CSS styles just as you'd do normally with any other font.' and an 'Example:' section showing a CSS rule: `h1 { font-family: 'Metrophobic', Arial, serif; font-weight: 400; }`.

Figura 3.96. Sitio web Google web fonts

Este sería un ejemplo de un vínculo CSS:

```
<link href='http://fonts.googleapis.com/css?family=Paprika'
rel='stylesheet' type='text/css'>
```

Y la aplicación en la hoja de estilo:

```
h1, h2 {
    font-family: 'Paprika', cursive;
}
```

Veamos el código HTML de la página:

```
<h1>Este es el título de mi página</h1>
<h2>El subtítulo de mi página</h2>
<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>
```

Este es el resultado visual:

*Este es el título de mi página*

*El subtítulo de mi página*

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.

A la hora de diseñar la interfaz de un sitio web, el sitio web **Font Combinator** (<http://font-combinator.com/>) te permite probar en línea tres fuentes tipográficas de Google: una para los títulos, una para los subtítulos y otra para el texto normal.

Selecciona las tipografías y el tamaño en la parte inferior de la interfaz.

## The Web Font Combinator

### A Web Typography Tool

This tool has been built to allow previewing of font combinations in a *fast, browser-based* manner. There have been numerous printed books through the years that allowed a designer to put a headline font next to a body font, and this is an attempt to recreate that for the web.

**Directions:** You can edit any of the type on this page in order to preview any particular text. In the controls at the bottom, select the element you want to modify, and then *play!* You can change the font, size, line height and color of an element, as well as hide an element altogether.

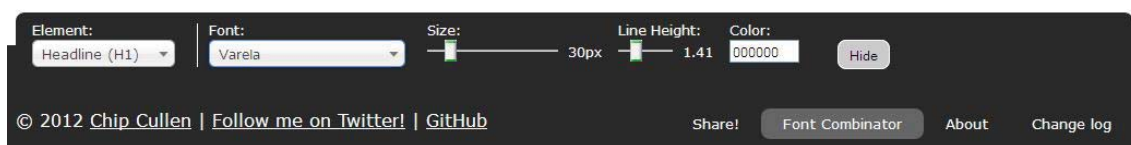


Figura 3.97. Sitio web Font Combinator

Otro sitio web interesante es **Font comparer** (<http://www.fontcomparer.com/>). En este sitio podrás comparar un gran número de tipografías de Google y luego recuperar el vínculo de la tipografía que desees. Para ello, pasa el cursor por encima del nombre de una tipografía y haz clic en el vínculo **Get font**.

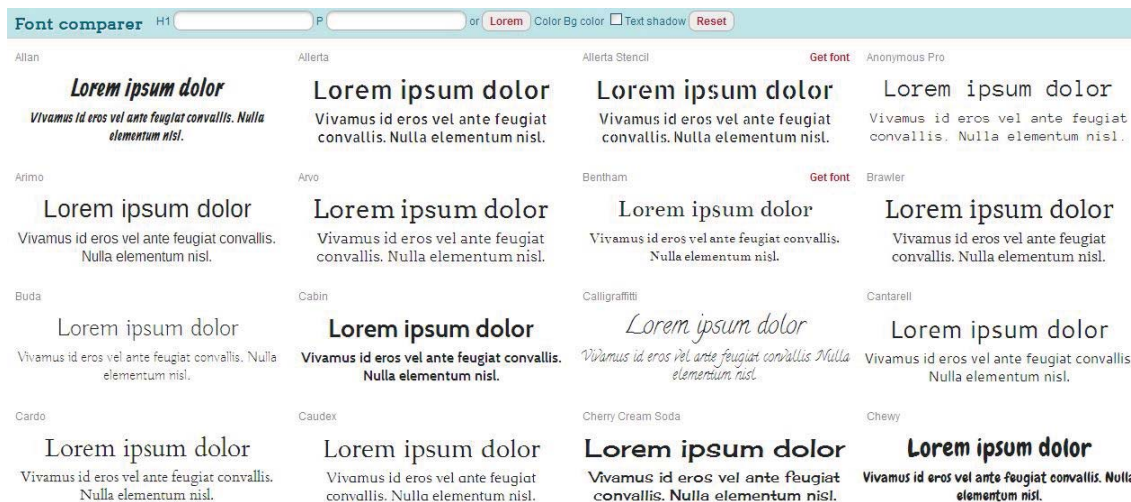


Figura 3.98. Sitio web Font Comparer

## 7.5. Textos con sombra

La nueva propiedad CSS3 `text-shadow` permite aplicar una sombra a un texto. Esta propiedad puede usar diversos valores:

- El primer valor es el desplazamiento horizontal de la sombra.
- El segundo valor es el desplazamiento vertical de la sombra.
- El tercer valor es el tamaño del difuminado de la sombra.
- El cuarto valor es el color de la sombra.
- El quinto valor es la dirección de la sombra: `inset`, hacia el interior (por defecto, la sombra será exterior).

Veamos un ejemplo sencillo:

```
h1 {  
  text-shadow: 8px 8px 5px rgb(100,50,200);  
}
```

- desplazamiento horizontal de 8 píxeles,
- desplazamiento vertical de 8 píxeles,
- difuminado de 5 píxeles,
- color violeta.

Este es el resultado visual:

**Título de la página con una sombra**

Una vez más, disponemos de multitud de soluciones en línea para general texto con sombras.

El sitio web **WestCIV** (<http://www.westciv.com/>) pone a tu disposición completas herramientas para generar estilos CSS3. La herramienta **Text Properties** (<http://www.westciv.com/tools/text-properties/index.html>) presenta una completa interfaz en la que podrás dar formato al texto con las propiedades CSS3. En la zona **Text and Font Properties**, encontrarás las propiedades clásicas que permiten cambiar el formato del texto. En la zona **Text Columns**, podrás organizar las cajas en columnas y, en la zona **Shadow**, podrás aplicar sombras al texto.



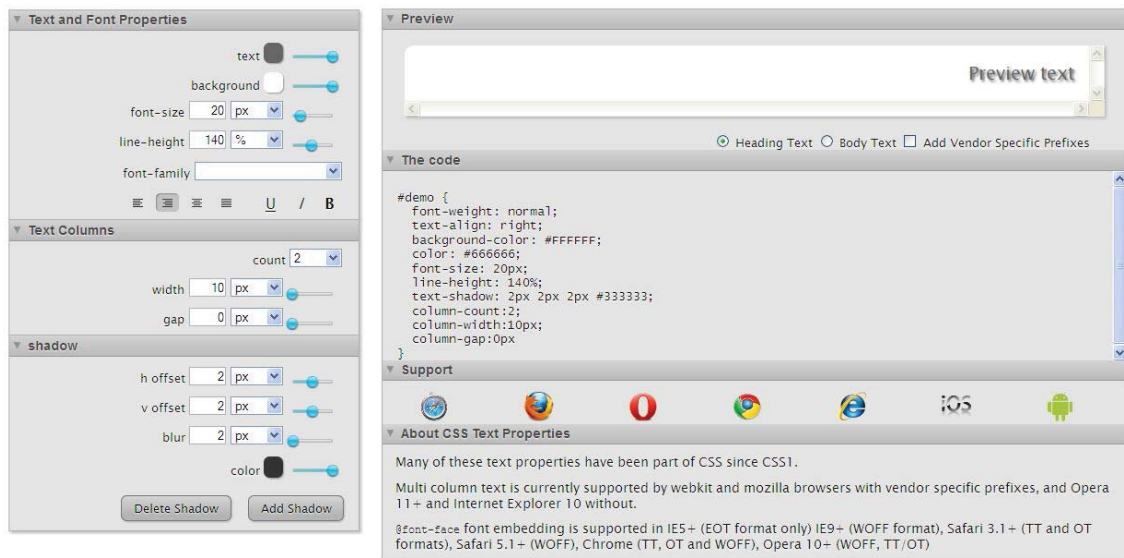


Figura 3.99. Sitio web WestCIV

El sitio web **CSS3Gen** (<http://css3gen.com/text-shadow/>) pone a tu disposición una sencilla interfaz donde podrás generar el código CSS3 que necesitas.



Figura 3.100. Sitio web CSS3Gen

El sitio web **WordPressThemShock** también propone un generador de texto sombreado (<http://www.wordpressthemeshock.com/css-text-shadow/>).

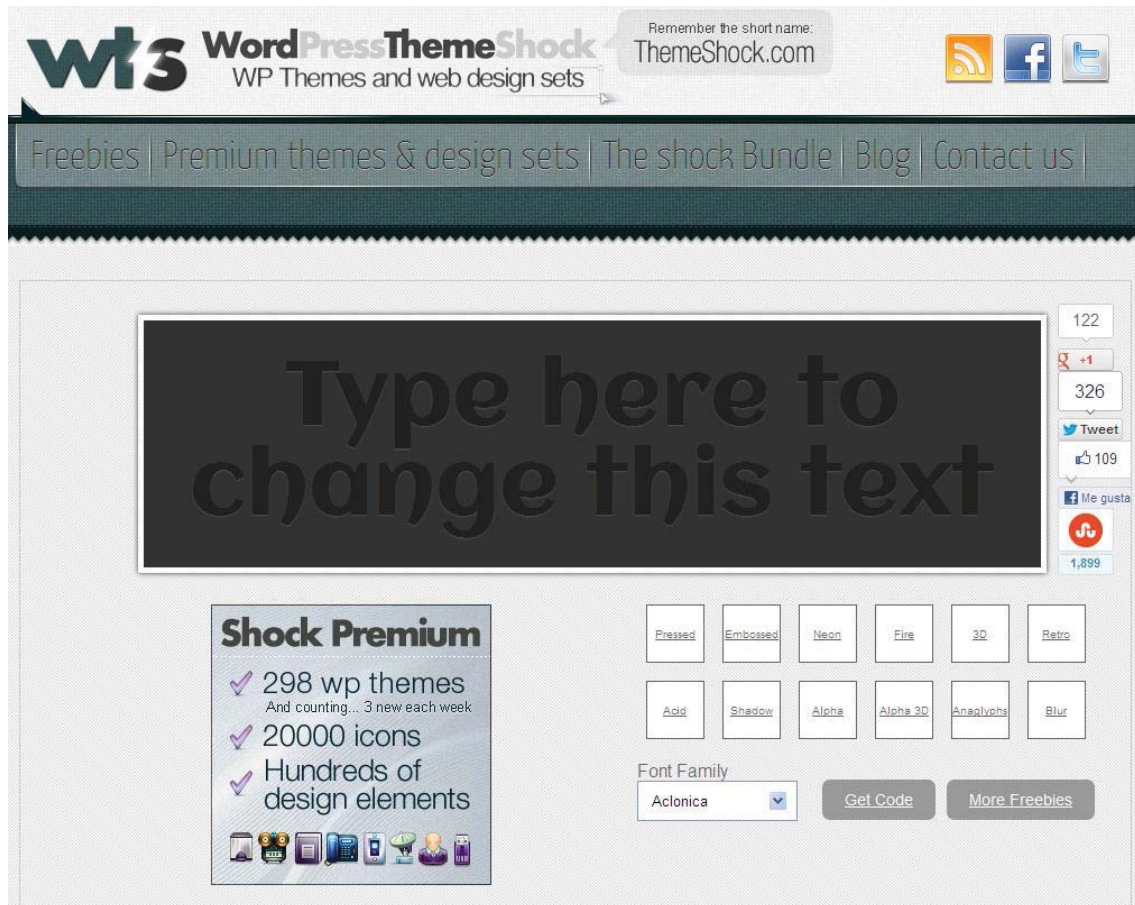


Figura 3.101. Sitio web WordPressThemeShock



## 8. LISTAS

### 8.1. Estilos básicos

#### 8.1.1. 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`.

<b>list-style-type</b>	Tipo de viñeta
<b>Valores</b>	<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>
<b>Se aplica a</b>	Elementos de una lista
<b>Valor inicial</b>	<code>disc</code>
<b>Descripción</b>	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`:

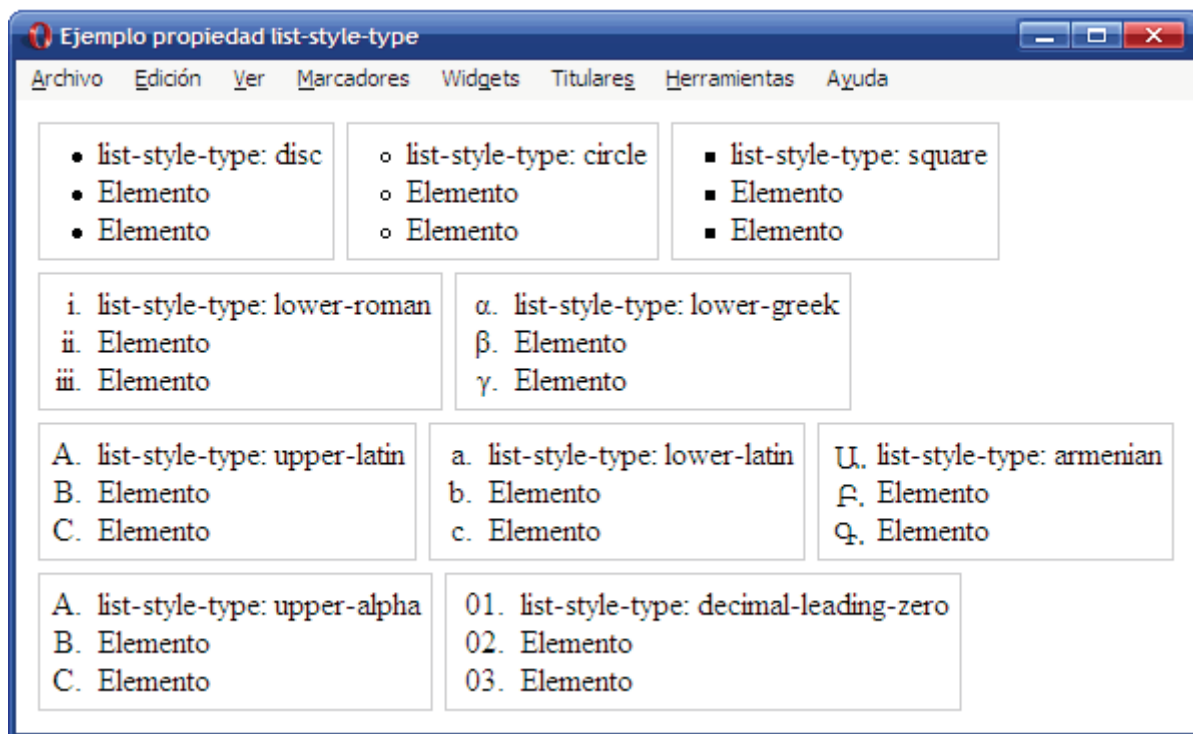


Figura 3.102. Ejemplo de propiedad `list-style-type`

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.

<b>list-style-position</b>	Posición de la viñeta
<b>Valores</b>	<code>inside</code>   <code>outside</code>   <code>inherit</code>
<b>Se aplica a</b>	Elementos de una lista
<b>Valor inicial</b>	<code>outside</code>
<b>Descripción</b>	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:

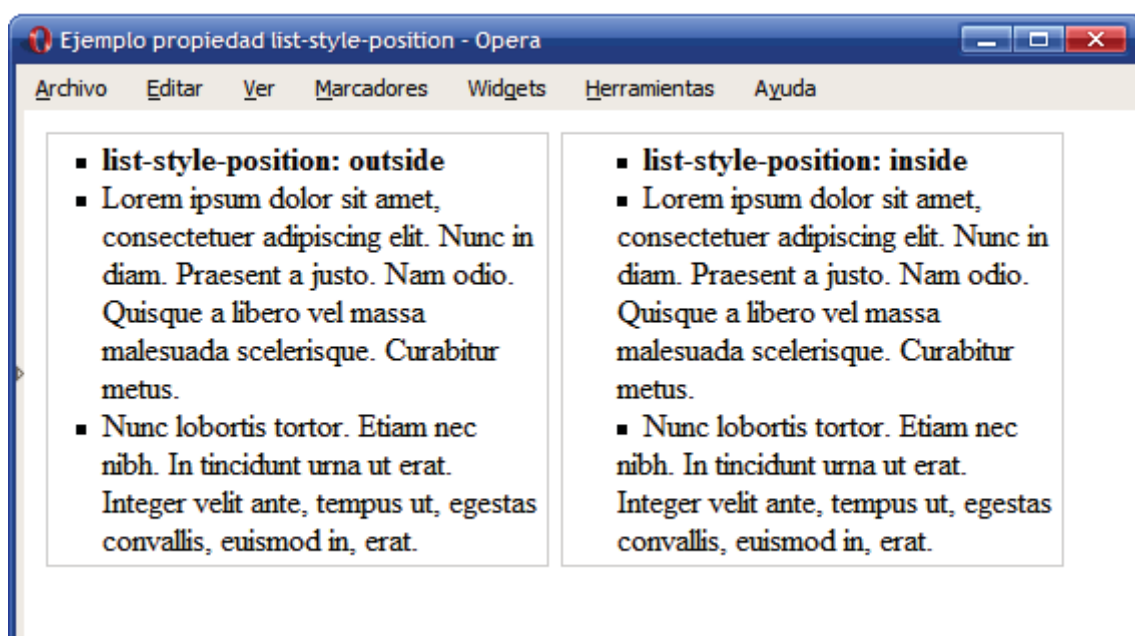


Figura 3.103. Ejemplo de propiedad `list-style-position`

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.

<b>list-style-image</b>	Imagen de la viñeta
<b>Valores</b>	<url>   none   inherit
<b>Se aplica a</b>	Elementos de una lista
<b>Valor inicial</b>	none
<b>Descripción</b>	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 3.104. 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`.

<b>list-style</b>	Estilo de una lista
<b>Valores</b>	( <list-style-type>    <list-style-position>    <list-style-image> )   inherit
<b>Se aplica a</b>	Elementos de una lista
<b>Valor inicial</b>	-
<b>Descripción</b>	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; }
```

### 8.1.2. 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 3.105. 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 3.106. 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 3.107. 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 3.108. Menú vertical: añadiendo bordes

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

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

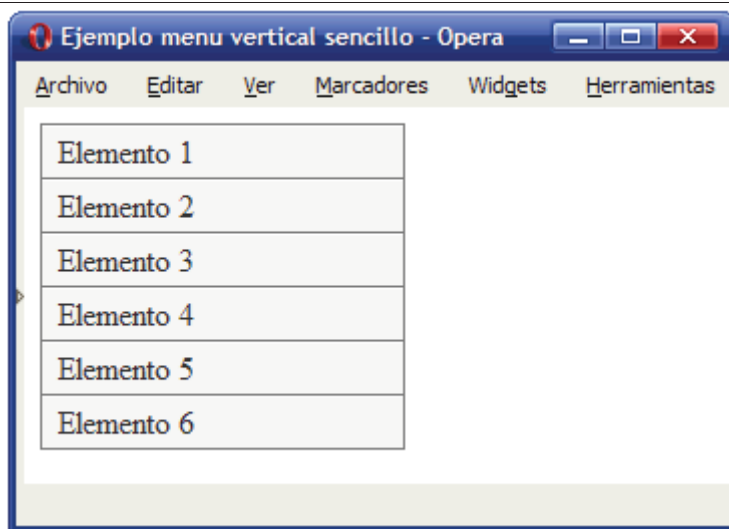


Figura 3.109. Aspecto final del menú vertical sencillo creado con CSS

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

## 8.2. Estilos avanzados

### 8.2.1. 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 definirlos para los elementos `<li>` como en el ejemplo anterior. En cualquier caso, es indiferente el

elemento en el que se aplican los estilos que definen el aspecto de cada opción del menú.

Código HTML del menú horizontal:

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

Código CSS del menú vertical anterior:

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

Aspecto final del menú horizontal:

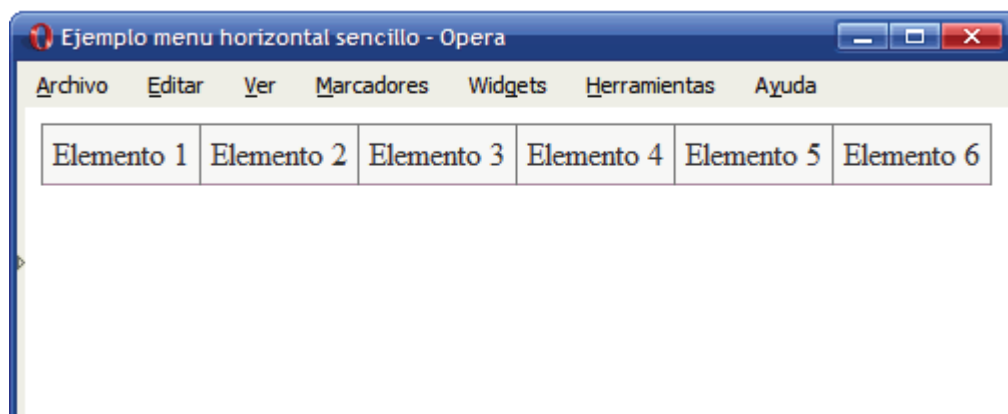


Figura 3.110. Menú horizontal sencillo creado con CSS

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

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

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



Figura 3.111. Menú horizontal: definiendo anchura y bordes

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

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

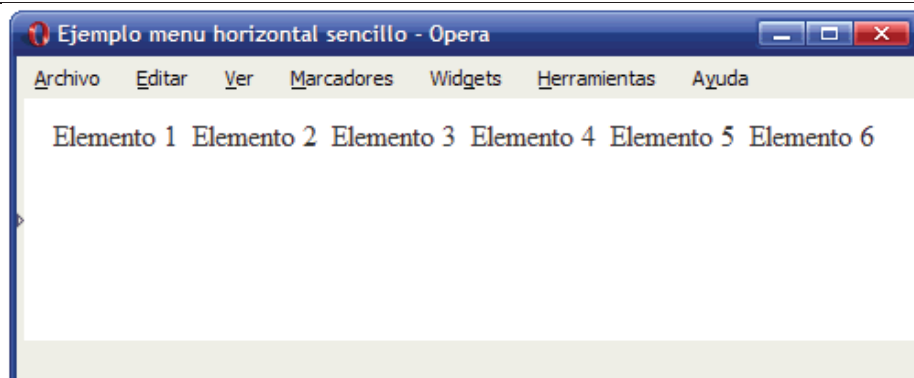


Figura 3.112. Menú horizontal: propiedades float y display

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;  
}
```

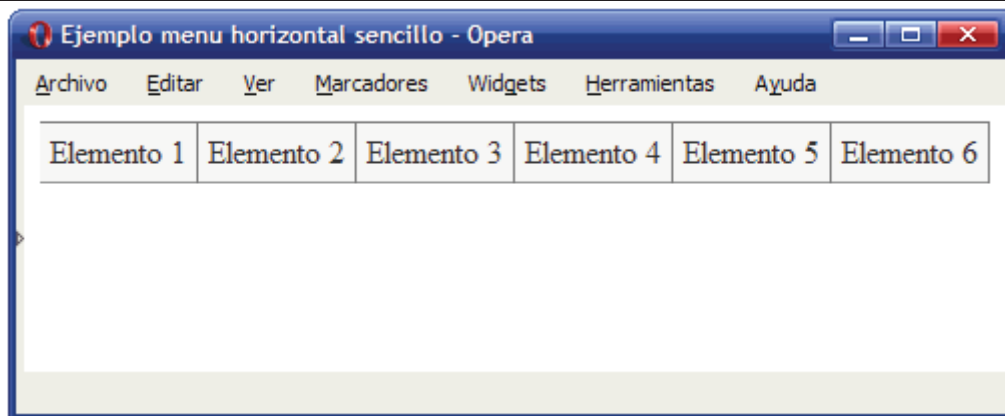


Figura 3.113. Menú horizontal: relleno y borde de los elementos

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

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

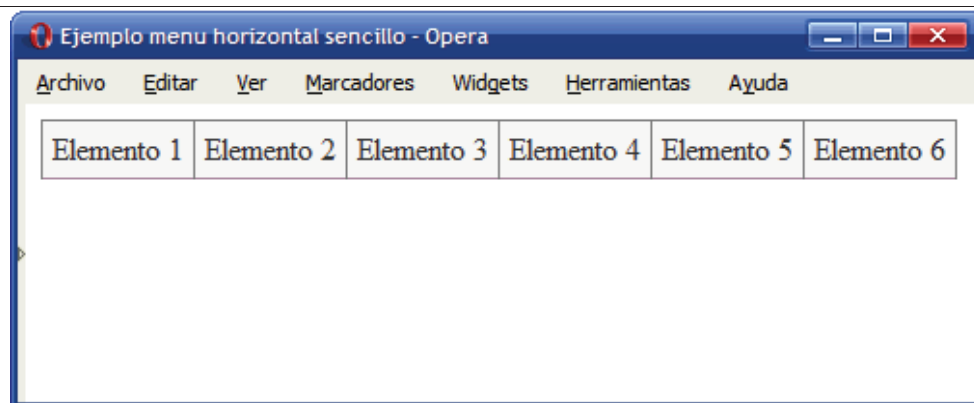


Figura 3.114. Aspecto final del menú horizontal sencillo creado con CSS

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;  
}
```

### 8.2.2. Menú horizontal con pestañas

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



Figura 3.115. Ejemplos de menús horizontales con pestañas creados con CSS

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



### 8.2.3. Menú horizontal avanzado

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

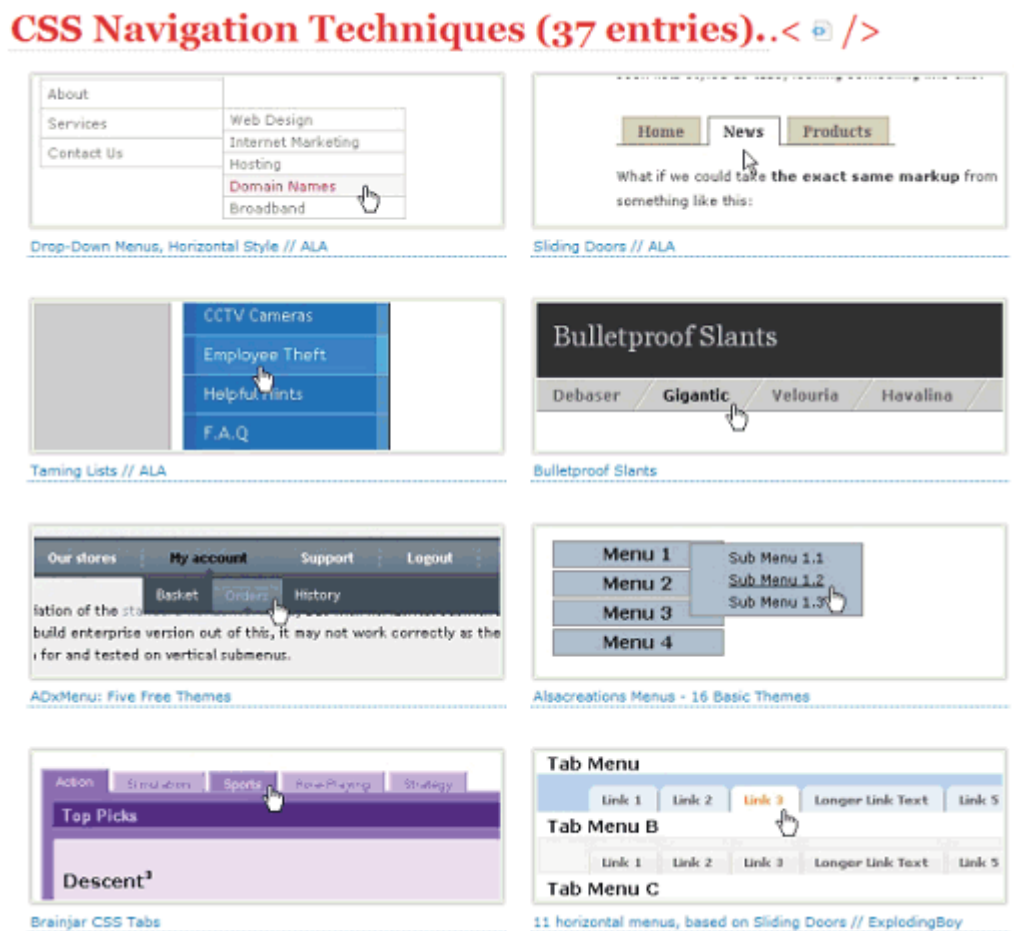


Figura 3.116. Ejemplos de menús horizontales y verticales complejos creados con CSS

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

## 9. TABLAS

### 9.1. 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:

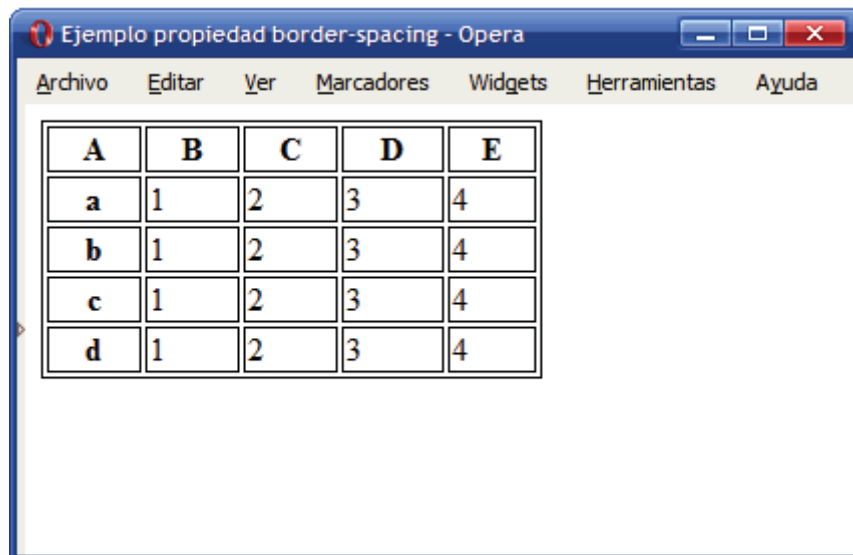


Figura 3.117. Aspecto por defecto de los bordes de una tabla

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

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

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

<b>border-collapse</b>	Fusión de bordes
<b>Valores</b>	<code>collapse</code>   <code>separate</code>   <code>inherit</code>
<b>Se aplica a</b>	Todas las tablas
<b>Valor inicial</b>	<code>separate</code>
<b>Descripción</b>	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 3.118. Ejemplo de propiedad border-collapse

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

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

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

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

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

<b>border-spacing</b>	Espaciado entre bordes
<b>Valores</b>	<medida> <medida>?   inherit
<b>Se aplica a</b>	Todas las tablas
<b>Valor inicial</b>	0
<b>Descripción</b>	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 3.119. 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.

## 9.2. Estilos avanzados

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

<b>empty-cells</b>	Tratamiento de las celdas vacías
<b>Valores</b>	<code>show</code>   <code>hide</code>   <code>inherit</code>
<b>Se aplica a</b>	Celdas de una tabla
<b>Valor inicial</b>	<code>show</code>
<b>Descripción</b>	Define el mecanismo utilizado para el tratamiento de las celdas vacías de una tabla

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

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

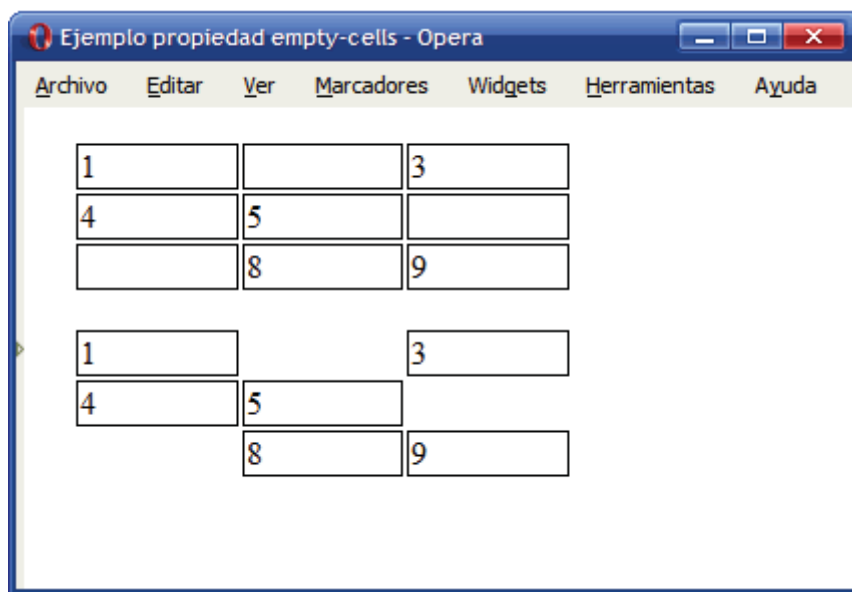


Figura 3.120. Ejemplo de propiedad `empty-cells`

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

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

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

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



```
.especial {  
  caption-side: bottom;  
}  
  
<table class="especial" summary="Tabla genérica">  
  <caption>Tabla 2.- Título especial</caption>  
  <tr>  
    <td>1</td>  
    <td>2</td>  
    <td>3</td>  
  </tr>  
  ...  
</table>
```

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

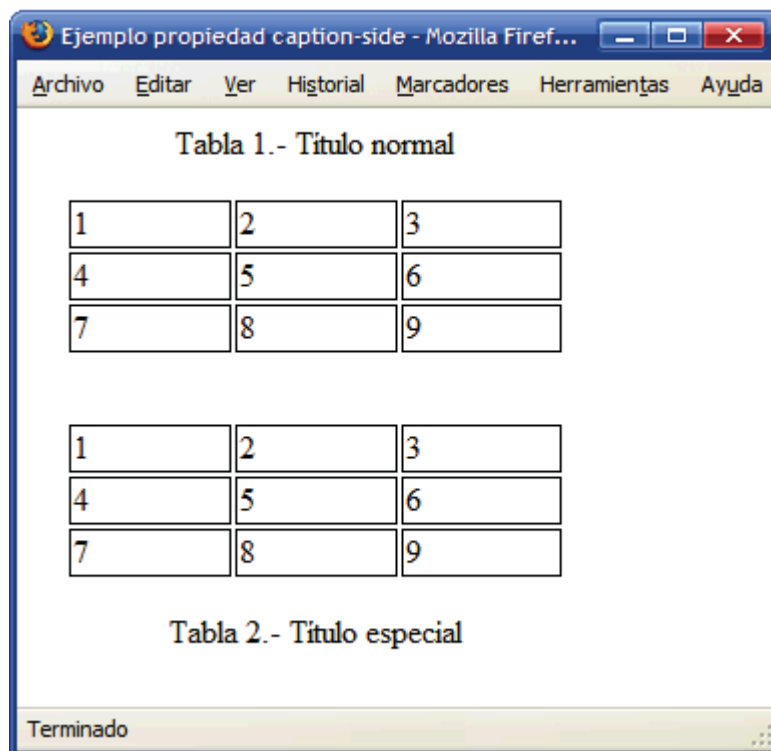


Figura 3.121. Ejemplo de propiedad caption-side

## 10. FORMULARIOS

### 10.1. Estilos básicos

#### 10.1.1. 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.

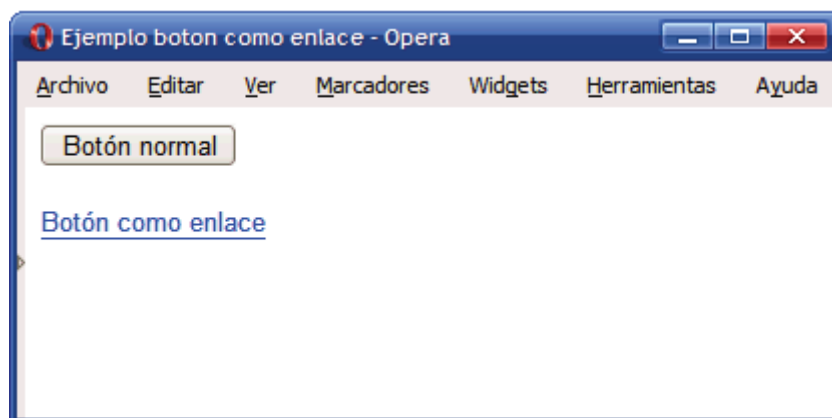


Figura 3.122. Mostrando un botón como si fuera un enlace

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" />
```

#### 10.1.2. Mejoras en los campos de texto

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

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

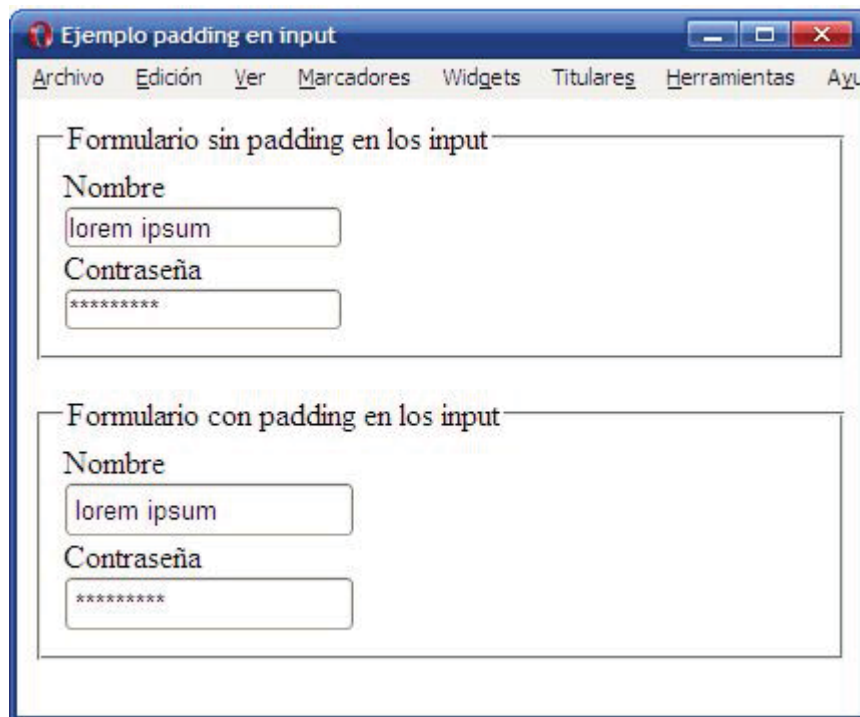


Figura 3.123. Mejorando el aspecto de los formularios gracias al padding

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

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

### 10.1.3. Labels alineadas y formateadas

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

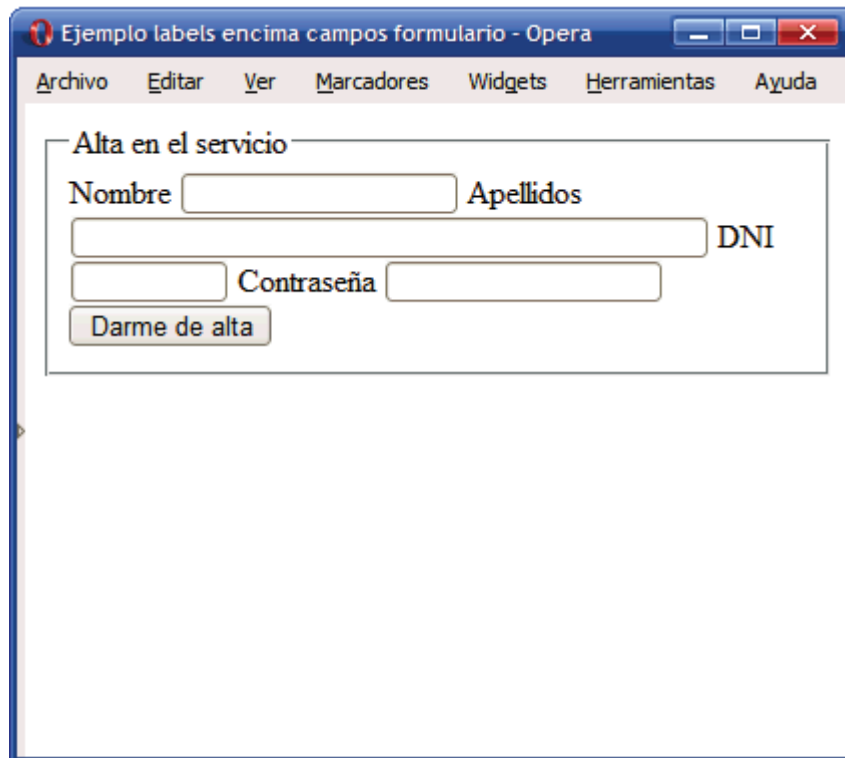


Figura 3.124. Aspecto por defecto que muestran los formularios

El código HTML del ejemplo anterior es el siguiente:

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

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

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

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

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

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

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

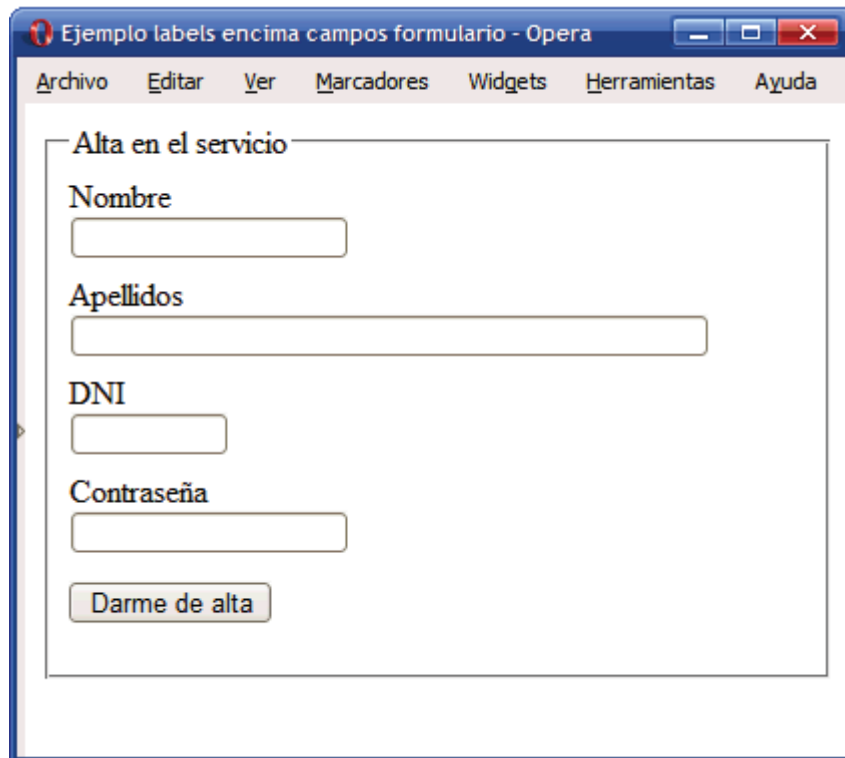


Figura 3.125. Mostrando las etiquetas label encima de los campos del formulario

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

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

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

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

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

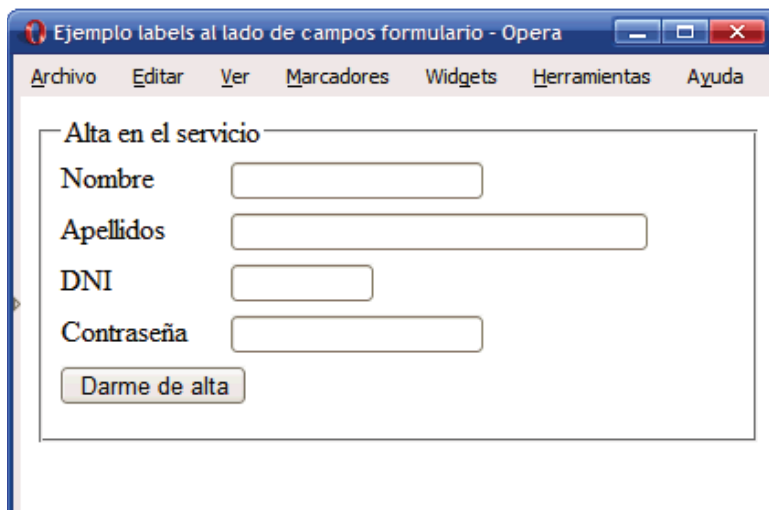


Figura 3.126. Mostrando las etiquetas label alineadas con los campos del formulario

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

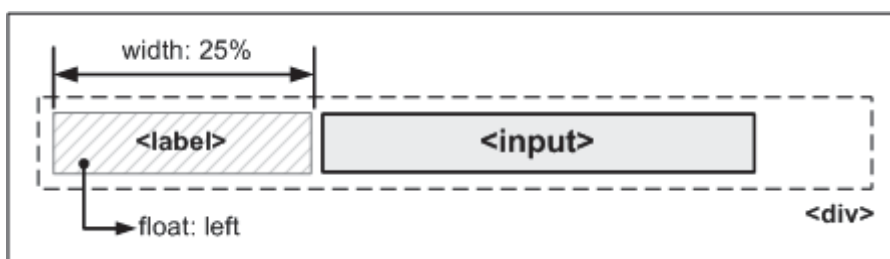


Figura 3.127. Esquema de la técnica de alineación de etiquetas label y campos de formulario

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

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

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

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



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

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

## 10.2. Estilos avanzados

### 10.2.1. Formulario en varias columnas

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

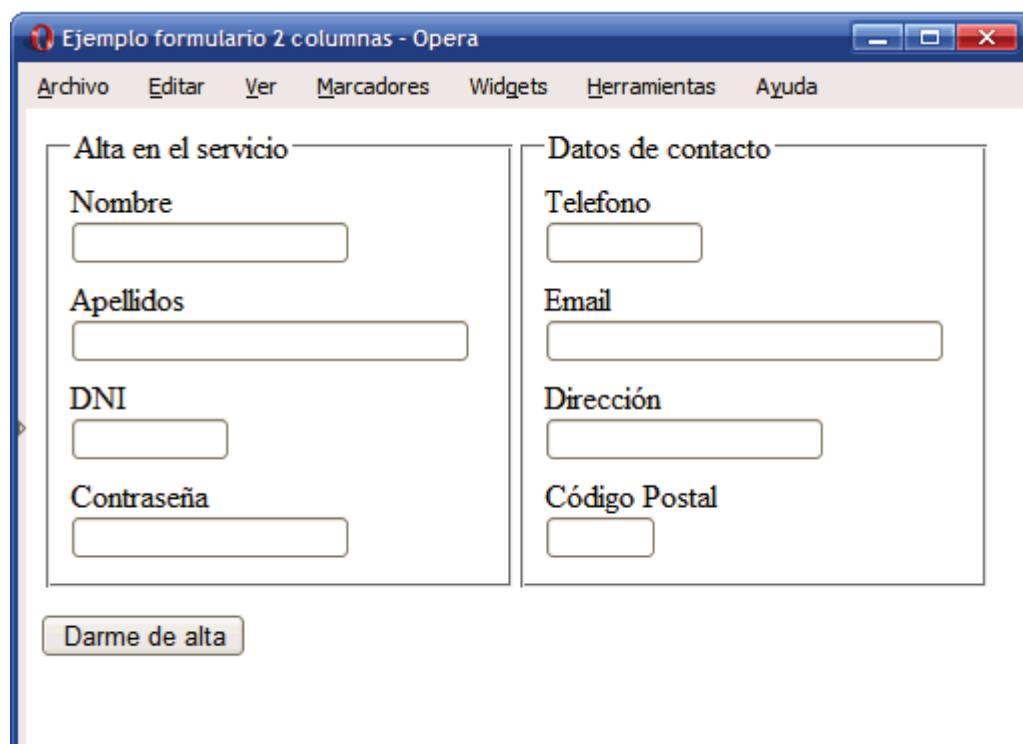


Figura 3.128. Ejemplo de formulario a dos columnas

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

```
form fieldset {  
    float: left;  
    width: 48%;  
}  
  
<form>  
    <fieldset>  
        ...  
    </fieldset>  
    ...  
</form>
```

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

### 10.2.2. Resaltar el campo seleccionado

Una de las mejoras más útiles para los formularios HTML consiste en resaltar de alguna forma especial el campo en el que el usuario está introduciendo datos. Para ello, CSS define la pseudo-clase `:focus`, vista ya anteriormente, 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:

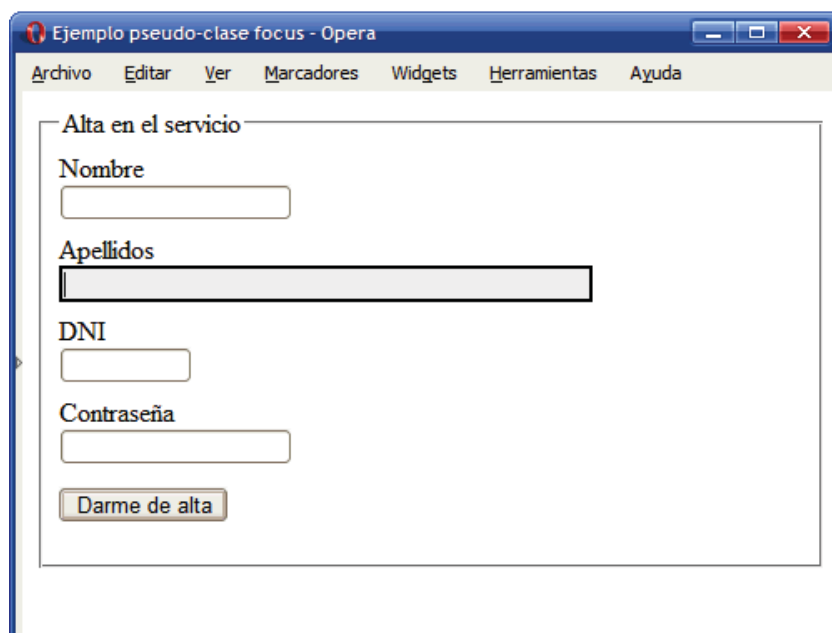
The image shows a screenshot of a web browser window titled "Ejemplo pseudo-clase focus - Opera". The browser's menu bar includes "Archivo", "Editar", "Ver", "Marcadores", "Widgets", "Herramientas", and "Ayuda". The main content area displays a form titled "Alta en el servicio". The form contains four text input fields labeled "Nombre", "Apellidos", "DNI", and "Contraseña", followed by a button labeled "Dar de alta". The "Apellidos" field is currently selected, indicated by a thick black border and a light gray background, which is the visual effect of the CSS :focus pseudo-class.

Figura 3.129. 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;  
}
```

## 10.3. CSS3 para los formularios

### 10.3.1. Redimensionar un campo

La nueva propiedad CSS3 `resize` permite autorizar a los usuarios para que puedan cambiar la dimensión de un campo de texto. Esta propiedad admite cinco valores:

- `none`: no se autoriza la redimensión.
- `both`: se autoriza la redimensión vertical y horizontal.
- `horizontal`: solamente se autoriza la redimensión horizontal.
- `vertical`: solamente se autoriza la redimensión vertical.
- `inherit`: hereda la propiedad del elemento padre.

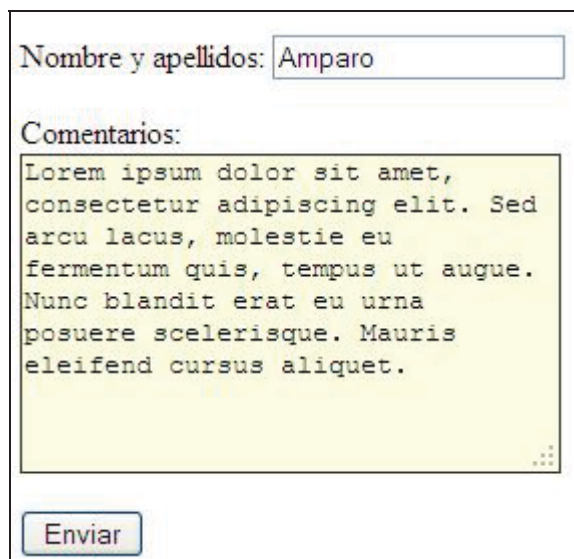
Veamos el formulario:

```
<form id="test" method="#" action="#" >  
  <p>  
    <label for="nombre">Nombre y apellidos:</label>  
    <input type="text" id="nombre" />  
  </p>  
  <p>  
    <label for="comentario">Comentarios:</label>  
    <br />  
    <textarea name="comentario" id="comentario"></textarea>  
  </p>  
  <p>  
    <input type="submit" name="enviar" id="enviar" value="Enviar" />  
  </p>  
</form>
```

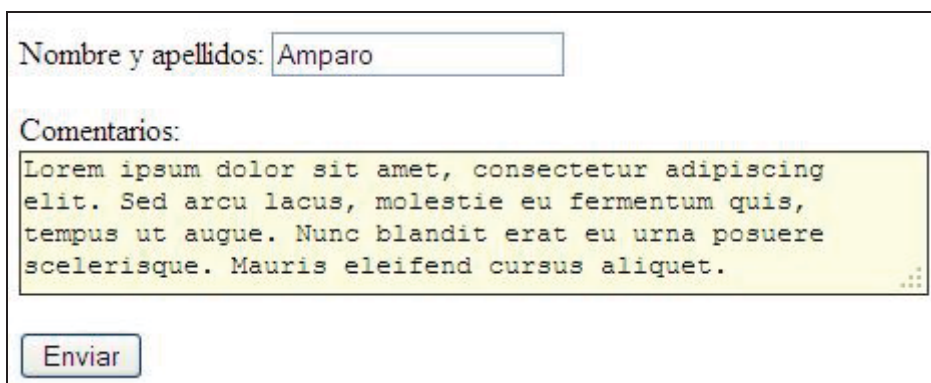
Veamos el estilo CSS para el elemento `<textarea>`:

```
#comentario {  
  width: 200px;  
  height: 90px;  
  border: 1px solid #333;  
  background-color: lightyellow;  
  resize: both;  
}
```

Así se visualiza con Firefox:



Se puede ajustar el tamaño tanto vertical como horizontalmente:



### 10.3.2. Las pseudo-clases para los formularios.

CSS3 introduce novedades muy interesantes con las pseudo-clases específicas para los elementos de los formularios. Vamos a poder aplicar estilos diferentes a los elementos activos de un formulario, los elementos deshabilitados y los elementos seleccionados. De este modo el usuario verá lo que ha hecho y qué elementos están habilitados o deshabilitados.

Veamos esas nuevas pseudo-clases `:enabled`, `:disabled` y `:checked`.

#### 1. El formulario y los estilos CSS

A continuación tenemos un formulario con dos campos de inserción de datos: el primer campo, que permite insertar el nombre, está habilitado (`enabled`); el segundo campo, que sirve para insertar la edad, está deshabilitado (`disabled`).

Luego tenemos dos botones de opción para el sexo (hombre o mujer).

Y, por último, una casilla de selección.

Vamos ahora a diferenciar visualmente los elementos del formulario en función del estado: habilitado, deshabilitado o seleccionado.

Este es el código HTML del formulario:

```
<form id="form1" method="#" action="#">
<p>
  <label for="nombre">Su nombre y apellidos: </label>
  <input type="text" id="nombre" />
</p>
<p>
  <label for="edad">Su edad: </label>
  <input type="text" id="edad" disabled="disabled"/>
</p>
<p>
  <input type="radio" name="sexo" id="hombre" value="hombre" />
  <label for="hombre">hombre</label><br/>
  <input type="radio" name="sexo" id="mujer" value="mujer" />
  <label for="mujer">mujer</label>
</p>
<p>
  <input type="checkbox" id="acepto" />
  <label for="acepto">Acepto las condiciones de uso.</label>
</p>
</form>
```

Y estos son los estilos CSS:

```
#acepto:checked+label {
  background-color: gold;
  font-weight: bold;
}
#hombre:checked+label, #mujer:checked+label {
  Font-style: italic;
}
:enabled {
  Background-color: lightgreen;
}
:disabled {
  Background-color: lightcoral;
}
```

El primer selector se dirige a la etiqueta (label) de la casilla de selección. (#acepto) y se aplicará cuando esta aparezca seleccionada (checked). Claro está, nos gustaría

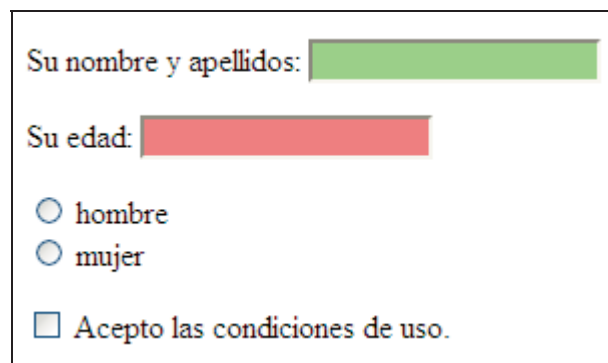
resaltar la etiqueta de la casilla, y no la casilla, por lo que hemos usado un selector adyacente.

El segundo selector se dirige al botón de opción seleccionado.

Los otros dos estilos añaden, simplemente, un fondo de color en función del estado del campo.

## 2. Resultado

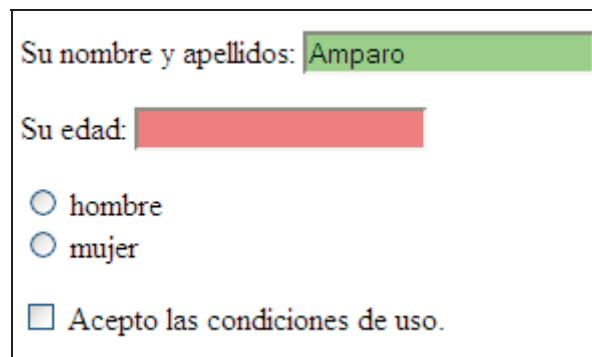
Así se visualizará el formulario cuando el usuario no haya realizado ninguna acción.



Formulario inicial:

- Su nombre y apellidos:
- Su edad:
- ☐ hombre
- ☐ mujer
- ☐ Acepto las condiciones de uso.

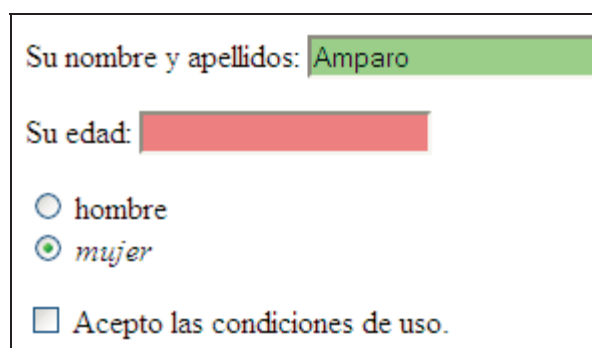
El usuario podrá introducir un valor en el primer campo activo (con fondo verde), pero no en el segundo, que se encuentra deshabilitado (fondo rojo):



Formulario con valor:

- Su nombre y apellidos:
- Su edad:
- ☐ hombre
- ☐ mujer
- ☐ Acepto las condiciones de uso.

El usuario ha seleccionado un botón de opción, así que su etiqueta aparece en cursiva:



Formulario con selección:

- Su nombre y apellidos:
- Su edad:
- ☐ hombre
- ☒ *mujer*
- ☐ Acepto las condiciones de uso.

El usuario ha activado la casilla de selección, así que la etiqueta aparece en negrita con un fondo dorado:

Su nombre y apellidos:

Su edad:

☐ hombre  
☒ *mujer*

☒ **Acepto las condiciones de uso.**

### 3. Otra pseudo-clase

Existe otro estado para los botones de opción y las casillas de selección, se trata del estado intermedio: `:indeterminate`. Este estado indica que el elemento no está ni seleccionado, ni no seleccionado.

### 4. Los campos obligatorios y facultativos

Puedes cambiar el diseño de los campos obligatorios y facultativos. Utiliza para ello las pseudo-classes `:required` y `:optional`.

Los campos obligatorios, que requieren que se haya insertado un valor, deben presentar el atributo booleano `required`. Los que no sean obligatorios, serán por defecto facultativos, sin que haga falta precisarlo.

Veamos un ejemplo de formulario:

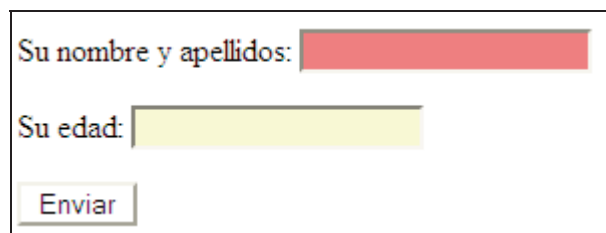
```
<form id="form1" method="#" action="#">
  <p>
    <label for="nombre">Su nombre y apellidos: </label>
    <input type="text" name="nombre" id="nombre" required />
  </p>
  <p>
    <label for="edad">Su edad: </label>
    <input type="text" name="edad" id="edad" />
  </p>
  <p>
    <input type="submit" id="enviar" value="Enviar" />
  </p>
</form>
```

Y estos son los estilos CSS:



```
input:required {  
    background-color: lightcoral;  
}  
input:optional {  
    background-color: lightgoldenrodyellow;  
}  
input#enviar {  
    background: none;  
}
```

Y el resultado visual obtenido:



Visualización de un formulario web. El campo "Su nombre y apellidos:" tiene un fondo rojo claro. El campo "Su edad:" tiene un fondo amarillo claro. Hay un botón "Enviar" sin fondo.

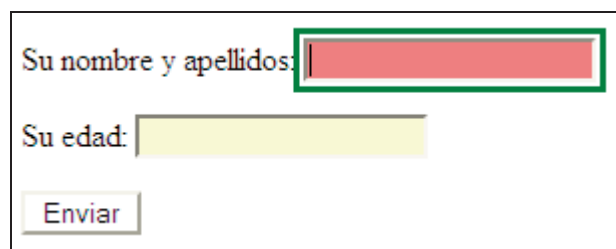
## 5. El formato de :focus

La pseudo-clase `:focus` permite resaltar el campo que se está usando en ese momento con un borde; `outline`. Los estilos CSS3 introducen una nueva propiedad, `outline-offset`, que corresponde a la distancia entre el límite de la caja y el límite del borde.

Veamos un ejemplo muy sencillo:

```
input:focus {  
    outline: solid 3px green;  
    outline-offset: 2px;  
}
```

Y el resultado visual:



Visualización del mismo formulario web, pero el campo "Su nombre y apellidos:" ahora tiene un borde verde grueso, indicando que está enfocado.

### 10.3.3. La validación de los datos introducidos

Ya vimos en el tema de HTML cómo podíamos hacer que un campo fuese obligatorio con el atributo `required` y cuál era la reacción por defecto de los navegadores. Ahora vamos a aprender a modificar el diseño de la validación de los formularios con los estilos CSS3.

#### 1. El formulario

Vamos a crear un formulario con tres campos y un botón de envío.

Este es el código HTML del formulario:

```
<form id="registro" method="#" action="#">
  <p>
    <label for="nombre">Su nombre: </label>
    <input type="text" id="nombre" required />
  </p>
  <p>
    <label for="email">Su e-mail: </label>
    <input type="email" id="email" required />
  </p>
  <p>
    <label for="edad">Su edad: </label>
    <input type="text" id="edad" pattern="\d{2}" />
  </p>
  <p>
    <input type="submit" id="enviar" value="Enviar" />
  </p>
</form>
```

Los dos primeros campos son obligatorios (`required`) y el último incluye una validación con una expresión regular. Se ha insertado un único botón de envío.

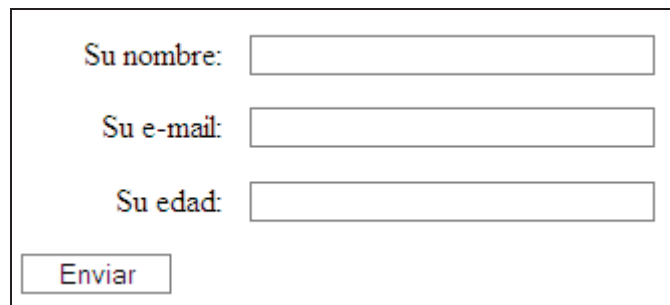
#### 2. El diseño del formulario

Estos son los estilos que se han usado para aplicar un diseño sencillo a los elementos del formulario:

```
input {
  border: solid 1px gray;
  width: 200px;
}
label, input {
  display: inline-block;
}
label {
  width: 100px;
  text-align: right;
```

```
margin-right: 10px;
}
input[type=submit] {
width: 75px;
background: none;
}
```

Este es el resultado visual en Firefox:



Su nombre:

Su e-mail:

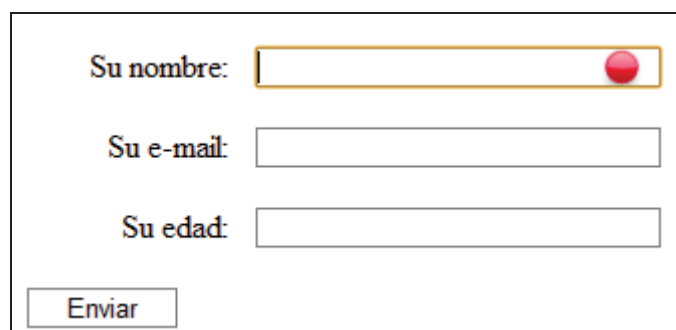
Su edad:

### 3. Los campos no válidos

A continuación tenemos un campo de tipo `email` y un campo con un `pattern`. La validación de los campos se realiza de forma continua, constantemente. Vamos a ver cómo funciona la validación cuando el campo está activo, con el elemento `:focus` y las pseudo-classes `:valid` y `:invalid`. Se ha establecido que aparezca un punto rojo a la derecha del campo cuando el valor no sea válido y un punto verde cuando dicho sea válido.

```
input:focus:invalid {
background: url(rojo.png) no-repeat 98% center;
}
input:focus:valid {
background: url(verde.png) no-repeat 98% center;
}
```

Cuando el usuario hace clic en el campo **Su nombre**, el valor no es válido aún, ya que por el momento no se ha insertado ningún valor. El campo está vacío y, claro, se trata de un campo obligatorio. Por ese motivo aparece el punto rojo.



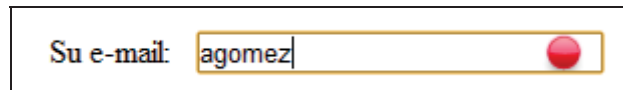
Su nombre:

Su e-mail:

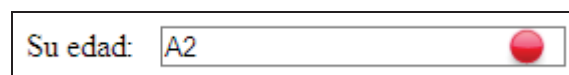
Su edad:

En cuanto el usuario comienza a insertar caracteres, el valor es validado y debería aparecer el punto verde.

La validación del e-mail sigue el mismo principio. Mientras el símbolo arroba no se encuentre entre los caracteres insertados, no se validará el valor del campo.



Teniendo en cuenta el pattern del campo de la edad, si se inserta un valor que no sean dos cifras, no se validará.



En la siguiente captura, todos los valores insertados son válidos.

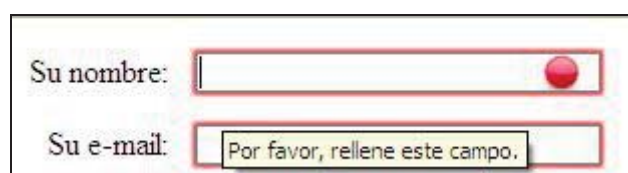


Por último, para evitar que el punto verde aparezca junto al botón de envío cuando se haga clic utilice la pseudo-clase `:not` para excluirlo mediante su código de identificación.

```
input:focus:valid:not(#enviar) {  
    background: url(verde.png) no-repeat 98% center;  
}
```

#### 4. Los mensajes de error

Como sabemos, los navegadores utilizan sus propios elementos de interfaz para mostrar los mensajes de error que aparecen cuando el valor insertado no es válido. Un ejemplo con Firefox:



Vamos ahora a personalizar ese mensaje de error.

En nuestro formulario vamos a añadir el elemento `<span>` inmediatamente después del elemento `<input>` del primer campo **Su nombre y apellidos**. El contenido de `<span>` es el texto que se mostrará cuando el valor introducido no sea válido.

```
<p>
  <label for="nombre">Su nombre y apellidos: </label>
  <input type="text" id="nombre" required />
  <span class="noValido">Este campo no puede dejarse
  vacío</span>
</p>
```

Habrás notado que `<span>` usa la clase `noValido`.

Veamos los estilos CSS que le hemos aplicado a la clase `noValido`:

```
span.noValido {
  display: none;
  position: relative;
  left: 250px;
  top: 5px;
  width: 230px;
  border: solid 1px red;
  background-color: lightgoldenrodyellow;
  padding: 3px;
  text-align: center;
}
```

La propiedad importante es la visualización, `display`, que se encuentra en `none`, para no mostrar el mensaje de error. Todo lo demás es la aplicación de las propiedades habituales.

Ahora tenemos que hacer que aparezca ese mensaje si se produce un error, cuando el campo esté activo.

Vamos a usar el selector:

```
input:focus:invalid + span.noValido {
  display: block;
}
```

- `input`: para un elemento de formulario de tipo campo de inserción de datos.
- `:focus`: esta pseudo-clase indica que el campo está activo.
- `:invalid`: esta pseudo-clase indica que el campo no es válido.

- `+` es el selector de adyacente directo que seleccionará al hermano inmediato de un elemento. En nuestro ejemplo, seleccionará el elemento inmediatamente después del campo `input`, es decir, el elemento `span` (como se puede comprobar en el código facilitado anteriormente).
- `span.noValido`: se aplica a los elementos `<span>` que utilizan la clase `noValido`.

Hemos usado la propiedad `display` en `block` para mostrar el mensaje con el diseño que acabamos de establecer.

Veamos cómo funciona el mensaje de advertencia.

El mensaje aparecerá cuando el usuario no haya introducido ningún dato en el campo **Su nombre y apellidos**.

Un formulario con el label "Su nombre y apellidos:" seguido de un campo de entrada vacío. Debajo del campo, un mensaje de error "Este campo no puede dejarse vacío" está resaltado en un recuadro amarillo con un borde rojo.

En cuanto se introduzca un carácter, el mensaje se ocultará.

Seguiremos el mismo principio para el campo de la dirección de e-mail:

```
<p>  
  <label for="email">Su e-mail: </label>  
  <input type="email" id="email" required />  
  <span class="noValido">La dirección de e-mail no es  
  correcta</span>  
</p>
```

El mensaje de error para la dirección de e-mail:

Un formulario con el label "Su e-mail:" seguido de un campo de entrada vacío. Debajo del campo, un mensaje de error "La dirección e-mail no es correcta" está resaltado en un recuadro amarillo con un borde rojo.

## 5. Los campos válidos

Ahora queremos indicarle a los usuarios que el valor que han insertado es válido. Vamos a tomar como ejemplo el campo e-mail.

Añadamos un elemento `<span>` para insertar el mensaje de validación.

```
<p>
  <label for="email">Su e-mail: </label>
  <input type="email" id="email" required />
  <span class="noValido">La dirección de e-mail no es
correcta</span>
  <span class="valido">La dirección e-mail es correcta</span>
</p>
```

Añadamos los estilos CSS para el formato del mensaje de validación:

```
span.valido {
  display: none;
  position: relative;
  left: 250px;
  top: 5px;
  width: 230px;
  border: solid 1px green;
  background-color: lightgreen;
  padding: 3px;
  text-align: center;
}
```

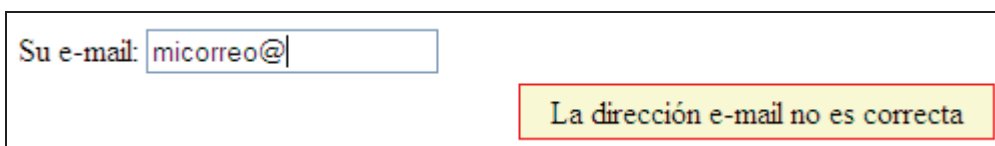
Ahora vamos a hacer que aparezca el mensaje de validación. En la sintaxis anterior habíamos utilizado el selector de adyacente directo: `+`, pero ahora tenemos dos `<span>` detrás de `<input>`. Así que esta vez vamos a usar el selector adyacente general `~`.

El estilo CSS modificado sería:

```
input:focus:invalid ~ span.noValido, input:focus:valid ~ span.valido {
  display: block;
}
```

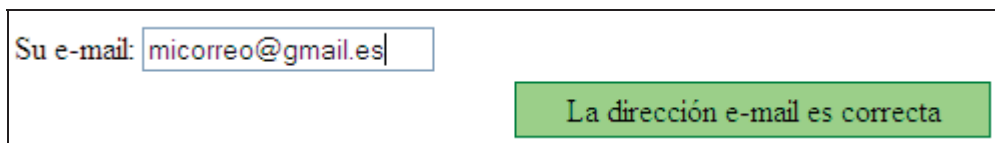
Veamos cómo funciona el campo de inserción de la dirección de e-mail:

Este campo, de momento, no es válido, así que se muestra el mensaje de error:



The screenshot shows a text input field with the label "Su e-mail:" and the text "micorre@" inside. To the right of the input field, there is a yellow rectangular box with a red border containing the text "La dirección e-mail no es correcta".

Ahora la dirección de e-mail es válida, así que se muestra el mensaje de validación:



The screenshot shows the same text input field, but now it contains the text "micorre@gmail.es". To the right of the input field, there is a green rectangular box with a green border containing the text "La dirección e-mail es correcta".



## 6. Los valores fuera de los límites

Disponemos de otras dos pseudo-clases para validar los números y las fechas. La pseudo-clase `:in-range` se aplica a los valores que se encuentran dentro de los límites definidos y la pseudo-clase `:out-of-range` se aplica a los valores que excedan de esos límites.

Veamos una aplicación sencilla con un formulario que contiene un campo de tipo number, con un límite inferior de 18 y un límite superior de 30:

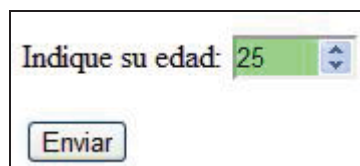
```
<form id="test" method="#" action="#">
<p>
  <label for="edad">Indique su edad: </label>
  <input type="number" min="18" max="30" step="2" value="25"
    id="edad" />
</p>
<p>
  <input type="submit" id="enviar" value="Enviar" />
</p>
</form>
```

Estos son los estilos CSS:

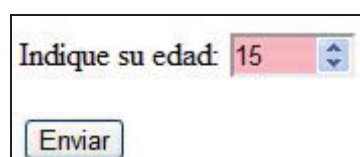
```
<style>
  input#edad:out-of-range {
    background-color: lightpink;
  }
  input#edad:in-range {
    background-color: lightgreen;
  }
</style>
```

Y este sería el resultado visual:

La edad indicada se encuentra dentro de los límites, así que se muestra un fondo verde:

Una captura de pantalla de un formulario web. Encabezado: 'Indique su edad:'. A la derecha, un campo de entrada de tipo 'number' con el valor '25'. El campo tiene un fondo verde claro. Debajo del campo, hay un botón 'Enviar'.

La edad indicada se encuentra fuera de los límites, así que se muestra un fondo rosa:

Una captura de pantalla de un formulario web. Encabezado: 'Indique su edad:'. A la derecha, un campo de entrada de tipo 'number' con el valor '15'. El campo tiene un fondo rosa claro. Debajo del campo, hay un botón 'Enviar'.

### 10.3.4. Crear botones con símbolos

En los formularios, puede que alguna vez necesitemos crear botones con símbolos o con pictogramas. Veamos cómo podemos crearlos fácilmente con las pseudos-clases `:before` y `:after`. Los ejemplos están adaptados del sitio web: <http://www.paulund.co.uk/css-buttons-with-icons-but-no-images>

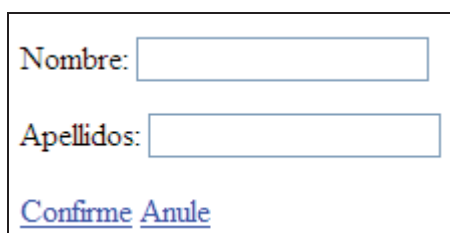
#### 1. El formulario

Vamos a crear un formulario clásico con dos botones, uno para validar el formulario y otro para anularlo. Sin embargo, no vamos a usar botones, sino vínculos, ya que la técnica que vamos a usar, con pseudos-elementos, no se puede aplicar a los botones.

Este es el formulario inicial:

```
<form id="registro" method="#" action="#">
  <p>
    <label for="nombre">Nombre: </label>
    <input type="text" name="nombre" id="nombre" />
  </p>
  <p>
    <label for="apellidos">Apellidos: </label>
    <input type="text" name="apellidos" id="apellidos" />
  </p>
  <p>
    <a href="#">Confirme</a>
    <a href="#">Anule</a>
  </p>
</form>
```

Así se visualizará sin estilos:



Nombre:

Apellidos:

[Confirme](#) [Anule](#)

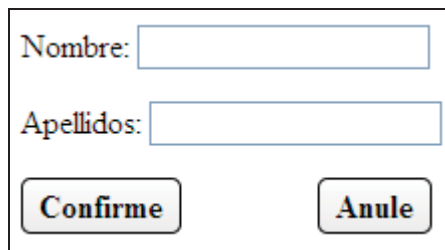
#### 2. Crear una clase para los botones

La técnica consiste en crear una clase para todos los botones que vayamos a usar en el formulario. Esta clase permitirá diseñar un botón con todos los elementos de formato habituales. Podemos usar las propiedades CSS porque los vínculos aparecen como bloque en línea: `display: inline-block`.

Con esta clase aplicaremos a los botones un ligero degradado y bordes redondeados.

```
.botones {  
  /*La caja*/  
  display: inline-block;  
  border: 1px solid #000;  
  padding: .2em .5em;  
  margin: 0 4em 0 0;  
  /*El degradado*/  
  background-image: -moz-linear-gradient(top, #fff 0%, #eee 100%);  
  background-image: -webkit-linear-gradient(top, #fff 0%, #eee 100%);  
  background-image: -o-linear-gradient(top, #fff 0%, #eee 100%);  
  background-image: linear-gradient(top, #fff 0%, #eee 100%);  
  /*Los bordes redondeados*/  
  -moz-border-radius: .3em;  
  -webkit-border-radius: .3em;  
  border-radius: .3em;  
  /*El texto*/  
  font-weight: bold;  
  font-size: 1em;  
  text-decoration: none;  
  color: black;  
}
```

Así se visualizará el formulario con los estilos:



Nombre:

Apellidos:

### 3. El pseudo-elemento ::before

Vamos a utilizar un pseudo-elemento que ya conocíamos. El pseudo-elemento `::before` (fíjate en la nueva sintaxis de los pseudos-elementos con los dos caracteres `::`) permite añadir texto delante del elemento al que está asociado. El pseudo-elemento `::after` hace lo mismo, pero detrás de dicho elemento.

Recordemos que podemos utilizar la sintaxis `:before` y `:after` si tenemos problemas de compatibilidad con el navegador. De hecho utilizaremos esta sintaxis.

Vamos a aplicar el pseudo-elemento `:before` a la clase genérica de los botones `.botones`, para que el estilo sea más sencillo.

```
.botones:before {  
  font-size: 1em;  
  padding: 0 .5em 0 0;  
}
```

#### 4. El contenido del pseudo-elemento ::before

Ahora tendremos que indicar el contenido que vamos a añadir delante de la etiqueta del botón, el texto del vínculo [a](#). debemos especificar a qué elemento se le añadirá ese contenido, ya que cada botón tendrá un contenido diferente.

En nuestro ejemplo, vamos a crear dos selectores específicos `.validar:before` y `.anular:before`, con un estilo específico para cada botón.

Segundo parámetro: se trata del pictograma que vamos a insertar. La norma Unicode UTF-8 nos propone cientos de símbolos y de pictogramas (<http://utf8-chartable.de/unicode-utf8-table.pl?start=9728&number=1024&names=-&utf8=string-literal>)

También se pueden utilizar las entidades de caracteres HTML clásicas: <http://www.w3.org/TR/html4/sgml/entities.html>

Para la validación del formulario, queremos insertar el “símbolo de validación”, que tiene el código de visualización `\2714`.

Para la anulación del formulario, queremos usar el símbolo de la “cruz de eliminación”, que tiene el código de visualización `\2717`.

Estos son los dos estilos:

```
.validar:before {  
    content: "\2714";  
}  
.anular:before {  
    content: "\2717";  
}
```

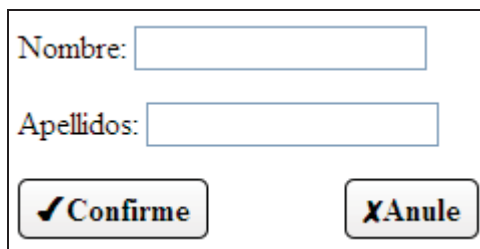
#### 5. Aplicar las clases

En la última etapa, ahora vamos a aplicar las clases que acabamos de crear a los elementos [a](#) de los vínculos del formulario. Los dos vínculos utilizan la clase genérica de los botones `botones` y cada uno de ellos utilizará la clase que le corresponda para la inserción del contenido `validar` y `anular`.

```
<a href="#" class="botones validar">Confirme</a>  
<a href="#" class="botones anular">Anule</a>
```

#### 6. La visualización del formulario

Así se visualizará el formulario:



Nombre:

Apellidos:

### 10.3.5. Los generadores de botones en línea

Los estilos CSS3 nos aportan mejoras reales en cuanto a la creación de botones, hasta tal punto que ya no es necesario usar programas de diseño para crearlos. Los botones con imágenes sólo presentan inconvenientes: aumentan considerablemente el tamaño de los archivos, las actualizaciones del diseño son más lentas, afectan negativamente al posicionamiento natural y la accesibilidad.

Para crear botones con un diseño moderno rápidamente, la web nos ofrece multitud de generadores en línea. Veamos algunos de ellos:

#### 1. CSS3 Button

Este sitio web (<http://css3button.net/>) tiene una interfaz agradable en la que podremos trabajar con:

- el degradado de fondo,
- los bordes y las esquinas redondeadas,
- el espaciamiento,
- las sombras externas e internas,
- el color del texto,
- las sombras del texto.

La visualización se actualizará automáticamente y podrás recuperar el código en la zona **CSS CODE**.

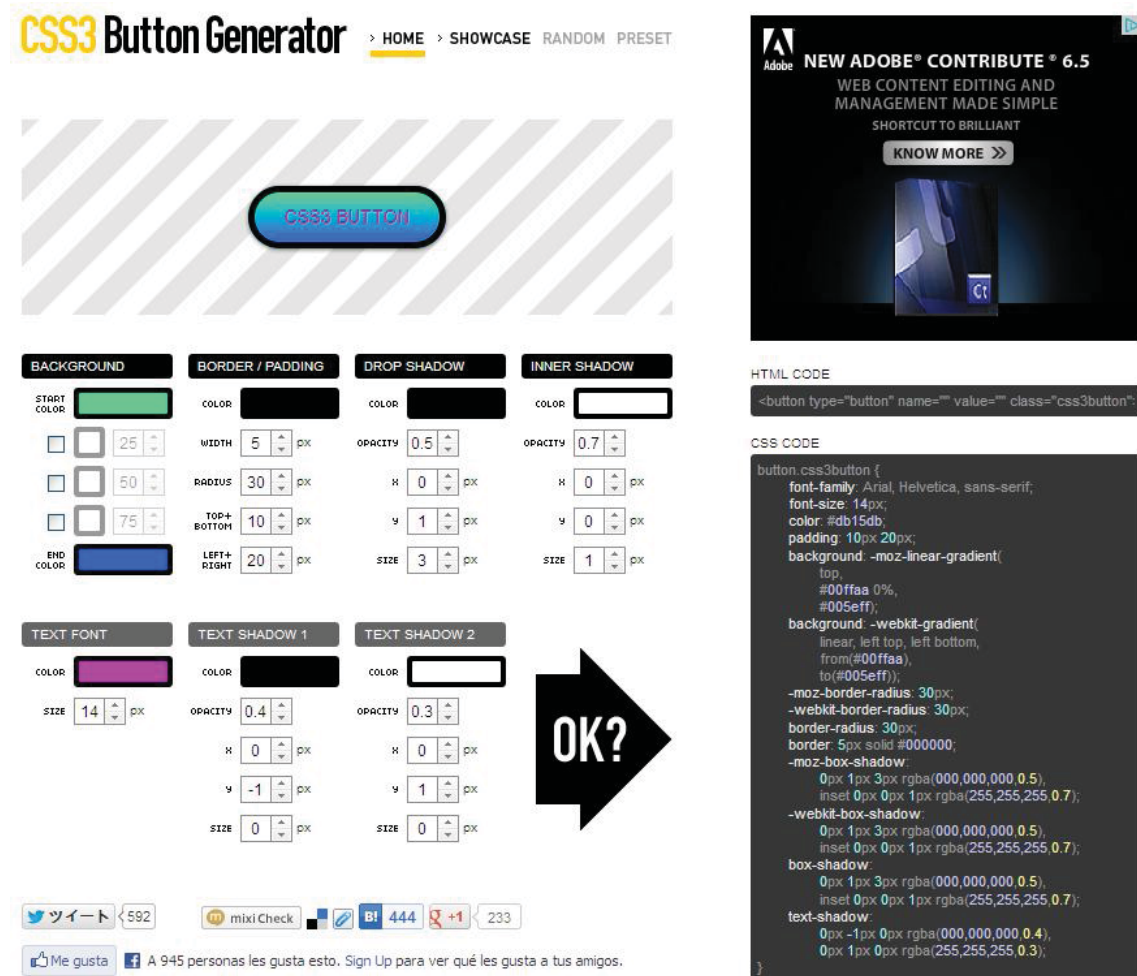


Figura 3.130. Ejemplo de botón generado en la web CSS3 Button

## 2. CSS3 Button Generator

Este sitio web (<http://css3buttongenerator.com/>) presenta una interfaz en acordeón en la que podrás seleccionar las propiedades CSS que desees aplicar:

- el texto sombreado,
- el espaciado y las sombras,
- los bordes y las esquinas redondeadas,
- el degradado del fondo,
- el degradado del fondo para `:hover`.

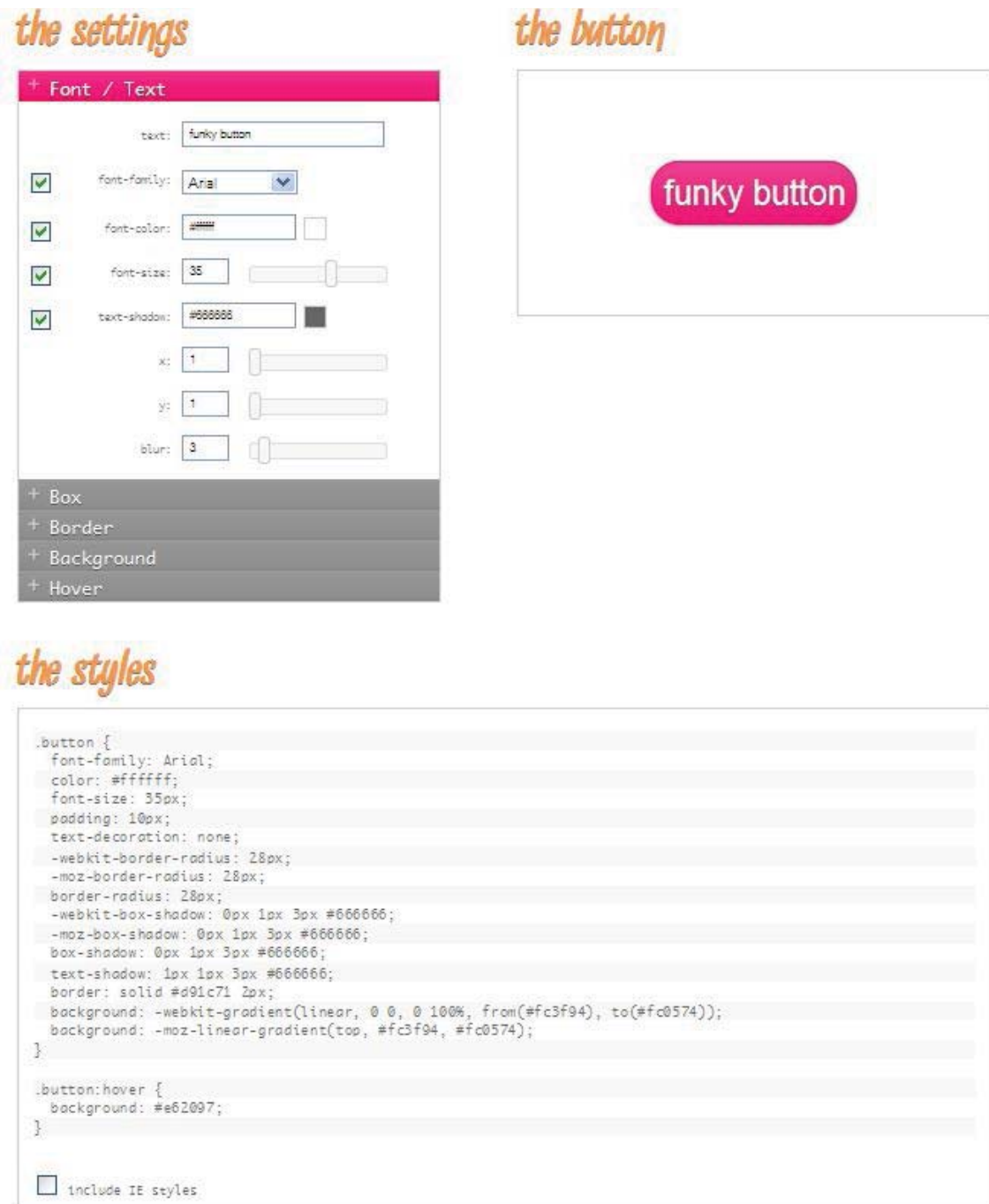


Figura 3.131. Ejemplo de botón generado en la web CSS3ButtonGenerator

### 3. CSS3 Button Generator

Este sitio web (<http://cssdrive.com/css3button>) propone crear botones de una forma muy completa. Podrás trabajar con:

- todos los parámetros del texto,
- la caja del botón: degradado para el fondo, ancho, espaciamiento...



- los bordes, las esquinas redondeadas y la sombra,
- las transformaciones, las transiciones y los estilos para `:hover`.

Nuevamente, podrás ver el resultado en tiempo real, así como el código CSS generado.



#### CSS Code

Figura 3.132. Ejemplo de botón generado en la web CSSDrive

#### 4. CSS3 Button Creator

El sitio web (<http://cssbuttongenerator.com/>) propone una interfaz innovadora y muy interactiva.

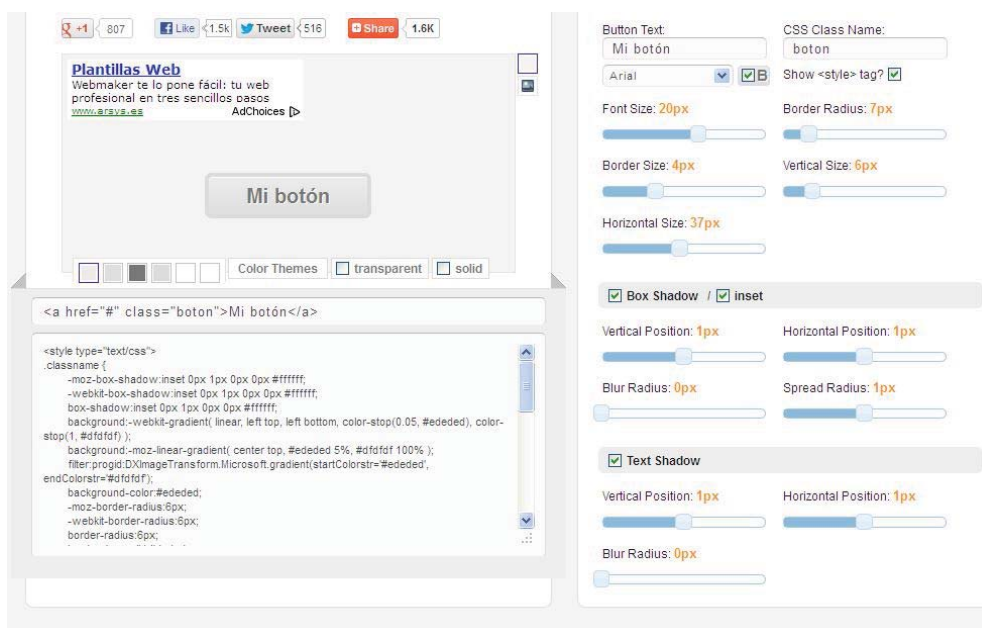


Figura 3.133. Ejemplo de botón generado en la web cssbuttongenerator

## 11. ESTRUCTURA DE LAS PÁGINAS WEB

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. Por este motivo, hace varios años la estructura de las páginas HTML se definía mediante tablas.

El desarrollo de CSS ha permitido que se puedan realizar los mismos diseños sin utilizar tablas HTML. Las principales ventajas de diseñar la estructura de las páginas web con CSS en vez de con tablas HTML son las siguientes:

- **Mantenimiento:** una página diseñada exclusivamente con CSS es mucho más fácil de mantener que una página diseñada con tablas. Cambiar el aspecto de una página creada con CSS es tan fácil como modificar unas pocas reglas en las hojas de estilos. Sin embargo, realizar la misma modificación en una página creada con tablas supone un esfuerzo muy superior y es más probable cometer errores.
- **Accesibilidad:** las páginas creadas con CSS son más accesibles que las páginas diseñadas con tablas. De hecho, los navegadores que utilizan las personas discapacitadas (en especial las personas invidentes) pueden tener dificultades con la estructura de las páginas complejas creadas con tablas HTML. No obstante, diseñar una página web exclusivamente con CSS no garantiza que la página sea accesible.
- **Velocidad de carga:** el código HTML de una página diseñada con tablas es mucho mayor que el código de la misma página diseñada exclusivamente con CSS, por lo que tarda más tiempo en descargarse. En cualquier caso, si el usuario accede al sitio con una conexión de banda ancha y la página es de un tamaño medio o reducido, las diferencias son casi imperceptibles.
- **Semántica:** aunque resulta obvio, las tablas HTML sólo se deben utilizar para mostrar datos cuya información sólo se entiende en forma de filas y columnas. Utilizar tablas para crear la estructura completa de una página es tan absurdo como utilizar por ejemplo la etiqueta `<ul>` para crear párrafos de texto.

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

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

## 11.1. 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 3.134. Página de anchura fija centrada mediante CSS



Figura 3.135. Página de anchura fija centrada mediante CSS



Figura 3.136. 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 3.137. Página de anchura variable centrada mediante CSS



Figura 3.138. Página de anchura variable centrada mediante CSS



Figura 3.139. Página de anchura variable centrada mediante CSS

## 11.2. 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.

Aunque centrar una página web horizontalmente es muy sencillo, centrarla verticalmente es mucho más complicado. Afortunadamente, no es muy común que una página web aparezca centrada de forma vertical. El motivo es que la mayoría de páginas web son más altas que la ventana del navegador, por lo que no es posible centrarlas verticalmente.

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.

La solución correcta para centrar verticalmente una página web se basa en el posicionamiento absoluto e implica realizar un cálculo matemático sencillo. A

continuación se muestra el esquema gráfico de los cuatro pasos necesarios para centrar una página web en la ventana del navegador:

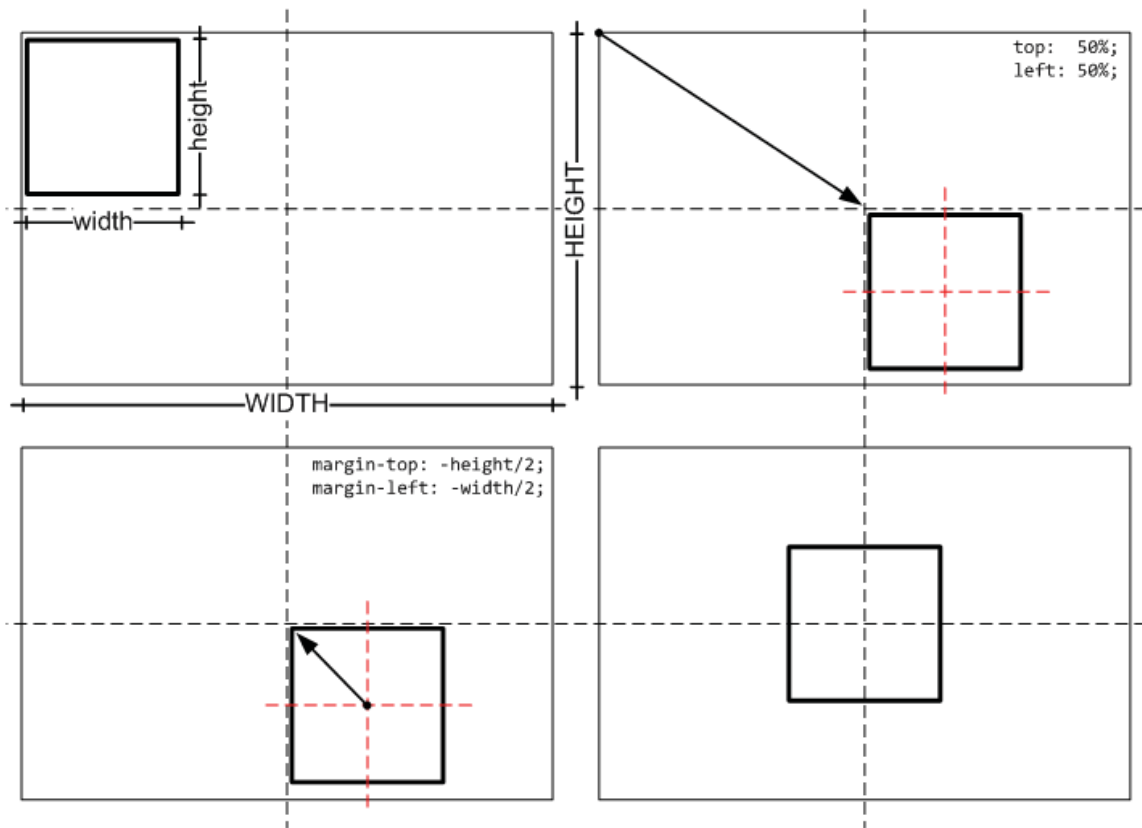


Figura 3.140. Pasos necesarios para centrar verticalmente una página web

En primer lugar, se asigna una altura y una anchura al elemento que encierra todos los contenidos de la página. En la primera figura, los contenidos de la página tienen una anchura llamada `width` y una altura llamada `height` que son menores que la anchura y altura de la ventana del navegador. En el siguiente ejemplo, se supone que tanto la anchura como la altura de la página es igual a `500px`:

```
#contenedor {  
  width: 500px;  
  height: 500px;  
}  
  
<body>  
  <div id="contenedor">  
    <h1>Lorem ipsum dolor sit amet</h1>  
    ...  
  </div>  
</body>
```



A continuación, se posiciona de forma absoluta el elemento `contenedor` y se asigna un valor de `50%` tanto a la propiedad `top` como a la propiedad `left`. El resultado es que la esquina superior izquierda del elemento `contenedor` se posiciona en el centro de la ventana del navegador:

```
#contenedor {  
  width: 500px;  
  height: 500px;  
  
  position: absolute;  
  top: 50%;  
  left: 50%;  
}
```

Si la página se debe mostrar en el centro de la ventana del navegador, es necesario desplazar hacia arriba y hacia la izquierda los contenidos de la página web. Para determinar el desplazamiento necesario, se realiza un cálculo matemático sencillo. Como se ve en la tercera figura del esquema anterior, el punto central de la página debe desplazarse hasta el centro de la ventana del navegador.

Como se desprende de la imagen anterior, la página web debe moverse hacia arriba una cantidad igual a la mitad de su altura y debe desplazarse hacia la izquierda una cantidad equivalente a la mitad de su anchura. Utilizando las propiedades `margin-top` y `margin-left` con valores negativos, la página se desplaza hasta el centro de la ventana del navegador.

```
#contenedor {  
  width: 500px;  
  height: 500px;  
  
  position: absolute;  
  top: 50%;  
  left: 50%;  
  
  margin-top: -250px; /* height/2 = 500px / 2 */  
  margin-left: -250px; /* width/2 = 500px / 2 */  
}
```

Con las reglas CSS anteriores, la página web siempre aparece centrada verticalmente y horizontalmente respecto de la ventana del navegador. El motivo es que la anchura/altura de la página son fijas (propiedades `width` y `height`), el desplazamiento necesario para centrarla también es fijo (propiedades `margin-top` y `margin-left`) y el desplazamiento hasta el centro de la ventana del navegador se calcula dinámicamente gracias al uso de porcentajes en las propiedades `top` y `left`.

Para centrar una página sólo verticalmente, se debe prescindir tanto del posicionamiento horizontal como del desplazamiento horizontal:

```
#contenedor {  
  width: 500px;  
  height: 500px;  
  
  position: absolute;  
  top: 50%;  
  
  margin-top: -250px;    /* height/2 = 500px / 2 */  
}
```

## 11.3. Estructura o layout

### 11.3.1. Diseño a 2 columnas con cabecera y pie de página

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

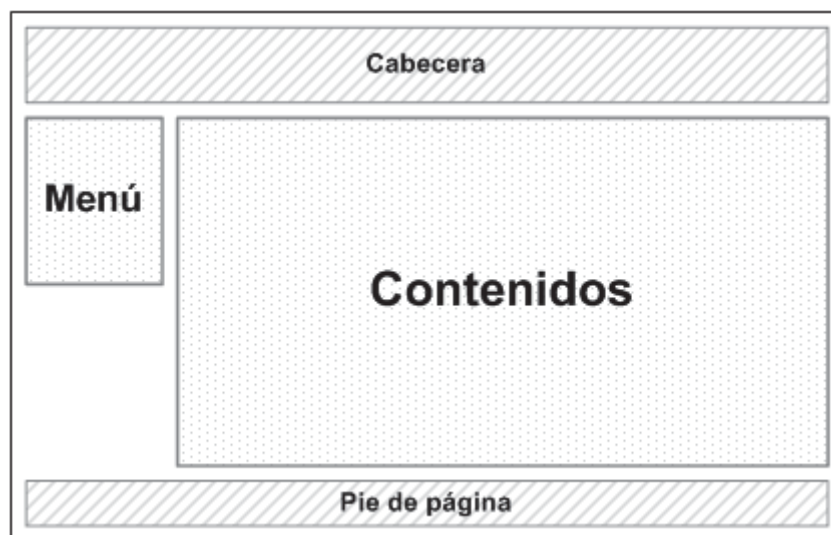


Figura 3.141. Esquema del diseño a 2 columnas con cabecera y pie de página

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

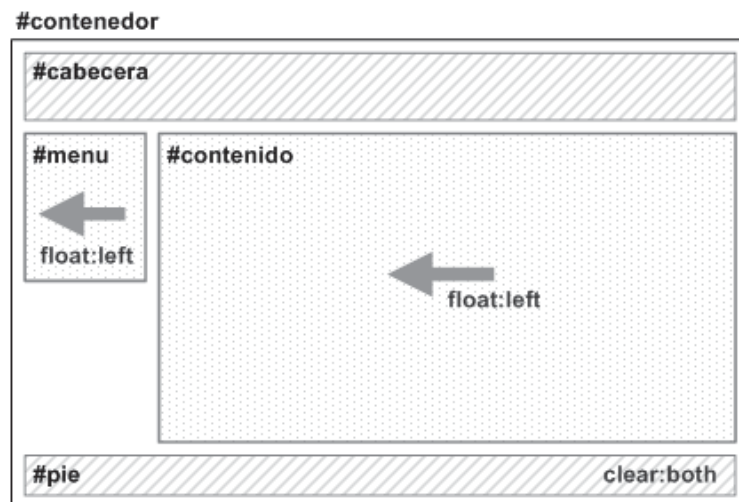


Figura 3.142. Propiedades CSS necesarias en el diseño a dos columnas con cabecera y pie de página

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

```
#contenedor {  
    width: 700px;  
}  
#cabecera {  
}  
#menu {  
    float: left;  
    width: 150px;  
}  
#contenido {  
    float: left;  
    width: 550px;  
}  
#pie {  
    clear: both;  
}  
  
<body>  
<div id="contenedor">  
    <div id="cabecera">  
    </div>  
  
    <div id="menu">  
    </div>  
  
    <div id="contenido">  
    </div>  
  
    <div id="pie">  
    </div>  
</div>  
</body>
```

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

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

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

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

### 11.3.2. Diseño a 3 columnas con cabecera y pie de página

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

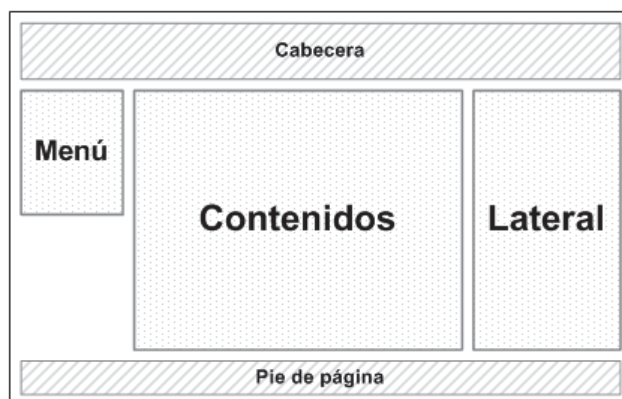


Figura 3.143. Esquema del diseño a tres columnas con cabecera y pie de página

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

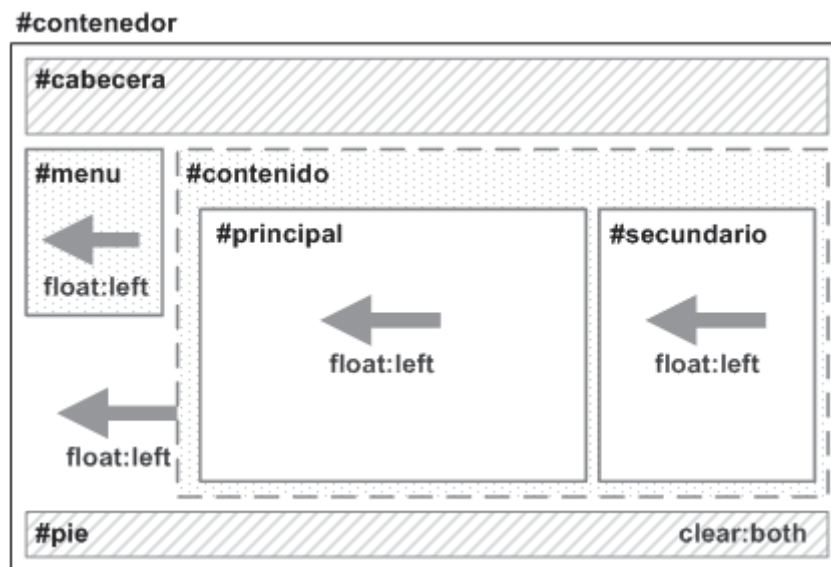


Figura 3.144. Propiedades CSS necesarias en el diseño a 3 columnas con cabecera y pie de página

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

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

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

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

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

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

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

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

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

## 11.4. 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`.

<b>max-width</b>	Anchura máxima
<b>Valores</b>	<code>&lt;medida&gt;</code>   <code>&lt;porcentaje&gt;</code>   <code>none</code>   <code>inherit</code>
<b>Se aplica a</b>	Todos los elementos salvo filas y grupos de filas de tablas
<b>Valor inicial</b>	<code>none</code>
<b>Descripción</b>	Permite definir la anchura máxima de un elemento
<b>min-width</b>	Anchura mínima
<b>Valores</b>	<code>&lt;medida&gt;</code>   <code>&lt;porcentaje&gt;</code>   <code>inherit</code>
<b>Se aplica a</b>	Todos los elementos salvo filas y grupos de filas de tablas
<b>Valor inicial</b>	<code>0</code>
<b>Descripción</b>	Permite definir la anchura mínima de un elemento
<b>max-height</b>	Altura máxima
<b>Valores</b>	<code>&lt;medida&gt;</code>   <code>&lt;porcentaje&gt;</code>   <code>none</code>   <code>inherit</code>
<b>Se aplica a</b>	Todos los elementos salvo columnas y grupos de columnas de tablas
<b>Valor inicial</b>	<code>none</code>
<b>Descripción</b>	Permite definir la altura máxima de un elemento

<b>min-height</b>	Altura mínima
<b>Valores</b>	<medida>   <porcentaje>   inherit
<b>Se aplica a</b>	Todos los elementos salvo columnas y grupos de columnas de tablas
<b>Valor inicial</b>	0
<b>Descripción</b>	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**.

## 11.5. Estilos avanzados

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

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

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



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

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

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

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

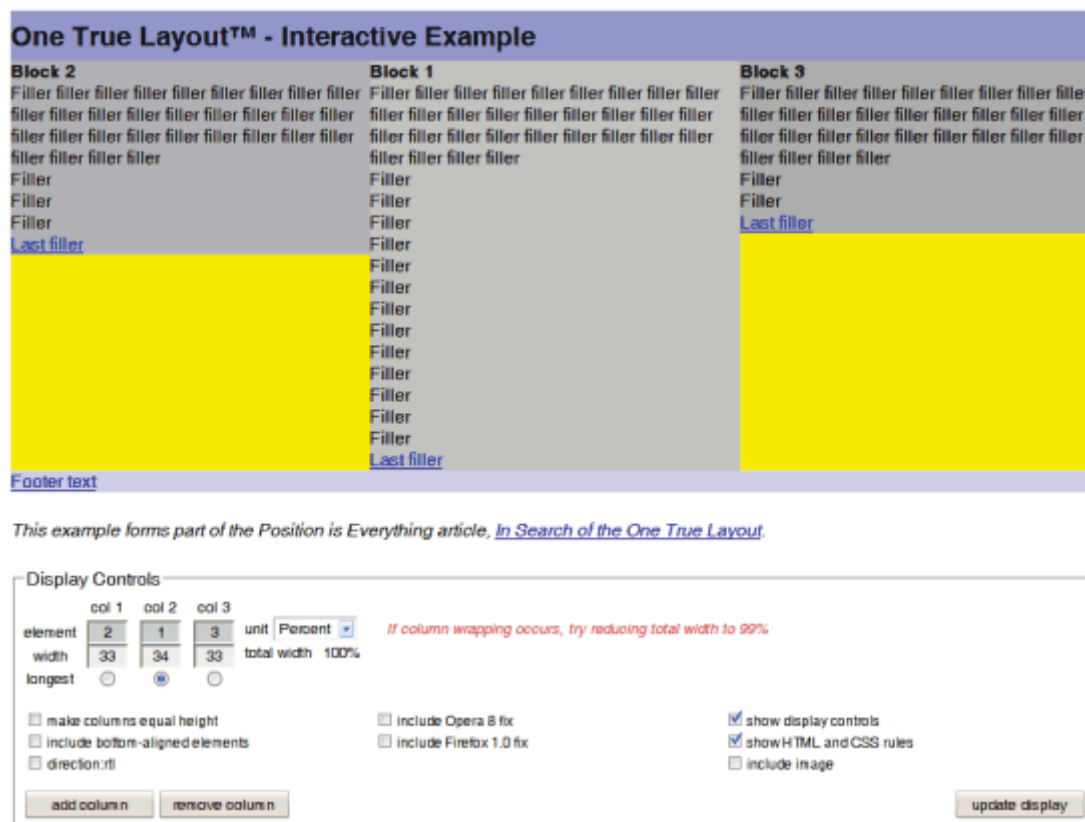


Figura 3.145. Herramienta online para diseñar layouts de varias columnas con CSS

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

## 12. OTROS

### 12.1. Propiedades shorthand

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

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

<b>font</b>	Tipografía
<b>Valores</b>	<code>( ( &lt;font-style&gt;    &lt;font-variant&gt;    &lt;font-weight&gt; )? &lt;font-size&gt; ( / &lt;line-height&gt; )? &lt;font-family&gt; )   caption   icon   menu   message-box   small-caption   status-bar   inherit</code>
<b>Se aplica a</b>	Todos los elementos
<b>Valor inicial</b>	-
<b>Descripción</b>	Permite indicar de forma directa todas las propiedades de la tipografía de un texto
<b>margin</b>	Margen
<b>Valores</b>	<code>( &lt;medida&gt;   &lt;porcentaje&gt;   auto ) {1, 4}   inherit</code>
<b>Se aplica a</b>	Todos los elementos salvo algunos casos especiales de elementos mostrados como tablas
<b>Valor inicial</b>	-
<b>Descripción</b>	Establece de forma directa todos los márgenes de un elemento

<b>padding</b>	Relleno
<b>Valores</b>	( <medida>   <porcentaje> ){1, 4}   inherit
<b>Se aplica a</b>	Todos los elementos excepto algunos elementos de tablas como grupos de cabeceras y grupos de pies de tabla
<b>Valor inicial</b>	-
<b>Descripción</b>	Establece de forma directa todos los rellenos de los elementos
<b>border</b>	Estilo completo de todos los bordes
<b>Valores</b>	( <medida_borde>    <color_borde>    <estilo_borde> )   inherit
<b>Se aplica a</b>	Todos los elementos
<b>Valor inicial</b>	-
<b>Descripción</b>	Establece el estilo completo de todos los bordes de los elementos
<b>background</b>	Fondo de un elemento
<b>Valores</b>	( <background-color>    <background-image>    <background-repeat>    <background-attachment>    <background-position> )   inherit
<b>Se aplica a</b>	Todos los elementos
<b>Valor inicial</b>	-
<b>Descripción</b>	Establece todas las propiedades del fondo de un elemento

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

## 12.2. Versión para imprimir

La mayoría de sitios web de calidad ofrecen al usuario la posibilidad de imprimir sus contenidos mediante una versión específica para impresora de cada página.

Estas versiones optimizadas para impresora eliminan los contenidos superfluos, modifican o eliminan las imágenes y colores de fondo y sobre todo, optimizan los contenidos de texto para facilitar su lectura.

CSS simplifica al máximo la creación de una versión para imprimir gracias al concepto de los medios CSS. Como se sabe, los estilos CSS que se aplican a los contenidos pueden variar en función del medio a través del que se acceden (pantalla, televisor, móvil, impresora, etc.)

De esta forma, realizar una versión para imprimir de una página HTML es tan sencillo como crear unas cuantas reglas CSS que optimicen sus contenidos para conseguir la mejor impresión.

El sitio web <http://www.alistapart.com/> es un excelente ejemplo de cómo los sitios web de calidad crean versiones específicas para impresora de las páginas web originales. Cuando se visualiza un artículo de ese sitio web en una pantalla normal, su aspecto es el siguiente:

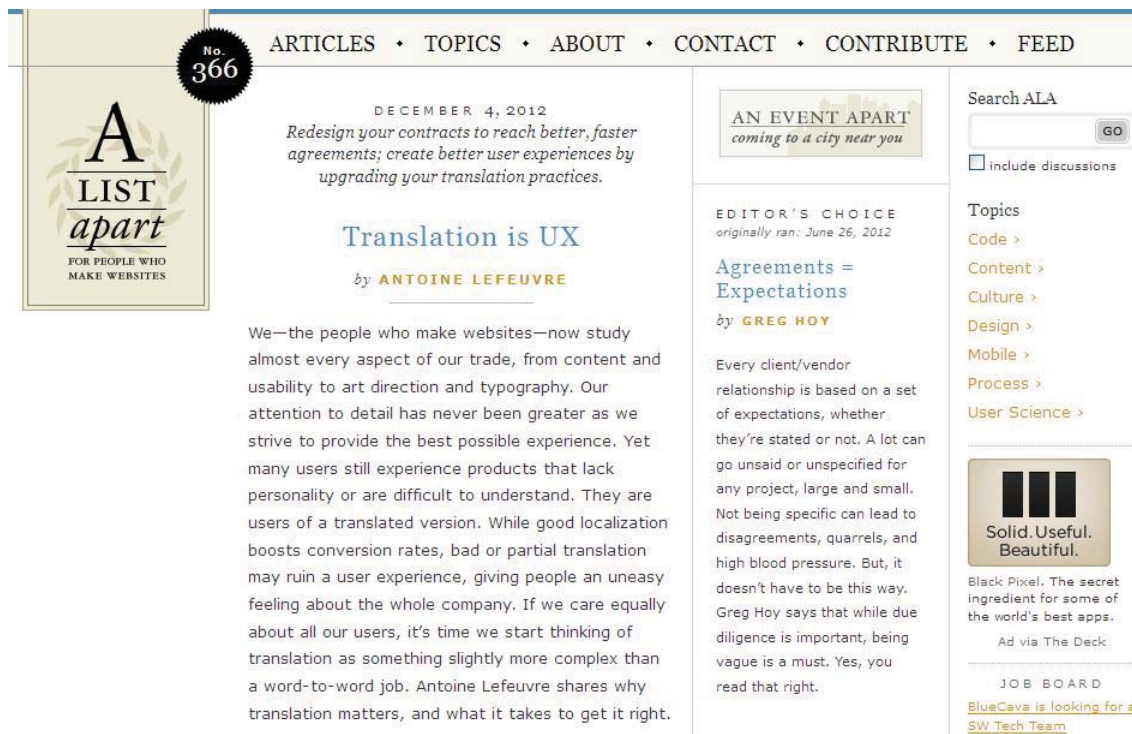


Figura 3.146. Aspecto de un artículo de A List Apart como se ve en la pantalla

Además de sus contenidos, las páginas de los artículos muestran el logotipo del sitio, el menú principal de navegación y una barra lateral con varias utilidades como un buscador.

Sin embargo, cuando se imprime la página del mismo artículo, su aspecto es el que muestra la siguiente imagen:



Figura 3.147. Aspecto de un artículo de A List Apart como se ve cuando se imprime

La página impresa elimina todos los contenidos superfluos como los menús de navegación, el buscador y el formulario para añadir comentarios en el artículo. Además, modifica la estructura de la página para que la zona de contenidos ocupe toda la anchura de la página y el espacio se aproveche mejor.

Crear una versión para imprimir similar a la mostrada en el ejemplo anterior es una tarea que no lleva más de unos pocos minutos.

Las reglas CSS de la versión para imprimir se aplican solamente al medio `print`. Por lo tanto, en primer lugar se crea una nueva hoja de estilos y al enlazarla se especifica que sólo debe aplicarse en las impresoras:

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

Normalmente, las hojas de estilos para la pantalla se aplican a todos los medios (por utilizar el valor `media="all"` al enlazarla o por no indicar ningún valor en el atributo `media`). Por este motivo, cuando se imprime una página se aplican los mismos estilos que se aplican al visualizarla en la pantalla.

Aprovechando este comportamiento, las hojas de estilos para impresoras son muy sencillas, ya que sólo deben modificar algunos estilos aplicados en el resto de hojas de estilos.

Por este motivo, normalmente las hojas de estilos para impresora se construyen siguiendo los pasos que se muestran a continuación:

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

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

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

2) Corregir la estructura de la página:

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

Normalmente, las páginas web complejas están formadas por varias columnas posicionadas mediante la propiedad `float`. Si al imprimir la página se eliminan las columnas laterales, es conveniente reajustar la anchura y el posicionamiento de la zona de contenidos y de otras zonas que sí se van a imprimir.

3) Modificar los colores y tipos de letra:

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

Aunque el uso de impresoras en color es mayoritario, suele ser conveniente imprimir todo el texto de las páginas de color negro, para ahorrar costes y para aumentar el contraste cuando se imprime sobre hojas de color blanco. También suele ser conveniente modificar el tipo de letra y escoger uno que facilite al máximo la lectura del texto.

### 12.2.1. Imprimiendo los enlaces

Uno de los principales problemas de las páginas HTML impresas es la pérdida de toda la información relativa a los enlaces. En principio, imprimir los enlaces de una página es absurdo porque no se pueden utilizar en el medio impreso.

Sin embargo, lo que puede ser realmente útil es mostrar al lado de un enlace la dirección a la que apunta. De esta forma, al imprimir la página no se pierde la información relativa a los enlaces.

CSS incluye una propiedad llamada `content` que permite crear nuevos contenidos de texto para añadirlos a la página HTML. Si se combina la propiedad `content` y el *pseudo-elemento* `:after`, es posible insertar de forma dinámica la dirección a la que apunta un enlace justo después de su texto:

```
a:after {  
  content: " (" attr(href) ") ";  
}
```

El código CSS anterior añade después de cada enlace de la página un texto formado por la dirección a la que apunta el enlace mostrada entre paréntesis. Si se quiere añadir las direcciones antes de cada enlace, se puede utilizar el *pseudo-elemento* `:before`:

```
a:before {  
  content: " (" attr(href) ") ";  
}
```

## 12.3. Personalizar el cursor

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

<b>cursor</b>	Puntero del ratón
<b>Valores</b>	<code>( (&lt;url&gt; ,)* ( auto   crosshair   default   pointer   move   e-resize   ne-resize   nw-resize   n-resize   se-resize   sw-resize   s-resize   w-resize   text   wait   help   progress ) )   inherit</code>
<b>Se aplica a</b>	Todos los elementos
<b>Valor inicial</b>	auto
<b>Descripción</b>	Permite personalizar el puntero del ratón

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

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











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












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





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

Los valores preestablecidos para el puntero se muestran a continuación:



Puntero	Navegadores que lo soportan
 cursor: default	Todos
 cursor: crosshair	Todos
 cursor: hand	Solo Internet Explorer
 cursor: pointer	Todos salvo Internet Explorer
 cursor:pointer; cursor: hand	Todos
 cursor: move	Todos
 cursor: text	Todos
 cursor: wait	Todos
 cursor: help	Todos
 cursor: n-resize	Todos

 cursor: ne-resize	Todos
 cursor: e-resize	Todos
 cursor: se-resize	Todos
 cursor: s-resize	Todos
 cursor: sw-resize	Todos
 cursor: w-resize	Todos
 cursor: nw-resize	Todos
 cursor: progress	Solo Internet Explorer
 cursor: not-allowed	Solo Internet Explorer
 cursor: no-drop	Solo Internet Explorer
 cursor: vertical-text	Solo Internet Explorer

 cursor: all-scroll	Solo Internet Explorer
 cursor: col-resize	Solo Internet Explorer
 cursor: row-resize	Solo Internet Explorer
 cursor: url(...)	Indica la URL de un fichero con el cursor a utilizar. Consejo: define siempre un cursor genérico al final de la lista en caso de que ninguno de los definidos por URL funcione.

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

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

## 12.4. Hacks y filtros

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

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

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

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

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

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

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

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

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

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

El anterior ejemplo solamente carga la hoja de estilos `basico_ie6.css` si el navegador es la versión 6 o superior de Internet Explorer, ya que `gte` se interpreta como *"greater than or equal"* ("igual o mayor que"). Otros valores disponibles son `gt` (*"greater than"* o "mayor que"), `lt` (*"less than"* o "menor que") y `lte` (*"less than or equal"* o "igual o menor que").

```
<!--[if gt IE 7]>
  Mayor que Internet Explorer 7
<![endif]-->

<!--[if gte IE 7]>
```

```
Mayor o igual que Internet Explorer 7
<![endif]-->

<!--[if lt IE 8]>
  Menor que Internet Explorer 8
<![endif]-->

<!--[if lte IE 7]>
  Igual o menor que Internet Explorer 7
<![endif]-->
```

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

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

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

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

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

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

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

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

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

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

## 12.5. Prioridad de las declaraciones CSS

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

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

Además de la hoja de estilos del navegador, cada usuario puede crear su propia hoja de estilos y aplicarla automáticamente a todas las páginas que visite con su navegador. Se trata de una opción muy útil para personas discapacitadas visualmente, ya que pueden aumentar el contraste y el tamaño del texto de todas las páginas para facilitar su lectura.

La forma en la que se indica la hoja de estilo del usuario es diferente en cada navegador:

- Internet Explorer:
  1. Pincha sobre el menú `Herramientas` y después sobre la opción `Opciones de Internet`
  2. En la pestaña `General` que se muestra, pulsa sobre el botón `Accesibilidad` que se encuentra dentro de la sección `Apariencia`
  3. En la nueva ventana aparece la sección `Hoja de estilos del usuario`, activa la opción `Formatear los documentos con mi hoja de estilos` y selecciónala pulsando sobre el botón `Examinar...`

4. Pulsa **Aceptar** hasta volver al navegador
- Firefox:
  5. Guarda tu hoja de estilos en un archivo llamado **userContent.css**
  6. Entra en el directorio de tu perfil de usuario de Firefox. En los sistemas operativos Windows este directorio se encuentra normalmente en **C:\Documents and Settings\[tu\_usuario\_de\_windows]\Datos de programa\Mozilla\Firefox\Profiles\[cadena\_aleatoria\_de\_letras\_y\_numeros].default**
  7. Copia la hoja de estilos **userContent.css** en el directorio **chrome** de tu perfil
  8. Reinicia el navegador para que se apliquen los cambios
- Safari:
  9. Pincha sobre el menú **Editar** y después sobre la opción **Preferencias**
  10. Selecciona la sección **Avanzado**
  11. Pincha sobre la lista desplegable llamada **Hoja de estilos** y selecciona la opción **Otra...**
  12. En la ventana que aparece, selecciona tu hoja de estilos
- Opera:
  13. Pincha sobre el menú **Herramientas** y después sobre la opción **Preferencias**
  14. Selecciona la pestaña **Avanzado** y pulsa sobre el botón **Opciones de estilo...**
  15. Pulsa sobre el botón **Seleccionar...** para seleccionar la hoja de estilos
  16. Pulsa **Aceptar** hasta volver al navegador

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



Figura 3.148. Orden en el que se aplican las diferentes hojas de estilos

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

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

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

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

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

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

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



*Figura 3.149. Orden en el que se aplican las diferentes hojas de estilos cuando se utiliza la palabra reservada `important`*

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

## 12.6. Validador

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



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

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

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



Figura 3.150. Validador CSS del W3C

## 12.7. Recomendaciones generales sobre CSS

### 12.7.1. Atributos ID y class

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

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

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

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

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

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

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

Por este motivo, se recomienda que los valores asignados a `id` y `class` indiquen la función del elemento y no estén relacionados con su aspecto o su posición:

Valores no recomendados	Valores recomendados
negrita	importante
arial15	titular
verdanaPequena	normal
menulzquierdo	menuSecundario
letraRoja	error

Elegir el valor adecuado para los atributos `id` o `class` es sencillo: si el aspecto de un elemento cambia, el valor de `id` o `class` debe seguir siendo adecuado. Por tanto, evita utilizar valores relacionados con su posición (`izquierdo`, `derecho`, `primero`, `segundo`, `superior`, etc.), color (`textoRojo`, `subrayadoGrisClaro`, etc.) o tipo de letra (`verdana10`, `arial15px`, etc.)

Técnicamente, los valores de los atributos `id` y `class` deben cumplir las siguientes restricciones:

- Sólo pueden empezar por un guión medio (-), un guión bajo (\_) o una letra.
- El resto de caracteres, pueden ser números, guiones medios (-), guiones bajos (\_) y letras.
- Los navegadores distinguen entre mayúsculas y minúsculas.
- Aunque es posible utilizar letras como ñ y acentos, no se recomienda hacerlo porque no es seguro que funcione correctamente en todas las versiones de todos los navegadores.

### 13.7.2. CLASSitis y DIVitis

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

Ejemplo de *divitis* y *classitis*:

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

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

### 13.7.3. Estructuración del código CSS

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

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

- Estilos básicos (<body>, tipo de letra por defecto, márgenes de <ul>, <ol> y <li>, etc.)
- Estilos de la estructura o layout (anchura, altura y posición de la cabecera, pie de página, zonas de contenidos, menús de navegación, etc.)
- Enlaces (estilos normales, estilos :hover, etc.)
- Estilos de cada una de las zonas (elementos de la cabecera, titulares y texto de la zona de contenidos, enlaces, listas e imágenes de las zonas laterales, etc.)

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

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

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

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

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

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

/* Links
```

```
-----*/  
a:link,  
a:visited {  
    text-decoration:none;  
    color:#e45a49;  
}  
  
/* Header  
-----*/  
#header {  
    ...  
}  
...  
  
/* Main navigation  
-----*/  
ul#nav {  
    ...  
}  
...
```

Ejemplo de código CSS estructurado de <http://uxmag.com/>

```
/*  
    UX Magazine  
    Design | Technology | Strategy | Common Sense  
    -----  
    Description:   Base setup styles  
    Filename:      uxm.base.css  
    Version:       1.9  
    Date:          Feb 9, 2011  
    ----- */  
  
/*  
    Base Body Styles  
    ----- */  
  
/*  
    Print Styles  
    ----- */  
  
/*  
    Top Bar Styles  
    ----- */  
  
/*  
    Slogan  
    ----- */  
/*  
    Search Form  
    ----- */  
/*  
    Channels  
    ----- */
```

#### 12.7.4. División de los estilos en varios archivos CSS

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

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

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

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

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

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

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

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

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

## 13. LAS TRANSFORMACIONES CSS3

El módulo de las transformaciones fue desarrollado en un primer momento por los programadores del motor de visualización WebKit (de Safari y Chrome). En la actualidad, las transformaciones se describen en dos módulos de trabajo del W3C:

- Las transformaciones 2D: <http://www.w3.org/TR/css3-2d-transforms/> (en Working Draft el 11 de septiembre de 2012).
- Las transformaciones 3D: <http://www.w3.org/TR/css3-3d-transforms/> (en Working Draft el 11 de septiembre de 2012). Nosotros no nos vamos a ocupar de este último módulo, ya que es demasiado complejo y muy poco compatible con los navegadores.

Las transformaciones permiten modificar la visualización de los elementos HTML de una página web con las propiedades CSS en un plano bidimensional. Esta es la definición del W3C: *“CSS 2D Transforms allows elements rendered by CSS to be transformed in two-dimensional space.”*

Una vez que el elemento HTML haya sido publicado de manera “tradicional”, podrá “soportar” transformaciones de rotación, de desplazamiento, de deformación, de escala y de perspectiva.

Una vez más, los prefijos propietarios de los navegadores serán indispensables.

### 13.1. La transformación

Para aplicar una transformación tendremos que utilizar la propiedad `transform`. Esta propiedad utilizará a su vez diversas funciones dependiendo de la transformación en cuestión.

#### 1. El punto de referencia

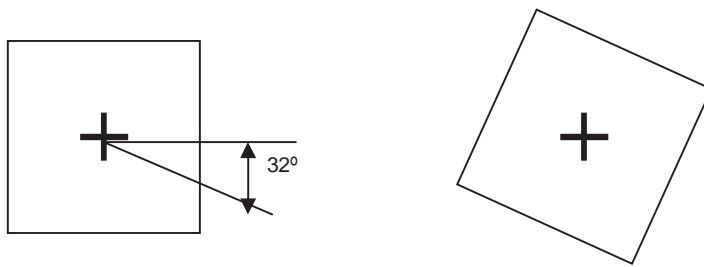
De manera predeterminada, todas las transformaciones tomarán como punto de referencia el centro del elemento. Ese punto de referencia sirve de valor inicial en los cálculos de las transformaciones.

Nosotros podemos cambiar ese punto de referencia con la propiedad `transform-origin`. El valor insertado indicará el nuevo punto de referencia.

Los valores posibles son:

- Porcentajes: por defecto, el valor es de `50%` `50%`, es decir, el centro del elemento.
- Palabras clave: `left`, `center`, `right`, `top`, `center`, `bottom`.
- Valores expresados en píxeles.

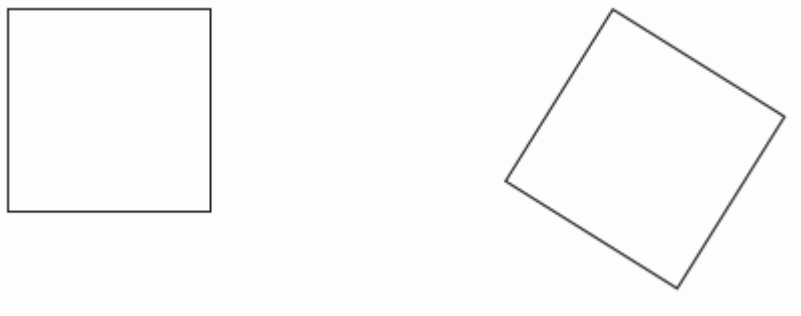
Este sería el resultado de una rotación con el punto de referencia predeterminado, es decir, en el centro del elemento.



Veamos un ejemplo de modificación del punto de origen a la esquina superior izquierda del elemento seguido de una rotación:

```
.rotar {  
  -moz-transform-origin: left top;  
  -webkit-transform-origin: left top;  
  -o-transform-origin:left top;  
  transform-origin: left top;  
  -moz-transform: rotate(32deg);  
  -webkit-transform: rotate(32deg);  
  -o-transform: rotate(32deg);  
  transform: rotate(32deg);  
}
```

Este sería el resultado obtenido al modificar del punto de referencia:





## 13.2. El desplazamiento

### 1. En los dos ejes

La función `translate` permite realizar un desplazamiento, una translación, de una distancia determinada, en función de la posición inicial y del punto de referencia.

El siguiente ejemplo desplaza la imagen 250 píxeles horizontalmente y 50 píxeles verticalmente.

```
.desplazamiento {  
    -moz-transform: translate(250px, 50px) ;  
    -webkit-transform: translate(250px, 50px) ;  
    -o-transform: translate(250px, 50px) ;  
    transform: translate(250px, 50px) ;  
}
```

Hemos aplicado la clase a una imagen:

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

Veamos el resultado de la transformación:



### 2. En un eje solamente

Podemos usar las dos funciones de desplazamiento en un único eje:

- `translateX`: para un desplazamiento en el eje horizontal.
- `translateY`: para un desplazamiento en el eje vertical.

Ejemplo de un desplazamiento horizontal de 20 píxeles:

```
.desplazamiento {  
    -moz-transform: translateX(20px) ;  
    -webkit-transform: translateX(20px) ;  
    -o-transform: translateX(20px) ;  
    transform: translateX(20px) ;  
}
```

## 13.3. El cambio de escala

### 1. La escala proporcional

La función `scale` permite transformar el tamaño de un elemento según una escala de `0` a `1`, en la que `1` es el tamaño inicial.

La clase del ejemplo siguiente permite aplicar una transformación proporcional según una escala del 50% con respecto al tamaño inicial.

```
.escala {  
  -moz-transform: scale(.5);  
  -webkit-transform: scale(.5);  
  -o-transform: scale(.5);  
  transform: scale(.5);  
}
```

### 2. La escala no proporcional

Si usted indica dos valores, el primero hará referencia al cambio de escala horizontal y, el segundo, al cambio de escala vertical.

En el ejemplo se ha aplicado un cambio de escala del 50% horizontalmente y del 20% verticalmente.

```
.escala {  
  -moz-transform: scale (.5,.2) ;  
  -webkit-transform: scale(.5,.2) ;  
  -o-transform: scale(.5,.2);  
  transform: scale(.5,.2);  
}
```

### 3. El cambio de escala en una única dirección

Puede usar las funciones:

- `scaleX`: para el cambio en la escala horizontal,
- `scaleY`: para el cambio de escala vertical.

En este ejemplo, el cambio de la escala es exclusivamente horizontal:

```
.escala {  
  -moz-transform: scaleX(.5) ;  
  -webkit-transform: scaleX(.5) ;  
  -o-transform: scaleX(.5);  
  transform: scaleX(.5);  
}
```

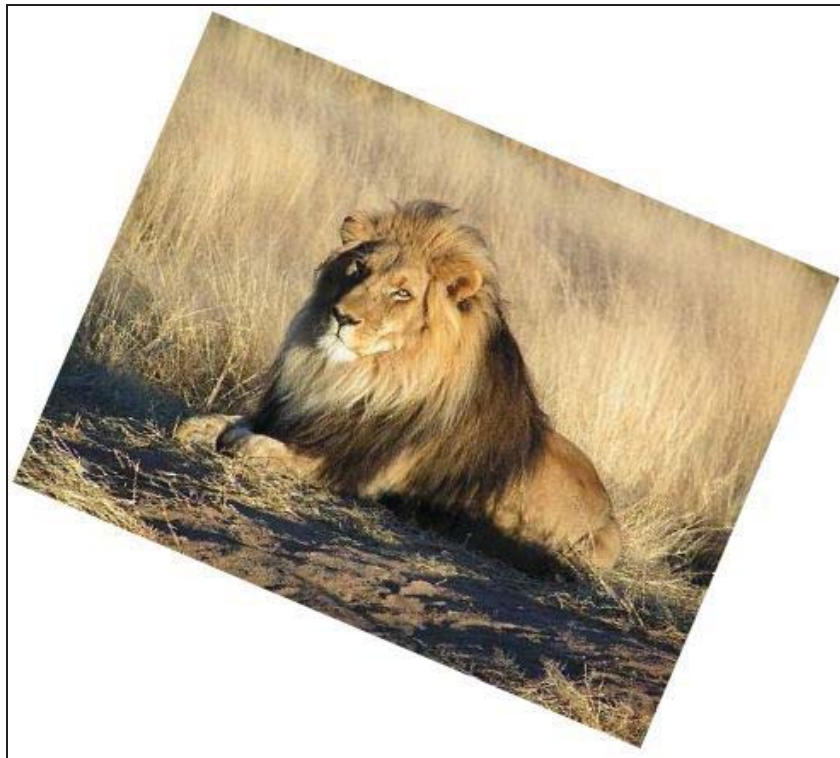
## 13.4. La rotación

La función `rotate` permite realizar la rotación de un elemento. La unidad se puede expresar en grados `deg` o en radianes `rad`.

Este es un ejemplo de una rotación de 23 grados:

```
.rotar {  
  -moz-transform: rotate(23deg) ;  
  -webkit-transform: rotate(23deg) ;  
  -o-transform: rotate(23deg);  
  transform: rotate(23deg);  
}
```

El resultado de la transformación:



## 13.5. La deformación

### 1. La deformación en los dos ejes

La función `skew` permite realizar una deformación del elemento en los dos ejes. La unidad puede expresarse en grados `deg` o en radianes `rad`.

Este es un ejemplo de una deformación horizontal de 20 grados y una deformación vertical de 5 grados:

```
.deformar {  
  -moz-transform: skew(20deg,-5deg) ;  
  -webkit-transform: skew(20deg,-5deg) ;  
  -o-transform: skew(20deg,-5deg);  
  transform: skew(20deg,-5deg);  
}
```

El resultado de la transformación:



## 2. La deformación en un único eje

Podrás usar las funciones:

- `skewX`: para la deformación horizontal,
- `skewY`: para la deformación vertical.

En este ejemplo, la transformación es exclusivamente horizontal:

```
.deformar {  
  -moz-transform: skewX(-20deg) ;  
  -webkit-transform: skewX(-20deg) ;  
  -o-transform: skewX(-20deg);  
  transform: skewX(-20deg);  
}
```

El resultado visual:





### 13.6. Aplicar todas las transformaciones

Es posible aplicar tantas transformaciones como queramos, bastará con que indiquemos las funciones que queramos usar.

Veamos un ejemplo con todas las transformaciones:

```
.transformar {  
-moz-transform: scale(.7) rotate(3deg) translateX(5px) skewX(-5deg);  
-webkit-transform: scale(.7) rotate(3deg) translateX(5px) skewX(-5deg);  
-o-transform: scale(.7) rotate(3deg) translateX(5px) skewX(-5deg);  
transform: scale(.7) rotate(3deg) translateX(5px) skewX(-5deg);  
}
```

Así se verá en la pantalla:



## 13.7. Los generadores en línea

Existen multitud de generadores en línea que te ayudarán a redactar su código CSS3 para que sea compatible con los navegadores habituales.

### 1. CSS 3.0 Maker

CSS 3.0 **Maker** (<http://www.css3maker.com/css3-transform.html>) te permite aplicar las cuatro transformaciones que acabamos de ver:



Figura 3.151. Transformación realizada en la página de CSS 3.0 Maker

### 2. CSS3 Generator

CSS3 **Generator** (<http://css3generator.com/>) le permite acceder a las transformaciones a través de la opción **Transform**.



Figura 3.152. Transformación realizada en la página de CSS3 Generator

### 3. WestCIV

**WestCIV** (<http://www.westciv.com/tools/transforms/index.html>) te permite acceder a todas las transformaciones que hemos presentado.

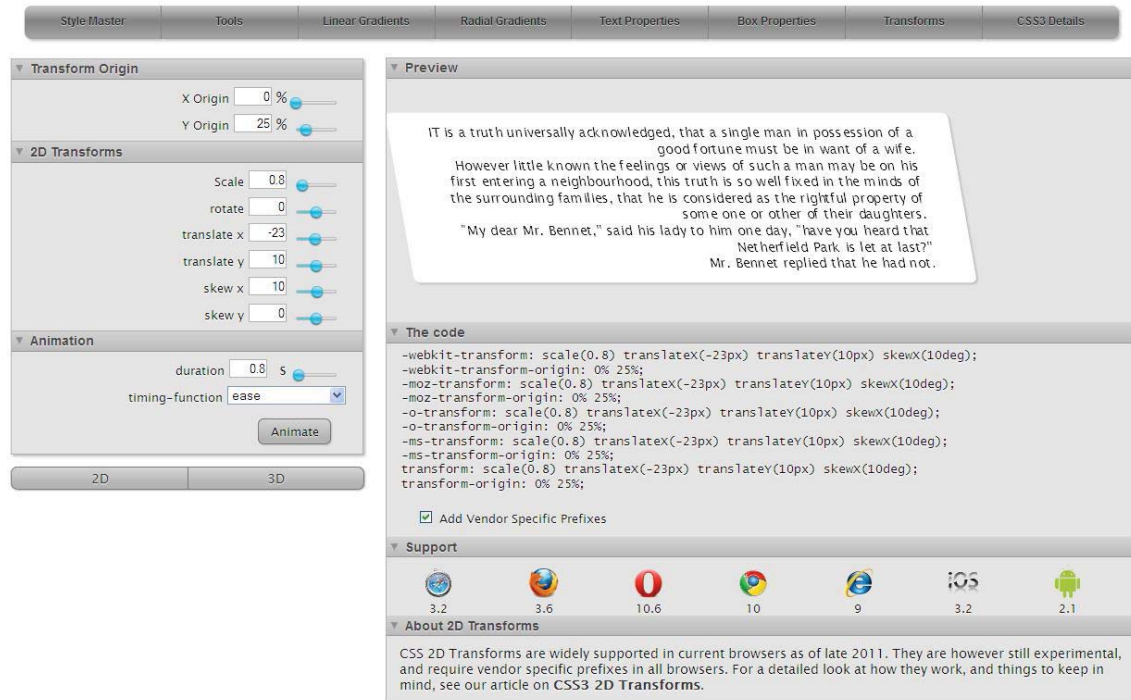


Figura 3.153. Transformación realizada en la página de Westciv

## 14. LAS TRANSICIONES CON CSS3

Es en cierta medida el “sueño” de todos los integradores web: crear “efectos” de transición gracias al CSS, sin un ápice de JavaScript y sin tener que pedir ayuda a un programador web. Pues bien, con CSS3 es posible, con el módulo **Transition**, aún en *Working Draft* el 3 de abril de 2012: <http://www.w3.org/TR/css3-transitions/>

Ahora es posible pasar de un valor CSS a otro, con una transición determinada, cuando se detecte un evento en un elemento. Y una vez que la transición haya terminado, el elemento recuperará sus parámetros CSS iniciales.

Una vez más, los prefijos propietarios de los navegadores serán necesarios.

### 14.1. Aplicar transiciones

#### 1. Las transiciones

Para activar una transición, es necesario que se detecte un evento con, por ejemplo, una pseudo-clase `:hover`, `:active` o `:focus`.

Deberemos indicar qué parámetros desea utilizar con la propiedad `transition-property`.

A continuación deberá indicar la duración de la transición con la propiedad `transition-duration`.

#### 2. Las propiedades CSS

Esta es la lista de propiedades CSS disponibles en el módulo **Transition**:

Nombre de la propiedad	Tipo
background-color	color
background-image	only gradients
background-position	percentage, length
border-bottom-color	color
border-bottom-width	length
border-color	color
border-left-color	color
border-left-width	length
border-right-color	color
border-right-width	length



border-spacing	length
border-top-color	color
border-top-width	length
border-width	length
bottom	length, percentage
color	color
crop	rectangle
font-size	length, percentage
font-weight	number
grid-*	various
height	length, percentage
left	length, percentage
letter-spacing	length
line-height	number, length, percentage
margin-bottom	length
margin-left	length
margin-right	length
margin-top	length
max-height	length, percentage
max-width	length, percentage
min-height	length, percentage
min-width	length, percentage
opacity	number
outline-color	color
outline-offset	integer
outline-width	length
padding-bottom	length
padding-left	length
padding-right	length
padding-top	length
right	length, percentage
text-indent	length, percentage
text-shadow	shadow
top	length, percentage
vertical-align	keywords, length, percentage

visibility	visibility
width	length, percentage
word-spacing	length, percentage
z-index	integer
zoom	number

La propiedad `transition-property` admite además dos palabras clave predefinidas:

- `transition-property: none` indica que ninguna propiedad está implicada en las transiciones.
- `Transition-property: all` indica que todas las propiedades podrían ser utilizadas en las transiciones.

## 14.2. Aplicar una transición de desplazamiento

### 1. Objetivo

Vamos a realizar una transición muy sencilla: una imagen que se desplace horizontalmente.

### 2. El código necesario

Tenemos una imagen dentro de una caja `<div>`:

```
<div id="desplazamiento"></div>
```

Vemos que se ha creado un selector de identificación para esa caja `<div>`. Ahora vamos a posicionar ese elemento:

```
#desplazamiento{  
    position: absolute;  
    left: 20px;  
    top: 20px;  
}
```

### 3. Las propiedades de la transición

Con la propiedad `transition-property` vamos a indicar cuál es la propiedad CSS que queremos modificar: `transition-property: left;`.

Con la propiedad `transition-duration` vamos a indicar la duración de la transición: `transition-duration: 2s;`. Las unidades que se pueden usar son los segundos `s` y los milisegundos `ms`.

Tendremos el siguiente selector:

```
#desplazamiento {  
    position: absolute;  
    left: 20px;  
    top: 20px;  
    -moz-transition-property: left;  
    -moz-transition-duration: 2s;  
    -webkit-transition-property: left;  
    -webkit-transition-duration: 2s;  
    -o-transition-property: left;  
    -o-transition-duration: 2s;  
    transition-property: left;  
    transition-duration: 2s;  
}
```

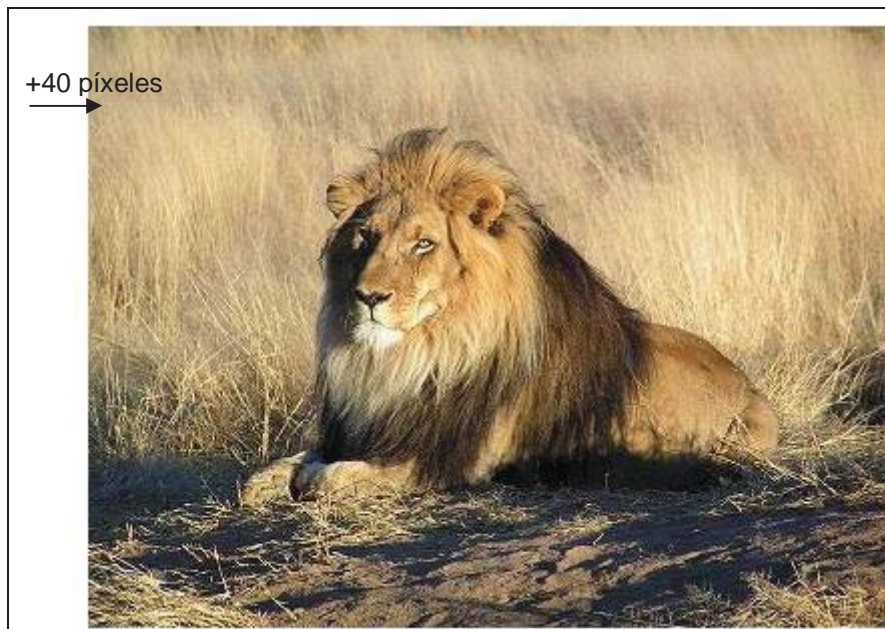
Ahora, para terminar, vamos a indicar el nuevo valor de la propiedad `left` y el evento que va a provocar la transición.

En nuestro ejemplo, será al pasar el cursor por encima (`:hover`) cuando se aplicará la transición. El desplazamiento se va a realizar hasta la posición horizontal de 40 píxeles.

```
#desplazamiento:hover {  
    left: 40px;  
}
```

#### 4. El resultado

Este sería el resultado visual, la imagen se desliza 40 píxeles en sentido horizontal.



## 14.3. Otras propiedades de las transiciones

### 1. El movimiento

La propiedad `transition-timing-function` permite definir el movimiento de la transición. Esta admite seis valores representados con palabras clave:

- `ease`: la transición es lenta al principio y luego se acelera cuando va a terminar (es el valor predeterminado).
- `linear`: la transición mantiene una velocidad constante.
- `ease-in`: la transición se lleva a cabo lentamente al principio y luego se acelera, cuando va a terminar.
- `ease-out`: la transición se lleva a cabo rápidamente al principio y luego disminuye su velocidad, cuando va a terminar.
- `ease-in-out`: la transición se lleva a cabo lentamente al principio, luego se acelera, y vuelve a disminuir la velocidad cuando va a terminar.
- `cubic-bezier`: permite configurar su propia curva de Bézier para definir el movimiento.

Estas son las curvas de Bézier utilizadas para esos efectos cinéticos:

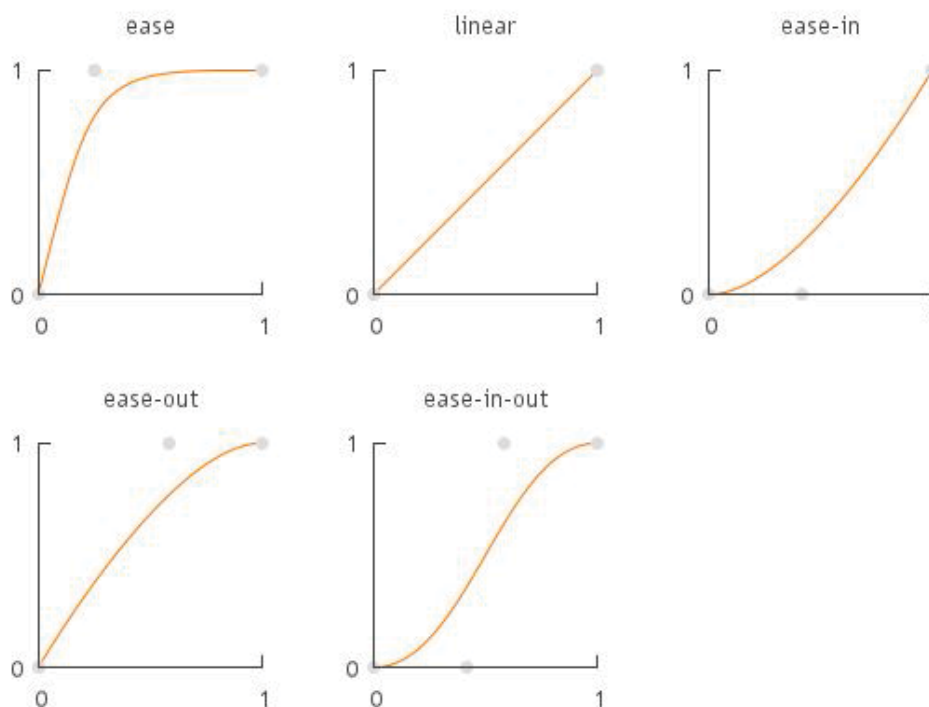


Figura 3.154. Curvas de Bézier utilizadas para efectos cinéticos

Fuente de los gráficos: <http://www.felix-girault.fr/blog/css/les-transitions-css3/>

Esta es la sintaxis que vamos a usar:

```
#desplazamiento {  
    position: absolute;  
    left: 20px;  
    top: 20px;  
    -moz-transition-property: left;  
    -moz-transition-duration: 2s;  
    -moz-transition-timing-function: ease-in;  
    -webkit-transition-property: left;  
    -webkit-transition-duration: 2s;  
    -webkit-transition-timing-function: ease-in;  
    -o-transition-property: left;  
    -o-transition-duration: 2s;  
    -o-transition-timing-function: ease-in;  
    transition-property: left;  
    transition-duration: 2s;  
    transition-timing-function: ease-in;  
}
```

## 2. El inicio de la transición

La propiedad `transition-delay` permite definir cuándo se iniciará la transición, cuánto tiempo después de la detección del evento.

Esta sintaxis hace que la transición se inicie 2 segundos después de que se detecte el evento: `transition-delay:2s;.`

Esta sería la sintaxis completa:

```
#desplazamiento {  
    position: absolute;  
    left: 20px;  
    top: 20px;  
    -moz-transition-property: left;  
    -moz-transition-duration: 2s;  
    -moz-transition-timing-function: ease-in;  
    -moz-transition-delay: 2s;  
    -webkit-transition-property: left;  
    -webkit-transition-duration: 2s;  
    -webkit-transition-timing-function: ease-in;  
    -webkit-transition-delay: 2s;  
    -o-transition-property: left;  
    -o-transition-duration: 2s;  
    -o-transition-timing-function: ease-in;  
    -o-transition-delay: 2s;  
    transition-property: left;  
    transition-duration: 2s;  
    transition-timing-function: ease-in;  
    transition-delay: 2s;  
}
```

### 3. La sintaxis abreviada

Se puede usar la sintaxis abreviada `transition` indicando a continuación las funciones separadas con un espacio simple, en este orden:

- `transition-property`
- `transition-duration`
- `transition-timing-function`
- `transition-delay`

Veamos un ejemplo de sintaxis abreviada:

```
#desplazamiento {  
    position: absolute;  
    left: 20px;  
    top: 20px;  
    -moz-transition: left 2s ease-in 2s;  
    -webkit-transition: left 2s ease-in 2s;  
    -o-transition: left 2s ease-in 2s;  
    transition: left 2s ease-in 2s;  
}
```

### 4. Las transiciones múltiples

Se pueden aplicar varias transiciones. Simplemente hay que separar cada transición con una coma e indicar el plazo para que cada transición comience en el momento oportuno.

Vamos a crear una transición múltiple con tres propiedades modificadas:

- el ancho `width`, que dura 3 segundos.
- la altura `height`, que dura 2 segundos y comienza 3 segundos más tarde.
- la opacidad `opacity`, que dura 4 segundos y comienza 6 segundos más tarde.

Vamos a trabajar a partir de una caja `<div>` que contiene, simplemente, un párrafo de texto.

Estos son los estilos CSS:

```
p.texto {  
    color: white;  
    font-weight: bold;  
    text-align: center;  
    line-height: 30px;  
}  
  
#multiple {
```

```
position: absolute;
left: 20px;
width: 200px;
height: 60px;
background-color: darkblue;
-moz-transition: width 3s linear, height 2s linear 3s, opacity
4s linear 6s;
-webkit-transition: width 3s linear, height 2s linear 3s,
opacity 4s linear 6s;
-o-transition: width 3s linear, height 2s linear 3s, opacity 4s
linear 6s;
transition: width 3s linear, height 2s linear 3s, opacity 4s
linear 6s;
}
#multiple:hover {
width: 400px;
height: 120px;
opacity: .3;
}
```

Este es el código de la caja <div>:

```
<div id="multiple">
  <p class="texto">En un lugar de la Mancha...</p>
</div>
```

Así se verá en la pantalla:

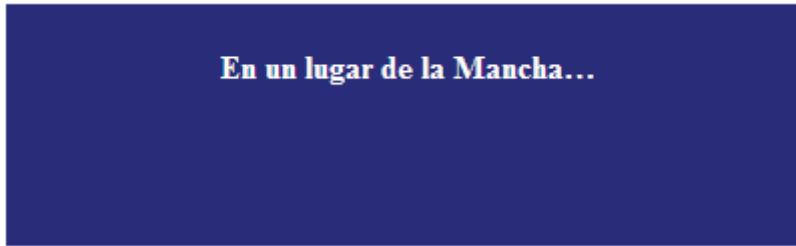
El cursor no se encuentra sobre la caja <div>, no se ha detectado ningún evento, no pasa nada:



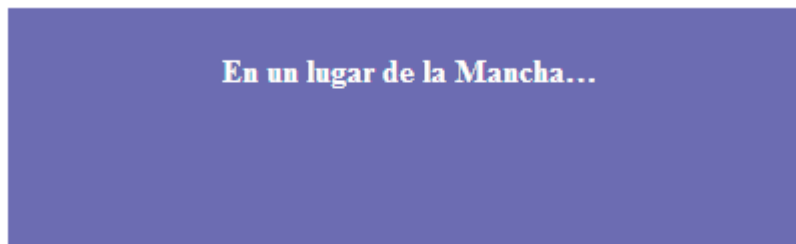
El cursor se encuentra encima (:hover) de la caja <div>, el ancho (property) se modifica durante 3 segundos (duration).



Luego, 3 segundos más tarde (delay), la altura de la caja (property) aumenta durante 2 segundos (duration):



Para terminar, 6 segundos más tarde (delay), se reduce la opacidad de la caja (property) durante 4 segundos (duration):



## 14.4. Los generadores en línea

### 1. CSS3 Generator

Si seleccionamos **Transitions** en el menú desplegable, **CSS3 Generator** (<http://css3generator.com/>) podremos trabajar con cinco propiedades (Property). Podrás especificar la duración (**Duration**) y el movimiento (**Function**).



Figura 3.155. Transición CSS3 configurada en la página CSS3 Generator



## 2. CSS 3.0 Maker

Con **CSS 3.0 Maker** (<http://www.css3maker.com/css3-transition.html>) solamente podremos trabajar con algunas propiedades de las transiciones. Es posible definir el evento (**Select Action**), la duración (**Transition Duration**) y el movimiento (**Transition Timing**).



Figura 3.156. Transición CSS3 configurada en la página CSS3 Maker

## 3. Crear las curvas de Bézier

Como ya vimos, la propiedad transition-timing-function nos permite aplicar nuestras propias curvas de Bézier con la palabra clave cubic-bezier: <http://cubic-bezier.com/>.

En la parte izquierda tienes un gráfico de Bézier en el que podrás cambiar los valores desplazando los círculos. La “fórmula” cubic-bezier se mostrará dinámicamente en la parte superior de la pantalla. También encontrarás un comparador de movimiento y las curvas predefinidas.

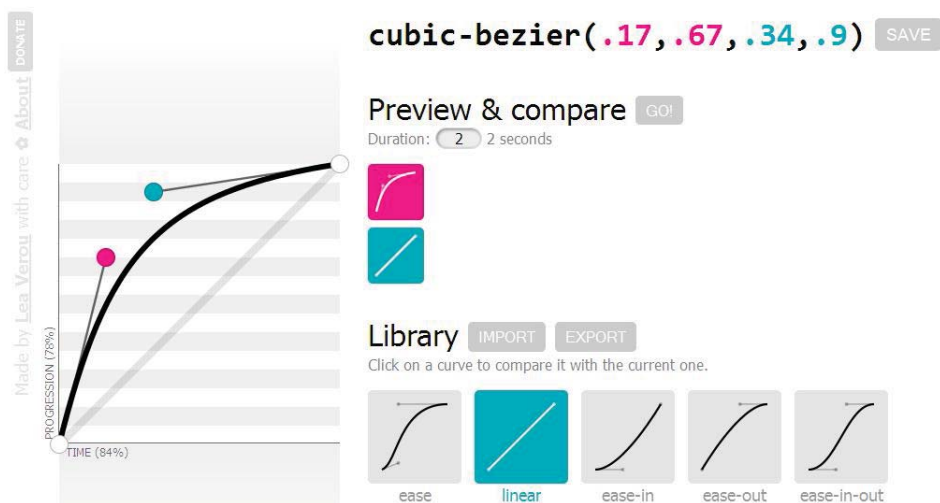


Figura 3.157. Sitio web para crear curvas de Bezier (cubic-bezier.com)