

# Unidad 4



Centro Don Bosco  
Villamuriel de Cerrato

# CSS

**(cubre CSS1, CSS2 y CSS3)**

Apuntes realizados para la asignatura de FP Grado Superior:  
**Lenguajes de Marcas y Sistemas de Gestión de Información**  
del ciclo Administración de Sistemas Informáticos en Red

**Autor: Jorge Sánchez Asenjo** ([www.jorgesanchez.net](http://www.jorgesanchez.net))  
**Versión del documento: 2.2, Año 2013**



Esta obra está bajo una licencia de Reconocimiento-NoComercial-CompartirIgual de Creative Commons.  
Para ver una copia de esta licencia, visite: <http://creativecommons.org/licenses/by-nc-sa/3.0/deed.es>





## Atribución-NoComercial-CompartirIgual 3.0 Unported (CC BY-NC-SA 3.0)

Esto es un resumen fácilmente legible del [Texto Legal \(la licencia completa\)](http://creativecommons.org/licenses/by-nc-sa/3.0/legalcode).

<http://creativecommons.org/licenses/by-nc-sa/3.0/legalcode>

### Usted es libre de:

Compartir - copiar, distribuir, ejecutar y comunicar públicamente la obra  
hacer obras derivadas

### Bajo las condiciones siguientes:



**Atribución** — Debe reconocer los créditos de la obra de la manera especificada por el autor o el licenciante (pero no de una manera que sugiera que tiene su apoyo o que apoyan el uso que hace de su obra).



**No Comercial** — No puede utilizar esta obra para fines comerciales.



**Compartir bajo la Misma Licencia** — Si altera o transforma esta obra, o genera una obra derivada, sólo puede distribuir la obra generada bajo una licencia idéntica a ésta.

### Entendiendo que:

**Renuncia** — Alguna de estas condiciones puede **no aplicarse** si se obtiene el permiso del titular de los derechos de autor

**Dominio Público** — Cuando la obra o alguno de sus elementos se halle en el **dominio público** según la ley vigente aplicable, esta situación no quedará afectada por la licencia.

**Otros derechos** — Los derechos siguientes no quedan afectados por la licencia de ninguna manera:

- Los derechos derivados de **usos legítimos** u otras limitaciones reconocidas por ley no se ven afectados por lo anterior.
- Los derechos **morales** del autor;
- Derechos que pueden ostentar otras personas sobre la propia obra o su uso, como por ejemplo **derechos de imagen** o de privacidad.



## índice

<b>(4.1)</b> las limitaciones de HTML	7
(4.1.1) introducción	7
(4.1.2) la ayuda de CSS	7
<b>(4.2)</b> versiones de CSS	8
<b>(4.3)</b> sintaxis básica de CSS	8
(4.3.1) comentarios	9
<b>(4.4)</b> inserción de código CSS	9
(4.4.1) en una etiqueta concreta de HTML	9
<b>(4.4.2)</b> en el elemento de cabecera <b>style</b>	10
(4.4.3) en un archivo externo	11
(4.4.4) orden de aplicación de estilos	12
(4.4.5) herencias	13
<b>(4.5)</b> selectores	13
(4.5.1) introducción	13
(4.5.2) selección de elementos HTML	13
(4.5.3) selección de clases	14
(4.5.4) selección de identificadores	15
(4.5.5) selector de limitación	15
(4.5.6) selector universal	15
(4.5.7) selección por atributos	16
(4.5.8) selectores jerárquicos	17
(4.5.9) pseudoclases	20
<b>(4.6)</b> formas de indicar valores de propiedades CSS	21
(4.6.1) introducción	21
(4.6.2) unidades de medida numéricas	21
(4.6.3) indicación de color	22
(4.6.4) indicación de URL	27
<b>(4.7)</b> formato de fuente	27
(4.7.1) propiedades de las fuentes	27
(4.7.2) propiedades de la fuente (tipografía)	29
<b>(4.8)</b> formato del texto	33
(4.8.1) word-spacing	33
(4.8.2) letter-spacing	33
(4.8.3) text-decoration	33
(4.8.4) vertical-align	34
(4.8.5) text-align	34
(4.8.6) text-indent	35
(4.8.7) text-transform	35



(4.8.8) direction	35
(4.8.9) text-overflow	35
(4.8.10) text-shadow	36
<b>(4.9) fondo</b>	<b>36</b>
(4.9.1) color de fondo. <i>background-color</i>	36
(4.9.2) imagen de fondo. <i>background-image</i>	36
(4.9.3) repetición del fondo. <i>background-repeat</i>	37
(4.9.4) posición de la imagen. <i>background-position</i>	37
(4.9.5) desplazamiento del fondo. <i>background-attachment</i>	38
<b>(4.9.6) background</b>	<b>38</b>
<b>(4.10) listas</b>	<b>39</b>
(4.10.1) list-style-type	39
(4.10.2) list-style-image	40
(4.10.3) list-style-position	40
(4.10.4) opciones de <b>display</b> referidas a listas	40
<b>(4.11) configuración de bordes y espacios alrededor de los elementos</b>	<b>42</b>
(4.11.1) elementos del formato de caja	42
(4.11.2) configuración del borde	43
<b>(4.12) tablas</b>	<b>48</b>
<b>(4.13) posicionamiento</b>	<b>50</b>
(4.13.1) introducción	50
(4.13.2) flotación	51
(4.13.3) tamaño del elemento	52
(4.13.4) modo de posicionamiento	52
(4.13.5) establecer coordenadas de posicionamiento	54
(4.13.6) coordenada <i>z</i>	56
(4.13.7) propiedades de visibilidad	58
<b>(4.14) propiedad <b>display</b></b>	<b>61</b>
<b>(4.15) formatos avanzados de caja (box)</b>	<b>65</b>
<b>(4.16) generación de contenido</b>	<b>67</b>
(4.16.1) propiedad <i>content</i>	67
(4.16.2) numeración automática	68
<b>(4.17) propiedades avanzadas de CSS3</b>	<b>70</b>
(4.17.1) el problema de la compatibilidad. prefijos	70
(4.17.2) opacidad	70
(4.17.3) relleno con gradientes	70
(4.17.4) transformaciones	71
(4.17.5) transiciones	72
(4.17.6) animaciones	74
<b>(4.18) aplicar CSS a documentos XML</b>	<b>76</b>

# (4) hojas de estilo CSS

## (4.1) las limitaciones de HTML

### (4.1.1) introducción

Como lenguaje para dar formato a las páginas web, HTML es muy limitado. En sus primeras versiones todas las posibilidades de formato se conseguían mediante etiquetas. Así existían etiquetas para modificar el tipo de letra (**font**), uso de maquetaciones complejas (**frame**), etc. Es decir ante una nueva necesidad, se inventaba una nueva etiqueta.

Hoy en día (y ya desde hace muchos años) se ha entendido que ese no es el modelo lógico para hacer páginas web. Es decir se ha entendido que HTML es un lenguaje que no es muy apropiado para definir el formato de la página. Las razones de no usar HTML para realmente dar formato a la página son:

- Podemos reutilizar el formato fácilmente para diferentes páginas manteniendo un formato coherente y armonioso en todo el sitio web. Con las etiquetas HTML no es posible, habría que volver a utilizar las etiquetas ya definidas.
- Permite usar HTML sólo como lenguaje semántico. De otro modo perderíamos el significado del texto. Esto es algo esencial para que los buscadores distingan el significado del texto de las páginas y afinen mejor en los resultados de las búsquedas que ofrecen al usuario.
- De usar sólo HTML, habría infinidad de etiquetas y una gran probabilidad de que cada navegador use las suyas propias (como ha ocurrido durante tantos años e incluso ahora sigue ocurriendo con algunas etiquetas).
- El formato y el contenido son independientes, con lo que se puede utilizar un lenguaje más apropiado para definir el formato. Es más, si en el futuro se inventan mejores lenguajes, sería fácil adaptarlos a las páginas web.

### (4.1.2) la ayuda de CSS

CSS es la abreviatura de **Cascade Style Sheets** (Hojas de Estilo en Cascada) y se trata de un lenguaje de texto que se incrusta en las páginas web para modificar el formato de la página. Actúa sobre HTML haciendo que las etiquetas HTML se muestren en el navegador con el formato que se indique.

Es capaz de actuar sobre todas las etiquetas del mismo tipo o sobre unas concretas. Se puede almacenar en un archivo aparte que después se puede usar para varias páginas a la vez. De modo que si cambiamos algo en el estilo, al instante se reflejará en todas las páginas.

CSS por lo tanto facilita la homogeneidad de las páginas y su mantenimiento. Hoy en día se considera una técnica imprescindible para dar formato a las páginas web. Además se puede aplicar también a código XML.

## (4.2) versiones de CSS

CSS se ideó a mediados de los años 90 y se ha ido estandarizando. A día de hoy se habla de tres versiones de CSS:

- **CSS1.** Es la versión original de CSS. Estandarizada en 1996 por la W3C incluye formatos de texto, párrafo, márgenes, lista, tamaños de imágenes,...
- **CSS2.** Es estándar desde 1998. Amplía el CSS anterior para incluir sobre todo posicionamiento (manejo de capas), además de tipos de medios (que permite definir distintos tipos de páginas web según los diferentes medios que la usen, pantallas, impresoras, reconocedores de voz...).

La especificación 2.1 es el último estándar. Modificó errores de la anterior.

- **CSS3.** Se lleva trabajando en ella desde 1998 y es ahora cuando parece que se está convirtiendo en el nuevo estándar. En realidad se compone de una serie de módulos que definen diferentes especificaciones que sumadas a CSS2 (con la que sigue siendo compatible) dan lugar a posibilidades muy avanzadas de formato. Como manejo del contenido, sombreados y rellenos avanzados, transparencias, transiciones, nuevos selectores,... De hecho en total hay unos 30 módulos, varios de ellos son ya considerados recomendación oficial.

En conjunto aún está en fase de borrador, se pueden observar sus trabajos en [www.w3.org/Style/CSS/current-work](http://www.w3.org/Style/CSS/current-work)

En estos apuntes se comenta de manera bastante profunda la especificación CSS2 y parte de las especificaciones CSS3. La mayoría de navegadores soportan íntegramente hasta CSS2 y parcialmente CSS3. Algunas páginas como <http://css3test.com/> permiten comprobar qué parte (incluso qué porcentaje) de CSS3 reconoce el navegador.

## (4.3) sintaxis básica de CSS

CSS es un lenguaje distinto de HTML es muy sencillo, pero su dificultad radica en que es muy extenso. CSS sigue esta sintaxis

```
selector {  
    propiedad1:valor1;  
    propiedad2:valor2;  
    ...  
}
```

El selector es la parte de CSS que nos permite indicar a qué elemento (o elementos) de la página web se va a aplicar el formato CSS. Para indicar dicho formato se marcan propiedades del elemento (cuyo nombre debe de ser reconocido por CSS) a las que asignamos valores. Esas propiedades permiten modificar el formato del elemento y los valores indican cuál es su nuevo formato.



Ejemplo:

```
p{
  font-size:14pt;
  color:red;
}
```

Ese código CSS provocaría que los elementos de tipo p salgan con tamaño de letra de 14 puntos y color rojo.

### (4.3.1) comentarios

Dentro del código CSS se pueden colocar comentarios. Para ello el texto del comentario se encierra entre los símbolos */\** y *\*/*. Ejemplo:

```
p {
  line-height:10pt;
  /*El siguiente código marcará el texto subrayado*/
  text-decoration:underline;
  text-align:center;
}
```

## (4.4) inserción de código CSS

### (4.4.1) en una etiqueta concreta de HTML

Todos los elementos de HTML (*p*, *strong*, *abbr*, *h1*,...) pueden utilizar un atributo llamado **style**, dentro de este atributo podemos añadir código CSS. Ejemplo:

```
<!doctype html>
<html lang="es">
  <head>
    <meta charset="UTF-8">
    <title></title>
  </head>
  <body>
    <p>Este texto sale de color negro (normal)</p>
    <p style="color:red;">Este texto sale de color rojo </p>
  </body>
</html>
```

En este caso el código CSS no lleva selector alguno, porque el estilo se aplica al elemento en el que se incrustó el código CSS. El resto del documento no queda afectado por el código CSS.

### (4.4.2) en el elemento de cabecera style

En este caso el código CSS se inserta debajo del elemento HTML **style** que se colocará en la zona de cabecera (**head**). Este código afectará a toda la página web. Ejemplo:

```
<!doctype html>
<html lang="es">
  <head>
    <meta charset="UTF-8">
    <title></title>

    <style type="text/css">
      p{
        color:red;
      }
    </style>

  </head>
  <body>
    <p>Este texto sale de color rojo</p>
    <p>Este texto sale de color rojo también </p>
  </body>
</html>
```

Ahora todos los elementos de tipo p se mostrarán en color rojo ya que es ese selector sobre el que se ha indicado el formato.

El elemento **style** usa fundamentalmente dos atributos:

- **type**. Que siempre contiene el valor **text/css**. En el caso de que apareciera otro lenguaje de estilos su contenido sería el tipo MIME correspondiente a ese lenguaje.
- **media**. Identifica a qué tipo de dispositivo se aplican los estilos. Podemos diseñar diferentes estilos en función de los dispositivos. Sus posibilidades son:
  - **all**. Opción por defecto que significa que los estilos se aplicarán a cualquier tipo de dispositivo en el que se esté viendo la página.
  - **screen**. Para pantallas a color
  - **print**. Para ser impreso.
  - **tty**. Pantallas de texto (como los terminales **ssh** por ejemplo o los de teletexto)
  - **tv**. Pantallas de televisión
  - **projection**. Proyector de vídeo
  - **handheld**. Dispositivos de mano
  - **speech**. Sintetizadores de voz. Hasta la versión 2 de CSS se usaba **aural** en lugar de **speech**, por lo que la mayoría de diseñadores lo sigue utilizando.
  - **braille**. Dispositivo de lectura táctil braille para personas invidentes.
  - **embossed**. Para dispositivos de impresión braille.

Otra forma (más utilizada) de indicar diferentes medios es utilizar dentro del código CSS la directiva **@media**, ejemplo:

```
<style type="text/css" >
  @media screen{
    p{
      color:green;
    }
  }
  @media print{
    p{
      color:#333333;
    }
  }
</style>
```

Con ese código conseguimos que por pantalla el texto de los párrafos salga de color verde, pero al imprimir salga de color gris. **@media** es una instrucción CSS que se puede utilizar en cualquier momento

#### (4.4.3) en un archivo externo

Es el caso más habitual. Se trata de crear un archivo de texto con extensión CSS al que se le inserta código CSS. La ventaja es que el mismo archivo nos sirve para aplicar sus estilos a diferentes documentos y si variamos simplemente el archivo CSS, todos los documentos que usen esa hoja de estilo cambiarán automáticamente.

Para poder colocar el código css existente en una página web HTML, hay que usar el elemento link (dentro del elemento head).

Por ejemplo si hemos creado un archivo de texto css llamado *estilos.css* con este contenido:

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

Suponiendo que lo guardemos en el mismo directorio que esta página web, el código siguiente provocaría que los dos párrafos se muestren en rojo.

```
<!doctype html>
<html lang="es">
  <head>
    <meta charset="UTF-8">
    <title></title>
    <link rel="stylesheet" href="estilo1.css" type="text/css">
  </head>
  <body>
    <p>Este texto sale de color rojo</p>
    <p>Este texto sale de color rojo también </p>
  </body>
</html>
```

La etiqueta **link** tiene los atributos:

- **href**. Con el que se indica la ruta de la hoja de estilos que se está incluyendo.
- **rel**. Que siempre contiene el texto **stylesheet** (para indicar que estamos incluyendo una hoja de estilos).

Además se pueden incluir los atributos **type** y **media** comentados en el punto anterior.

#### (4.4.4) orden de aplicación de estilos

Es posible que algunas páginas web utilicen varios estilos referidos al mismo elemento. Ejemplo:

```
<!doctype html>
<html lang="en">
  <head>
    <meta charset="UTF-8">
    <title>Document</title>
    <style type="text/css" media="screen">
      p{
        color:blue;
      }
    </style>
  </head>
  <body>
    <p style="color:red;">Hola</p>
  </body>
</html>
```

La cuestión es de color saldrá el texto **Hola** (que dentro del elemento p al que hacen referencia dos definiciones). La respuesta es de color rojo, porque tiene prioridad la definición más restrictiva. Es decir, en orden de preferencia se ejecuta:

- Primero se aplican los estilos del navegador. Es decir el formato predefinido del navegador. Todos los navegadores poseen un estilo predefinido, que dicta con que tamaño por defecto se muestra el texto, los colores, el tipo de letra,... Estos estilos son los que se ejecutan en primer lugar. Pero cualquier estilo definido fuera del navegador, tiene preferencia.
- Después se aplican los estilos externos (los que se incorporan con la etiqueta **link**)
- Después los que proceden de la etiqueta **style**.
- Después los que se definan internamente en el elemento (mediante el atributo **style**).
- En caso de dos estilos referidos al mismo elemento y definidos en el mismo ámbito (por ejemplo ambos procedentes de archivos externos e incluidos con el elemento **link**) tiene preferencia el último que se utilice en el código (es decir ganan los estilos del segundo **link**).

No obstante se puede alterar la preferencia utilizando una palabra clave: **!important**. Los estilos marcados con ella tienen preferencia sobre cualquier otro. Ejemplo:

```
p{  
  color:green !important;  
}
```

El color verde para los párrafos tendrá preferencia sobre cualquier redefinición de estilos sobre el elemento **p**.

### (4.4.5) herencias

Hay que tener en cuenta que hay etiquetas que son *padre* de otras. Es decir etiquetas que contienen a otras. En el ejemplo:

```
<p>Arturo Herrero: <em>Los años veinte</em></p>
```

La etiqueta **p** es padre de la etiqueta **em** (**em** está dentro de **p**). Esto hace que **em** herede todo el estilo que posea **p** y además añada el suyo propio. Por ejemplo, si hemos definido:

```
p{  
  color:blue;  
  font-size:12pt  
}  
em{  
  font-size:14pt;  
}
```

En el ejemplo anterior, *los años veinte* tendrán color azul y tamaño 14.

## (4.5) selectores

### (4.5.1) introducción

Los estilos CSS se aplican hacia el elemento HTML que indiquemos. Este elemento puede ser una etiqueta HTML, pero también podemos hacer selecciones más elaboradas. Hacer buenas selecciones nos permitirá conseguir hojas de estilos muy sofisticadas y coherentes. Veremos a continuación las posibilidades.

### (4.5.2) selección de elementos HTML

Podemos aplicar un estilo a un elemento concreto de HTML. Como en los ejemplos anteriores:

```
p{  
  color:blue;  
  font-size:12pt  
}
```

En principio con ese código todos los elementos de tipo **p** de la página saldrán de color azul.



Podemos incluso aplicar el estilo a varias etiquetas a la vez:

```
h1,h2,h3{  
  color:blue;  
}
```

Los títulos de tipo **h1**, **h2** o **h3** saldrán de color azul.

### (4.5.3) selección de clases

Una de las primeras formas que tiene CSS para diferenciar elementos del mismo tipo (por ejemplo un párrafo de otro) son las clases.

Una clase es un identificador que asignamos a una serie de propiedades y valores CSS. Se configuran de esta forma:

```
selector.nombreclase{  
  propiedad1:valor1;  
  propiedad2:valor2;  
  ...  
}
```

Por ejemplo:

```
p.clase1{  
  color: #339999;  
  background-color: #D6D6D6;  
}
```

Para que un párrafo (necesariamente marcado con la etiqueta **p**) adopte este estilo hay que indicarlo gracias a un atributo presente en todos los elemento HTML; se trata del atributo **class**. Ejemplo:

```
<p class="clase1">  
  Este texto sale con el formato indicado por la clase <em>clase1</em>  
</p>
```

Definir la clase sin indicar a qué selector pertenece:

```
.clase1{  
  color: #339999;  
  background-color: #D6D6D6;  
}
```

En ese caso cualquier elemento HTML podrá utilizar esa clase. Ejemplo:

```
<h1 class="clase1">  
  Título con estilo clase1  
</h1>
```

#### (4.5.4) selección de identificadores

La idea es parecida a la de las clases, pero ahora el nombre que indiquemos se referirá al atributo identificador de un elemento, el cual se marca con el conocido atributo **id** de HTML (presente en todas las etiquetas).

Los identificadores no se pueden repetir en el mismo documento, por lo que el formato indicado sólo se aplicará a un elemento de la página.

Ejemplo:

```
#parrafo1{  
  color: #339999;  
  background-color: #D6D6D6;  
}
```

Y la forma de aplicar:

```
<p id="parrafo1">  
  texto del párrafo coloreado  
</p>
```

#### (4.5.5) selector de limitación

Permite indicar que el estilo definido se aplica a una determinada etiqueta pero cuando sea hija de otra que especificada. Ejemplo:

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

Se aplica a los elementos de tipo **p** cuando están dentro de elementos de celda (**td**).

#### (4.5.6) selector universal

Existe un selector que permite aplicar un estilo a todas las etiquetas. Es el asterisco. Ejemplo:

```
*{  
  color:black;  
}
```

No se suele utilizar de esa forma ya que es demasiado indiscriminada. Pero sí se utiliza para elementos de este tipo:

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

Permite colorear en rojo el texto de los párrafos cuando esta etiqueta se use en tablas (es decir, dentro de elementos **table**); no importará si entre **table** y **p** hay otras etiquetas (sean del tipo que sean).

### (4.5.7) selección por atributos

Permite aplicar estilos a un elemento cuando este usa un atributo sobre el que toma un determinado valor. Para ello se indica el atributo entre corchetes, seguido del signo de igualdad y el valor entre comillas. Ejemplo:

```
<head>
  <meta charset="UTF-8">
  <title>Document</title>
  <style type="text/css" >

    p[lang="en"]{
      font-style: italic;
    }

  </style>
</head>
<body>
  <p lang="es">texto que sale de forma normal</p>
  <p lang="en">texto que sale en cursiva</p>
</body>
```

Se puede mezclar este tipo de definiciones con clases o definiciones por identificador:

```
p.clase1[lang="en"]{
  font-style: italic;
}
```

En este caso el estilo definido se aplica a párrafos de clase 1 que estén marcados con el valor **en** en el atributo **lang** (es decir que estén en inglés).

Se puede incluso utilizar más de un atributo:

```
p [lang="en"][spellcheck="true"]{
  font-style: italic;
}
```

En este caso se aplica el estilo para los elementos de tipo p que usen los atributos **lang** y **spellcheck** con los valores indicados.

También podemos indicar el estilo simplemente para los elementos que usen el atributo independientemente de su valor:

```
p [lang]{
  font-style: italic;
}
```

Por otro lado gracias a CSS3 disponemos de estas posibilidades:

sintaxis	significado
elemento[atributo~="valor"]	Elementos que usen el atributo indicado que contengan el valor aunque separado de otros valores por espacios
elemento[atributo\$="valor"]	Elementos que utilicen el atributo y cuyo contenido finalice con el valor indicado
elemento[atributo^="valor"]	Elementos que utilicen el atributo y cuyo contenido empiece con el valor indicado

sintaxis	significado
elemento[atributo*="valor"]	Elementos que utilicen el atributo indicado y contengan (en cualquier parte) el atributo indicado

### (4.5.8) selectores jerárquicos

Podemos entender HTML como un documento formado de manera jerárquica, donde hay elementos que contienen otros elementos formando una estructura de árbol. Esta es una parte importante del funcionamiento tanto de una página web HTML como de un documento XML. Esta visión permite seleccionar elementos que casen con una cierta jerarquía. De este modo esta página web:

```
<!DOCTYPE html>
<html lang="es-ES">
  <head>
    <meta charset="UTF-8">
    <title></title>
  </head>
  <body>
    <h1>Monumentos de Palencia</h1>
    <p>Palencia dispone de numerosos monumentos interesantes. Debido a su
      importante actividad medieval posee una gran cantidad de edificios
      religiosos entre los que destaca la <strong>Catedral de Palencia</strong>
      así como la iglesia de San Miguel con su peculiar torre de defensa y el
      monumento más conocido de la ciudad: El Cristo del Otero</p>
    <p>A finales del siglo <em>XIX</em> y principios del <em>XX</em>
      aparecieron edificios suntuosos y civiles que han embellecido una buena
      parte de la ciudad, en especial la transitada Calle Mayor.</p>
    <h2>Edificios religiosos</h2>
    <ul>
      <li>Cristo del Otero</li>
      <li>Catedral Mayor</li>
      <li>Iglesia de San Miguel</li>
      <li>Iglesia de San Lázaro</li>
      <li>Convento de San Pablo</li>
      <li>Iglesia de la Compañía</li>
    </ul>
  </body>
</html>
```

Podemos entender que sus elementos forman este diagrama:

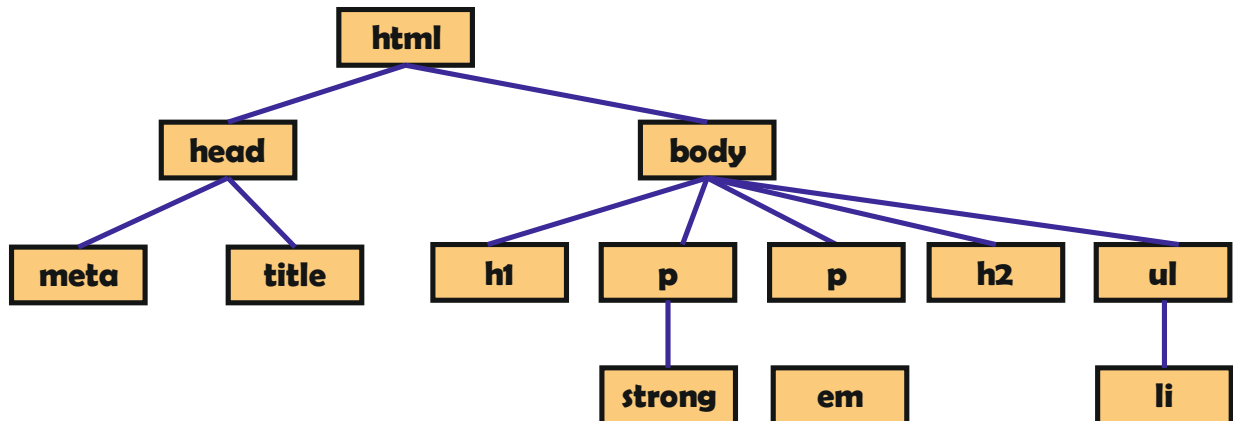


Ilustración 1, Estructura jerárquica de una página HTML

En él se observa como los elementos **body** y **head** son hijos de **html**. Mientras que **meta** y **title** son hijos de **head** (luego nietos de **html**). Los nodos al mismo nivel forman hermanos (**h1**, **h2**, **p** y **ul** en este esquema son hermanos).

Para especificar una relación padre/hijo, vimos antes como indicarla de esta forma:

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

Y esto significa lo mismo que:

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

El signo > indica una relación jerárquica de padre a hijo. Otras expresiones posibles son:

sintaxis	significado
elemento1 + elemento2	El estilo se aplica al <b>elemento2</b> cuando es hermano del <b>elemento1</b> y además el <b>elemento1</b> precede inmediatamente al <b>elemento2</b> .
elemento1 ~ elemento2	Se aplica al <b>elemento2</b> cuando es hermano del <b>elemento1</b> y éste le precede, aunque no sea inmediatamente.



sintaxis	significado
elemento: <b>nth-child</b> (n)	<p>Se aplica al elemento indicado cuando sea el hijo número <b>n</b> de su elemento padre. <b>n</b> puede ser un número (por ejemplo 3, para el tercer hijo), o una expresión más compleja como:</p> <ul style="list-style-type: none"> <li>• <b>2n+1</b>. Se aplica a los hijos con número impar de orden</li> <li>• <b>odd</b>. Igual que la anterior</li> <li>• <b>even</b>. A los pares</li> <li>• <b>3n+1</b>. Cada tres elementos hijos</li> </ul> <p>Ejemplo:</p> <pre><b>tr:nth-child(2n+1) td</b>{     background-color: green; }</pre> <p>Colorea de verde una los elementos <b>td</b> (celdas) que se encuentren en filas (elemento <b>tr</b>) cuyo número de hijo sea múltiplo de tres. Es decir pinta de verde una fila de cada tres.</p>
elemento: <b>nth-last-child</b> (n)	Igual que el anterior pero cuenta el orden de atrás hacia delante.
elemento: <b>nth-of-type</b> (n)	<p>Funciona como <b>nth-child</b> pero ahora se refiere al elemento número <b>n</b> (con n teniendo todas las posibilidades comentadas anteriormente) siendo n el número de hijo de ese tipo.</p> <p>Es decir, si tenemos dentro de <b>body</b> un párrafo de tipo h1 y otro de tipo p; ambos son hijos de <b>body</b>, p sería el segundo hijo (<b>nth-child(2)</b>) pero es el primero de su tipo (<b>nth-of-type(1)</b>).</p>
elemento: <b>nth-last-of-type</b> (n)	Como el anterior pero cuenta <b>n</b> desde el final.
elemento: <b>first-child</b>	Se aplica al elemento cuando es el primer hijo
elemento: <b>last-child</b>	Se aplica al elemento cuando es el último hijo
elemento: <b>first-of-type</b>	Primer descendiente de su tipo
elemento: <b>last-of-type</b>	Último descendiente de su hijo
elemento: <b>empty</b>	Se aplica cuando el elemento está vacío
elemento: <b>only-child</b>	Se aplica cuando el elemento es el único hijo
elemento: <b>only-of-type</b>	Se aplica cuando el elemento es el único hijo de ese tipo

Podemos incluso hacer combinaciones avanzadas. Ejemplo:

```
ul > li + li {
    color: green;
}
```

Se aplica a los elementos **li** que estén dentro de elementos **ul** y además estén inmediatamente precedidos por otro elemento **li**.

### (4.5.9) pseudoclasas

Las pseudoclasas permiten asociar estilos a un selector cuando le ocurre una determinada circunstancia. Las pseudoclasas más famosas son las que se aplican a los enlaces (elemento `a`). Las cuales son:

- **a:link**. Se aplica para los enlaces no visitados
- **a:visited**. Enlaces visitados
- **a:active**. Enlaces activos (aquellos sobre los que hacemos clic)
- **a:hover**. Se aplica cuando el ratón pasa por encima del enlace

Estas pseudoclasas permiten un cierto dinamismo en la página HTML. Actualmente además es posible utilizarlas en otros elementos distintos de la etiqueta `a` (como `p`, `div`, etc.) lo que da enormes posibilidades.

Otras pseudoclasas son:

pseudoclase	significado
<b>:focus</b>	Cuando el elemento obtiene el foco. Muy útil en formularios.
<b>:lang(código)</b>	Se aplica cuando el elemento esté marcado con el lenguaje indicado por su código ( <i>es</i> para español, <i>en</i> para inglés,...)
<b>:enabled</b>	Cuando está habilitado (útil en formularios)
<b>:disabled</b>	Cuando está deshabilitado (útil en formularios)
<b>:checked</b>	Para controles de formulario de tipo <b>radio</b> o <b>checkbox</b> cuando estén activados
<b>:before</b>	Para indicar contenido anterior al párrafo. Siempre se suele usar con la propiedad <b>content</b> para añadir contenido al elemento.
<b>:after</b>	Para indicar contenido después del elemento.
<b>:first-line</b>	Aplica estilo a la primera línea del elemento.
<b>:first-letter</b>	Aplica el estilo a la primera letra del elemento.

Las pseudoclasas se pueden combinar, de modo que se pueden construir cosas como:

```
input:focus:hover {  
    background-color: yellow;  
}
```

Se pondrá color de fondo amarillo, cuando el cuadro de tipo input tenga el foco y el ratón pase por encima de él. O cosas más complejas como:

```
input.clase1:focus:hover[type="password"] {  
    background-color: yellow;  
}
```

En este caso se aplica el fondo amarillo cuando el cursor esté encima de un control de texto de tipo contraseña que además tenga el foco y sea de *clase1*.

## (4.6) formas de indicar valores de propiedades CSS

### (4.6.1) introducción

Numerosas propiedades CSS tienen la necesidad de indicar valores. Por ejemplo el tamaño de la letra requiere un número. La cuestión es que CSS permite que dicho número se puede indicar con diferentes medidas (píxeles, pulgadas, centímetros, etc.)

La idea actual en cuanto a CSS es que permita dar formato a documentos que no sólo se muestren en pantalla, sino en todo tipo de dispositivos de salida. Las unidades de medida más adecuadas, por lo tanto, pueden variar y de ahí que se nos permita elegir.

La forma de indicarlas es colocarlas detrás de la cantidad que usemos. Por ejemplo **12in** significa doce pulgadas, mientras que **12cm** significa doce centímetros.

### (4.6.2) unidades de medida numéricas

- **in.** Pulgadas. Medida muy habitual en el mundo anglosajón, equivale a 25,4 milímetros.
- **cm.** Centímetro
- **mm.** Milímetro
- **pt.** Puntos. Medida muy utilizada en tipografía. Un punto tipográfico equivale a 1/72 pulgadas. Como todos los sistemas operativos le utilizan en los tipos de letra, lo cierto es que es de uso común para utilizar tamaños de fuentes.
- **pc.** Pica. Una pica la forman doce puntos. Y seis puntos equivalen a una pulgada.
- **ex.** Tamaño relativo respecto a la letra equis minúscula (**x**). Así **2x** indica que el tipo de letra aumenta hasta ocupar el doble la letra x mayúscula. Con **0.5x** ocuparía la mitad
- **em.** Tamaño relativo, en este caso, respecto a la letra M mayúscula
- **ch.** Altura relativa al número cero (0) en la tipografía actual. No se debe utilizar, ya que la mayoría de navegadores no le reconoce
- **px.** Píxeles. Esta medida es relativa respecto al dispositivo de salida. Ya que en cada dispositivo el tamaño del píxel varía. Se usa mucho en elementos grandes (capas, tablas, ....)
- **%.** Porcentaje. Es relativo respecto del tamaño del elemento padre del elemento al que le ponemos esta medida. Así 50% para una tabla dentro del elemento **body** ocuparía la mitad de la pantalla, pero dentro de una capa ocuparía la mitad del tamaño de la capa.
- **deg.** Grados. Utilizada para ángulos.
- **rad.** Radianes, también utilizada para ángulos. 360° son 2π radianes.
- **grad.** Gradianes. 400 gradianes son 360°.
- **turn.** También para ángulos. Indica vueltas, una vuelta serían 360 grados. 0.5turn es media vuelta (o sea 180°)
- **s.** Segundos. Se usa para indicar intervalos de tiempo

- **ms.** Milisegundos
- **Hz.** Hertzios. Usada para indicar frecuencias (para indicar valores de sonido o voz)
- **KHz.** Kilohertzios.
- **dpi.** Puntos por pulgada. Se usa para indicar resoluciones (para especificar, por ejemplo, la resolución mínima del dispositivo que debe tener para mostrar los estilos)
- **dpcm.** Tiene la misma utilidad que la anterior, pero indica puntos por centímetro.
- **dppx.** Puntos por píxel.

### (4.6.3) indicación de color

Hay numerosas etiquetas con capacidad de mostrar colores. Para indicar un color, CSS dispone de estas posibilidades:

- **Notación hexadecimal.** Se trata de la notación más utilizada en CSS. Consiste en una cifra hexadecimal, precedida del símbolo **#**. Las dos primeras cifras hexadecimales indican el nivel de rojo, las dos siguientes el nivel de verde y las dos últimas de verde; por ejemplo **#FF0000** es el código del rojo puro. Ejemplo<sup>1</sup>:

Color	Código hexadecimal	Código mediante función RGB
	#000000	rgb(0,0,0)
	#FF0000	rgb(255,0,0)
	#00FF00	rgb(0,255,0)
	#0000FF	rgb(0,0,255)
	#FFFF00	rgb(255,255,0)
	#00FFFF	rgb(0,255,255)
	#FF00FF	rgb(255,0,255)
	#C0C0C0	rgb(192,192,192)
	#FFFFFF	rgb(255,255,255)

- **Mediante función RGB.** Consiste en usar el formato **rgb(r,g,b)** donde **r** es el nivel de rojo (entre 0 y 255), **g** es el nivel de verde y **b** el de azul.
- **Mediante RGB con porcentaje.** Funciona como la anterior, pero el nivel de rojo, verde y azul se indican con tanto por ciento. Ejemplo: **rgb(50%,25%,12%)** (el color es un rojo anaranjado)
- **Mediante función RGB y transparencia.** Permite utilizar en la función RGB un cuarto parámetro que es un valor de transparencia (conocido como parámetro **alpha**). Este parámetro puede tener un valor entre 0 (transparencia total) y 1.0 (opacidad total). Es parte de CSS3 y sólo está disponible en versiones recientes del navegador.
- **Mediante función HSL.** Permite seleccionar colores utilizando tres valores:
  - **Tono (Hue).** Un valor de 0 a 360 que indica el giro en la rueda de colores. Por ejemplo el cero es el rojo, el 120 el verde y el 240 el azul.

<sup>1</sup> Extraído de [http://www.w3schools.com/cssref/css\\_colors.asp](http://www.w3schools.com/cssref/css_colors.asp)

- **Saturación.** Un número del 0% al 100%, que indica el nivel de saturación. Más saturación indica un color más vivo. Una saturación del 0% significa pasar el color a escala de grises.
- **Luminosidad.** Un número del 0% al 100%. Indica el nivel de luminosidad del color: más luminoso significa más claridad para el color (0% significa negro).

Este modo está disponible en los últimos navegadores compatibles con CSS3. Es el modelo más parecido a la forma que tiene el ser humano de percibir el color.

■ **Mediante función HSLA.** Añade al modelo anterior un cuarto valor (un número decimal entre 0.0 y 1.0) que sirve para indicar transparencia (nivel alfa).

■ **Por el nombre.** Permite indicar el color por su nombre estándar. Hay diecisiete colores estándares para especificar colores son:


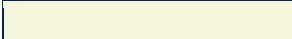







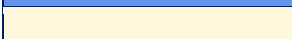






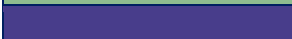


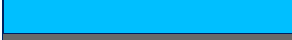



Color	Nombre	Código Hexadecimal
	White	#FFFFFF
	Silver	#C0C0C0
	Gray	#808080
	Black	#000000
	Red	#FF0000
	Maroon	#800000
	Yellow	#FFFF00
	Olive	#808000
	Lime	#00FF00
	Green	#008000
	Aqua	#00FFFF
	Teal	#008080
	Blue	#0000FF
	Navy	#000080
	Fuchsia	#FF00FF
	Purple	#800080

Estos colores se reconocen por cualquier navegador web. Además la especificación CSS 3 referente a los colores ([CSS3 color module](#)), reconoce los colores **X11**. Esta lista de colores se creó para el sistema X Windows de Unix y se fue adoptando por los navegadores y por diversos lenguajes gráficos (como SVG). En la actualidad son reconocidos por la mayoría de navegadores.

Hay diferencias entre el original X11 y la recomendación de nombres estándar de la W3C (que es la adoptada en CSS3). La lista completa de nombres de colores es:










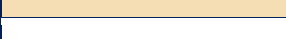
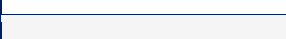



Color	Name	Hexadecimal	Modo RGB	Modo HSL
	Alice Blue	#F0F8FF	RGB(94%,97%,100%)	HSL(208,100%,97%)
	Antique White	#FAEBD7	RGB(98%,92%,84%)	HSL(34,78%,91%)
	Aqua	#00FFFF	RGB(0%,100%,100%)	HSL(180,100%,50%)
	Aquamarine	#7FFFD4	RGB(50%,100%,83%)	HSL(160,100%,75%)



Color	Name	Hexadecimal	Modo RGB	Modo HSL
	Azure	#F0FFFF	RGB(94%,100%,100%)	HSL(180,100%,97%)
	Beige	#F5F5DC	RGB(96%,96%,86%)	HSL(60,56%,91%)
	Bisque	#FFE4C4	RGB(100%,89%,77%)	HSL(33,100%,88%)
	Black	#000000	RGB(0%,0%,0%)	HSL(0,0%,0%)
	Blanched Almond	#FFEBCD	RGB(100%,92%,80%)	HSL(36,100%,90%)
	Blue	#0000FF	RGB(0%,0%,100%)	HSL(240,100%,50%)
	Blue Violet	#8A2BE2	RGB(54%,17%,89%)	HSL(271,76%,53%)
	Brown	#A52A2A	RGB(65%,16%,16%)	HSL(0,59%,41%)
	Burlywood	#DEB887	RGB(87%,72%,53%)	HSL(34,57%,70%)
	Cadet Blue	#5F9EA0	RGB(37%,62%,63%)	HSL(182,26%,50%)
	Chartreuse	#7FFF00	RGB(50%,100%,0%)	HSL(90,100%,50%)
	Chocolate	#D2691E	RGB(82%,41%,12%)	HSL(25,75%,47%)
	Coral	#FF7F50	RGB(100%,50%,31%)	HSL(16,100%,66%)
	Cornflower	#6495ED	RGB(39%,58%,93%)	HSL(219,79%,66%)
	Cornsilk	#FFF8DC	RGB(100%,97%,86%)	HSL(48,100%,93%)
	Crimson	#DC143C	RGB(86%,8%,24%)	HSL(348,83%,47%)
	Cyan	#00FFFF	RGB(0%,100%,100%)	HSL(180,100%,50%)
	Dark Blue	#00008B	RGB(0%,0%,55%)	HSL(240,100%,27%)
	Dark Cyan	#008B8B	RGB(0%,55%,55%)	HSL(180,100%,27%)
	Dark Goldenrod	#B8860B	RGB(72%,53%,4%)	HSL(43,89%,38%)
	Dark Gray	#A9A9A9	RGB(66%,66%,66%)	HSL(0,0%,66%)
	Dark Green	#006400	RGB(0%,39%,0%)	HSL(120,100%,20%)
	Dark Khaki	#BDB76B	RGB(74%,72%,42%)	HSL(56,38%,58%)
	Dark Magenta	#8B008B	RGB(55%,0%,55%)	HSL(300,100%,27%)
	Dark Olive Green	#556B2F	RGB(33%,42%,18%)	HSL(82,39%,30%)
	Dark Orange	#FF8C00	RGB(100%,55%,0%)	HSL(33,100%,50%)
	Dark Orchid	#9932CC	RGB(60%,20%,80%)	HSL(280,61%,50%)
	Dark Red	#8B0000	RGB(55%,0%,0%)	HSL(0,100%,27%)
	Dark Salmon	#E9967A	RGB(91%,59%,48%)	HSL(15,72%,70%)
	Dark Sea Green	#8FBC8F	RGB(56%,74%,56%)	HSL(120,25%,65%)
	Dark Slate Blue	#483D8B	RGB(28%,24%,55%)	HSL(248,39%,39%)
	Dark Slate Gray	#2F4F4F	RGB(18%,31%,31%)	HSL(180,25%,25%)
	Dark Turquoise	#00CED1	RGB(0%,81%,82%)	HSL(181,100%,41%)
	Dark Violet	#9400D3	RGB(58%,0%,83%)	HSL(282,100%,41%)
	Deep Pink	#FF1493	RGB(100%,8%,58%)	HSL(328,100%,54%)
	Deep Sky Blue	#00BFFF	RGB(0%,75%,100%)	HSL(195,100%,50%)
	Dim Gray	#696969	RGB(41%,41%,41%)	HSL(0,0%,41%)
	Dodger Blue	#1E90FF	RGB(12%,56%,100%)	HSL(210,100%,56%)
	Firebrick	#B22222	RGB(70%,13%,13%)	HSL(0,68%,42%)
	Floral White	#FFFAF0	RGB(100%,98%,94%)	HSL(40,100%,97%)
	Forest Green	#228B22	RGB(13%,55%,13%)	HSL(120,61%,34%)

Color	Name	Hexadecimal	Modo RGB	Modo HSL
	Fuchsia	#FF00FF	RGB(100%,0%,100%)	HSL(300,100%,50%)
	Gainsboro	#DCDCDC	RGB(86%,86%,86%)	HSL(0,0%,86%)
	Ghost White	#F8F8FF	RGB(97%,97%,100%)	HSL(240,100%,99%)
	Gold	#FFD700	RGB(100%,84%,0%)	HSL(51,100%,50%)
	Goldenrod	#DAA520	RGB(85%,65%,13%)	HSL(43,74%,49%)
	Gray	#808080	RGB(50%,50%,50%)	HSL(0,0%,50%)
	Green (W3C)	#008000	RGB(0%,50%,0%)	HSL(120,100%,25%)
	Green Yellow	#ADFF2F	RGB(68%,100%,18%)	HSL(84,100%,59%)
	Honeydew	#F0FFF0	RGB(94%,100%,94%)	HSL(120,100%,97%)
	Hot Pink	#FF69B4	RGB(100%,41%,71%)	HSL(330,100%,71%)
	Indian Red	#CD5C5C	RGB(80%,36%,36%)	HSL(0,53%,58%)
	Indigo	#4B0082	RGB(29%,0%,51%)	HSL(275,100%,26%)
	Ivory	#FFFFF0	RGB(100%,100%,94%)	HSL(60,100%,97%)
	Khaki	#F0E68C	RGB(94%,90%,55%)	HSL(54,77%,75%)
	Lavender	#E6E6FA	RGB(90%,90%,98%)	HSL(240,67%,94%)
	Lavender Blush	#FFF0F5	RGB(100%,94%,96%)	HSL(340,100%,97%)
	Lawn Green	#7CFC00	RGB(49%,99%,0%)	HSL(90,100%,49%)
	Lemon Chiffon	#FFFACD	RGB(100%,98%,80%)	HSL(54,100%,90%)
	Light Blue	#ADD8E6	RGB(68%,85%,90%)	HSL(195,53%,79%)
	Light Coral	#F08080	RGB(94%,50%,50%)	HSL(0,79%,72%)
	Light Cyan	#E0FFFF	RGB(88%,100%,100%)	HSL(180,100%,94%)
	Light Goldenrod	#FAFAD2	RGB(98%,98%,82%)	HSL(60,80%,90%)
	Light Gray	#D3D3D3	RGB(83%,83%,83%)	HSL(0,0%,83%)
	Light Green	#90EE90	RGB(56%,93%,56%)	HSL(120,73%,75%)
	Light Pink	#FFB6C1	RGB(100%,71%,76%)	HSL(351,100%,86%)
	Light Salmon	#FFA07A	RGB(100%,63%,48%)	HSL(17,100%,74%)
	Light Sea Green	#20B2AA	RGB(13%,70%,67%)	HSL(177,70%,41%)
	Light Sky Blue	#87CEFA	RGB(53%,81%,98%)	HSL(203,92%,76%)
	Light Slate Gray	#778899	RGB(47%,53%,60%)	HSL(210,14%,53%)
	Light Steel Blue	#B0C4DE	RGB(69%,77%,87%)	HSL(214,41%,78%)
	Light Yellow	#FFFFE0	RGB(100%,100%,88%)	HSL(60,100%,94%)
	Lime (W3C)	#00FF00	RGB(0%,100%,0%)	HSL(120,100%,50%)
	Lime Green	#32CD32	RGB(20%,80%,20%)	HSL(120,61%,50%)
	Linen	#FAF0E6	RGB(98%,94%,90%)	HSL(30,67%,94%)
	Magenta	#FF00FF	RGB(100%,0%,100%)	HSL(300,100%,50%)
	Maroon	#7F0000	RGB(50%,0%,0%)	HSL(0,100%,25%)
	Medium Aquamarine	#66CDAA	RGB(40%,80%,67%)	HSL(160,51%,60%)
	Medium Blue	#0000CD	RGB(0%,0%,80%)	HSL(240,100%,40%)
	Medium Orchid	#BA55D3	RGB(73%,33%,83%)	HSL(288,59%,58%)
	Medium Purple	#9370DB	RGB(58%,44%,86%)	HSL(260,60%,65%)
	Medium Sea Green	#3CB371	RGB(24%,70%,44%)	HSL(147,50%,47%)

Color	Name	Hexadecimal	Modo RGB	Modo HSL
	Medium Slate Blue	#7B68EE	RGB(48%,41%,93%)	HSL(249,80%,67%)
	Medium Spring Green	#00FA9A	RGB(0%,98%,60%)	HSL(157,100%,49%)
	Medium Turquoise	#48D1CC	RGB(28%,82%,80%)	HSL(178,60%,55%)
	Medium Violet Red	#C71585	RGB(78%,8%,52%)	HSL(322,81%,43%)
	Midnight Blue	#191970	RGB(10%,10%,44%)	HSL(240,64%,27%)
	Mint Cream	#F5FFFA	RGB(96%,100%,98%)	HSL(150,100%,98%)
	Misty Rose	#FFE4E1	RGB(100%,89%,88%)	HSL(6,100%,94%)
	Moccasin	#FFE4B5	RGB(100%,89%,71%)	HSL(38,100%,86%)
	Navajo White	#FFDEAD	RGB(100%,87%,68%)	HSL(36,100%,84%)
	Navy	#000080	RGB(0%,0%,50%)	HSL(240,100%,25%)
	Old Lace	#FDF5E6	RGB(99%,96%,90%)	HSL(39,85%,95%)
	Olive	#808000	RGB(50%,50%,0%)	HSL(60,100%,25%)
	Olive Drab	#6B8E23	RGB(42%,56%,14%)	HSL(80,61%,35%)
	Orange	#FFA500	RGB(100%,65%,0%)	HSL(39,100%,50%)
	Orange Red	#FF4500	RGB(100%,27%,0%)	HSL(16,100%,50%)
	Orchid	#DA70D6	RGB(85%,44%,84%)	HSL(302,59%,65%)
	Pale Goldenrod	#EEE8AA	RGB(93%,91%,67%)	HSL(55,67%,80%)
	Pale Green	#98FB98	RGB(60%,98%,60%)	HSL(120,93%,79%)
	Pale Turquoise	#AFEEEE	RGB(69%,93%,93%)	HSL(180,65%,81%)
	Pale Violet Red	#DB7093	RGB(86%,44%,58%)	HSL(340,60%,65%)
	Papaya Whip	#FFEDD5	RGB(100%,94%,84%)	HSL(37,100%,92%)
	Peach Puff	#FFDAB9	RGB(100%,85%,73%)	HSL(28,100%,86%)
	Peru	#CD853F	RGB(80%,52%,25%)	HSL(30,59%,53%)
	Pink	#FFC0CB	RGB(100%,75%,80%)	HSL(350,100%,88%)
	Plum	#DDA0DD	RGB(87%,63%,87%)	HSL(300,47%,75%)
	Powder Blue	#B0E0E6	RGB(69%,88%,90%)	HSL(187,52%,80%)
	Purple	#7F007F	RGB(50%,0%,50%)	HSL(300,100%,25%)
	Red	#FF0000	RGB(100%,0%,0%)	HSL(0,100%,50%)
	Rosy Brown	#BC8F8F	RGB(74%,56%,56%)	HSL(0,25%,65%)
	Royal Blue	#4169E1	RGB(25%,41%,88%)	HSL(225,73%,57%)
	Saddle Brown	#8B4513	RGB(55%,27%,7%)	HSL(25,76%,31%)
	Salmon	#FA8072	RGB(98%,50%,45%)	HSL(6,93%,71%)
	Sandy Brown	#F4A460	RGB(96%,64%,38%)	HSL(28,87%,67%)
	Sea Green	#2E8B57	RGB(18%,55%,34%)	HSL(146,50%,36%)
	Seashell	#FFF5EE	RGB(100%,96%,93%)	HSL(25,100%,97%)
	Sienna	#A0522D	RGB(63%,32%,18%)	HSL(19,56%,40%)
	Silver	#C0C0C0	RGB(75%,75%,75%)	HSL(0,0%,75%)
	Sky Blue	#87CEEB	RGB(53%,81%,92%)	HSL(197,71%,73%)
	Slate Blue	#6A5ACD	RGB(42%,35%,80%)	HSL(248,54%,58%)
	Slate Gray	#708090	RGB(44%,50%,56%)	HSL(210,13%,50%)

Color	Name	Hexadecimal	Modo RGB	Modo HSL
	Snow	#FFFAFA	RGB(100%,98%,98%)	HSL(0,100%,99%)
	Spring Green	#00FF7F	RGB(0%,100%,50%)	HSL(150,100%,50%)
	Steel Blue	#4682B4	RGB(27%,51%,71%)	HSL(207,44%,49%)
	Tan	#D2B48C	RGB(82%,71%,55%)	HSL(34,44%,69%)
	Teal	#008080	RGB(0%,50%,50%)	HSL(180,100%,25%)
	Thistle	#D8BFD8	RGB(85%,75%,85%)	HSL(300,24%,80%)
	Tomato	#FF6347	RGB(100%,39%,28%)	HSL(9,100%,64%)
	Turquoise	#40E0D0	RGB(25%,88%,82%)	HSL(174,72%,57%)
	Violet	#EE82EE	RGB(93%,51%,93%)	HSL(300,76%,72%)
	Wheat	#F5DEB3	RGB(96%,87%,70%)	HSL(39,77%,83%)
	White	#FFFFFF	RGB(100%,100%,100%)	HSL(0,0%,100%)
	White Smoke	#F5F5F5	RGB(96%,96%,96%)	HSL(0,0%,96%)
	Yellow	#FFFF00	RGB(100%,100%,0%)	HSL(60,100%,50%)
	Yellow Green	#9ACD32	RGB(60%,80%,20%)	HSL(80,61%,50%)

#### (4.6.4) indicación de URL

Muchas propiedades de CSS necesitan poder indicar direcciones URL a recursos necesarios para las páginas (direcciones de enlaces, imágenes, etc.). Para ello se utiliza una función llamada URL a la que (entre paréntesis) se le indica la dirección URL. Ejemplos (cambio de la imagen de fondo):

```
body{  
    background-image: url(imágenes/fondo1.jpg);  
}  
  
body{  
    background-imagen:url(www.libreria.com/imagenes/img1.jpg);  
}
```

### (4.7) formato de fuente

#### (4.7.1) propiedades de las fuentes

Indudablemente la tipografía (llamada también formato de fuente o, simplemente, fuente) es uno de los aspectos más importantes de la estética de un documento escrito. Manejar adecuadamente el estilo de la letra puede hacer que nuestra página sea mucho más atractiva.

CSS dispone de numerosas propiedades para modificar los tipos de letra. Muchas de sus propiedades tienen que ver con la siguiente imagen:

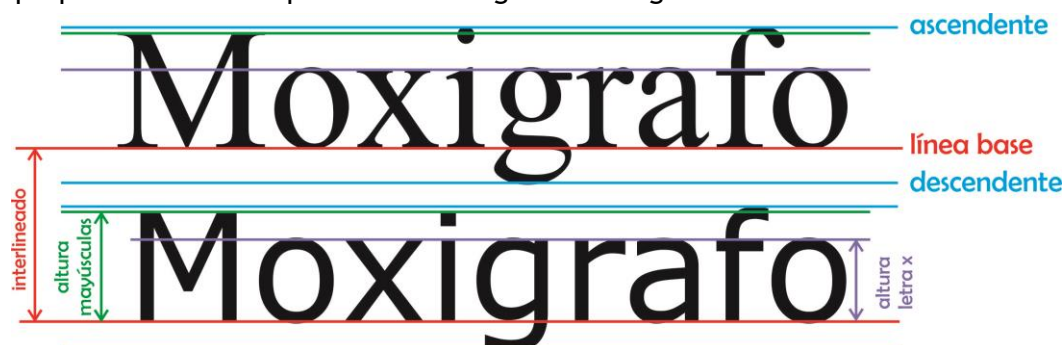


Ilustración 2, Elementos de la tipografía

En la imagen anterior se observan las líneas básicas que permiten encajar las letras. Además hay otros elementos a tener en cuenta:

- **Línea base.** Es la línea principal donde se colocan las letras. Es la línea que permite colocar adecuadamente en la horizontal al texto.
- **Altura de la x.** Mide el tamaño de la letra equis minúscula, que sirve de referencia para todas las letras minúsculas (aunque algunas como la **g** y la **a** del primer ejemplo la puedan rebasar).
- **Ascendente.** Línea superior que indica el máximo que puede ocupar las letras (en la imagen es el límite superior de la letra **f**).
- **Descendente.** Línea inferior, posición máxima inferior que puede ocupar el texto (la letra **g** en el ejemplo es la que marca esa posición).
- **Altura mayúsculas.** Línea superior máxima de las letras mayúsculas (se suele tomar la de la letra **M** mayúscula).
- **Interlineado.** Distancia entre dos líneas, se toma utilizando las líneas base.
- **Interletrado.** Distancia horizontal entre letra y letra. Ampliarla significaría distancias más cada letra de una palabra.
- **Kerning.** Es la distancia entre letra y letra pero aplicando distancias distintas en función de las letras. Así en la imagen anterior se observa que la distancia entre la **M** y la letra **o**, no es la misma que entre la **o** y la **x** (la **x** se mete un poco más dentro de la **o**), ya que de esa forma el texto es más estético.
- **Cuerpo.** La mayor parte de las tipografías actuales tiene tamaños (en horizontal) distintos de las letras. Es decir no ocupa lo mismo la letra **M** que la letra **N** (la **N** es más corta). Los llamados tipos de letra monoespaciados (como la letra Courier), hacen que cada carácter ocupe lo mismo (al estilo de las máquinas de escribir antiguas) en ellas la **M** ocupa lo mismo que la **N**.
- **Letras vectoriales y letras en mapa de bits.** Hoy en día todas las tipografías son de tipo vectorial, esto significa que no se guarda una imagen de cada letra, sino que de cada letra se guarda la fórmula para dibujarla. Esto permite que al ampliar las letras nunca perdamos resolución (al contrario de lo que ocurría en los ordenadores antiguos).

Las primeras letras vectoriales fueron las pertenecientes al lenguaje **PostScript** de **Adobe** incluidos en los ordenadores **Apple Macintosh**. **Microsoft** creó tipos de letra **TrueType** para Windows y ambas compañías actualmente desarrollan los tipos **OpenType** que se espera se convierta en un estándar abierto.



En la actualidad se da por hecho que todas las fuentes son vectoriales.

- **Familia.** Hay dos grandes familias de letras las letras de tipo Serif y las Sans-Serif. Las primeras se distinguen por colocar *patas* en algunas letras. Es el caso de la letra **Times**, que escribiría así: **MÉTODO** (obsérvense las patas de la **M** y la **T**), mientras que en forma sans serif, por ejemplo como la letra Arial, se escribiría así: **MÉTODO**.

Además de estas dos famosas familias, CSS permite distinguir otras tres familias. En total son:

- **Serif.** Ya explicada, por ejemplo: **Times**, **Garamond**, **Georgia**, **Bodoni**, **Bitstream Cyberbit**, **Baskerville**, **Bookman Old Style**..
- **Sans-serif.** Ya explicada, por ejemplo: **Arial**, **Verdana**, **Helvetica**, **Trebuchet**, **Gill Sans**, **Futura**, **Tahoma**, **Geneva**, ..
- **Cursive.** Son letras normalmente inclinadas y muy ornamentadas que simulan la escritura manual. Por ejemplo: *Zapf-Chancery*, **Comic Sans**, *Script MJ*, *Catfish Script*, *Adobe Poetica*, *Monotype Corsiva*, *Brush Script*..
- **Fantasy.** Letras muy ornamentadas que no están pensadas para el texto normal de un documento, pero sí se podría utilizar para sus títulos (pero sólo para los que deseemos recargar). Por ejemplo: **WESTERN**, **Impact**, **COPPERPLATE**, **Artistik**, **Britannic Bold**, **ALGERIAN**, **HERCULANUM**...
- **Monospaced.** Letras monoespaciadas, cada letra ocupa lo mismo. Simulan escritura con máquina de escribir antigua. Ejemplos: **Courier**, **Lucida Console**, **Consolas**, **Monaco**, **Liberation Mono**, **Andale Mono**...

## (4.7.2) propiedades de la fuente (tipografía)

### font-size

Tamaño de la fuente en pantalla. Se puede especificar de tres maneras:

- **En modo absoluto.** Hace referencia a tamaños predefinidos.

valor	significado
<b>xx-small</b>	Fuente muy pequeña
<b>x-small</b>	Fuente pequeña
<b>small</b>	Fuente un poco pequeña
<b>medium</b>	Fuente normal.
<b>large</b>	Fuente un poco grande
<b>x-large</b>	Fuente grande
<b>xx-large</b>	Fuente muy grande

- **En modo relativo.** En este caso se aumenta o disminuye el tamaño de la letra sobre el tamaño que tenía la letra en el elemento que contiene al del estilo (elemento padre). Valores:

valor	significado
<b>smaller</b>	Más pequeña
<b>larger</b>	Más grande

- **Modo exacto.** En este caso se indica el tamaño de la letra con su valor numérico. Inmediatamente tras este número se indica la medida en la que se debe medir el número. Ejemplos: **12px**, **2mm**, **12pt**, **1.2em**, **120%**,...

### font-family

Indica el tipo de letra. En definitiva, la fuente. El problema es que no todas las fuentes están disponibles en todos los sistemas, por ello se suelen indicar varias opciones separadas por comas; de modo que si la primera no está disponible, se usa la siguiente. Ejemplo:

```
p{  
  font-family:"AvantGarde Bk", Arial, Helvetica, sans-serif;  
}
```

### font-weight

Peso de la fuente (grosor). Valores posibles:

- **normal.** Espesor normal.
- **bold.** Negrita
- Número. Que puede ser: **100**, **200**, **300**, **400**, **500**, **600**, **700**, **800** y **900**. Ningún navegador soporta tantos pesos. Por lo que sólo funcionan bien las opciones **normal** y **bold**

### font-style

Estilo de letra. Puede ser; **normal**, **italic** (cursiva) u **oblique** (normalmente se representa igual que la anterior).

### font-variant

VERSALES (SMALL-CAPS). Valores: **normal** y **small-caps**.

### line-height

Permite calibrar el interlineado (la distancia entre cada línea). Se puede especificar de estas formas:

- Un número. En ese caso dicta la distancia multiplicando este número por la distancia normal. Es decir si indicamos **2**, el interlineado será doble, si indicamos **1.5** será un 50% mayor de lo normal.
- Un número seguido de una unidad de medida. Si indicamos **16px**, entonces estamos indicando la distancia exacta entre cada línea (que será de 16 píxeles).
- 

### font

Permite desde una sola propiedad cambiar en un solo golpe todas las anteriores. Su sintaxis es:

```
font: font-style font-variant font-weight font-size/line-height font-family;
```

El orden tiene que ser estrictamente ese, pero algunas propiedades se pueden dejar sin utilizar. Ejemplos:

```
/*Letra cursiva y negrita de tipo Comic con opciones a Arial y Helvetica*/
```

```
font: italic bold 16pt "Comic Sans MS", Arial, Helvetica, sans-serif;
```

```
/* Letra cursiva con versalitas de tamaño 18 puntos y 24 de puntos de distancia entre cada línea */
```

```
font: italic small-caps 18pt/24pt;
```

```
/* Letra cursiva */
```

```
font: italic;
```

También nos permite utilizar las fuentes de los elementos del sistema operativo en el que estemos, concretamente:

- **caption.** Letra utilizada para los títulos.
- **icon.** Letra de los iconos
- **menu.** Letra de los menús
- **message-box.** Letra de los cuadros de diálogo
- **small-caption.** Letra de los controles pequeños.
- **status-bar.** Letra de las barras de estado.

## color

Color de la fuente. Utilizando cualquiera de los códigos de color explicados en el apartado de dicado a las unidades y medidas.

## @font-face

Uno de los problemas más importantes sobre la tipografía es querer utilizar un tipo de letra sobre la que no hay seguridad que exista en el dispositivo de la persona que está viendo la página HTML.

CSS3 ha introducido el uso de esta directiva en el código CSS para precargar el tipo de letra necesario. Para ello deberemos utilizar el archivo que contiene el tipo de letra en sí. Dicho archivo puede ser **ttf** (letras de tipo **True Type**) u **otf** (**Open Type**) (Explorer utiliza el formato de letra **eot**) (**Embedded Open Type**).

Por ejemplo supongamos que queremos utilizar el tipo de letra **Artistik** y disponemos (teniendo en cuenta que los tipos de letra pueden tener derechos exclusivos de uso, por lo que habrá que asegurarse de que uso es libre o bien pagar los derechos para su uso) de archivo **Artistik.ttf** que contiene el tipo de letra en sí. El código para que todos los párrafos la usen (con la seguridad de que sí saldría) es:

```
@font-face{
  font-family:artistik;
  src: url('Artistik.ttf'), url('Artistik.eot');
}

p {
  font-family: artistik;
}
```

En la parte **@font-face** hay que indicar al menos dos propiedades:

- **font-family**. Para dar nombre de familia a la letra que estamos importando. Ese nombre es el que luego utilizarán los selectores para elegir la letra que hemos cargado (en el ejemplo es lo que usa **p** para dar formato con esa letra a los párrafos).
- **src**. En la que se indica una o más URL (separadas por comas) que apuntan a archivos de letras que se cargarán en la página para que se vea correctamente la letra.

Además, opcionalmente se puede usar:

- **font-weight**. La propiedad ya vista anteriormente, que nos permitiría indicar si el tipo de letra es para negrita (**bold**) o para texto normal.
- **font-style**. Ya vista anteriormente para indicar si la letra es cursiva o no.
- **font-stretch**. Indica el grado de condensación de la letra (por defecto se toma normal):
  - **normal**
  - **condensed**
  - **ultra-condensed**
  - **extra-condensed**
  - **semi-condensed**
  - **expanded**
  - **semi-expanded**
  - **extra-expanded**
  - **ultra-expanded**

El problema es que sólo es soportada esta característica por los navegadores más modernos y además algunos leen unos tipos de letras y otras no. Pero indudablemente en el futuro será una opción de uso muy habitual, porque asegura la misma apariencia de letra en cualquier máquina.

Otra cosa a tener en cuenta es que hay que cargar el archivo de fuentes y eso puede llevar cierto tiempo, durante el cual no se ve texto alguno o sale con un formato diferente.

## (4.8) formato del texto

Son propiedades que afectan al texto, fundamentalmente cambian el formato de los párrafos y aspectos del texto que no se refieren a su tipografía.

### (4.8.1) word-spacing

Indica la distancia entre las palabras del texto. Usa las mismas medidas que la propiedad **font-size**. Ejemplo de texto con `word-spacing:20px;`

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Sed quo  
nam dignissimos eum ex perferendis dolorem dolor fuga quasi.  
Praesentium provident obcaecati porro soluta mollitia quae sapiente  
explicabo necessitatibus beatae id tempora ab eligendi omnis quibusdam  
eius exercitationem doloremque minima aut accusantium natus temporibus  
repellendus deleniti repellat dolorum possimus quasi ipsam magni nisi  
voluptatibus voluptas illum numquam placeat inventore veritatis voluptate  
est ratione. Sint veritatis suscipit maiores voluptatem minus at  
quibusdam possimus voluptate rem reiciendis explicabo perspiciatis quo  
tempora laudantium aspernatur illo molestias aliquid assumenda harum  
similique enim officia vitae dolores ducimus ipsum culpa dicta corrupti  
vel numquam sapiente repudiandae.

### (4.8.2) letter-spacing

Indica la distancia entre las letras del texto. Es similar a la anterior, pero ahora referida a la distancia horizontal entre caracteres. Ejemplo con `letter-spacing:20px;`

L o r e m i p s u m d o l o r s i t  
a m e t , c o n s e c t e t u r  
a d i p i s i c i n g e l i t . S e d  
q u o n a m d i g n i s s i m o s e u m  
e x p e r f e r e n d i s d o l o r e m  
d o l o r f u g a q u a s i .  
P r a e s e n t i u m p r o v i d e n t  
o b c a e c a t i p o r r o s o l u t a  
m o l l i t i a q u a e s a p i e n t e  
e x p l i c a b o n e c e s s i t a t i b u s  
b e a t a e i d t e m p o r a a b  
e l i g e n d i o m n i s  
q u i b u s d a m e i u s  
e x e r c i t a t i o n e m  
d o l o r e m q u e m i n i m a a u t  
a c c u s a n t i u m n a t u s

### (4.8.3) text-decoration

Se indican posibles efectos en el texto. Valores:

- **underline**. Subrayado (línea por debajo del texto)
- **overline**. Línea por encima del texto.
- **line-through**. Tachado, línea que atraviesa el texto.
- **blink**. Parpadeo. No funciona en casi ningún navegador (sí en Firefox).

### (4.8.4) vertical-align

Posición vertical del texto (o imagen) respecto a su contenedor. Es muy versátil porque permite tanto alinear en vertical un texto respecto, por ejemplo, a la celda de la tabla en la que se encuentre; como indicar superíndices y subíndices. Posibilidades:

- **baseline**. En la línea base inferior del texto
- **sub**. Subíndice
- **super**. Superíndice
- **top**. Arriba respecto al elemento más alto de la línea
- **text-top**. En la línea superior del texto
- **middle**. Medio respecto a la altura del texto o contenedor en el que estemos
- **bottom**. Abajo respecto al elemento más alto de la línea
- **text-bottom**. En la línea inferior del texto
- **Porcentaje**. Porcentaje respecto al texto (por ejemplo 120%)

### (4.8.5) text-align

Alineación horizontal del texto. Puede ser: **left** (izquierda), **right** (derecha), **center** (centrada) o **justify** (justificada a derecha e izquierda).

Lorem ipsum dolor sit amet, consectetur adipisicing elit. Sed quo nam dignissimos eum ex perferendis dolorem dolor fuga quasi. Praesentium provident obcaecati porro soluta mollitia quae sapiente explicabo necessitatibus beatae id tempora ab eligendi omnis quibusdam eius exercitationem doloremque minima

Alineación izquierda: **text-align:left;**

Lorem ipsum dolor sit amet, consectetur adipisicing elit. Sed quo nam dignissimos eum ex perferendis dolorem dolor fuga quasi. Praesentium provident obcaecati porro soluta mollitia quae sapiente explicabo necessitatibus beatae id tempora ab eligendi omnis quibusdam eius exercitationem doloremque minima

Alineación izquierda: **text-align:center;**

Lorem ipsum dolor sit amet, consectetur adipisicing elit. Sed quo nam dignissimos eum ex perferendis dolorem dolor fuga quasi. Praesentium provident obcaecati porro soluta mollitia quae sapiente explicabo necessitatibus beatae id tempora ab eligendi omnis quibusdam eius exercitationem doloremque minima

Alineación izquierda: **text-align:right;**

Lorem ipsum dolor sit amet, consectetur adipisicing elit. Sed quo nam dignissimos eum ex perferendis dolorem dolor fuga quasi. Praesentium provident obcaecati porro soluta mollitia quae sapiente explicabo necessitatibus beatae id tempora ab eligendi omnis quibusdam eius exercitationem doloremque minima

Alineación izquierda: **text-align:justify;**

### (4.8.6) text-indent

Sangría de la primera línea del párrafo. Distancia extra que se deja a la primera línea respecto al resto de líneas. Por ejemplo si indicamos, `text-indent:50px;` el resultado sería (para un párrafo al que se aplique ese código):

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Sed quo nam dignissimos eum ex  
perferendis dolorem dolor fuga quasi. Praesentium provident obcaecati porro soluta mollitia quae  
sapiente explicabo necessitatibus beatae id tempora ab eligendi omnis quibusdam eius exercitationem  
doloremque minima aut accusantium natus temporibus repellendus deleniti repellat dolorum  
possimus quasi ipsam magni nisi voluptatibus voluptas illum numquam placeat inventore veritatis  
voluptate est ratione. Sint veritatis suscipit maiores voluptatem minus at quibusdam possimus  
voluptate rem reiciendis explicabo perspiciatis quo tempora laudantium aspernatur illo molestias  
aliquid assumenda harum similique enim officia vitae dolores ducimus ipsum culpa dicta corrupti vel  
numquam sapiente repudiandae.

### (4.8.7) text-transform

Permite modificar el texto para que se muestre en mayúsculas o minúsculas.

- **capitalize.** La primera letra en Mayúsculas
- **uppercase.** Mayúsculas
- **lowercase.** Minúsculas
- **none.** No hace ninguna transformación

### (4.8.8) direction

Procede de CSS2, especifica la dirección en la que se escribe el texto. Posibilidades:

- **ltr.** *Left to right*, de izquierda a derecha
- **rtl.** *Right to left*, de derecha a izquierda (utilizada en lenguas como el árabe)

### (4.8.9) text-overflow

Parte de CSS3. Indica que hacer con el texto cuando está dentro de un contenedor (como una capa) y no tiene el tamaño suficiente para mostrar todo el texto. Posibilidades:

- **clip.** El texto sale recortado. Sólo se ve el texto que cabe en la capa, el resto no se muestra
- **ellipsis.** Como la anterior, pero se ponen ... al final del texto recortado.

### (4.8.10) text-shadow

Se trata de una propiedad CSS3. Permite colocar una sombra al texto para darle efecto de volumen. Tiene esta sintaxis:

**text-shadow:** color distanciaX distanciaY desenfoque;

- **color.** Es el color de la sombra
- **distanciaX.** Es el desplazamiento horizontal que tendrá la sombra (puede ser positivo o negativo)
- **distanciaY.** Es el desplazamiento vertical que tendrá la sombra (puede ser positivo o negativo)
- **desenfoque.** Es opcional e indica cuánto se va a desenfocar la sombra. Se indica una cantidad que cuanto mayor sea, más desenfocará el fondo.

## (4.9) fondo

### (4.9.1) color de fondo. *background-color*

La propiedad **background-color** permite establecer un color de fondo al elemento al que se aplique la propiedad. Si se aplica al elemento **body**, toda la página tendrá ese color de fondo.

### (4.9.2) imagen de fondo. *background-image*

La propiedad **background-image** permite establecer una imagen de fondo. Esta imagen se superpone al color de fondo, de modo que si la imagen no se puede cargar (porque la ruta a ella no se ha indicado), entonces aparece el color de fondo.

La imagen de fondo se repite las veces necesarias (creando un mosaico), hasta que se rellena el elemento.

Ejemplo de código completo:

```
<!doctype html>
<html lang="es">
<head>
  <meta charset="UTF-8">
  <title>Document</title>
  <style>
    .fondo{
      background-color: maroon;
      background-image: url(granito.png);
      color:white;
    }
  </style>
</head>
```



```
<body>
  <p class="fondo">Lorem ipsum dolor sit amet, consectetur adipisicing elit. Sed
  quo nam dignissimos eum ex perferendis dolorem dolor fuga quasi. Praesentium
  provident obcaecati porro soluta mollitia quae sapiente explicabo necessitatibus
  beatae id tempora ab eligendi omnis quibusdam eius exercitationem doloremque
  minima aut accusantium natus temporibus repellendus deleniti repellat </p>
  <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit. Dolorum excepturi
  ut quaerat magnam error perferendis obcaecati. Voluptatem voluptatum odit quas
  dicta cumque reprehenderit necessitatibus perferendis minus laudantium neque
  molestias quo.</p>
</body>
</html>
```

Resultado:

Lorem ipsum dolor sit amet, consectetur adipisicing elit. Sed quo nam dignissimos eum ex perferendis dolorem dolor fuga quasi. Praesentium provident obcaecati porro soluta mollitia quae sapiente explicabo necessitatibus beatae id tempora ab eligendi omnis quibusdam eius exercitationem doloremque minima aut accusantium natus temporibus repellendus deleniti repellat

Lorem ipsum dolor sit amet, consectetur adipisicing elit. Dolorum excepturi ut quaerat magnam error perferendis obcaecati. Voluptatem voluptatum odit quas dicta cumque reprehenderit necessitatibus perferendis minus laudantium neque molestias quo.

### (4.9.3) repetición del fondo. background-repeat

En principio, el fondo se repite en todas las direcciones hasta rellenar el elemento al que se aplica la imagen de fondo.

Posibles valores de la propiedad:

- **repeat.** Es el valor por defecto. la imagen se repite en todas las direcciones (efecto mosaico), hasta rellenar completamente el elemento.
- **repeat-x.** La repetición de la imagen se hace sólo en horizontal.
- **repeat-y.** La repetición de la imagen se hace sólo en vertical.
- **no-repeat.** La imagen no se repite aparecerá sólo una vez

### (4.9.4) posición de la imagen. background-position

Por defecto la imagen se coloca desde la esquina superior izquierda de la página (posición 0,0) y desde ahí se repite (si se ha indicado repetición de la imagen con la propiedad anterior) o no.

La posición desde la que la imagen se coloca inicialmente se puede modificar. La forma de hacerlo es así:

```
background-position: posicionHorizontal posicionVertical
```

Podemos indicar ambos valores de esta forma:

- **posicionHorizontal**. Se puede especificar las palabras: **left** (izquierda), **right** (derecha) o **middle** (centro). También se puede indicar una coordenada horizontal concreta (por ejemplo **12px**) o relativa (**5%**).
- **posicionVertical**. Se puede especificar las palabras: **top** (arriba), **bottom** (abajo) o **center** (centro). También se puede indicar una coordenada vertical concreta (por ejemplo **20px**) o relativa (**15%**).

Así por ejemplo el código:

```
background-position: right bottom;
```

La imagen de fondo (si no hay repetición de la misma) se mostrará en la esquina inferior derecha del elemento en el que se ponga.

#### (4.9.5) desplazamiento del fondo. **background-attachment**

Cuando se desplaza el contenido de una página web, el fondo se mueve con el resto de la página. Mediante esta propiedad podremos hacer que el fondo quede fijo mientras que sólo el resto de elementos se mueven. Los posibles valores para la propiedad son:

- **scroll**. Valor por defecto. El fondo y el texto se mueven juntos cuando el usuario se desplaza por la página
- **fixed**. El fondo es fijo y sólo se mueve el texto.

#### (4.9.6) **background**

Fija en una sola propiedad todas las propiedades de fondo. Sintaxis:

```
background: background-color background-image background-repeat  
background-attachment background-position
```

Ejemplo:

```
background: maroon url('fondo1.gif') no-repeat fixed left bottom;
```

Coloca color marrón de fondo, la imagen **fondo1.gif**, que sólo aparece una vez, en la parte inferior izquierda y que se queda fijo en el fondo cuando el usuario se desplaza por la pantalla.

## (4.10) listas

CSS permite varios formatos pensados para listas. En general para los elementos que contienen las etiquetas **ul** y **ol**.

### (4.10.1) list-style-type

La propiedad **list-style-type** permite especificar el tipo de elemento de numeración de la lista. Normalmente los navegadores muestran un círculo relleno en las listas no numeradas (las que se realizan mediante la etiqueta **ul**) y números romanos en las listas numeradas (etiqueta **ol**).

Esta propiedad cambia esos símbolos para permitir elegir el que deseemos. La cuestión es que (aunque se podría) no es recomendable hacer que la etiqueta **ol** muestre símbolos no numéricos y que la etiqueta **ul** muestre símbolos numéricos, para mantener la coherencia semántica en el lenguaje HTML.

En todo caso los posibles valores de esta propiedad son:

- **armenian**. La lista quedará encabezada por números del alfabeto armenio.
- **circle**. La lista quedará encabezada por círculos.
- **cjk-ideographic**. Se usan símbolos numéricos ideográficos chinos.
- **decimal**. Números decimales.
- **decimal-leading-zero**. Números decimales empezando por cero.
- **disc**. La lista quedará encabezada por círculos sin rellenar.
- **georgian**. Números del alfabeto georgiano.
- **hebrew**. Números del alfabeto hebreo.
- **hiragana**. Números del alfabeto hiragana japonés.
- **hiragana-iroha**.
- **katakana**. Números del alfabeto katakana japonés.
- **katakana-iroha**.
- **lower-alpha**. Letras minúsculas del alfabeto actual.
- **lower-greek**. Letras griegas minúsculas.
- **lower-latin**. Letras minúsculas latinas.
- **lower-roman**. Números romanos en minúsculas.
- **none**. Sin números.
- **square**. Cuadrados rellenos.
- **upper-alpha**. Letras mayúsculas del alfabeto actual.
- **upper-latin**. Letras mayúsculas latinas.
- **upper-roman**. Números romanos en mayúsculas.

### (4.10.2) list-style-image

En lugar de utilizar uno de los símbolos anteriores, se puede indicar una imagen con la que se rellenará la lista. La imagen puede tener cualquiera de los formatos habituales en las páginas web ([gif](#), [jpg](#), [png](#)). Lógicamente el tamaño debe de ser apropiado; si es muy grande la lista quedará totalmente descuadrada.

Ejemplo de uso:

```
list-style-image:url('cuadrado.gif');
```

El problema es que los navegadores nbo

### (4.10.3) list-style-position

Sólo tiene dos posibles valores referidos a la posición del texto respecto de la imagen.

- **inside**. El símbolo de numeración es interior a los márgenes del elemento en el que se coloca. Es la opción por defecto.
- **outside**. El símbolo de numeración es exterior.

### (4.10.4) opciones de display referidas a listas

CSS dispone de una propiedad muy potente llamada **display**. Esta propiedad modifica el comportamiento de los elementos y permite indicar otras posibilidades. En el caso de las listas hay dos valores de display que son interesantes:

- **display:inline**. El valor [inline](#) aplicado a una lista hace que la lista no se muestre separada y acompañada de los símbolos de lista, sino que se muestra en la misma línea separados con un espacio cada elemento. Ejemplo:

```
<!doctype html>
<html lang="es">
<head>
  <meta charset="UTF-8">
  <title>Document</title>
  <style>
    li{
```

```
      display:inline;
```

```
    }
  </style>
</head>
<body>
  <ol>
    <li>Azúcar</li>
    <li>Sal</li>
    <li>Alubias</li>
    <li>Refresco</li>
  </ol>
</body>
</html>
```

Resultado:

Lista de la compra

Azucar Sal Alubias Refresco

■ **display:list-style**. Es justo el contrario del anterior permite a otros elementos usarse como parte de una lista. Ejemplo:

```
<!doctype html>
<html lang="es">
<head>
  <meta charset="UTF-8">
  <title>Document</title>
  <style>
```

```
    p{
      display:list-item;
    }
```

```
</style>
</head>
<body>
  <p>Lista de la compra</p>
  <ol>
    <p>Azúcar</p>
    <p>Sal</p>
    <p>Alubias</p>
    <p>Refresco</p>
  </ol>
</body>
</html>
```

El resultado es:

Lista de la compra

1. Azucar
2. Sal
3. Alubias
4. Refresco

La etiqueta **p** se comporta en el ejemplo como si fuera una etiqueta **li**, sin embargo respetará el tipo de letra, los espacios,.. es decir todas las características restantes de los elementos **p**. Se observa también que cuando **p** no está dentro de una etiqueta de lista (como **ul** u **ol**) funciona de manera habitual (como se observa en el párrafo [Lista de la compra](#)).

## (4.11) configuración de bordes y espacios alrededor de los elementos

### (4.11.1) elementos del formato de caja

Se suele denominar **formato de caja** (o **formato de cuadro**) a la parte de CSS encargada del formato referente al rectángulo imaginario que envuelve a un elemento de una página HTML. Este rectángulo imaginario contiene complemente al elemento, así como al borde (si lo hay) y a una serie de espacios exteriores al elemento que se pueden configurar.

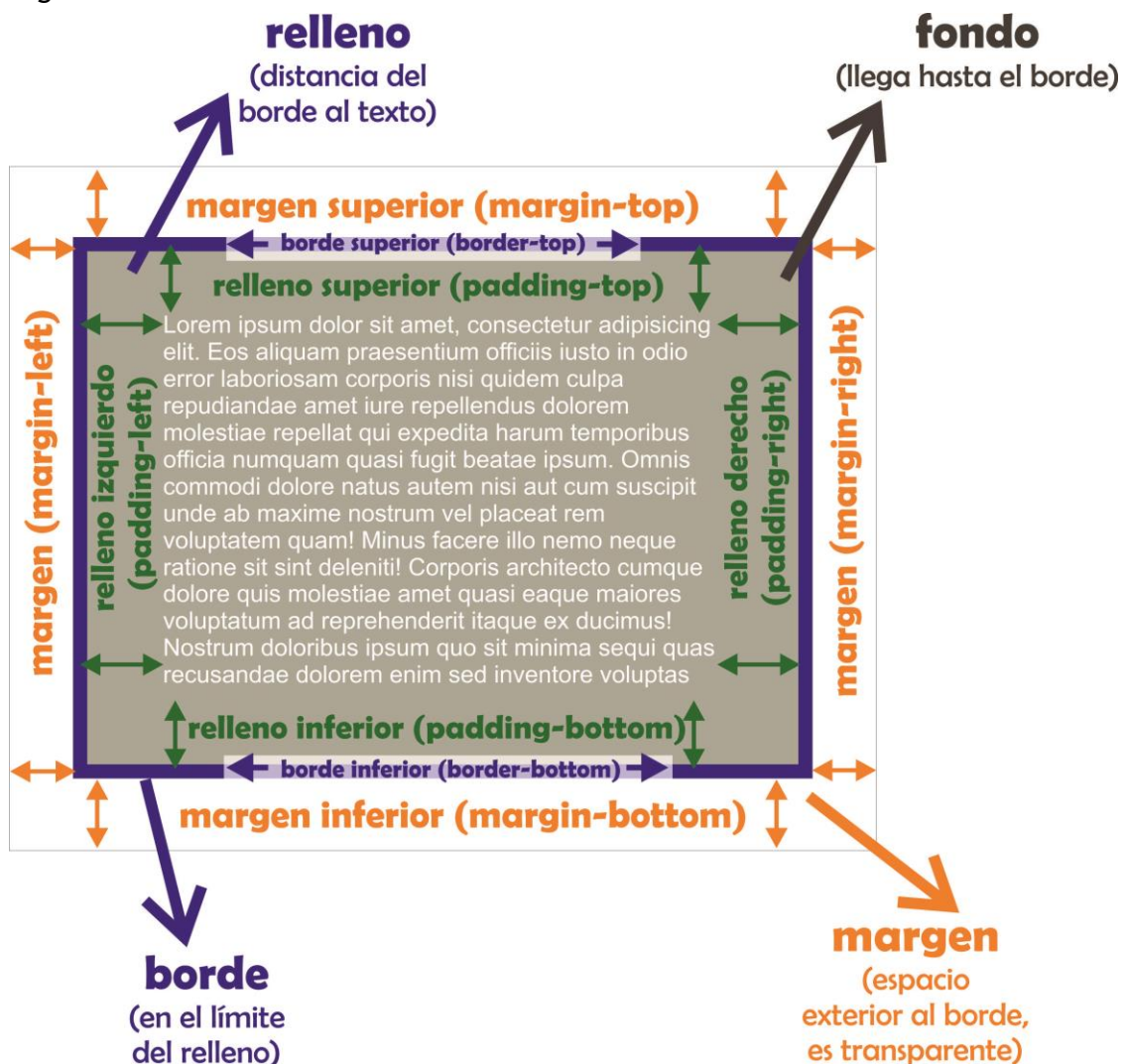


Ilustración 3, Componentes del formato de cuadro de un elemento CSS

En la Ilustración 3 se describen los elementos del formato de cuadro en CSS. En apartados anteriores ya hemos visto como configurar el fondo de un elemento. Hay que observar que dicho fondo termina en el borde del elemento.

El espacio que va desde el contenido del elemento (en la ilustración sería el texto coloreado de blanco) hasta el borde, que quedará relleno con el fondo que se haya configurado.

El margen es un espacio exterior al borde. En dicho espacio no se ve el borde, es transparente, por lo que se rellenará con el fondo del elemento contenedor (el elemento padre del actual). Es decir si el elemento es una etiqueta **p** que cuelga directamente de **body**, en el espacio el margen veremos el fondo del elemento **body** (si le hay).

### (4.11.2) configuración del borde

#### configuración del borde (CSS1 y CSS2)

El borde posee tres propiedades fundamentales: color, estilo y grosor. Además podremos configurar independientemente cada borde (izquierdo, derecho, superior e inferior). Por lo que disponemos de todas posibilidades:

■ **border-width**. Anchura del borde, indicada en la unidad deseada. Por ejemplo **12px** indicaría un borde de doce píxeles. También podemos indicar estos valores:

- **thin**. Borde fino
- **medium**. Borde de grosor medio
- **thick**. Borde grueso

También podemos indicar a la vez diferentes grosores. Para ello podemos especificar cuatro valores a la vez en lugar de uno. El orden para indicarlos es arriba, derecha, abajo e izquierda (siguiendo las agujas del reloj desde las doce). Ejemplo:

**border-width:** thick thin thick opx;

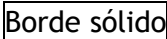





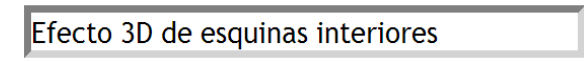
indica bordes superior e inferior gruesos, a la derecha fino y a la izquierda no dibujará ningún borde.

No es necesario indicar todos los bordes, ejemplos:

**border-width:**thick thin; /\* superior e inferior grueso, izquierdo y derecho fino \*/

**border-width:** thick thin opx; /\* superior grueso, izquierdo y derecho fino e inferior sin borde \*/

■ **border-style**. Modifica el estilo del borde. Esta propiedad afecta a todos los bordes (superior, derecho, inferior e izquierdo). Posibilidades:

- **solid**. 
- **dashed**. 
- **dotted**. 
- **double**. 
- **groove**. 
- **ridge**. 
- **inset**.  El efecto depende del color elegido



- **outset.** Efecto 3D de esquinas interiores El efecto depende del color elegido
- **none.** Sin borde.

Como ocurre con **border-width**, podemos indicar varios estilos a la vez. Ejemplos:

```
border-style:solid dashed dotted dotted; /* superior sólido, derecho rallado, inferior e izquierdo punteado */
```

```
border-style: solid dashed dotted; /* superior sólido, izquierdo y derecho rallado e inferior punteado */
```

- **border-color.** Color del borde. Al igual que en las dos propiedades anteriores podemos asignar colores distintos a cada borde (siguiendo lo ya comentado).
- **border-top-width, border-top-color, border-top-style.** Indica respectivamente la anchura, el color y el estilo del borde superior.
- **border-left-width, border-left-color, border-left-style.** Indica respectivamente la anchura, el color y el estilo del borde izquierdo.
- **border-bottom-width, border-bottom-color, border-bottom-style.** Indica respectivamente la anchura, el color y el estilo del borde inferior.
- **border-right-width, border-right-color, border-right-style.** Indica respectivamente la anchura, el color y el estilo del borde derecho.
- **border-top.** Permite indicar a la vez las tres propiedades para el borde izquierdo. El orden es: grosor, estilo y color. Ejemplo:

```
border-top: 2px solid red;
```

No estamos obligados a rellenar las tres propiedades podemos rellenar sólo una o dos.

- **border-right.** Permite indicar, a la vez, las tres propiedades del borde derecho.
- **border-bottom.** Permite indicar, a la vez, las tres propiedades del borde inferior
- **border-left.** Permite indicar, a la vez, las tres propiedades del borde izquierdo
- **border.** Permite indicar el grosor, estilo y color de los cuatro bordes a la vez. Se usa igual que las anteriores pero en este caso lo que configuremos implica a todos los bordes.

### bordes redondeados (CSS3)

En CSS3 ha aparecido la posibilidad de crear bordes redondeados así las esquinas de los bordes pueden formar un radio de un círculo cuyo tamaño podremos especificar. Es posible incluso indicar diferentes redondeados para las distintas esquinas.



La pega es que es un formato CSS3 por lo que algunos navegadores no lo reconocen. Propiedades relacionadas:

- **border-radius.** Permite indicar bordes redondeados (también afecta al sombreado que saldrá con los bordes redondeados. Ejemplo de uso:

```
p{  
  background-color: lightgray;  
  border:thick double black;  
  border-radius:10px;  
}
```

Resultado:

Lorem ipsum dolor sit amet, consectetur adipisicing elit. Mollitia excepturi earum veritatis sint aperiam quam expedita adipisci alias vitae explicabo ipsa consectetur ratione quae dolorum repudiandae minus veniam nisi consequuntur.

Permite indicar a la vez varios tamaños de redondeo en el orden de izquierda a derecha y de arriba abajo. Ejemplo:

```
p{  
  background-color: lightgray;  
  border:thick double black;  
  border-radius:10px 5px 0px 20px;  
}
```

Lorem ipsum dolor sit amet, consectetur adipisicing elit. Mollitia excepturi earum veritatis sint aperiam quam expedita adipisci alias vitae explicabo ipsa consectetur ratione quae dolorum repudiandae minus veniam nisi consequuntur.

- **border-bottom-left-radius.** Permite indicar un tamaño de radio para redondear el borde inferior izquierdo.
- **border-bottom-right-radius.** Permite indicar un tamaño de radio para redondear el borde inferior derecho.
- **border-top-left-radius.** Permite indicar un tamaño de radio para redondear el borde superior izquierdo.
- **border-top-right-radius.** Permite indicar un tamaño de radio para redondear el borde superior derecho.

## bordes usando imágenes

También en CSS3 ha aparecido la posibilidad de indicar una imagen para que cubra el borde. La idea es crear una imagen (el formato idóneo es PNG) en la que se pinten los bordes. Se estirará la imagen en la zona que no son las esquinas (el tamaño de las esquinas se puede indicar) para cubrir todo el tamaño del elemento que se está bordeando con la imagen.

Normalmente la imagen tiene solamente coloreado los bordes y el resto de la imagen son píxeles que se considerarán transparentes (por eso el formato más habitual para utilizar con este tipo de imagen es el PNG).

Un ejemplo de imagen preparada para pintarse como borde es esta:



Todo lo que aparece de blanco en la imagen, en realidad tiene color transparente y así a través de ellos se observará el fondo.

La idea es indicar la parte de la imagen dedicada a las esquinas usando su tamaño (en la imagen cada cuadrado de las esquinas mide 18 píxeles). De esa forma la imagen quedará troceada en nueve partes (las cuatro esquinas y la parte superior, inferior, izquierda, derecha y central de la imagen).

Para la zona que no es la esquina, indicaremos como haremos para rellenar completamente el elemento. Es decir, en la imagen anterior, los círculos azules no valen para rellenar el elemento completo al que estemos dando borde con la imagen. Por ello indicaremos al navegador que repita cada círculo (**round**), o bien que le estire (**stretch**).

Sintaxis de la propiedad **border-image**:

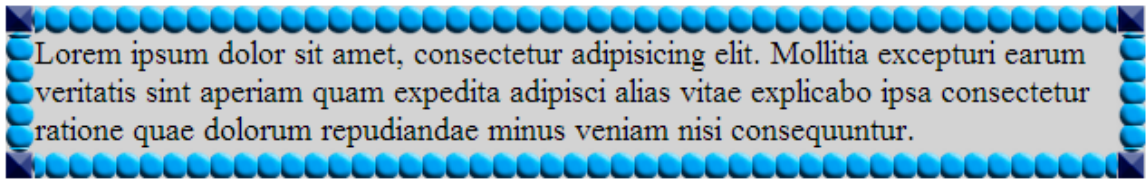
**border-image:** origenDeLaImagen tamañoEsquinas modoEstirar;

- **origenDeLaImagen**. Es la ruta a la imagen
- **tamañoEsquinas**. Es lo que mide cada esquina empezando por la superior izquierda y luego yendo hacia la derecha y hacia abajo. Si miden lo mismo, sólo se indica un número (normalmente no se indica la unidad porque siempre se entiende en píxeles)
- **modoEstirar**. Puede ser **round** (repetir hasta llenar) o **stretch** (estirar). Podemos indicar dos valores, el primer se referirá a la forma de rellenar en horizontal y el segundo será en vertical.

Ejemplos (suponiendo que la imagen anterior es borde2.png y que los cuadraditos miden 18 píxeles x 19 píxeles):

```
p{  
  background-color: lightgray;  
  border-width:15px;  
  -webkit-border-image: url(borde2.png) 18 19 round; /*para Safari*/  
  -o-border-image: url(borde2.png) 18 19 round; /* para Opera */  
  border-image: url(borde2.png) 18 19 round;  
}
```

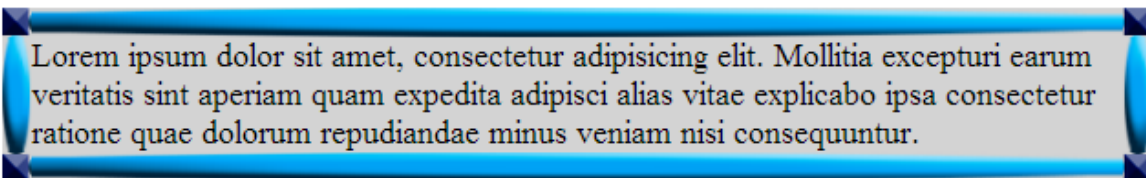
Resultado:



Otro:

```
p{  
  background-color: lightgray;  
  border-width:15px;  
  -webkit-border-image: url(borde2.png) 18 19 stretch; /*para Safari*/  
  -o-border-image: url(borde2.png) 18 19 stretch ; /* para Opera */  
  border-image: url(borde2.png) 18 19 stretch ;  
}
```

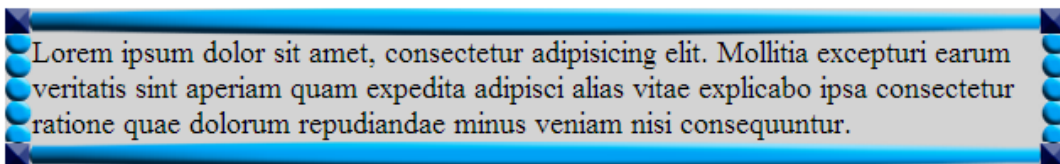
Resultado:



Y otro:

```
p{  
  background-color: lightgray;  
  border-width:15px;  
  -webkit-border-image: url(borde2.png) 18 19 stretch round; /*para  
Safari*/  
  -o-border-image: url(borde2.png) 18 19 stretch round; /* para Opera */  
  border-image: url(borde2.png) 18 19 stretch round;  
}
```

Resultado:



## (4.12) tablas

Las tablas, por su particularidad poseen propiedades que, aunque se pueden utilizar con cualquier elemento, están pensadas específicamente para ellas.

El manejo de tablas es un poco distinto con CSS. La anchura de las tablas sigue pudiéndose con la propiedad **width** de la etiqueta **table** y los bordes de la tabla se manejan también con las propiedades HTML.

Muchos diseñadores trabajan de esta forma con HTML indican la anchura de la tabla y la posibilidad de tener bordes (propiedad **border** de HTML) y con CSS se pueden establecer propiedades concretas para los bordes (como **border-color**, **border-style** o **border-width**).

Otra opción (quizá más coherente aunque falle en algunos navegadores) es usar sólo CSS. Aunque frustre un poco (sobre todo al principio) la forma en la que se comportan los navegadores.

Las propiedades CSS específicas para tablas son:

- **border-collapse**. Permite indicar cómo manejar los bordes adyacentes entre elementos. Posibles valores:
  - **collapse**. Se unen los bordes adyacentes para formar un único borde
  - **separate**. Los bordes adyacentes se mantienen separados. Es el valor por defecto.

Ejemplo:

```
table{  
  width:100%;  
}  
td{  
  border:2px solid black;  
}
```


La tabla se muestra en modo **separate**. Sin embargo indicando modo **collapse** en la etiqueta **table**:

```
table{  
  width:100%;  
  border-collapse:collapse;  
}  
td{  
  border:2px solid black;  
}
```

Resultado:


Hay que fijarse en que es en el elemento contenedor (**table** porque es el contenedor de toda la tabla) en el que se indica esta propiedad.

- **border-spacing.** Indica el espacio entre bordes. Admite dos medidas, la primera se utiliza para la distancia horizontal y la segunda para la vertical. Si se utiliza una medida se referirá a ambas distancias. Ejemplo:

**border-spacing:10px 5px;**

Ejemplo:

```
<!doctype html>
<html lang="es">
  <head>
    <meta charset="UTF-8">
    <title>Document</title>
    <style>

      table{
        width:100%;
        border-spacing: 10px 25px;
        border:2px solid black;
      }
      td{
        border:2px solid black;
      }

    </style>
  </head>
  <body>
    <table>
      <tr>
        <td>&nbsp;</td>
        <td>&nbsp;</td>
        <td>&nbsp;</td>
        <td>&nbsp;</td>
      </tr>
      <tr>
        <td>&nbsp;</td>
        <td>&nbsp;</td>
        <td>&nbsp;</td>
        <td>&nbsp;</td>
      </tr>
    </table>
  </body>
</html>
```

```
<tr>
  <td>&nbsp;</td>
  <td>&nbsp;</td>
  <td>&nbsp;</td>
  <td>&nbsp;</td>
</tr>
</table>
</html>
```

Resultado:


- **caption-side**. Propiedad utilizada para la etiqueta **caption** que contiene el título de una tabla. Permite indicar si queremos que el título esté por encima de la tabla (valor **top**, valor por defecto) o por debajo (valor **bottom**)
- **empty-cells**. Permite especificar si los bordes y sombreados de la tabla se aplican a las celdas vacías. Valores:
  - **show**. Se aplican bordes y sombreados a esas celdas (valor por defecto)
  - **hide**. No se aplican los bordes y sombreados a las celdas vacías.
- **table-layout**. Indica la forma en la que se determina la anchura de la tabla y las celdas. Posibles valores:
  - **auto**. Valor por defecto. Aunque hayamos indicado anchura para tabla y/o celdas, su tamaño se determina por el contenido de las mismas (si el contenido requiere mayor anchura, así se hará).
  - **fixed**. Predomina la anchura establecida por las propiedades y no por el contenido

## (4.13) posicionamiento

### (4.13.1) introducción

Una de las grandes peticiones de los diseñadores web era la posibilidad de poder colocar contenido en una posición concreta de la página, haciendo que los contenidos no se vayan mostrando en el orden en el que aparecen en el código, sino que podamos indicar una posición concreta.

Fue la principal aportación de CSS2 y actualmente se considera que la labor de maquetar la página que antes hacían las tablas, ahora se puede establecer de mejor manera con estas propiedades. Estas propiedades dan lugar a lo que se conoce como **capas**, elementos que permiten una distribución de contenidos más libre. La etiqueta **div** suele ser la encargada de crear tablas, aunque con HTML5 se empieza a aconsejar que sean otros elementos los que hagan esta labor, ya que **div** no es una etiqueta

semántica (no dice qué tipo de contenido posee, si es un artículo, un título, un menú,...).

### (4.13.2) flotación

La propiedad **float** es muy útil para contenidos multimedia como imágenes o tablas. Permite indicar cómo se comportan los elementos siguientes sobre aquel al que le establecemos la propiedad.

Normalmente todos los elementos **no flotan**. Es decir cuando colocamos, por ejemplo, una imagen, el contenido que va a continuación aparece en su línea base, dejando un hueco poco estético.

Podemos hacer que ese contenido contornee la imagen a su derecha (**float:left**) o a su izquierda (**float:right**). Posibles valores de la propiedad **float**:

- **left**. El elemento **flota** a la izquierda. El contenido inmediatamente siguiente le contornea a su derecha.
- **right**. El elemento **flota** a la derecha. El contenido inmediatamente siguiente le contornea a su izquierda.
- **none**. El elemento no flota (valor por defecto).



Ilustración 4, Imagen sin flotación (**float:none**), el texto aparece después

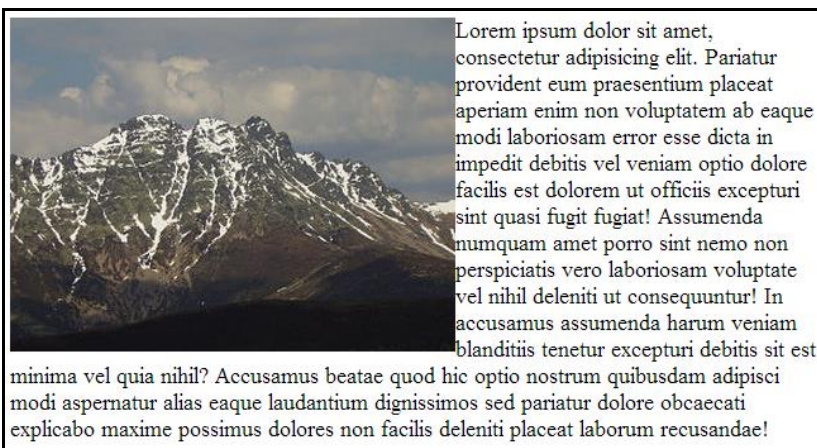


Ilustración 5, Imagen con flotación izquierda (**float:left**)



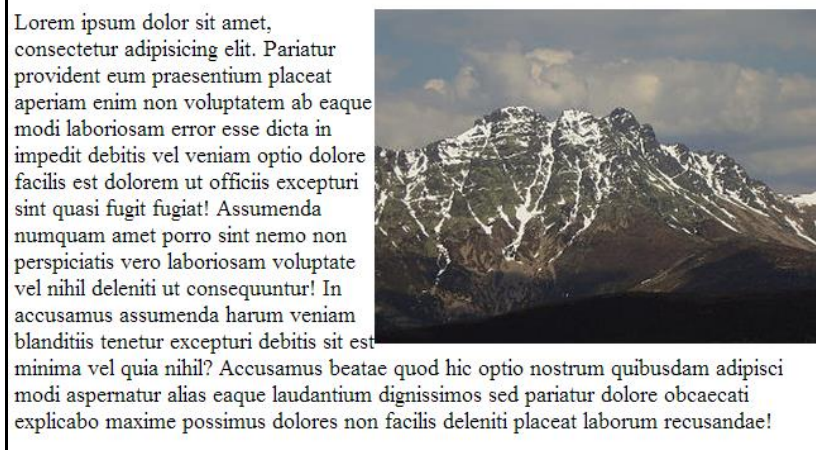


Ilustración 6, Imagen con flotación derecha (*float:right*)

### (4.13.3) tamaño del elemento

Dos propiedades permiten establecer el tamaño de un elemento:

- **width**. Anchura del elemento. Su valor por defecto es **auto**, lo que implica que el elemento se hará lo suficientemente ancho para que quepa su contenido. De otro modo indicaremos su tamaño mediante un tamaño exacto (en píxeles) o en porcentaje.  
El porcentaje se referirá a la anchura de su contenedor, es decir si indicamos un 50%, estaremos diciendo que el elemento ocupa la mitad del espacio disponible en el elemento padre del actual. Si se lo decimos a un elemento **p** que está dentro de **body**, dicho elemento ocupará la mitad total de la anchura de la página.
- **height**. Altura del elemento. Tiene las mismas posibilidades que el elemento anterior.

### (4.13.4) modo de posicionamiento

La propiedad **position** es la fundamental para establecer el posicionamiento, permite indicar si el elemento al que se aplica realmente se comportará como una capa de contenidos que se colocará libremente en la página. Posibles valores:

- **static**. Es el valor por defecto. Si se establece así, el elemento al que se aplique aparece en la página en la posición que dicte el flujo natural (de izquierda a derecha y de arriba abajo) del código. Es decir, ignorará las propiedades de posicionamiento (**left**, **top**,...)
- **fixed**. La posición del elemento se fija en base a las coordenadas de la ventana, independientemente de dónde esté contenido el elemento. Es decir toma como referencia la ventana del documento de modo que la esquina superior izquierda será la coordenada 0,0.
- **relative**. Funciona como **static** sólo que permite flotar el contenido las coordenadas elegidas. En modo **relative**, el elemento marcado con este posicionamiento, deja el hueco que le correspondería en modo **static**. No se usa mucho en maquetaciones de páginas
- **absolute**. Funciona como **fixed**, salvo que toma como referencia de las coordenadas, los límites del componente que contiene al actual.

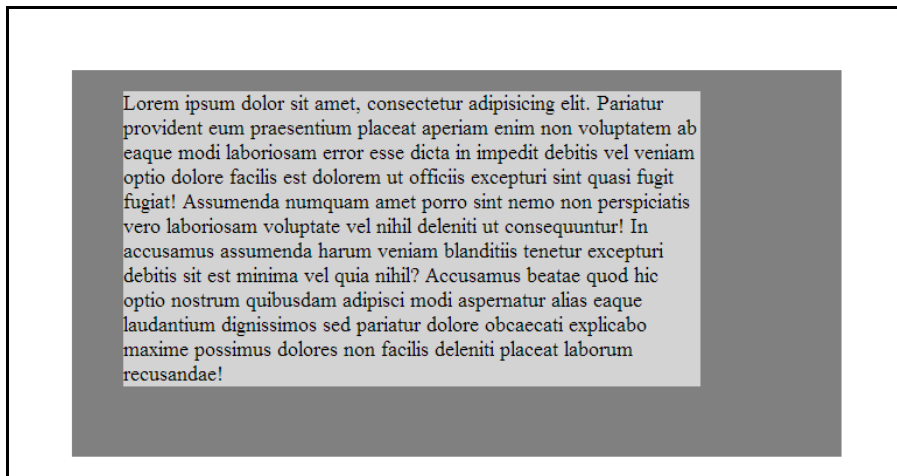


Es decir si tenemos un elemento (por ejemplo de tipo **div**) llamado *interior* dentro de otro llamado *padre*; si utilizamos **position:absolute**; dentro del CSS de la capa interior, entonces las coordenadas de la capa interior tomarán los límites de la capa padre como referencia. Poniendo en la capa interior: **left:30px**; separamos esta capa treinta píxeles del margen izquierdo de la capa padre (en lugar de separarnos del borde de la ventana como hace **fixed**).

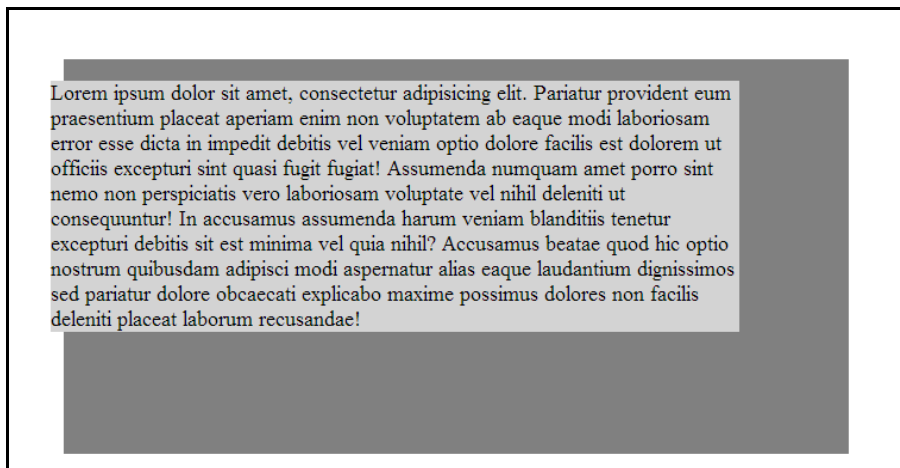
Ejemplo (posicionamiento relativo):

```
<!doctype html>
<html lang="es">
<head>
  <meta charset="UTF-8">
  <title>Document</title>
  <style>
    div{
      position:fixed;
      left:50px;
      top:50px;
      width:600px;
      height:300px;
      background-color: gray;
    }
    p{
      position:absolute;
      background-color: lightgray;
      width:75%;
      left:40px;
    }
  </style>
</head>
<body>
  <div>
    <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit. Pariatur
    provident eum praesentium placeat aperiam enim non voluptatem ab eaque modi
    laboriosam error esse dicta in impedit debitis vel veniam optio dolore facilis est
    dolorem ut officiis excepturi sint quasi fugit fugiat! Assumenda numquam amet
    porro sint nemo non perspiciatis vero laboriosam voluptate vel nihil deleniti ut
    consequuntur! In accusamus assumenda harum veniam blanditiis tenetur excepturi
    debitis sit est minima vel quia nihil? Accusamus beatae quod hic optio nostrum
    quibusdam adipisci modi aspernatur alias eaque laudantium dignissimos sed
    pariatur dolore obcaecati explicabo maxime possimus dolores non facilis deleniti
    placeat laborum recusandae!</p>
  </div>
</body>
</html>
```

Mostraría este resultado:



Si modificamos el CSS y cambiamos la línea remarcada a **position:fixed**:



Ahora el párrafo se sale de su contenedor porque las coordenadas se refieren a la ventana y no se calculan a partir de la etiqueta **div** que contiene al párrafo.

#### **(4.13.5) establecer coordenadas de posicionamiento**

Cuatro propiedades se encargan de mostrar un elemento en una posición exacta en la página:

- **left**. Permite indicar la coordenada izquierda del elemento. Tomada desde el margen izquierdo
- **top**. Coordenada superior. Tomada desde el borde superior.
- **bottom**. Coordenada inferior. Distancia al borde inferior
- **right**. Coordenada derecha. Distancia al borde derecho.

Las coordenadas se calculan desde la referencia de coordenadas establecidas por la propiedad **position**. Si es **fixed** se toma los bordes de la ventana. Es incompatible usar **bottom** y **right** y, a la vez, los tamaños con **width** y **height**.

Ejemplo:

```
<!doctype html>
<html lang="es">
  <head>
    <meta charset="UTF-8">
    <title>Document</title>
    <style>
      div{
        position: fixed;
        background-color: yellow;

        left:50px;
        top:120px;
        right:80px;
        bottom:100px;
      }
    </style>
  </head>
  <body>
    <div>
      <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit. Pariatur
provident eum praesentium placeat aperiam enim non voluptatem ab eaque modi
laboriosam error esse dicta in impedit debitis vel veniam optio dolore facilis est
dolorem ut officiis excepturi sint quasi fugit fugiat! Assumenda numquam amet
porro sint nemo non perspiciatis vero laboriosam voluptate vel nihil deleniti ut
consequuntur! In accusamus assumenda harum veniam blanditiis tenetur excepturi
debitis sit est minima vel quia nihil? Accusamus beatae quod hic optio nostrum
quibusdam adipisci modi aspernatur alias eaque laudantium dignissimos sed
pariatur dolore obcaecati explicabo maxime possimus dolores non facilis deleniti
placeat laborum recusandae!</p>
    </div>
  </body>
</html>
```

Resultado:



Ilustración 7, Funcionamiento de las coordenadas de posicionamiento en modo *fixed*

#### (4.13.6) coordenada **z**

La pantalla es un medio en dos dimensiones y por tanto la posición de los elementos se indica con las coordenadas **x** e **y**. Sin embargo se puede simular la tercera dimensión (la profundidad) mediante una coordenada ficticia **z**.

Realmente lo que aporta es la posibilidad de que los elementos se solapen y podamos decidir qué elemento está por encima, de modo que el que tenga la coordenada **z** más alta diríamos que está más cerca del espectador y por lo tanto tiene prioridad; es decir, quedará **encima** de los otros.

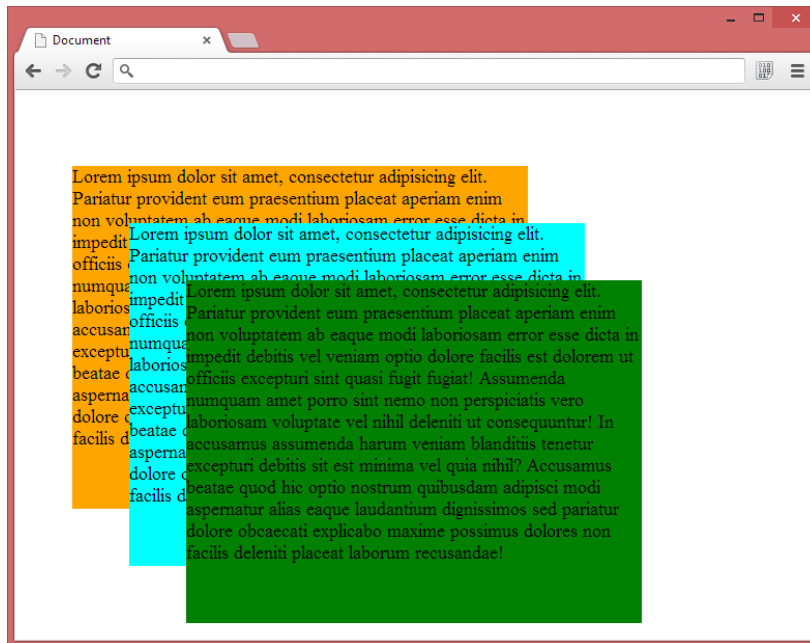
Es el mismo efecto que logran los sistemas operativos gráficos como Windows u OS X cuando tenemos dos ventanas solapadas; una de ellas tapa a la otra entendiendo que está **por delante**.

La propiedad que permite controlar la coordenada **z** es **z-index** a la que se le otorga un número entero de modo que cuanto mayor sea, más quedará por delante de los otros elementos. El texto normal (de elementos que no tengan marcado un **z-index**) está en el nivel cero; con lo cual, si indicamos un nivel negativo en la propiedad **z-index** de un elemento, entenderemos que por detrás del contenido normal (no se verá el elemento, quedará tapado).

Ejemplo:

```
#capa1{
  position:fixed;
  left:50px;
  top:50px;
  width:400px;
  height:300px;
  background-color:cyan;
  z-index:1;
}
#capa2{
  position:fixed;
  left:100px;
  top:100px;
  width:400px;
  height:300px;
  background-color:cyan;
  z-index:1;
}
#capa3{
  position:fixed;
  left:150px;
  top:150px;
  width:400px;
  height:300px;
  background-color:cyan;
  z-index:1;
}
```

Si creamos tres elementos con esos identificadores y el mismo contenido, el resultado es:



#### (4.13.7) propiedades de visibilidad

Hay dos propiedades que se ocupan de la visibilidad del contenido de las capas, son:

- **overflow.** Permite decidir la política que debe adoptar el navegador cuando un contenido rebasa el tamaño previsto de su contenedor. Posibles valores:
  - **visible.** El contenido se muestra aun cuando rebase el tamaño previsto.

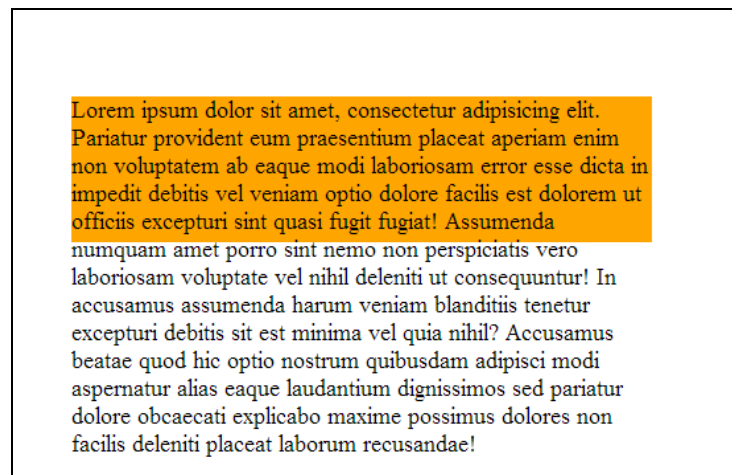


Ilustración 8, *overflow: visible*, el texto rebasa el tamaño de la capa pero se muestra

- **hidden.** Oculta el contenido que rebasa el tamaño prefijado para la capa:

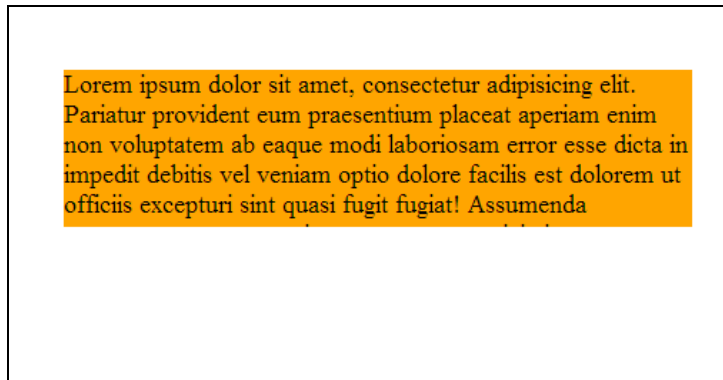


Ilustración 9, *overflow:hidden*, sólo se muestra el texto que cabe en la capa

- **scroll.** Se muestra barras de desplazamiento para poder acceder al contenido que supera el tamaño establecido para la capa.

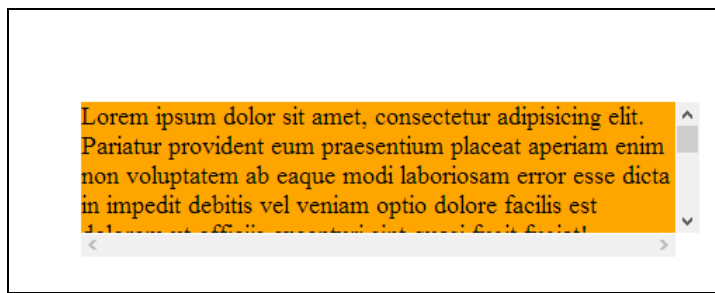


Ilustración 10, *overflow:scroll*, una barra de desplazamiento permite acceder al texto oculto

- **auto.** Es igual que el anterior, pero las barras sólo se muestran si el contenido no cabe en el tamaño fijado para la capa; si sí cabe no se muestra barra de desplazamiento alguna.
- **visibility.** Permite ocultar entero el elemento y su contenido.
  - **visible.** El elemento y su contenido se muestran. Es la opción por defecto.
  - **hidden.** El elemento queda oculto. Se emplea para hacer efectos.
- **clip.** Permite recortar el contenido de un elemento. Para ello indica un rectángulo con cuatro coordenadas: **top**, **right**, **bottom**, **left** mediante la función **rect**. Todo el contenido fuera de esas coordenadas quedará oculto.



Ejemplo de uso:

```
#capa1{  
  position:fixed;  
  left:50px;  
  top:50px;  
  width:400px;  
  height:250px;  
  clip:rect(20px,170px,70px,50px);  
  background-color:orange;  
  z-index:2;  
}
```

Resultado:



Ilustración 11, Ejemplo de recorte

## (4.14) propiedad display

**display** es una propiedad aparecida en CSS2 que permite establecer la forma en la que se muestra por pantalla un elemento. Está especialmente indicada para documentos distintos a HTML (XML por ejemplo) en los que el contenido no tiene por defecto una forma de mostrarse.

En HTML permite decidir la forma en la que se muestra un elemento. Por ejemplo, en HTML los elementos de tipo **p** (párrafos) se entienden que se mostrarán como párrafos independientes, es decir, con cada etiqueta **p** comienza un nuevo párrafo o bloque. No irán seguidos dos elementos de párrafo consecutivos.

Sin embargo **display** puede cambiar la forma de funcionar de un elemento, por ejemplo en esta página:

```
<!doctype html>
<html lang="es">
  <head>
    <meta charset="UTF-8">
    <title>Propiedad display</title>
    <style>
      p{
        display:inline;
      }
    </style>
  </head>
  <body>
    <p>Uno</p>
    <p>Dos</p>
    <p>Tres</p>
    <p>Cuatro</p>
    <p>Cinco</p>
    <p>Seis</p>
  </body>
</html>
```

Se mostrará el contenido de esta forma:

Uno Dos Tres Cuatro Cinco Seis

Ya que **display** considera ahora que la etiqueta **p** irá seguida en la misma línea (por su valor **inline**).

Los posibles valores de la propiedad **display** son:

- **none**. El elemento se muestra a su manera habitual
- **block**. El elemento se muestra como bloque separado (al igual que el elemento **p**).
- **inline**. Elemento que se muestra seguido del anterior y el siguiente (como el elemento **span**)

- **list-item**. El elemento se considerará un ítem de una lista (al igual que la etiqueta **li**)
- **box**. (También vale **flex-box**), muestra el elemento
- **table**. Considera al elemento una tabla
- **table-caption**. Considera al elemento un título de tabla
- **table-cell**. Considera al elemento una celda de una tabla
- **table-row**. Considera que el atributo es una fila.
- **table-row-group**. Considera al elemento un grupo de filas.
- **table-column**. Considera al elemento una columna
- **table-column-group**. Considera al elemento un grupo de columnas.
- **table-footer-group**. Considera al elemento un pie de tabla (como **tfoot**)
- **table-header-group**. Considera al elemento una cabecera de tabla (como **thead**)
- **inline-table**. Elemento **inline** interior a una tabla.
- **inline-block**. El elemento forma un bloque, pero que es interior a su contenedor. Es como si fuera el cuadro de una imagen.

Ejemplo: (**inline-block**):

```
<!doctype html>
<html lang="es">
  <head>
    <meta charset="UTF-8">
    <title>Propiedad display</title>
    <style>
      #marco{
        background-color:lightgray;
        border:2px solid black;
        display:inline-block;
        width:50%;
      }
      p{
        border:1px solidgray;
      }
    </style>
  </head>
  <body>
    <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit. Enim aliquam iste culpa
    aperiam deserunt quia aliquid qui saepe quisquam rem consequatur necessitatibus
    voluptates laborum architecto quaerat et facilis nobis eveniet quae officiis. Harum
    temporibus cupiditate asperiores
```

```
<span id="marco">
    Lorem ipsum dolor sit amet, consectetur adipisicing elit.
    Numquam
    laudantium non iure soluta rem ipsa hic tenetur dolores aspernatur
    natus eius praesentium quod iusto consequuntur aut quo commodi
    repellendus fuga?
</span>
arum rerum in quaerat deserunt numquam recusandae nam sed nostrum vel odit
voluptates eius animi enim vitae iure deleniti quasi nulla voluptas reiciendis ullam
mollitia molestiae voluptatibus praesentium nihil. Quas atque commodi molestiae
facilis architecto dicta sit quia aliquam perspiciatis expedita voluptatem veritatis
rerum nulla mollitia accusantium! Quod temporibus explicabo aliquid a officiis
labore sequi nesciunt commodi eos nostrum distinctio reprehenderit? Consectetur
obcaecati ad! deserunt totam officiis nesciunt voluptas saepe suscipit quo
nostrum voluptatum voluptatem quod! Assumenda nobis consequatur blanditiis iure
doloremque fuga excepturi aut.</p>
<p>Lorem ipsum dolor sit amet, consectetur adipisicing elit. Sequi beatae
obcaecati assumenda temporibus voluptate aspernatur vitae? Alias a libero nisi
aspernatur asperiores ea deserunt molestias obcaecati facere consectetur voluptates
excepturi!</p>
</body>
</html>
```

Resultado:

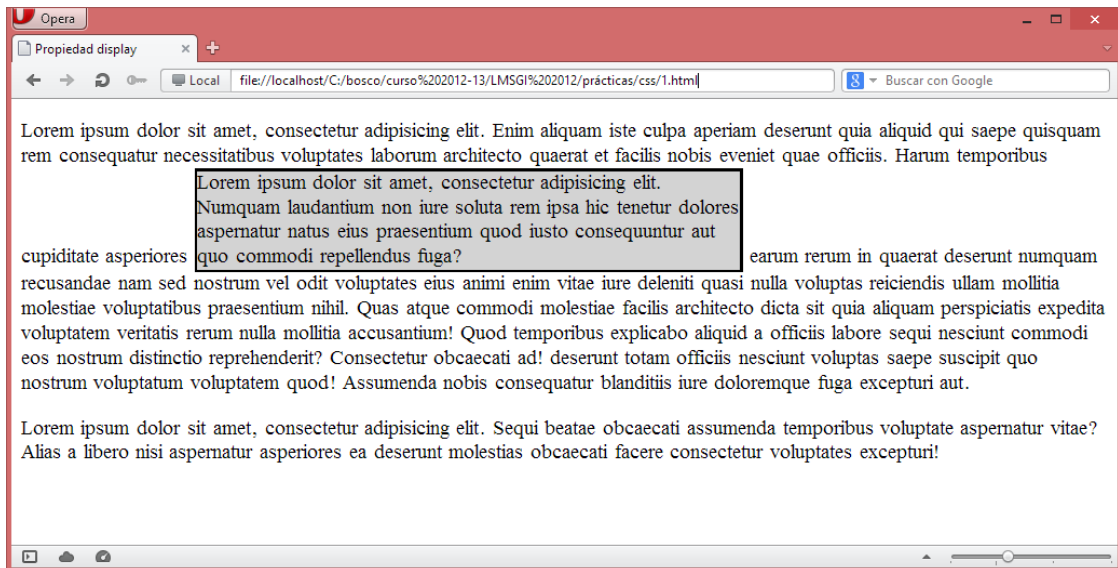


Ilustración 12, propiedad **display** con valor **inline-block**, el cuadro de color gris es un elemento **span** al que se le ha dado categoría de bloque interior.

Otro ejemplo (creación de tablas sin usar la etiqueta **table**):

```
<!doctype html>
<html lang="es">
  <head>
    <meta charset="UTF-8">
    <title>Propiedad display</title>
    <style>
```

```
    div{
        display:table;
    }
    div div{
        display:table-row;
    }

    div div div{
        display:table-cell;
    }
```

```
</style>
</head>
<body>
  <div id="contenedor">
    <div id="fila1">
      <div id="c1">
```

Lorem ipsum dolor sit amet, consectetur adipisicing elit. Voluptas officia neque tempora vel commodi aliquam sapiente officiis quae atque ducimus ex mollitia obcaecati expedita tempore possimus doloribus sed minima deleniti aspernatur recusandae eligendi quia ipsam. Sunt architecto necessitatibus atque quis eaque omnis voluptatum pariatur facere sequi rerum voluptatem animi commodi.

```
      </div>
      <div id="c2">
```

Eius iusto ipsum iste voluptatibus quis eaque placeat labore voluptates voluptas debitis suscipit doloremque ex facilis aliquam aperiam voluptatum quas! Architecto aliquid necessitatibus temporibus veritatis blanditiis quaerat. Sequi totam doloremque tenetur distinctio aliquid vitae eius similique.

```
      </div>
      <div id="c3">
```

Ullam tempora quam eveniet consequatur accusantium explicabo quisquam molestias facere sed aliquid. Molestiae ut esse rem nostrum praesentium voluptatibus modi culpa asperiores obcaecati magni dolores nobis delectus ratione odio consequatur consectetur ipsam atque enim ipsa exercitationem optio quo est nemo quam et quis porro blanditiis repudiandae laborum aliquam consequuntur non expedita provident nulla sint maiores quae! Consectetur consequuntur illo voluptas reprehenderit. Odio doloremque architecto doloribus quidem in at quae ab!

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

Resultado:

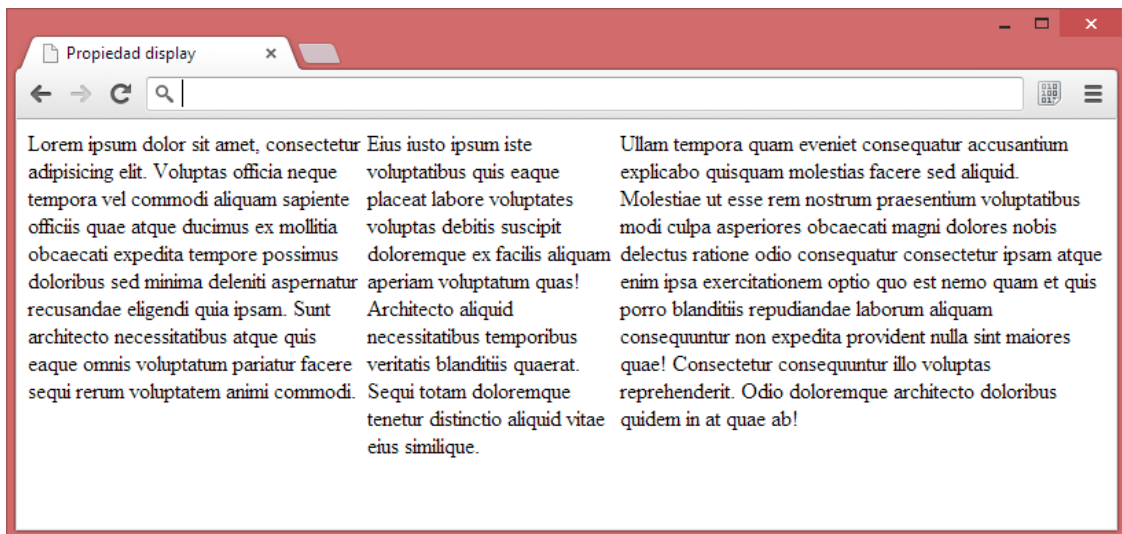


Ilustración 13, Columnas conseguidas gracias al uso de *display* con *table*, *table-row* y *table-cell*

## (4.15) formatos avanzados de caja (box)

Todos ellos son parte de CSS3, por lo que no todos los navegadores los soportan.

- **box-shadow.** Soportada por todos los navegadores actuales. Permite añadir una sombra al elemento. Dicha sombra se muestra por fuera de los bordes (en el área de margen). Para establecer la sombra hay que indicar:
  - **Color de la sombra.**
  - **Desplazamiento en el eje X.**
  - **Desplazamiento en el eje Y.**
  - **Desenfoque de la sombra.** Es una cantidad (normalmente en píxeles) que permite indicar cuánto queremos desenfocar. Es opcional

Ejemplo:

```
p{  
  background-color: lightgray;  
  margin:30px;  
  box-shadow: gray 10px 10px 5px;  
}
```

Resultado:

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Mollitia excepturi earum veritatis sint aperiam quam expedita adipisci alias vitae explicabo ipsa consectetur ratione quae dolorum repudiandae minus veniam nisi consequuntur.

- **overflow-x.** Permite especificar lo que ocurre en el caso de que el contenido de un elemento sea más ancho que el elemento que le contiene (porque el elemento tienen una anchura menor). Posibilidades:
  - **visible.** Se muestra el contenido aunque sobrepase la anchura del elemento.
  - **hidden.** Se oculta el contenido que sobrepase la anchura del elemento.
  - **scroll.** Se muestra barra de desplazamiento horizontal para mostrar el elemento.
  - **auto.** Si la anchura es insuficiente para mostrar el contenido se mostrará una barra de desplazamiento.
- **overflow-y.** Hace lo mismo que el anterior, pero ahora referido a la altura del elemento. Es decir, decide lo que hay que hacer en el caso de que el contenido de un elemento sea mayor en el eje vertical que el elemento que lo contiene. Las posibilidades son las mismas que en el caso anterior.
- **overflow-style.** Indica qué tipo de desplazamiento se usará en el caso de que en alguno, en ambos, casos anteriores se haya decidido un scroll. Posibilidades:
  - **scrollbar.** Barra de desplazamiento normal.
  - **marquee.** El contenido se desplaza automáticamente el sólo para mostrarse por completo (al estilo de una marquesina).
  - **move.** El usuario podrá arrastrar el contenido sin usar barras de desplazamiento para mostrar el contenido.

Ningún navegador actual soporta esta propiedad. De hecho, hoy en día son los propios navegadores los que deciden cuál es el método idóneo de desplazamiento. No está nada claro que esta propiedad llegue a tener cabida en los navegadores.

- **rotation.** Propiedad (aún no soportada), que permite indicar un ángulo de rotación para el contenido.
- **rotation-point.** Al igual que la anterior, no está siendo soportada por ningún navegador. Permitiría establecer un punto de rotación (el punto alrededor del cual se giraría el contenido) indicando sus coordenadas.



## (4.16) generación de contenido

### (4.16.1) propiedad *content*

Con CSS2 apareció la posibilidad de que el propio CSS estableciera el contenido de una capa. Esto permite que el propio CSS escriba contenido HTML, cambiando un poco la óptica que se tiene de CSS en cuanto a ser un lenguaje que da formato.

La propiedad relacionada con esta facultad es:

- **content**. Según el estándar sólo se puede utilizar usando los selectores **:before** y **:after** que permiten establecer contenido que se mostrará antes y después del elemento al que se aplica. Ejemplo:

```
<!doctype html>
<html lang="es">
  <head>
    <meta charset="UTF-8">
    <title>Document</title>
    <style>
```

```
      .cita:before{
        content: "";
      }
      .cita:after{
        content: "";
      }
      .cita{
        font-style: italic;
      }
```

```
    </style>
  </head>
  <body>
    <p class="cita">Lorem ipsum dolor sit amet, consectetur adipisicing elit.
    Pariatur provident eum praesentium placeat aperiam enim non voluptatem ab eaque
    modi laboriosam error esse dicta in impedit debitis vel veniam optio dolore facilis
    est dolorem ut officiis excepturi sint quasi fugit fugiat! Assumenda numquam amet
    porro sint nemo non perspiciatis vero laboriosam voluptate vel nihil deleniti ut
    consequuntur! In accusamus assumenda harum veniam blanditiis tenetur excepturi
    debitis sit est minima vel quia nihil? Accusamus beatae quod hic optio nostrum
    quibusdam adipisci modi aspernatur alias eaque laudantium dignissimos sed
    pariatur dolore obcaecati explicabo maxime possimus dolores non facilis deleniti
    placeat laborum recusandae!</p>
    </body>
  </html>
```

En el ejemplo se establece que por delante del texto marcado con clase *cita* se pongan comillas, por detrás se hace lo mismo. Por lo que cualquier texto de clase *cita* saldrá entrecomillado y en cursiva:

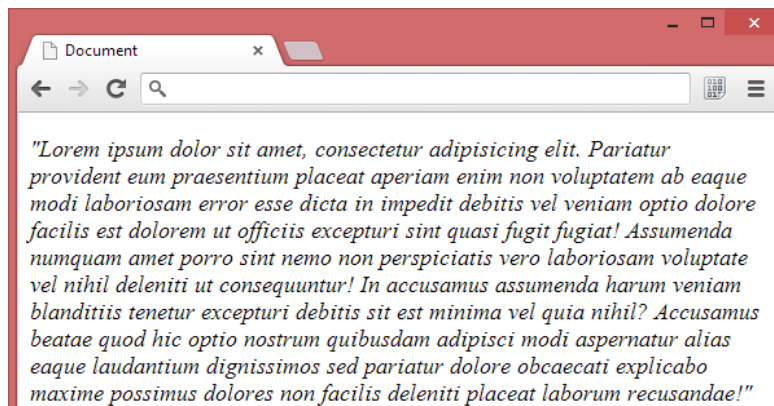


Ilustración 14, Texto entrecomillado automáticamente gracias a la propiedad *content*

Aunque se supone (según el estándar) que la propiedad *content* sólo se puede utilizar con selectores *:before* y *:after*, lo cierto es que algunos navegadores permiten que se utilicen para cualquier selector.

## (4.16.2) numeración automática

CSS2 trajo también la posibilidad de numerar automáticamente elementos gracias al uso de contadores. Para ello podemos usar una función llamada *counter* que nos permite utilizar unan *variable de contador*. La variable contadora se manipula con estas propiedades:

- **counter-reset**. Permite indicar un valor inicial para una variable contadora. Ejemplo:

```
counter-reset: cont 5;
```

En este caso la variable contadora *cont* se coloca inicialmente con el valor 5.

Si no se usa esta propiedad el contador se inicia a cero.

- **counter-increment**. Indica cómo se incremente el contador, el valor por defecto es 1. Pero podemos cambiarlo y darle el valor que deseemos:

```
counter-increment: cont 10; /* El contador irá contando de 10 en 10 */
```

Ejemplo:

```
<!doctype html>
<html lang="es">
<head>
  <meta charset="UTF-8">
  <title>Document</title>
  <style>
```

```
body{
  counter-reset:cont;
}
.cita:before{
  content: "Capítulo " counter(cont) " ";
  counter-increment: cont 2;
}
```

```
</style>
</head>
<body>
  <p class="cita">Lorem ipsum dolor sit amet, consectetur adipisicing elit.
  Impedit assumenda ratione repellat quaerat voluptate aperiam quis ipsum amet esse
  quia ullam eos quibusdam non maiores atque. Animi laborum repudiandae
  modi!maxime possimus dolores non facilis deleniti placeat laborum
  recusandae!</p>
  <p class="cita">Lorem ipsum dolor sit amet, consectetur adipisicing elit.
  Obcaecati quasi delectus quia quo mollitia ex vero veniam nostrum ipsum minima
  tenetur ad animi non soluta amet rem assumenda? Molestias officia.</p>
  <p class="cita">Lorem ipsum dolor sit amet, consectetur adipisicing elit. Nisi
  repellat mollitia numquam minima alias id debitis voluptas ut animi dolor!
  Distinctio laborum quasi quo soluta dignissimos odit nam dolores quibusdam?</p>
</html>
```

El resultado es:

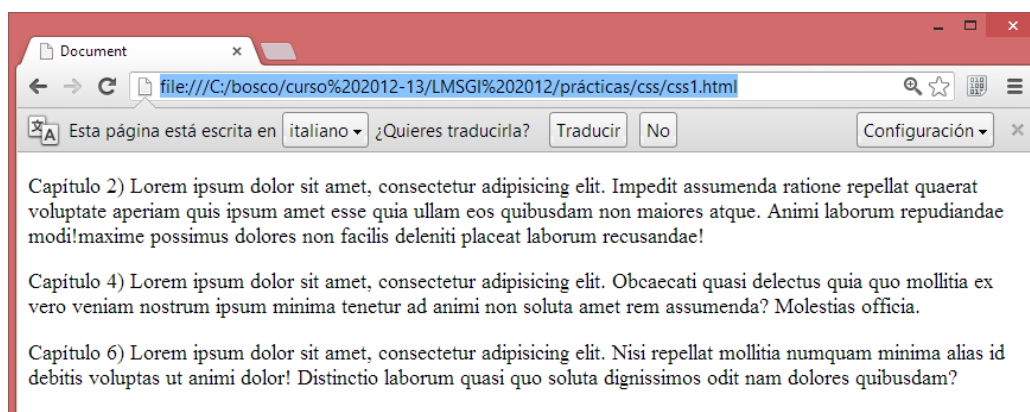


Ilustración 15, Página numerada mediante contadores

## (4.17) propiedades avanzadas de CSS3

### (4.17.1) el problema de la compatibilidad. prefijos

Debido a lo reciente de CSS3, numerosas propiedades y elementos de CSS3, no están disponibles en los navegadores. Además los propios navegadores incorporan propiedades que sólo pueden utilizar ellos.

Eso significa que la seguridad de que nos funcionen estas propiedades en todos los navegadores es complicada.

Por ello en muchas ocasiones las propiedades CSS3 tienen distintas variantes según el navegador. Es previsible que con el tiempo se use sólo la versión estándar, pero mientras tanto se usan versiones de las propiedades CSS usando un **prefijo**.

Los prefijos más utilizados actualmente son:

- **-webkit-**. Para los navegadores que usen el motor conocido como **Webkit** (como **Chrome** o **Safari**).
- **-ms-**. Para los navegadores de **Microsoft** (como **Internet Explorer**)
- **-o-**. Para el navegador **Opera**.
- **-moz-**. Para los navegadores **Mozilla** (como **Firefox**)

Así la propiedad **border-radius** para que funcione en el navegador Safari necesita de usar su propiedad específica que sería:

```
-webkit-border-radius: 12px;
```

Lo malo es que muchas propiedades para cada navegador funcionan de forma distinta, lo que nos obliga a conocer todas las variantes.

### (4.17.2) opacidad

Es una propiedad CSS3 muy soportada por los navegadores desde hace años. Impone un factor de opacidad al elemento. Ese factor se indica con un número del cero (transparencia total) a uno (totalmente opaco). Ejemplo:

```
opacity:.5; /* Deja al elemento semitransparent */
```

### (4.17.3) relleno con gradientes

Un gradiente es un degradado que permite colorear en lugar de con un color único, con una variación que va de un color a otro. CSS3 permite el uso de una función llamada **linear-gradient** que permite usarse por ejemplo en la propiedad **background-image** para establecer un gradiente.

Sintaxis de la función (estándar):

```
linear-gradient(direccion,primerColor,segundoColor,...)
```

Ejemplo:

```
background: linear-gradient(top,red,blue); /* degradado vertical del rojo al azul */  
background: linear-gradient(top,red,blue,yellow);  
/* degradado vertical del rojo al amarillo, pasando por el azul */  
background: linear-gradient(left,red,blue,yellow);  
/* El degradado ahora va de la izquierda a la derecha */  
background: linear-gradient(45deg,red,blue,yellow);  
/* El degradado tiene un ángulo de 45 grados */
```

Cada navegador tiene su versión de esta función que tiene más posibilidades. En cualquier caso, todos los ejemplos anteriores funcionan en los navegadores actuales si añadimos el prefijo correspondiente. Ejemplo:

```
p{  
  background: -webkit-linear-gradient(145deg, red, blue,yellow);  
  background: -moz-linear-gradient(145deg, red, blue,yellow);  
  background: -o-linear-gradient(145deg, red, blue,yellow);  
  background: -ms-linear-gradient(145deg, red, blue,yellow);  
  background: linear-gradient(145deg, red, blue,yellow);  
}
```

#### (4.17.4) transformaciones

Hay una propiedad llamada **transform** que permite aplicar transformaciones a una capa. De modo que se aplica sobre ella un efecto gráfico avanzado. Como valor de esta propiedad podemos indicar **none** para eliminar cualquier transformación o utilizar alguna de estas funciones (la mayoría no funcionan en los navegadores):

- **scale(x,y)**. Realiza un escalado sobre el elemento. El primer valor indica el factor de escalado horizontal (1 es el 100% y .5 escala al 50%) y el segundo el escalado vertical.
- **scaleX(x)**. Escala el elemento en horizontal.
- **scaleY(y)**. Escala el elemento en vertical.
- **rotate(ángulo)**. Rota el elemento el ángulo indicado (por ejemplo **45deg**)
- **rotateX(ángulo)**. Escala en horizontal el elemento pero rotándole si superamos los noventa grados. Le hace un efecto de espejo en horizontal si usamos **rotateX(180deg)**.
- **rotateY(ángulo)**. El mismo efecto pero referido al eje vertical.
- **skew(ánguloX, ánguloY)**. Estira el elemento usando un ángulo en horizontal y otro en vertical.
- **skewX(angle)**. Estira el elemento pero sólo en el eje horizontal
- **skewY(angle)**. Estira el elemento pero sólo en el eje vertical

## (4.17.5) transiciones

Sin duda es el efecto más espectacular a aplicar sobre un elemento mediante CSS3. Se trata de varias propiedades que permiten realizar que al actuar sobre un elemento, este modifique sus propiedades (a través de efectos de transición) poco a poco estableciendo una duración y una velocidad configurables.

Este código HTML:

```
<!doctype html>
<html lang="es">
<head>
  <meta charset="UTF-8">
  <title>Document</title>
  <style>
    div{
      background-color:red;
      width:50px;
      height:50px;
    }
    div:hover{
      width:300px;
    }
  </style>
</head>
<body>
  <div>&nbsp;</div>
</html>
```

Muestra en pantalla un rectángulo de color rojo; al arrimar el ratón sobre él (comportamiento **:hover**) el rectángulo crece hasta los trescientos píxeles.

Mediante las transiciones podemos hacer que ese cambio se realice progresivamente (de otro modo, el cambio aparece de golpe).

La propiedad fundamental para crear transiciones es **transition** cuya sintaxis es:

**transition:** propiedad tiempo funciónDeTiempo retardo;

No hay por qué rellenar todos los valores. Por otro lado hay que crear versiones específicas de esta propiedad para los motores concretos de los navegadores, con lo que la sintaxis real es:

**-webkit-transition:** propiedad tiempo funciónDeTiempo retardo;;  
**-moz-transition:** propiedad tiempo funciónDeTiempo retardo;;  
**-o-transition:** propiedad tiempo funciónDeTiempo retardo;;  
**-ms-transition:** propiedad tiempo funciónDeTiempo retardo;;  
**transition:** propiedad tiempo funciónDeTiempo retardo;

En los ejemplos siguientes no usaremos la sintaxis completa sino la perteneciente al estándar.

Así por ejemplo cambiando el CSS anterior por:

```
div{
  background-color:red;
  width:50px;
  height:50px;
}

div:hover{
  width:300px;
  -webkit-transition: 2s;
  -moz-transition: 2s;
  -o-transition: 2s;
  -ms-transition: 2s;
  transition: 2s;
}
```

Ya tendríamos un efecto de transición que dura dos segundos para pasar del tamaño anterior de la capa al nuevo simplemente arrimando el ratón.

Las propiedades que se pueden indicar son **width**, **height**, **background-color**,.. Es decir, en principio cualquier propiedad CSS. Pero como es lógico los navegadores sólo pueden realizar la transición sobre algunas. Al indicar sobre qué propiedad realizamos la transición sólo podemos indicar una. Pero podemos indicar **all** (valor por defecto) para que haga la transición sobre todas las propiedades a la vez.

La duración se suele establecer en segundos, al igual que el retardo. El retardo es el tiempo que tarda el efecto de transición en empezar a realizarse. Si no se indica, el efecto comienza inmediatamente, si se indica por ejemplo **1s** el efecto tardará un segundo en empezar.

Finalmente la función de transición es una de las siguientes:

- **lineal**. Es la que se emplea por defecto, la transición tiene siempre la misma velocidad.
- **ease**. La transición frena la velocidad al final.
- **ease-out**. La velocidad de la transición va poco a poco disminuyendo.
- **ease-in**. La velocidad de la transición empieza lenta y luego va acelerando.
- **ease-in-out**. La velocidad empieza frenada, acelera y vuelve a frenar al final.

Ejemplo:

```
transition: width 3s ease-in 1s;
/* Transición para modificar la anchura, que dura tres segundos,
   acelera al final y tarda un segundo en empezar */
```



### (4.17.6) animaciones

Las animaciones son parecidas a las transiciones con la diferencia de que están pensadas para que se ejecuten directamente. Es decir no son cambios de estado porque el usuario/a haga algo, sino que convierten elementos de la página web en elementos animados que se mueven libremente.

Las animaciones requiere primero de un bloque **@keyframes** (para navegadores Webkit hay que usar además la variante **@-webkit-keyframes**) que es el encargado de establecer las propiedades de la animación y de dar nombre a la misma. Dentro de ese bloque se establecen dos elementos:

- **from**. Con las propiedades iniciales del elemento al que se aplicas la animación.
- **to**. Con las propiedades finales.

Ejemplo:

```
@keyframes ani1{  
  from{  
    left:20px;  
    background-color: red;  
  }  
  to{  
    left:220px;  
    background-color:blue;  
  }  
}
```

También podemos establecer indicar en lugar de origen y final, pasos en porcentaje, por ejemplo:

```
@keyframes ani1{  
  from{  
    left:20px;  
    background-color: red;  
  }  
  50%{  
    background-color: green;  
  }  
  to{  
    left:220px;  
    background-color:blue;  
  }  
}
```

Ahora el color de fondo se va cambiando a verde antes de convertirse en azul

Hemos creado una animación de nombre `ani1`, que modificarás la posición izquierda del elemento y su color (del rojo al azul).

Ahora hay que aplicarla al elemento deseado. Para eso se usa la propiedad **animation** (en Webkit es **-webkit-animation**), que tiene esta sintaxis:

**animation:** nombre duración funciónDeTiempo retardo contador dirección;

El **nombre**, es el nombre de la animación a aplicar (por ejemplo podemos aplicar `ani1`). El **tiempo**, la **funcionDeTiempo** y el **retardo** funcionan igual que con las transiciones, explicadas en el apartado anterior.

El **contador** es el número de veces que se repetirá la animación. Puede ser un número o la palabra **infinite** que indica que se repite la animación continuamente. Por defecto este valor vale uno.

La dirección puede tomar los valores:

- **normal** (valor por defecto) que indica que la animación se ejecuta en la dirección programada
- **alternate** que permite que la animación se realice en sentido contrario en cada turno. Es decir, si nuestra animación consiste en mover un elemento de izquierda a derecha, en los turnos pares lo hará al revés (de derecha a izquierda).

Ejemplo completo:

```
<!doctype html>
<html lang="es">
<head>
  <meta charset="UTF-8">
  <title>Cuadrado móvil</title>
  <style>
    @keyframes ani1{
      from{
        left:20px;
        background-color: red;
      }
      to{
        left:220px;
        background-color:blue;
      }
    }
    /* Versión para Chrome y Safari */
    @-webkit-keyframes ani1{
      from{
        left:20px;
        background-color: red;
      }
      to{
        left:220px;
        background-color:blue;
      }
    }
  </style>
</html>
```

```
div{
    position:fixed;
    background-color:red;
    left:20px;
    width:50px;
    height:50px;
    -webkit-animation:ani1 2s ease-in-out 1s infinite alternate;
    animation:ani1 2s ease-in-out 1s infinite alternate;
}

</style>
</head>
<body>
    <div>&nbsp;</div>
</html>
```

Este código muestra un cuadrado rojo que se desplaza continuamente hacia la derecha y luego vuelve a la izquierda. Además cambia el color de rojo a azul a la vez que se mueve.

## (4.18) aplicar CSS a documentos XML

Aunque CSS se creó para páginas web creadas en HTML, lo cierto es que se puede aplicar también a documentos XML.

Para ello necesitamos crear el código CSS separado del documento XML y después adjuntarle a XML mediante la siguiente etiqueta:

```
<?xml-stylesheet type="text/css" href="urlALaHojaDeestilos.css"?>
```

Quizá la propiedad CSS más importante para que el XML salga de forma más adecuada sea **display** (véase (4.14), propiedad display). Los elementos XML no tienen asignado ninguna forma de salida en pantalla, por lo que la forma en la que los navegadores muestran el contenido es **inline**, es decir el contenido se muestra seguido sin saltos de línea ni de párrafo.

Por ello debemos indicar **display:block** en aquellos elementos cuyos contenidos se deben mostrar en párrafos separados. Es posible mostrarlos en formas de tabla, listas,... gracias a la potencia de la propiedad **display**.

Todo lo aprendido con CSS es perfectamente válido con elementos XML, de hecho la potencia al usar selectores, tiene más sentido en XML.

Ejemplo completo:

Archivo XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/css" href="documentos.css"?>
<documentos>
  <documento id="1">
    <titulo>Informe anual de ventas</titulo>
    <autor dep="12">Ana Madrigal</autor>
    <autor dep="12">Luis Gonzala</autor>
  </documento>
  <documento id="2">
    <titulo>Resumen de nuevos mercados</titulo>
    <autor dep="9">Iovanna Areas</autor>
    <autor dep="12">Estrella Eamos</autor>
  </documento>
  <documento id="3">
    <titulo>Informe logístico</titulo>
    <autor dep="12">Ana Madrigal</autor>
    <autor dep="7">Luis Manglano</autor>
    <autor dep="6">Carlos Hernández</autor>
  </documento>
</documentos>
```

Archivo CSS (documentos.css):

```
documento{
  font-family:Arial, sans-serif;
  font-weight:bold;
  font-size:14pt;
}

titulo{
  background-color:gray;
  color:white;
  display:block;
  font-size:20pt;
  font-family: Georgia,"Times New Roman",Times, serif;
}

autor{
  margin-left:20px;
  display: block;
}

autor[dep="12"]:after{
  content:" Departamento de ventas ";
  color:gray;
  font-style:italic;
}
```

Resultado:



Ilustración 16, Contenido XML formateado gracias a CSS