Tema 3. Hojas de estilo en cascada (CSS)

Tema 3. Hojas de Estilo en Cascada (CSS)

1. Introducción.

CSS es un lenguaje de hojas de estilo para controlar la presentación de documentos HTML, XHTML y XML. Entendemos por presentación, el aspecto con el que aparecerán los contenidos, es decir, el color, las fuentes, los márgenes, las posiciones, etc.

Actualmente no se usa HTML para dar un formato al documento. Las razones principales son:

- Si el formato de un documento se separa del contenido, es fácil reutilizar dicho formato para diferentes páginas manteniendo una coherencia en todos los documentos del sitio web.
- Si usamos HTML sólo como lenguaje semántico no perdemos el significado del texto. Esto tiene gran utilidad por ejemplo para los buscadores.
- Al ser independientes el contenido y el formato, siempre se podrá mejorar ambos aspectos de una forma menos compleja con otros nuevos lenguajes.

La definición del lenguaje CSS se ha recogido por el W3C en las siguientes recomendaciones:

- La primera recomendación se conoce como CSS nivel 1 y apareció en el año 1996.
- En el año 1998, se publicó la recomendación CSS nivel 2. Actualmente, la versión que utilizan todos los navegadores es la CSS 2.1 (http://www.w3.org/TR/CSS2/).
- La siguiente recomendación, conocida como CSS nivel 3, continúa en desarrollo desde 1998 y, por el momento, sólo se han publicado borradores, aunque la mayoría de los exploradores actuales van soportando las especificaciones que continuamente se van desarrollando.

1.1. Normas básicas de composición.

CSS es un lenguaje de texto que se compone de diferentes **reglas**. Cada regla se forma mediante un **selector** seguido entre llaves de una **declaración**.

El selector indica el elemento o elementos del documento al que se aplica la regla CSS y la declaración especifica los estilos que se aplican a los elementos indicados en el selector.

Cada declaración está compuesta por una o más **propiedades** con sus **valores** correspondientes.

Una propiedad permite modificar el aspecto de una característica del elemento y un valor indica el nuevo valor de la característica modificada en el elemento.

El siguiente es un ejemplo de regla CSS:

```
h1 {color:red;}
```

h1 es el selector y *{color:red;}* es la declaración formada por una propiedad, *color*, y un valor, *red*.

Una de las características más importantes de CSS es la herencia, de modo que, en general, las propiedades de los elementos se heredan de los elementos que los contienen, salvo si se les especifica una propiedad particular. De todas formas, la interpretación de la herencia puede variar según el navegador.

1.2. Formas de incluir CSS en un documento XHTML.

Existen tres formas de incluir hojas de estilo en un documento XHTML:

a) **CSS interno**. Utilizando la etiqueta <style> dentro de la cabecera <head>. Por ejemplo:

```
<head>
    ...
    <style type="text/css">
        p {color: red; font-family: sans-serif;}
    </style>
    ...
</head>
```

b) **CSS en línea**. En las propias etiquetas HTML, utilizando el atributo style="definición de estilo". Por ejemplo:

```
<body>
    …
    <p style = "color: red; font-family: sans-serif;">Las Alpujarras granadinas se encuentran en las laderas altas de Sierra Nevada, mirando al Mediterráneo, a espaldas de la estación de esquí.</p>
    …
</body>
```

c) **CSS externo**. Este caso es el más habitual. Se trata de crear un archivo de texto con extensión *.css* con las reglas CSS. La ventaja es que un único archivo sirve para aplicar sus estilos a diferentes documentos, y si cambiamos de archivo CSS, todos los documentos que usen ese nuevo documento CSS cambiarán al nuevo estilo. Para ello disponemos de dos posibilidades:

- Utilizando un archivo externo y enlazándolo mediante la etiqueta <link>:

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

- Invocando al archivo externo desde la etiqueta <style> poniendo, como primera línea una declaración de tipo @import:

```
<style type="text/css">
    @import ("url_archivo_css");
    … resto de definición de estilo …
</style>
```

1.3. Orden en la aplicación de estilos.

Es posible que algunos documentos HTML utilicen varios estilos que hagan referencia a un mismo elemento. Así por ejemplo:

```
<!doctype html>
<html lang="en">
    <head>
        <meta charset="UTF-8">
        <title>estilos 1</title>
        <link rel="stylesheet" type = "text/css" href = "parrafo_verde.css" />
        <style type="text/css">
            p {color:blue; }
        </style>
    </head>
    <body>
        <p style="color:red;">¿Con qué color apareceré?</p>
    </body>
</html>
```

Nos podemos preguntar con qué color se presentará el contenido del párrafo. Pues rojo, ya que en este caso la regla que tiene mayor prioridad es la que parece en la etiqueta <p> dentro del cuerpo de la página. El orden de preferencia para las reglas es:

- Primero se aplican los estilos propios predefinidos del navegador.
- A continuación se aplican los estilos externos que indique la etiqueta <link>.
- Después los que procedan de la etiqueta <style>.
- Por último los que se definan en el propio elemento mediante el atributo style.

Si se da el caso de tener varios estilos definidos en el mismo ámbito a un mismo elemento, tiene más peso o preferencia el último que se presente en el código.

1.4. Medios CSS.

CSS permite indicar diferentes reglas para diferentes medios y/o dispositivos, por ejemplo, paginación y salto de página para medios impresos; volumen y tipo de voz para medios de audio, etc. Los medios que utiliza CSS son:

Medio	Descripción
all	Todos los medios definidos (valor por defecto).
braille	Dispositivos táctiles que emplean el sistema braille.
embosed	Impresoras braille.
handheld	Dispositivos de mano: móviles, PDA, etc.
print	Impresoras y navegadores en el modo "Vista previa para imprimir".
projection	Proyectores y dispositivos para presentaciones.
screen	Pantallas de ordenador.
speech	Sintetizadores para navegadores de voz utilizados por personas discapacitadas.
tty	Dispositivos textuales limitados, como teletipos y terminales de texto.
tv	Televisores o dispositivos de baja resolución.

Existen cuatro formas diferentes de indicar el medio en el que se deben aplicar los estilos CSS.

1.4.1. Medios definidos con las reglas de tipo @media.

Estas reglas permiten indicar de forma directa el medio o los medios en los que se aplicarán los estilos. Tan sólo hay que escribir la expresión *@media* seguida del medio o los medios implicados. Si los estilos se aplican a varios medios, se incluyen los nombres de todos ellos separados por comas.

```
<style type="text/css">
    @media screen {body {line-height: 0.8}}
    @media print {body {line-height: 1.2}}
</style>
```

1.4.2. Medios definidos con las reglas de tipo @import.

Cuando se usan estas reglas para enlazar archivos CSS externos, se puede especificar el medio en el que se aplican los estilos indicando el nombre del medio después de la URL del archivo CSS.

```
<style type="text/css">
    @import url ("estilos_basicos.css") screen;
<style>
```

Si los estilos del archivo CSS externo deben aplicarse en varios medios, se indican los nombres de todos ellos separados por comas. En el caso de que no se indique ningún medio, el navegador interpreta que éste es *all*, es decir, que los estilos se aplican en todos los medios.

1.4.3. Medios definidos con la etiqueta <link>.

Si se utiliza la etiqueta <link> para enlazar archivos CSS externos, se puede añadir el atributo *media* para indicar el medio o los medios en los que se aplican los estilos de cada archivo. Por ejemplo:

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

Si no se indica el medio CSS, se sobreentiende que los estilos se aplicarán a todos los medios.

1.4.4. Medios definidos mezclando varios métodos.

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

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

```
<style>
    @import url("estilos_seccion.css") screen;
    @media print {
        /* Estilos específicos para la impresora */
    }
</style>
```

1.5. Comentarios.

Entre las reglas y estilos de un archivo CSS se pueden incluir comentarios que mejoran la comprensión para el diseñador pero que los navegadores ignoran por completo.

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

```
/* Esto es un comentario */
```

Los comentarios pueden ocupar tantas líneas como sea necesario pero no se pueden anidar, es decir, no se puede escribir un comentario dentro de otro.

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

2. Unidades de medida y colores.

2.1. Unidades de medida.

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

Las unidades de medida CSS pueden ser de dos tipos:

- **Relativas**: definen su valor en relación con otra medida por lo que, para obtener su valor, se debe realizar alguna operación con el valor indicado.
- **Absolutas**: establecen de forma completa el valor de una medida, por lo que su valor real es exactamente el especificado.

2.1.1. Unidades relativas.

Estas unidades son más flexibles que las unidades absolutas porque se adaptan más fácilmente a los diferentes medios. Son las siguientes:

- *em*, relativa al tamaño de letra empleado. Aunque no es una definición exacta, el valor `1em` se puede aproximar por la altura de la letra "m" del tipo de letra que se esté empleando. Por ejemplo: `h1 { font-size: 2em; }` indica que el tamaño de letra para el texto referenciado por `<h1>` será el doble del tamaño que se esté usando.
- *ex*, también es relativa al tamaño de la letra empleada pero, esta vez, respecto a la altura de la letra "x". Su uso es poco frecuente.
- *px*, píxel. Dependiendo de la pantalla los pixeles tendrán un tamaño u otro.
- *%*, porcentaje, hace referencia a una proporción de otra medida.

Estas unidades se pueden utilizar en diferentes elementos de una misma página, como se puede ver en el siguiente ejemplo:

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

En la primera línea se establece un tamaño de letra base de 10 pixeles para toda la página. En la segunda, se asigna el tamaño `2.5em` al elemento `<h1>`, por lo que su tamaño de letra expresado en pixeles será de 2.5 x 10px = 25px.

2.1.2. Unidades absolutas.

CSS permite utilizar las siguientes unidades absolutas:

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

He aquí algunos ejemplos de utilización de estas unidades:

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

2.1.3. Recomendaciones.

Se recomienda el uso de unidades relativas ya que mejora la accesibilidad de la página y permite que los documentos se adapten fácilmente a cualquier medio y dispositivo.

Normalmente, se utilizan *píxel* y porcentajes para definir la anchura de las columnas y de los elementos de las páginas; y *em* y porcentajes para el tamaño de la letra de los textos.

El concepto del llamado *Responsive Web Design* se basa entre otras cosas en usar propiedades según el tamaño de la pantalla (*CSS Media Queries*), usar medidas porcentuales y establecer tamaños máximos para los elementos multimedia en vez de tamaños fijos.

2.2. Colores.

Los colores en CSS se pueden indicar de modos diferentes: palabras clave, colores del sistema, RGB/RGBA y HSL/HSLA De ellos, el más habitual es el RGB/RGBA en formato hexadecimal.

2.2.1. Palabras clave.

CSS 2.1 define 17 palabras clave para referirse a los colores básicos, que corresponden al nombre en inglés de cada color*: aqua, black, blue, fucsia, gray, green, lime, maroon, navy, olive, orange, purple, red, silver, teal, white, yellow*.

La especificación CSS3 referente a los colores reconoce los llamados colores **X11**. Es otra lista de colores que se creó para el sistema X Windows de Unix y que ha ido adoptándose por los navegadores y por diversos lenguajes gráficos, como SVG. Actualmente, la mayoría de los navegadores reconocen estos colores. Una lista con los nombres de estos colores se puede obtener en http://en.wikipedia.org/wiki/X11_color_names.

Aunque el uso de palabras clave es una forma muy sencilla de referirse a los colores por su nombre, este método apenas se utiliza porque permite representar una gama de colores muy limitada.

2.2.2. Colores del sistema.

Este método es similar al de las palabras clave, sin embargo, los valores usados en él son palabras del sistema operativo que designan elementos visuales del mismo, como pueden ser el borde de la ventana activa (*ActiveBorder*), el fondo de escritorio (*Background*), el texto de una ventana (*WindowText*), etc. La lista completa se puede consultar en http://www.w3.org/TR/CSS21/ui.html#system-colors.

De esta forma, podemos indicar que un determinado elemento de nuestra página se muestre en el mismo color en que se ve otro del sistema operativo. El siguiente ejemplo muestra el texto de un párrafo en el mismo color que el de los botones de los cuadros de diálogo:

```
p {color: ButtonFace; }
```

En http://en.wikipedia.org/wiki/Websafe#Web-safe_colors podemos encontrar información de todos los sistemas de colores e incluso como convertir un color de un sistema a otro.

2.2.3. RGB decimal.

El modelo RGB permite definir un color indicando las cantidades de las componentes rojo (*Red*), verde (*Green*) y azul (*Blue*) que se deben mezclar para obtenerlo. Estas cantidades pueden ir desde cero a un valor máximo. En concreto, en CSS, las componentes de los colores definidos mediante RGB decimal pueden tomar valores entre 0 y 255.

Si el valor de cada componente es cero, el color creado será el negro y, si es el máximo, se obtendrá el blanco. Por otro lado, para crear el color puro se utilizará el máximo valor para la cantidad de dicho color y valor cero para las cantidad de los otros componentes.

La sintaxis para indicar un color consiste en escribir *rgb()* y, dentro del paréntesis, los valores de las tres componentes separados por comas. En el siguiente ejemplo se establece el color del texto de un párrafo:

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

2.2.4. RGB porcentual.

El funcionamiento y la sintaxis de este método es el mismo que el del RGB decimal, excepto porque el valor de las componentes está comprendido entre 0% y 100%. El ejemplo anterior también se podría escribir así:

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

2.2.5. RGB hexadecimal.

En este caso, el valor de cada componente se expresa mediante un número hexadecimal comprendido entre 0 y FF. Sin embargo, la sintaxis para indicar un color concreto varía ligeramente puesto que simplemente se escribirá el símbolo # seguido de dos dígitos hexadecimales para cada una de las componentes.

El ejemplo de las secciones anteriores se escribe así:

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

Una ventaja de este formato es que permite comprimir sus valores cuando cada componente contiene un mismo dígito repetido. Por ejemplo: El color #AA0066 puede expresarse como #A06.

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

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

2.2.6. RGBA.

CSS3 define el color RGBA como una extensión al sistema de color RGB al que se le añade un cuarto componente llamado *canal alfa* que especifica la opacidad o transparencia del objeto al que se aplica el color. Este parámetro varía entre 0.0 (transparencia total) y 1.0 (totalmente opaco). Por ejemplo:

```
p {color: rgb(27%, 38%, 69%, 0.5); }
```

2.2.7. HSL y HSLA.

Los colores HSL se obtienen mediante los valores del tono (*hue*, valores de 0 a 360) saturación (valores porcentuales de 0 a 100) y luminosidad (valores porcentuales de 0 a 100). El color HSLA es una extensión al color HSL al que se le ha añadido un cuarto parámetro de transparencia. Ejemplo:

```
p { color: hsla(120, 100%, 50%, 0.1) }        /* Verde muy transparente */
```

Esta forma de expresar el color es propia de CSS3 y representa un modelo de percepción del color más parecido al de los humanos.

3. Selectores.

Como vimos anteriormente, una regla CSS está formada por una parte llamada "selector" y otra parte llamada "declaración". La declaración indica "*qué hay que hacer*" y el selector indica "*qué elementos van a ser afectados*".

A un mismo elemento HTML se le pueden asignar varias reglas CSS y cada regla CSS puede aplicarse a varios elementos. Es decir, una misma regla se puede aplicar a diferentes selectores y un mismo selector se puede utilizar en varias reglas.

El estándar CSS 2.1 incluye doce tipos diferentes de selectores que permiten seleccionar de forma muy precisa elementos individuales o conjuntos de elementos dentro de una página web. Sin embargo, la mayoría de los sitios web se pueden diseñar utilizando solamente los cinco selectores básicos.

3.1. Selectores básicos.

3.1.1. Selector universal.

Se utiliza para seleccionar todos los elementos de la página. Se indica mediante un asterisco y, a pesar de su sencillez, no se utiliza habitualmente ya que es raro que un mismo estilo se pueda aplicar a todos los elementos de una página.

El siguiente ejemplo elimina el margen y el relleno de todos los elementos HTML:

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

3.1.2. Selector de tipo o etiqueta.

Selecciona todos los elementos de la página cuya etiqueta HTML coincide con el valor del selector. Para utilizar este selector, solamente es necesario indicar el nombre de una etiqueta HTML correspondiente a los elementos que se quieren seleccionar.

El siguiente ejemplo selecciona todos los párrafos de la página:

```
p {  … declaraciones … }
```

Si se quiere aplicar los mismos estilos a dos etiquetas diferentes, se puede aplicar un selector múltiple a una sola regla. Para esto, se incluyen todos los selectores, separados por una coma. Por ejemplo:

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

En las hojas de estilos complejas es habitual agrupar las propiedades comunes de varios elementos en una única regla CSS y, posteriormente, definir las propiedades específicas de esos mismos elementos. Por ejemplo, tras el código anterior, podemos escribir:

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

3.1.3. Selector descendente.

Recordemos que un elemento es descendiente de otro cuando se encuentra entre las etiquetas de apertura y cierre de otro elemento. Pues bien, un selector descendiente es aquél que permite seleccionar elementos que se encuentran dentro de otros elementos. Se construye con dos o más selectores simples separados por un espacio en blanco.

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

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

Si éste se aplica al siguiente código HTML de una página:

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

El selector *p span* seleccionará tanto *texto1* como *texto2*. El motivo es que en el selector descendiente, un elemento no tiene que ser "*hijo directo*" de otro, sino que basta con que se encuentre dentro de él.

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

La sintaxis formal del selector descendente se muestra a continuación:

```
elemento1 elemento2 elemento3 … elementoN { regla; }
```

donde *elementoN* indica el elemento sobre el que se aplican los estilos y el resto indica el lugar en que se tiene que encontrar *elementoN* para que los estilos se apliquen realmente.

En el siguiente ejemplo, el selector descendente contiene cuatro elementos:

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

El estilo subrayado de esta regla se aplicará a los elementos de tipo que se encuentren dentro de elementos de tipo que, a su vez, estén contenidos en elementos de tipo <a> que estén dentro de elementos de tipo <p>.

3.1.4. Selector de clase.

Este selector permite aplicar estilo a uno o varios elementos de una página. En concreto, a todos aquellos que pertenezcan a la misma clase. Por ejemplo, si en el siguiente código HTML queremos que los textos del primer y tercer párrafo se muestren en rojo:

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

Ninguno de los selectores vistos hasta el momento nos lo permiten. Sin embargo, existe una solución que consiste en:

1. Utilizar el atributo *class* de HTML sobre el elemento para indicar directamente la regla CSS que se le debe aplicar:

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

2. Crear, en el archivo CSS, una nueva regla cuyo selector es el valor del atributo *class*, precedido por el carácter punto y con todos los estilos que se quieren asignar. En nuestro ejemplo:

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

El selector .destacado se interpreta como "cualquier elemento de la página cuyo atributo *class* sea igual a destacado".

La principal característica de este selector es que, en una misma página HTML, varios elementos diferentes pueden utilizar el mismo valor en el atributo *class*. En el siguiente ejemplo, se aplica el estilo destacado a varios elementos:

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

Por el contrario, a veces, es necesario restringir el alcance del selector de clase. Para ello, escribiremos el selector de tipo o etiqueta y el de clase. Supongamos que en el ejemplo anterior, sólo queremos aplicar el estilo a los párrafos que lleven el atributo *class* con valor destacado. Para ello, utilizaremos el selector:

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

Hay que distinguir, por tanto, entre estas expresiones:

p.destacado {...}	Todos los elementos de tipo "p" con atributo class="destacado".
p .destacado {...}	Todos los elementos con atributo class="destacado" que estén dentro de cualquier elemento de tipo "p".
p, .destacado{...}	Todos los elementos "p" de la página y todos los elementos con atributo class="destacado".

3.1.5. Selector por identificador.

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

El selector por identificador selecciona un único elemento a través del valor de su atributo *id*, cuyo valor nunca se puede repetir en dos elementos diferentes de una misma página.

La sintaxis de los selectores por identificador es análoga a la de los selectores de clase, salvo porque se utiliza el símbolo almohadilla (#), en vez del punto (.), como prefijo del nombre de la regla CSS.

En el siguiente ejemplo, el selector #destacado solamente selecciona el segundo párrafo:

```
#destacado {color: red;}

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

En general, se recomienda usar el selector por identificador cuando se quiere aplicar un estilo a un solo elemento específico de una página y el selector de clase cuando se quiere aplicar un estilo a varios elementos diferentes de la misma.

Cuando se trabaja sobre varias páginas suele resultar útil restringir el alcance de un selector por identificador. En el siguiente ejemplo se aplica la regla CSS sólo al elemento de tipo <p> que tenga un atributo *id* igual al indicado:

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

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

No debe confundirse el selector por identificador con los selectores anteriores:

p#destacado {...}	Todos los elementos de tipo "p" con atributo id="destacado"
p #destacado {...}	Todos los elementos con atributo id="destacado" que estén dentro de cualquier elemento de tipo "p".
p, #destacado{...}	Todos los elementos "p" de la página y todos los elementos con atributo id="destacado".

3.2. Selectores avanzados.

Aunque con los selectores básicos es posible diseñar prácticamente cualquier página web, tenemos la opción de utilizar los selectores avanzados para simplificar las hojas de estilos.

El navegador Internet Explorer 6 y sus versiones anteriores no soportan este tipo de selectores. Se puede consultar la lista de los selectores que soporta cada versión de cada navegador en:

http://dev.l-c-n.com/CSS3-selectors/browser-support.php

3.2.1. Selector de hijos.

Se utiliza para seleccionar un elemento que es hijo directo de otro elemento y se indica mediante el signo mayor que (>). Veamos un ejemplo:

```
p > span {color: blue; }
. . .
<p><span>texto1</span></p>
<p><a href="#"><span>Texto2</span></a></p>
```

El selector p > span se interpreta como *"cualquier elemento que sea hijo directo de un elemento <p>"*. Por ello, el primer elemento cumple la condición del selector, pero el segundo no la cumple porque aunque es descendiente, no es hijo directo de <p>.

3.2.2. Selector adyacente.

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

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

Con él se seleccionan todos los elementos de tipo *elemento2* que cumplan las dos siguientes condiciones:

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

En el siguiente ejemplo:

```
h1 + h2  {color: red;}
. . .
<body>
    <h1>Título1</h1>
    <h2>Subtítulo</h2>
    ...
    <h2>Otro subtítulo</h2>
</body>
```

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

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

3.2.3. Selector de atributos.

Permiten seleccionar elementos HTML en función de sus atributos y/o valores de esos atributos.

Existen varios tipos de selectores de atributos:

- *elemento[nombre_atributo]*, selecciona los elementos, que tienen establecido el atributo llamado *nombre_atributo*, independientemente de su valor. Por ejemplo: p[class] {color: blue;} presentaría en azul todos los párrafos que tuvieran declarado el atributo *class*.
- *elemento[nombre_atributo="valor"]*, selecciona los elementos que tienen establecido un atributo llamado *nombre_atributo* con valor igual a *valor*. Por ejemplo: p[class="externo"] {color: blue;} presentaría en azul los párrafos que tuvieran declarado el atributo *class* con valor *externo*.

Con CSS3 se dispone también de:

- *elemento[nombre_atributo~="valor"]*, selecciona los elementos que tienen establecido un atributo llamado *nombre_atributo* y cuyo valor es una lista de palabras separadas por espacios en blanco en la que al menos una de ellas es exactamente igual a *valor*. Por ejemplo: img[title~="foto"] {...}

- *elemento[nombre_atributo|="valor"]*, selecciona los elementos que tienen establecido un atributo llamado *nombre_atributo* y cuyo valor es exactamente igual a *valor* o comienza con *valor* seguido de un guión. Ejemplo: *[lang|="es"] {color: blue;}
- *elemento[nombre_atributo$="valor"]*, selecciona los elementos que tienen establecido un atributo llamado *nombre_atributo* y cuyo contenido finalice con *valor*.
- *elemento[nombre_atributo^="valor"]*, selecciona los elementos que tienen establecido un atributo llamado *nombre_atributo* y cuyo contenido comience con *valor*.
- *elemento[nombre_atributo*="valor"]*, selecciona los elementos que tienen establecido un atributo llamado *nombre_atributo* y cuyo contenido contenga el texto *valor* en cualquier parte.

3.2.4. Pseudos-clases.

Permiten seleccionar elementos en función del comportamiento posterior del usuario. Por ejemplo según los enlaces que visita o el movimiento del ratón. Algunas son:

- **:link**, aplica estilos a los enlaces que apuntan a páginas o recursos que aún no han sido visitados por el usuario.
- **:visited**, aplica estilos a los enlaces que apuntan a recursos que han sido visitados anteriormente por el usuario.
- **:hover**, aplica estilos a un elemento seleccionado por el usuario pero sin activarlo. Como por ejemplo cuando situamos el ratón encima de un objeto pero sin hacer clic.
- **:active**, aplica estilos al elemento que está siendo activado, por ejemplo cuando el usuario pulsa el botón izquierdo del ratón.
- **:focus**, aplica estilos al elemento que tiene el foco, por ejemplo un cuadro de texto en un formulario.
- **:checked.** Se suele usar para dar estilos a controles de formulario de tipo *radio* o *checkbox* cuando están seleccionados.
- **:enabled.** Para dar estilos a controles de formulario cuando están habilitados.
- **:disabled.** Para dar estilos a controles de formulario cuando están deshabilitados.

Ejemplos:

```
a:link {color:black;}
a:visited {color:blue;}
a:hover {color:green;}
a:active {color:red;}
input:focus {background-color:yellow;}
input:checked {background-color:#F00;}
input[type="text"]:disabled {background-color:#ddd;}
```

3.2.5. Pseudos-elementos.

Se usan para añadir efectos especiales a algunos selectores. Su sintaxis es:

```
selector:pseudo-elemento {propiedad:valor;}
```

Algunos de ellos son:

- **:first-line**. Se aplica a la primera línea de un texto.
- **:first-letter**. Se aplica a la primera letra de un texto.
- **:before**. Sirve para generar texto antes del contenido del elemento. Se suele usar con la propiedad *content*. (Esta propiedad permite generar nuevo contenido e insertarlo en la página HTML)
- **:after**. Sirve para generar texto después del contenido del elemento. Se suele usar con la propiedad *content*.

Ejemplos:

```
h1:before {content: "--- Saludos ---";}
h1:after {content: "desde Granada";}
p:first-line {font-size: 18px;}
p:first-letter {font-size: 2em;}
```

4. Modelo de cajas.

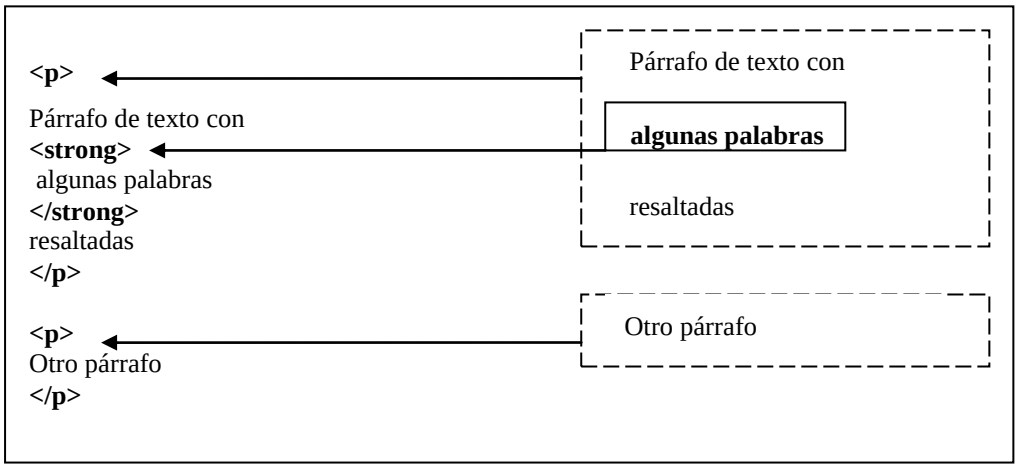
4.1. Descripción.

Este modelo es seguramente la característica más importante del lenguaje CSS ya que condiciona el diseño de todas las páginas web. En él, todos los elementos incluidos en una página HTML se representan mediante cajas rectangulares cuyo aspecto es controlado mediante CSS.

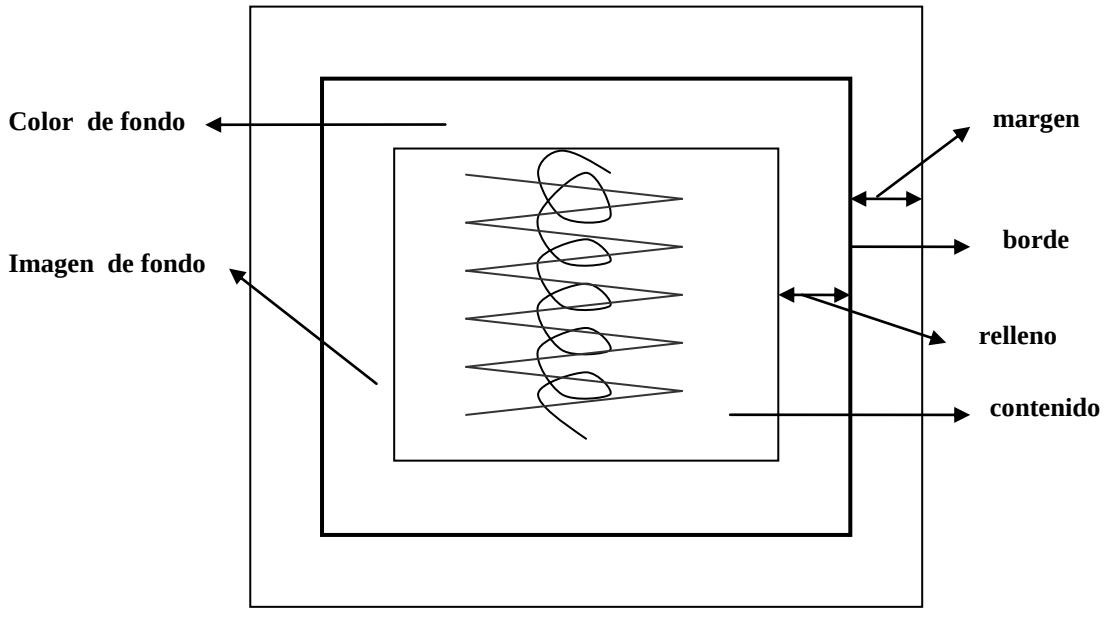
Las cajas se crean automáticamente. Cada vez que se inserta una etiqueta o elemento en la página, se crea una nueva caja rectangular que encierra los contenidos del elemento.

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

A continuación se muestra la creación automática de cajas para cada elemento definido en el código HTML de una página:



Una caja tiene los siguientes elementos:



4.2. Propiedades.

4.2.1. Anchura y altura.

La propiedad CSS que controla la anchura del contenido de una caja se denomina *width*.

width	**Anchura**
Valores	<medida> | <porcentaje> | auto | inherit
Se aplica a	Todos los elementos, salvo los elementos en línea que no sean imágenes, las filas de las tablas y los grupos de filas de tablas
Valor inicial	auto
Descripción	Establece la anchura de una caja

El valor *inherit* indica que la anchura del elemento se hereda de su elemento padre. El valor *auto* indica que el navegador debe calcular automáticamente la anchura del elemento, teniendo en cuenta sus contenidos y el sitio disponible en la página.

El siguiente ejemplo establece el valor de la anchura del elemento lateral:

```
#lateral {width: 200px; }
. . .
<div id="lateral">
    ...
</div>
```

La propiedad CSS que controla la altura del contenido de una caja se denomina *height*.

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

El siguiente ejemplo establece el valor de la altura del elemento de cabecera:

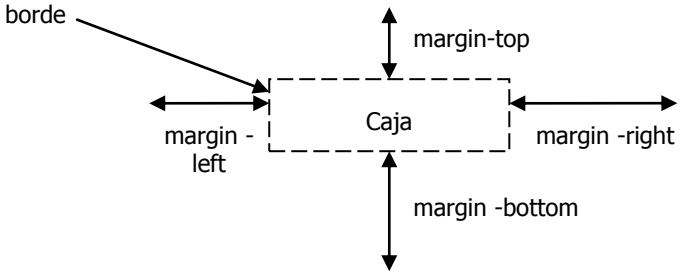
```
#cabecera {height: 60px; }
. . .
<div id="cabecera">
    ...
</div>
```

4.2.2. Margen y relleno.

El margen define la separación opcional existente entre la caja y el resto de cajas adyacentes. CSS define cuatro propiedades para controlar cada uno de los márgenes horizontales y verticales de un elemento.

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

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



El siguiente ejemplo añade un margen izquierdo a un párrafo:

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

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

La propiedad que permite definir de forma simultánea los cuatro márgenes se denomina *margin*:

margin	Margen
Valores	(<medida> | <porcentaje> | auto) {1, 4} | inherit
Se aplica a	Todos los elementos. Las propiedades margin-top y margin-button sólo se aplican a los elementos de bloque y las imágenes
Valor inicial	-
Descripción	Establece de forma directa todos los márgenes de un elemento

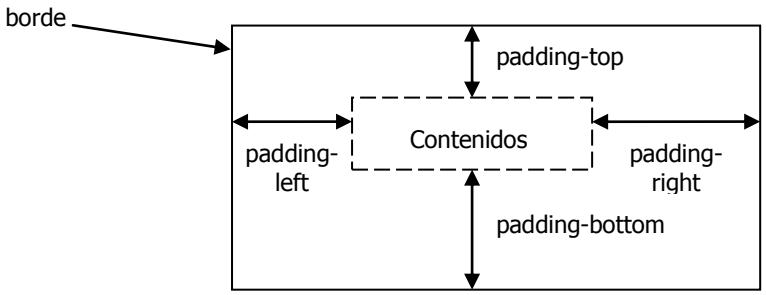
Ejemplos:

```
margin: 0 auto; /* centrado horizontal (margenes verticales a 0 y márgenes laterales automáticos.) */
margin: 0.5em 1em;  /* Establece los márgenes verticales a 0.5em y los laterales a 1em. */
margin: 0.5em 1em 1.5em 2em;  /* Establece los márgenes superior, derecho, inferior e izquierdo */
margin: 0.5em 1em 1.5em; /* establece el superior, los laterales y el inferior. */
```

El relleno define el espacio libre opcional existente entre el contenido y el borde de la caja. CSS define cuatro propiedades para controlar cada uno de los espacios de rellenos horizontales y verticales de un elemento.

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

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



La propiedad que permite definir de forma simultánea los cuatro tipos de relleno se denomina *padding*:

padding	Relleno
Valores	(<medida> | <porcentaje>) {1, 4} | inherit
Se aplica a	Todos los elementos, salvo algunos elementos de tablas como grupos de cabeceras y grupos de pies de tabla
Valor inicial	0
Descripción	Establece de forma directa todos los rellenos de los elementos

Ejemplos:

```
padding: 0.5em 1em;  /* Establece los rellenos verticales a 0.5em y los laterales a 1em. */
padding: 0.5em 1em 1.5em 2em;  /* Establece los rellenos superior, derecho, inferior e izquierdo */
padding: 1% 2% 3%; /* establece el superior, los laterales y el inferior. */
padding: 2px; /* Establece los cuatro rellenos a 2px; */
```

4.2.3. Bordes.

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

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

La medida más habitual para indicar la anchura de los bordes es el píxel, ya que permite un control preciso del grosor. Las palabras clave apenas se usan, ya que impiden mostrar bordes iguales en diferentes navegadores.

El color de los bordes se controla con estas cuatro propiedades:

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

CSS permite establecer el estilo de los bordes mediante estas cuatro propiedades:

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

CSS permite establecer todos los atributos de cada borde mediante una propiedad "*shorthand*" o "atajo":

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

Teniendo en cuenta que:

Expresión	Valores
<media_borde>	<medida> | thin | medium | thick
<color_borde>	<color> | transparent
<estilo_borde>	none | hidden | dotted | dashed | solid | double | groove | ridge | inset | outset

El siguiente ejemplo establece el color y el tipo del borde inferior, pero no su anchura:

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

CSS define la propiedad de tipo "*shorthand*" global para establecer el valor de todos los atributos de los 4 bordes de forma directa:

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

En CSS3 se pueden crear bordes redondeados indicando para las esquinas el valor del radio del círculo que forma el borde. El radio de cada borde puede tomar diferentes valores.

border-top-left-radius **border-bottom-right-radius** **border-bottom-left-radius** **border-top-right-radius**	Radio del borde superior izquierdo Radio del borde inferior derecho Radio del borde inferior izquierdo Radio del borde superior derecho
Valores	medida | %
Se aplica a	Todos los elementos
Valor inicial	-
Descripción	Establece el radio de cada uno de los cuatro bordes de los elementos

La propiedad que permite definir de forma simultánea el radio de los cuatro bordes se denomina *border-radius*:

border-radius	Radio de los bordes
Valores	<medida> | <porcentaje> {1, 4}
Se aplica a	Todos los elementos.
Valor inicial	-
Descripción	Establece de forma directa todos los radios de los bordes de un elemento

Ejemplo:

```
div
{
	border: 2px solid #a1a1a1;
	padding: 10px 40px;
	background:#ddd;
	width:300px;
	border-radius:25px;
}
```

También en CSS3 existe la posibilidad de usar una imagen para que cubra el borde, pero esta capacidad no la contemplaremos en este curso.

4.2.4. Fondos.

El fondo de la caja de un elemento puede ser un color simple o una imagen. Se muestra por detrás del contenido y del espacio de relleno. Si una caja define tanto un color como una imagen de fondo, la imagen tiene más prioridad y es la que se visualiza. No obstante, si la imagen de fondo no cubre totalmente la caja o si la imagen tiene zonas transparentes, también se visualiza el color de fondo.

Las propiedades que CSS define para establecer el fondo de cada elemento son *background-color*, *background-image*, *background-repeat*, *background-attachment*, *background-position* y *background* (shorthand).

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

background-image	Imagen de fondo
Valores	<url> | none | inherit
Se aplica a	Todos los elementos
Valor inicial	none
Descripción	Establece una imagen de fondo para los elementos

background-repeat	Repetición de la imagen de fondo
Valores	repeat | repeat-x | repeat-y | no repeat | inherit
Se aplica a	Todos los elementos
Valor inicial	repeat
Descripción	Establece la forma en la que se repiten las imágenes de fondo

El valor *repeat* indica que la imagen se debe repetir en todas las direcciones, siendo éste el comportamiento por defecto.

background-position	Posición de la imagen de fondo
Valores	(left top | left center |left bottom | right top | right center | right bottom | center top | center center |center bottom) || (x% y%) || (xpos ypos)
Se aplica a	Todos los elementos
Valor inicial	0% 0%
Descripción	Controla la posición en la que se muestra la imagen en el fondo del elemento

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

background-attachment	Comportamiento de la imagen de fondo
Valores	scroll | fixed | inherit
Se aplica a	Todos los elementos
Valor inicial	scroll
Descripción	Controla la forma en la que se visualiza la imagen de fondo: permanece fija cuando se hace scroll en la ventana del navegador o se desplaza junto con la ventana

Para hacer que una imagen de fondo se muestre fija al desplazar la ventana del navegador, se debe añadir la propiedad *fixed*.

La propiedad *background* permite establecer todas las propiedades en una sola declaración.

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

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

```
body { background: #00FF00 url('smiley.gif') no-repeat fixed center; }
```

5. Modelo de formato visual.

5.1. Elementos.

El estándar HTML clasifica a todos los elementos en dos grandes grupos:

- Elementos de bloque (*block elements*), siempre empiezan en una nueva línea y ocupan todo el espacio disponible hasta el final de la misma.
- Elementos en línea (*inline elements*), no empiezan necesariamente en una nueva línea y sólo ocupan el espacio necesario para mostrar sus contenidos.

Debido a este comportamiento, el tipo de un elemento influye de forma decisiva en la caja que el navegador crea para mostrarlo.

5.2. Posicionamiento.

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

Normalmente, cuando se posiciona una caja también es necesario desplazarla respecto de su posición original o respecto de otro origen de coordenadas. CSS define cuatro propiedades para controlar este desplazamiento:

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

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

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

Los tipos de posicionamiento son:

- Posicionamiento normal o estático (*static*): utilizado por defecto por los navegadores y en el que se ignoran los valores de las propiedades *top*, *right*, *bottom* y *left*.
- Posicionamiento relativo (*relative*): consiste en aplicar a una caja su posicionamiento normal y, después, desplazarla respecto a su posición original mediante las propiedades *top*, *right*, *bottom* y *left*.
- Posicionamiento absoluto (*absolute*): establece la posición de una caja de forma absoluta respecto a su elemento contenedor y el resto de los elementos de la página ignoran dicha posición.
- Posicionamiento fijo (*fixed*): convierte una caja en un elemento inmovible, de forma que su posición en la pantalla siempre es la misma, independientemente del resto de los elementos y de si el usuario sube o baja la página en la ventana del navegador.

- Posicionamiento flotante: desplaza las cajas todo lo posible hacia la izquierda o hacia la derecha de la línea en que se encuentran, para lo cual se utiliza la propiedad *float*.

float	Posicionamiento float
Valores	left | right | none | inherit
Se aplica a	Todos los elementos
Valor inicial	none
Descripción	Establece el tipo de posicionamiento flotante para el elemento

Para modificar el comportamiento por defecto del posicionamiento flotante y forzar a un elemento a mostrarse debajo de cualquier caja flotante se usa la propiedad *clear*. En ella se establece el lado que no debe ser adyacente a ninguna caja flotante.

clear	Despejar los elementos flotantes adyacentes
Valores	none | left | right | both | inherit
Se aplica a	Todos los elementos
Valor inicial	none
Descripción	Indica el lado del elemento que no debe ser adyacente a ninguna caja flotante

5.3. Visualización.

CSS define otras cuatro propiedades para controlar la visualización de los elementos: *display*, *visibility*, *overflow* y *z-index*. A través de ellas es posible ocultar y/o hacer invisibles las cajas de los elementos, por lo que son imprescindibles para realizar efectos avanzados y animaciones.

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

Si un elemento tiene la propiedad *display* como *none* no se visualizará y además el espacio que debería ocupar se pierde y es ocupado por otro elemento

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

Si un elemento tiene la propiedad *visibility* como *hidden* no se visualizará, pero mantiene el espacio que debería ocupar.

La propiedad *overflow* especifica cómo se visualiza un elemento cuando es mayor que la caja que lo contiene.

overflow	Parte sobrante de un elemento
Valores	visible | hidden | scroll | auto | inherit
Se aplica a	Elementos de bloque y celdas de tabla
Valor inicial	visible
Descripción	Permite controlar los contenidos sobrantes de un elemento.

CSS permite controlar la posición tridimensional de las cajas explícitamente posicionadas, es decir, indicar las cajas que se muestran delante o detrás de otras cajas cuando se producen solapamientos.

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

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

El valor más común de la propiedad *z-index* es un número entero. Aunque la especificación oficial permite los números negativos, en general, se considera el número 0 como el nivel más bajo. Cuanto más alto sea el valor numérico, más cerca del usuario se muestra la caja.

6. Texto.

6.1. Tipografía.

CSS define numerosas propiedades para modificar la apariencia del texto. De ellas, la propiedad básica es *color*, que permite establecer el color de la letra.

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

Aunque el color por defecto del texto depende del navegador, en general, los navegadores principales utilizan el color negro. Para establecer el color la letra de un texto se pueden utilizar cualquiera de las cinco formas que incluye CSS para definir un color. Veamos un ejemplo:

```
h1 {color: #148; }
p {color: pink; }
a, span {color: #4163E; }
div {color: rgb{34, 75, 93); }
```

Otra propiedad básica que define CSS se denomina *font-family* y se utiliza para indicar el tipo de letra con el que se muestra el texto.

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

El tipo de letra se puede indicar de dos formas:

- Mediante el nombre particular de una *familia* tipográfica, es decir, mediante el nombre del tipo de letra, como *Arial*, *Verdana*, *Garamond*, etc.
- Mediante el nombre genérico de una *familia* tipográfica, es decir, el nombre que designa el estilo del tipo de letra, como serif (similar a *Times New Romans*), sans-serif (tipo *Arial*), cursive (tipo *Comic Sans*), fantasy (tipo *Impact*) y monospace (tipo *Courier New*).

Para evitar el problema común de que el usuario no tenga instalada la fuente que quiere usar el diseñador, CSS permite indicar en la propiedad *font-family* más de un tipo de letra. El navegador probará en primer lugar con el primer tipo de letra indicado. Si el usuario la tiene instalada, el texto se muestra con ese tipo de letra; si no, el navegador irá probando con el resto hasta que encuentre alguna fuente instalada en el ordenador del usuario.

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

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

CSS permite utilizar una serie de palabras clave para indicar el tamaño de la letra del texto:

- Tamaño_absoluto: *xx-small*, *x-smal*, *small*, *medium*, *large*, *x-large*, *xx-large*.
- Tamaño_relativo: *larger* y *smaller*, que toman como referencia el tamaño de la letra del elemento padre.

La propiedad que controla la anchura de la letra es *font-weight*:

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

CSS permite establecer el estilo de la letra de un texto con la propiedad *font-style*:

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

CSS permite otra variación en el estilo del tipo de letra, controlado mediante la propiedad *font-variant*:

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

Por último, CSS proporciona una propiedad tipo "*shorthand*" denominada *font* y que permite indicar de forma directa algunas o todas las propiedades de la tipografía del texto:

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

6.2. Apariencia del texto.

CSS permite la alineación de texto mediante la propiedad *text-align*:

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

Esta propiedad no sólo alinea el texto que contiene el elemento, sino que también alinea todos sus contenidos como, por ejemplo, las imágenes.

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

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

<número> indica el múltiplo del tamaño de letra del elemento, por tanto, son equivalentes:

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

Además de la decoración que se puede aplicar a la tipografía, CSS define otros estilos y decoraciones para el texto en su conjunto a través de la propiedad *text-decoration*:

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

Una de las propiedades de CSS más desconocida y que permite variar de forma sustancial el aspecto del texto es *text-transform*:

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

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

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

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

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

CSS también permite controlar la separación entre las letras que forman las palabras y la separación entre las palabras que forman el texto. Para esto, se usan las propiedades *letter-spacing* y *word-spacing*, respectivamente:

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

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

7. Enlaces.

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

Con los atributos *id* o *class* no es posible aplicar distintos estilos a un mismo elemento en función de su estado. Por ello, CSS introduce un nuevo concepto llamado *pseudo-clases* y, en concreto, define cuatro diferentes:

- **:link**, aplica estilos a los enlaces que apuntan a páginas o recursos que aún no han sido visitados por el usuario.
- **:visited**, aplica estilos a los enlaces que apuntan a recursos que han sido visitados anteriormente por el usuario.
- **:hover**, aplica estilos al enlace sobre el que el usuario ha posicionado el puntero del ratón.
- **:active**, aplica estilos al enlace que está pinchando el usuario. Los estilos sólo se aplican desde que el usuario pincha el botón del ratón hasta que lo suelta, por lo que suelen ser unas pocas décimas de segundo.

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

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

Si se definen varias pseudo-clases diferentes sobre un mismo enlace, habrá que hacerlo en el orden en que se han descrito para que no se produzcan colisiones de estilos.

8. Imágenes.

CSS permite establecer la anchura y la altura de una imagen mediante las propiedades *width* y *height*, respectivamente, independientemente de su anchura y altura real. Veamos un ejemplo:

```
#destacada {
   width: 120px;
   height: 250px;
}
<img id="destacada" src="imagen.png" />
```

Usar anchuras y alturas diferentes a las reales produce deformaciones en las imágenes y el resultado estético es muy desagradable.

Por otra parte, establecer la anchura y altura para cada imagen mediante CSS es una práctica poco recomendable puesto que produce una sobrecarga de estilos. Por este motivo, aunque es una solución que no respeta la separación entre contenidos y presentación, se recomienda establecer la anchura y la altura de las imágenes mediante atributos de la etiqueta <img>.

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

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

9. Listas.

9.1. Viñetas personalizadas.

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

CSS define la propiedad *list-style-type* para controlar el tipo de viñeta que muestra una lista:

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

El valor *none* permite mostrar una lista sin viñetas, números o letras, por lo que es un valor muy utilizado en los menús de navegación de las páginas.

El resto de los valores se dividen en tres grupos:

- Valores gráficos: muestran las viñetas como círculos rellenos (*circle*), círculos vacíos (*disc*) o cuadrados (*square*).
- Valores numéricos: pueden ser *decimal*, *decimal-leading-zero*, *lower-roman*, *upper-roman*, *armenian* y *georgian*.
- Valores alfanuméricos: se controlan mediante *lower-latin*, *upper-latin*, *lower-alpha*, *upper-alpha* y *lower-greek*.

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

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

Para personalizar el aspecto de las viñetas se emplea la propiedad *list-style-image* que permite mostrar una imagen propia en vez de una viñeta automática.

list-style-image	Imagen de la viñeta
Valores	<url> | none | inherit
Se aplica a	Elementos de una lista
Valor inicial	none
Descripción	Permite reemplazar las viñetas automáticas por una imagen personalizada

CSS define una propiedad de tipo "*shorthand*" que permite establecer todas las propiedades de una lista de forma directa. El siguiente ejemplo indica que no se debe mostrar ni viñetas automáticas ni personalizadas: `ul { list-style: none; }`

9.2. Menús.

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

A continuación se muestra una transformación de una lista sencilla de enlaces de un menú de navegación:

```
<ul class="menu">
    <li><a href="#" >Elemento 1</a></li>
    <li><a href="#" >Elemento 2</a></li>
    <li><a href="#" >Elemento 3</a></li>
</ul>
```

La transformación de la lista en un menú se podría realizar de esta forma:

```
ul.menu {
    width: 180px;
    list-style: none;
    margin: 0;
    padding: 0;
    border: 1px solid #3A3A3A;
}
ul.menu li{
    border-bottom: 1px solid #3A3A3A;
    border-top: 1px solid #BBB;
    background: #F4F4F4;
}
ul.menu li a{
    padding: .2em 0 .2em  .5em;
    display: block;
    text-decoration: none;
    color:  #222;
}
```

- Definir anchura
- Eliminar viñetas, márgenes y espaciados
- Enmarcar la lista

- Establecer color de fondo y bordes a cada elemento del menú

Aplicar estilos a los enlaces:
- añadir relleno.
- mostrarlos como bloques
- modificar colores y decoración

10. Tablas.

El modelo de borde de las celdas de una tabla se selecciona mediante la propiedad *border-collapse*:

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

El modelo *collapse* fusiona de forma automática los bordes de las celdas adyacentes, mientras que el modelo *separate* fuerza a que cada celda muestre sus cuatro bordes.

Si se opta por el modelo *separate*, se puede utilizar la propiedad *border-spacing* para controlar la separación entre los bordes de cada celda.

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

Si solamente se indica una medida, se asignará a la separación horizontal y vertical. Si se indican dos medidas, la primera corresponderá a la separación horizontal y la segunda a la separación vertical.

Cuando se usa el modelo de bordes *separate*, se puede establecer el tratamiento que reciben las celdas vacías de una tabla mediante la propiedad *empty-cells*:

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

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

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

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

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

Se puede aplicar las pseudo-clase *:hover* para que el color de una fila varíe cuando el usuario pasa el ratón por encima de ella. En el ejemplo se muestra cómo se haría:

```
table tr:hover{ background: #FFFF66; }
```

Desafortunadamente, Internet Explorer 6 y sus versiones anteriores no soportan esta pseudo-clase en elementos que no sean enlaces.

11. Formularios.

CSS permite modificar el aspecto de los controles de un formulario. Por ejemplo, podemos cambiar el estilo de un botón:

```
.boton {
    border: 0;
    padding: 0;
    background-color: transparent;
    color: red;
    border-bottom: 1x solid red;
}
```

Por defecto, los campos de texto de los formularios no incluyen ningún espacio de relleno, por lo que el texto introducido por el usuario aparece pegado a los bordes del cuadro del texto. Añadiendo un pequeño *padding* a cada elemento <*input*>, se mejora notablemente el aspecto del formulario. Por ejemplo, podríamos escribir:

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

CSS permite resaltar el campo en el que el usuario está introduciendo datos mediante la pseudo-clase *:focus*. A continuación, se muestra un ejemplo:

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

12. Cursor.

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

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

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

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

13. Filtros y hacks.

13.1. Filtros.

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

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

A pesar de que utilizar filtros y "hacks" es una solución poco ortodoxa, en ocasiones es la única forma de conseguir que una página web muestre el mismo aspecto en todos los navegadores.

Los filtros permiten definir u ocultar ciertas reglas CSS para algunos navegadores específicos. Un caso especial de filtro son los *comentarios condicionales*. Se trata de un mecanismo propio de Internet Explorer que permite incluir hojas de estilos o definir reglas CSS específicas de una versión concreta.

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

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

Utilizando comentarios condicionales, también es posible incluir reglas CSS para versiones específicas de Internet Explorer:

```
<!- -[if gte IE 6]>
    <style type="text/css">
        @import (estilobasico6.css");
    </style>
<![endif]- ->
```

En el ejemplo anterior, solamente se carga la hoja de estilos *estilobasico6.css* si la versión de Internet Explorer es 6 o posterior, ya que *gte* se interpreta como "greater than or equal" ("mayor o igual que"). Otros valores disponibles son *gt* ("greater than" o "mayor que"), *lt* ("less than" o "menor que") y *lte* ("less than or equal" o "menor o igual que").

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

13.2. Hacks.

Los "hacks" permiten forzar el comportamiento de un navegador para que se comporte tal y como se espera. Sin embargo, se trata de una forma poco elegante de crear hojas de estilos.

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

En el siguiente ejemplo se muestra un borde inferior punteado en los <*div*> cuando la página se visualiza con cualquier navegador, excepto con Internet Explorer 6. Como en este navegador no se muestran correctamente los bordes punteados de 1 píxel de anchura, se decide mostrar un borde formado por una línea continua:

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

Otro "hack" muy conocido y utilizado por su sencillez es el "*underscore hack*" que consiste en que las propiedades cuyos nombres se indiquen con un guión bajo por delante, sólo son interpretadas por el navegador Internet Explorer 6 y sus versiones anteriores:

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

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

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

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