Manual de CSS, hojas de estilo

Autores del manual

Este manual ha sido realizado por los siguientes colaboradores de DesarrolloWeb.com:

Miguel Angel Alvarez
Director de DesarrolloWeb.com
http://www.desarrolloweb.com
(18 capítulos)

Serviweb
Diseño web Murcia
http://www.serviweb.es/
(1 capítulo)

Federico Elgarte
http://www.cssboulevar.com.ar/
(1 capítulo)

José Juan Corpas Martos
http://www.recursosflash.es
(2 capítulos)

Leonardo A. Correa
http://www.webnova.com.ar
(1 capítulo)

Fernando Campaña
Programación - Multimedia
http://www.rakidwam.com.ar/
(2 capítulos)

Manu Gutierrez
http://www.tufuncion.com
(1 capítulo)

Introducción a las CSS

El lenguaje HTML está limitado a la hora de aplicarle forma a un documento. Esto es así porque fué concebido para otros usos (científicos sobretodo), distinto a los actuales, mucho más amplios.

Para solucionar estos problemas los diseñadores han utilizado técnicas tales como la utilización de tablas imagenes transparentes para ajustarlas, utilización de etiquetas que no son estádares del HTML y otras. Estas "trampas" han causado a menudo problemas en las páginas a la hora de su visualizaciónen distintas plataformas.

Además, los diseñadores se han visto frustrados por la dificultad con la que, aun utilizando estos trucos, se encontraban a la hora de maquetar las páginas, ya que muchos de ellos venian maquetando páginas sobre el papel, donde el control sobre la forma del documento es absoluto.

Finalmente, otro antecedente que ha hecho necesario el desarrollo de esta tecnología consiste en que las páginas web tienen mezclado en su código HTML el contenido del documento con las etiquetas necesarias para darle forma. Esto tiene sus inconvenientes ya que la lectura del código HTML se hece pesada y difícil a la hora de buscar errores o depurar las páginas. Aunque, desde el punto de vista de la riqueza de la información y la utilidad de las páginas a la hora de almacenar su contenido, es un gran problema que estos textos est&aacuteen mezclados con etiquetas incrustadas para dar forma a estos: se degrada su utilidad.

En estas páginas de CSS pretendemos dar a conocer la tecnología con un enfoque práctico para que en pocos capítulos podáis usar las CSS de una manera depurada, reflejando toda nuestra experiencia en su uso. No pretendendemos explorar todos los aspectos de la tecnología ya que para realizar esto necesitariamos un la extensión de un libro entero.

> **Referencia:** Este manual trata los aspectos más teóricos de las hojas en cascada. En DesarrolloWeb.com también podemos encontrar otro manual con unos talleres prácticos de aplicación de las CSS.

*Artículo por **Miguel Angel Alvarez***

Características y ventajas de las CSS

El modo de funcionamiento de las CSS consiste en definir, mediante una sintaxis especial, la forma de presentación que le aplicaremos a:

- Un web entero, de modo que se puede definir la forma de todo el web de una sola vez.
- Un documento HTML o página, se puede definir la forma, en un pequeño trozo de código en la cabecera, a toda la página.
- Una porción del documento, aplicando estilos visibles en un trozo de la página.
- Una etiqueta en concreto, llegando incluso a poder definir varios estilos diferentes para una sola etiqueta. Esto es muy importante ya que ofrece potencia en nuestra programación. Podemos definir, por ejemplo, varios tipos de párrafos: en rojo, en azul, con margenes, sin ellos...

La potencia de la tecnología salta a la vista. Pero no solo se queda aquí, ya que además esta sintaxis CSS permite aplicar al documento formato de modo mucho más exacto. Si antes el HTML se nos quedaba corto para maquetar las páginas y teníamos que utilizar trucos para conseguir nuestros efectos, ahora tenemos muchas más herramientas que nos permiten definir esta forma:

- Podemos definir la distancia entre líneas del documento.
- Se puede aplicar identado a las primeras líneas del párrafo.
- Podemos colocar elementos en la página con mayor precisión, y sin lugar a errores.
- Y mucho más, como definir la visibilidad de los elementos, margenes, subrayados, tachados...

Y seguimos mostrandoos ventajas, ya que si con el HTML tan sólo podíamos definir atributos en las páginas con pixeles y porcentajes, ahora podemos definir utilizando muchas más unidades como:

- Pixels (px) y porcentaje (%), como antes.
- Pulgadas (in)
- Puntos (pt)
- Centímetros (cm)

Navegadores que lo soportan

Esta tecnología es bastante nueva, por lo que no todos los navegadores la soportan. En concreto, sólo los navegadores de Netscape versiones de la 4 en adelante y de Microsoft a partir de la versión 3 son capaces de comprender los estilos en sintaxis CSS. Además cabe destacar que no todos los navegadores implementan las mismas funciones de hojas de estilos, por ejemplo, Microsoft Internet Explorer 3 no soporta todo lo relativo a capas.

Esto quiere decir que debemos de usar esta tecnología con cuidado, ya que muchos usuarios no podrán ver los formatos que apliquemos a las páginas con CSS. Así pues, utilizad las hojas de estilos cuando estas no vayan a suponer un problema.

*Artículo por **Miguel Angel Alvarez***

Usos de las CSS I

Vamos ahora a describir los diferentes usos de las CSS introducidos en el anterior capítulo. Vamos por orden, describiendo los puntos según su dificultad e importancia.

CSS tiene una sintaxis propia, la veremos a través de ejemplos. Luego se verá con detalle

Hemos partido este capítulo en dos partes por su extensión y por haber varias formas distintas de aplicar estilos, aquí veremos las más sencillas y en el capítulo siguiente otras más complicadas pero más potentes.

Pequeñas partes de la página

Para definir estilos en secciones reducidas de una página se utiliza la etiqueta **<SPAN>**. Con su atributo **style** indicamos en sintaxis CSS las características de estilos. Lo vemos con un ejemplo, pondremos un párrafo en el que determinadas palabras las vamos a visualizar en

color verde.

```
<p>
Esto es un párrafo en varias palabras <SPAN style="color:green">en color verde</SPAN>. resulta muy fácil.
</p>
```

Que tiene como resultado:

Esto es un párrafo con varias palabras en color verde. resulta muy fácil.

Estilo definido para una etiqueta

De este modo podemos hacer que toda una etiqueta muestre un estilo determinado. Por ejemplo, podemos definir un párrafo entero en color rojo y otro en color azul. Para ello utilizamos el atributo **style**, que es admitido por todas las etiquetas del HTML (siempre y cuando dispongamos de un navegador compatible con CSS).

```
<p style="color:#990000">
Esto es un párrafo de color
rojo.
</p>
<p style="color:#000099">
Esto es un párrafo de color
azul.
</p>
```

Que tiene como resultado:

Esto es un párrafo de color rojo.

Esto es un párrafo de color azul.

Estilo definido en una parte de la página

Con la etiqueta **<DIV>** podemos definir secciones de una página y aplicarle estilos con el atributo **style**, es decir, podemos definir estilos de una vez a todo un bloque de la página.

```
<div style="color:#000099; font-weight:bold">
<h3>Estas etiquetas van en <i>azul y
negrita</i></h3>
<p>
Seguimos dentro del DIV, luego permanecen los etilos
</p>
</div>
```

Que tiene como resultado:

<h3>Estas etiquetas van en azul y negrita</h3>
Seguimos dentro del DIV, luego permanecen los etilos

Hasta aquí este capítulo, en el siguiente seguiremos viendo formas más avanzadas de usar las CSS.

*Artículo por **Miguel Angel Alvarez***

Usos de las CSS y II

Estilo definido para toda una página

Podemos definir, en la cabecera del documento, estilos para que sean aplicados a toda la página. Es una manera muy cómoda de darle forma al documento y muy potente, ya que estos estilos serán seguidos en toda la página y nos ahorraremos así muchas etiquetas HTML que apliquen forma al documento. Además, si deseamos cambiar los estilos de la página lo haremos de una sola vez.

Este ejemplo es más complicado, puesto que se utiliza la sintaxis CSS de manera más avanzada. Pero no te preocupes puesto que con los ejemplos irás aprendiendo su uso y más tarde comentaremos la sintaxis en profundidad.

En el ejemplo vemos que se utiliza la etiqueta <STYLE> colocada en la cabecera de la página para definir los distintos estilos del documento.

A grandes rasgos, entre de <STYLE> y </STYLE>, se coloca el nombre de la etiqueta que queremos definir los estilos y entre llaves -{}- colocamos en sintaxis CSS las características de estilos.

```
<html>
<head>
 <title>Ejemplo de estilos para toda una p&aacute;gina</title>
 <STYLE type="text/css">
 <!--
 H1 {text-decoration: underline; text-align:center}
 P {font-Family:arial,verdana; color: white; background-color: black}
 BODY {color:black;background-color: #cccccc; text-indent:1cm}
 // -->
 </STYLE>
</head>
<body>
<h1>P&aacute;gina con estilos</h1>
Bienvenidos...
<p>Siento ser tan hortera, pero esto es un ejemplo sin m&aacute;s
importancia</p>
</body>
</html>
```

Como se puede apreciar en el código, hemos definido que la etiqueta <H1> se presentará

- Subrayado
- Centrada

También, por ejemplo, hemos definido que el cuerpo entero de la página (etiqueta <BODY>) se le apliquen los estilos siguientes:

- Color del texto negro
- Color del fondo grisaceo
- Margen lateral de 1 centímetro

Caber destacar que si aplicamos estilos a la etiqueta <BODY>, estos serán heredados por el resto de las etiquetas del documento. Esto es así siempre y cuando no se vuelvan a definir esos estilos en las siguientes etiquetas, en cuyo caso el estilo de la etiqueta más concreta será el que mande. Puede verse este detalle en la etiqueta <P>, que tiene definidos estilos que ya fueron definidos para <BODY>. Los estilos que se tienen en cuenta son los de la etiqueta <P>, que es más concreta.

Por último, ha de apreciarse los comentarios HTML que engloban toda la declaración de estilos: <!--Declaración de estilos-->. Estos comentarios se utilizan para que los navegadores antiguos, que no comprenden la sintaxis CSS, no incluyan ese texto en el cuerpo de la página. Si no se pusiera, los navegadores antiguos (por ejemplo Netscape 3) escribirían ese "feo código" en la página.

Pulsa para ver el ejemplo anterior.

Hemos preparado la misma página, pero con declaraciones de estilos distintas, para que comprobéis las diferencias en la forma del documento con sólo unos cambios en sus estilos. **Puedes verla pinchando aquí.**

Estilo definido para todo un sitio web

Una de las características más potentes de la programación con hojas de estilos consiste en que, de una vez, podemos definir los estilos de todo un sitio web. Esto se consigue creando un archivo donde tan sólo colocamos las declaraciones de estilos de la página y enlazando todas las páginas del sitio con ese archivo. De este modo, todas las páginas comparten una misma declaración de estilos y, por tanto, si la cambiamos, cambiarán todas las páginas. Con las ventajas añadidas de que se ahorra en líneas de código HTML (lo que reduce el peso del documento) y se evita la molestia de definir una y otra vez los estilos con el HTML, tal como se comentó anteriormente.

Veamos ahora cómo el proceso para incluir estilos con un fichero externo.

1- Creamos el fichero con la declaración de estilos

Es un fichero de texto normal, que puede tener cualquer extensión, aunque le podemos asignar la extensión .css para aclararnos qué tipo de archivo es. El texto que debemos incluir debe ser escrito exclusivamente en sintaxis CSS, es decir, sería erroneo incluir código HTML en el: etiquetas y demás. Podemos ver un ejemplo a continuación.

```
P  {
 font-size : 12pt;
 font-family : arial,helvetica;
 font-weight : normal;
}
H1  {
 font-size : 36pt;
 font-family : verdana,arial;
 text-decoration : underline;
 text-align : center;
 background-color : Teal;
}
TD  {
 font-size : 10pt;
 font-family : verdana,arial;
 text-align : center;
 background-color : 666666;
}
BODY  {
 background-color :
#006600;
 font-family : arial;
 color : White;
}
```

2- Enlazamos la página web con la hoja de estilos

Para ello, vamos a colocar la etiqueta <LINK> con los atributos

- **rel="STYLESHEET"** indicando que el enlace es con una hoja de estilos
- **type="text/css"** porque ela archivo es de texto, en sintaxis CSS
- **href="estilos.css"** indica el nombre del fichero fuente de los estilos

Veamos una página web entera que enlaza con la declaración de estilos anterior:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<html>
<head>
 <link rel="STYLESHEET" type="text/css" href="estilos.css">
 <title>P&aacute;gina que lee estilos</title>
</head>
<body>
<h1>P&aacute;gina que lee estilos</h1>
Esta p&aacute;gina tiene en la cabecera la etiqueta necesaria para enlazar con la hoja de estilos. Es muy
f&aacute;cil.
<br>
<br>
<table width="300" cellspacing="2" cellpadding="2" border="0">
<tr>
   <td>Esto est&aacute; dentro de un TD, luego tiene estilo propio, declarado en el fichero externo</td>
</tr>
<tr>
   <td>La segunda fila del TD</td>
</tr>
</table>
</body>
</html>
```

El resultado conseguido se puede ver aquí

Reglas de importancia en los estilos

Los estilos se heredan de una etiqueta a otra, como se indicó anteriormente. Por ejemplo, si tenemos declarado en el <BODY> unos estilos, por lo general, estas declaraciones también afectatarán a etiquetas que estén dentro de esta etiqueta, o lo que es lo mismo, dentro de todo el cuerpo.

En muchas ocasiones más de una declaración de estilos afecta a la misma porción de la página. Siempre se tiene en cuenta la declaración más particular. Pero las declaraciones de estilos se pueden realizar de múltiples modos y con varias etiquetas, también entre estos modos hay una jerarquía de importancia para resolver conflictos entre varias declaracionesde estilos distintas para una misma porción de página. Se puede ver a continuación esta jerarquía, primero ponemos las formas de declaración más generales, y por tanto menos respetadas en caso de conflicto:

- Declaración de estilos con fichero externo. (Para todo un sitio web)
- Declaración de estilos para toda la página. (Con la etiqueta <STYLE> en la cabecera de la página)
- Estilos definidos en una parte de la página. (Con la etiqueta <DIV>)
- Definidos en una etiqueta en concreto. (Utilizando el atributo style en la etiqueta en cuestión)
- Declaración de estilo para una porción pequeña del documento. (Con la etiqueta <SPAN>)

Ya vimos cómo incluir estilos en la página, de todas las maneras posibles e hicimos un repaso con la lista anterior. Ahora estás en condiciones de empezar a usar las hojas de estilo en cascada para mejorar tus páginas y aumentar la productividad de tu trabajo. Pero estate

atento a los siguientes capítulos donde aprenderás las lecciones que te faltan para dominar bien la materia: conocer la sintaxis, los distintos atributos de estilos y otras cosas que mejorarán tus páginas.

*Artículo por **Miguel Angel Alvarez***

Otra manera de definir estilos en un archivo externo

Veamos ahora otra manera de importar una declaración externa de estilos CSS: @import url("archivo_a_importar.css"), que se utiliza para importar unas declaraciones de estilos guardadas en la ruta que se indica entre paréntesis. (las comillas son opcionales, pero los paréntesis son obligatorios, por lo menos, en Explorer).

Se debe incluir en la declaración de estilos global a una página, es decir entre las etiquetas <style type="text/css"> y </style>, que se colocan en la cabecera del documento.

Es importante señalar que la sentencia de importación del archivo CSS se debe escribir en la primera línea de la declaración de estilos, algo parecido al código siguiente.

```
<style type="text/css">
@import url ("estilo.css");
body{
   background-color:#ffffcc;
}
</style>
```

El funcionamiento es el mismo que si escribiésemos todo el fichero a importar dentro de las etiquetas de los estilos, con la salvedad de que, si redefinimos dentro del código HTML (entre las etiquetas </style>) estilos que habían quedado definidos en el archivo externo, los que se aplicarán serán los que hayamos redefinido.

Así, en el ejemplo anterior, aunque hubiésemos definido en estilo.css un color de fondo para la página, el color que prevalecería sería el definido a continuación de la importación: #ffffcc

La diferencia entre este tipo de importación del tipo y la que hemos visto anteriormente:

```
<link rel="stylesheel" type="text/css" href="hoja.css">
```

Es que @import url ("estilo.css") se suele utilizar cuando hay unas pautas básicas en el trabajo con los estilos (que se definen en un archivo a importar) y unos estilos específicos para cada página, que se definen a continuación, dentro del código HTML entre las etiquetas </style>, como es el caso del ejemplo visto anteriormente.

*Artículo por **Miguel Angel Alvarez***

Sintaxis CSS

Tal como se vió en los ejemplos la sintaxis es bastante sencilla y repetitiva. Vamos a verla:

- Para definir un estilo se utilizan atributos como font-size,text-decoration... segudos de dos puntos y el valor que le deseemos asignar. Podemos definir un estilo a base de definir muchos atributos separados por punto y coma.
  Ejemplo:
  font-size: 10pt; text-decoration: underline; color: black; (el último punto y coma de la lista de atributos es opcional)
- Para definir el estilo de una etiqueta se escribe la etiqueta seguida de la lista de atributos encerrados entre llaves.
  Ejemplo:
  H1{text-align: center; color:black}
- Los valores que se pueden asignar a los atributos de estilo se pueden ver en una tabla en el siguiente capítulo. Muchos estos valores son unidades de medida, por ejemplo, el valor del tamaño de un margen o el tamaño de la fuente. Las unidades de medida son las siguientes:

Puntos	pt
Pulgadas	in
Centímetros	cm
pixels	px

Hasta aquí, queda dicho todo lo relativo a la sintaxis. En el siguiente capítulo podrás encontrar una lista de los atributos de las hojas de estilo en cascada.

*Artículo por **Miguel Angel Alvarez***

Atributos de las hojas de estilo

Tanto para practicar en tu aprendizaje como para trabajar con las CSS lo mejor es disponer de una tabla donde se vean los distintos atributos y valores de estilos que podemos aplicarle a las páginas web.

Aquí puedes ver la tabla de los atributos CSS, tenla a mano cuando utilices las CSS.

Nombre del atributo	Posibles valores	Ejemplos
FUENTES - FONT		
color	valor RGB o nombre de color	color: #009900; color: red;
Sirve para indicar el color del texto. Lo admiten casi todas las etiqetas de HTML. No todos los nombres de colores son admitidos en el estandar, es aconsejable entonces utilizar el valor RGB.		
font-size	xx-small | x-small | small | medium | large | x-large | xx-large Unidades de CSS	font-size:12pt; font-size: x-large;
Sirve para indicar el tamaño de las fuentes de manera más rígida y con mayor exactitud.		
font-family	serif | sans-serif | cursive | fantasy | monospace Todas las fuentes habituales	font-family:arial,helvetica; font-family: fantasy;
Con este atributo indicamos la familia de tipografia del texto. Los primeros valores son genéricos, es decir, los exploradores las comprenden y utilizan las fuentes que el usuario tenga en su sistema. También se pueden definir con tipografías normales, como ocurría en html. Si el nombre de una fuente tiene espacios se utilizan comillas para que se entienda bien.		
font-weight	normal | bold | bolder | lighter | 100 | 200 | 300 | 400 | 500 | 600 | 700 | 800 | 900	font-weight:bold; font-weight: 200;

Sirve para definir la anchura de los caracteres, o dicho de otra manera, para poner negrillas con total libertad.
Normal y 400 son el mismo valor, así como bold y 700.

font-style	normal | italic | oblique	font-style:normal; font-style: italic;

Es el estilo de la fuente, que puede ser normal, itálica u oblícua. El estilo oblique es similar al italic.

PÁRRAFOS - TEXT

line-height	normal y unidades CSS	line-height: 12px; line-height: normal;

El alto de una línea,y por tanto, el espaciado entre líneas. Es una de esas características que no se podian mofificar utilizando HTML.

text-decoration	none | [underline || overline || line-through]	text-decoration: none; text-decoration: underline;

Para establecer la decoración de un texto, es decir, si está subrayado, sobrerayado o tachado.

text-align	left | right | center | justify	text-align: right; text-align: center;

Sirve para indicar la alineación del texto. Es interesante destacar que las hojas de estilo permiten el justificado de texto, aunque recuerda que no tiene por que funcionar en todos los sistemas.

text-indent	Unidades CSS	text-indent: 10px; text-indent: 2in;

Un atributo que sirve para hacer sangrado o márgenes en las páginas. Muy útil y novedosa.

text-transform	capitalize | uppercase | lowercase | none	text-transform: none; text-transform: capitalize;

Nos permite transformar el texto, haciendo que tenga la primera letra en mayúsculas de todas las palabrs, todo en mayúsculas o minúsculas.

FONDO - BACKGROUND

Background-color	Un color, con su nombre o su valor RGB	background-color: green; background-color: #000055;

Sirve para indicar el color de fondo de un elemento de la página.

Background-image	El nombre de la imagen con su camino relativo o absoluto	background-image: url(mármol.gif) ; background-image: url(http://www.x.com/fondo.gif)

Colocamos con este atributo una imagen de fondo en cualquier elemento de la página, se puede ver una **página de ejemplo**

BOX - CAJA

Margin-left	Unidades CSS	margin-left: 1cm; margin-left: 0,5in;

Indicamos con este atributo el tamaño del margen a la izquierda

Margin-right	Unidades CSS	margin-right: 5%; margin-right: 1in;

Se utiliza para definir el tamaño del margen a la derecha

Margin-top	Unidades CSS	margin-top: 0px; margin-top: 10px;

Indicamos con este atributo el tamaño del margen arriba de la página

Margin-bottom	Unidades CSS	margin-bottom: 0pt; margin-top: 1px;

Con el se indica el tamaño del margen en la parte de abajo de la página

Padding-left	Unidades CSS	padding-left: 0.5in; padding-left: 1px;

Indica el espacio insertado, por la izquierda, entre el borde del elemento-continente y el contenido de este. Es parecido a el atributo cellpadding de las tablas.
El espacio insertado tiene el mismo fondo que el fondo del elemento-continente.

Padding-right	Unidades CSS	padding-right: 0.5cm; padding-right: 1pt;

Indica el espacio insertado, en este caso por la derecha, entre el borde del elemento-continente y el contenido de este. Es parecido a el atributo cellpadding de las tablas.
El espacio insertado tiene el mismo fondo que el fondo del elemento-continente.

Padding-top	Unidades CSS	padding-top: 10pt; padding-top: 5px;

Indica el espacio insertado, por arriba, entre el borde del elemento-continente y el contenido de este.

Padding-bottom	Unidades CSS	padding-right: 0.5cm; padding-right: 1pt;

Indica el espacio insertado, en este caso por abajo, entre el borde del elemento-continente y el contenido de este.

Border-color	color RGB y nombre de color	border-color: red; border-color: #ffccff;
Para indicar el color del borde del elemento de la página al que se lo aplicamos. Se puede poner colores por separado con los atributos border-top-color, border-right-color, border-bottom-color, border-left-color.		
Border-style	none | dotted | solid | double | groove | ridge | inset | outset	border-style: solid; border-style: double;
El estilo del borde, los valores significan: none=ningun borde, dotted=punteado (no parece funcionar), solid=solido, double=doble borde, y desde groove hasta outset son bordes con varios efectos 3D.		
border-width	Unidades CSS	border-width: 10px; border-width: 0.5in;
El tamaño del borde del elemento al que lo aplicamos. Para ver otros ejemlos de Box **pulsar aquí**		
float	none | left | right	float: right;
Sirve para alinear un elemento a la izquierda o la derecha haciendo que el texto se agrupe alrededor de dicho elemento. Igual que el atributo align en imagenes en sus valores right y left.		
clear	none | right | left	clear: right;
Si este elemento tiene a su altura imagenes u otros elementos alineados a la derecha o la izquierda, con el atributo clear hacemos que se coloque en un lugar donde ya no tenga esos elementos a el lado que indiquemos. Para ver una página que utiliza float y clear **pulsar aquí**		

La especificación de estilos CSS es muy amplia, seguro que se queda en el tintero algún atributo de estilo, pero creo que la inmensa mayoría están, y por supuesto la selección de los más importantes.

*Artículo por **Miguel Angel Alvarez***

Trucos avanzados con CSS

Las hojas de estilos son un tema largo del que se han escrito libros enteros. Nosotros nos centramos en los temas prácticos y por ello vamos a acabar ya con este capítulo, en unos cuantos puntos

Definir estilos utilizando clases

Las clases nos sirven para crear definiciones de estilos que se pueden utilizar repetidas veces.

Una clase se puede definir entre las etiquetas <STYLE> (en la cabecera del documento), o en un archivo externo a la página. Para definirlas utilizamos la siguiente sintaxis, un punto seguido del nombre de la clase y entre llaves los atributos de estilos deseados. De esta manera:

.nombredelaclase {atributo: valor;atributo2:valor2; ...}

Una vez tenemos una clase, podemos utilizarla en cualquier etiqueta HTML. Para ello utilizaremos el atributo class, poniéndole como valor el nombre de la clase, de esta forma:

<ETIQUETA class="nombredelaclase">

Ejemplo de la utilización de clases

```
<html>
<head>
```

```
<title>Ejemplo de la utilizaci&oacute;n de clases</title>
<STYLE type="text/css">
 .fondonegroletrasblancas {background-color:black;color:white;font-size:12;font-family:arial}
 .letrasverdes {color:#009900}
</STYLE>
</head>
<body>
<h1 class=letrasverdes>Titulo 1</h1>
<h1 class=fondonegroletrasblancas>Titulo 2</h1>
<p class=letrasverdes>
Esto es un p&aacute;rrafo con estilo de letras verdes</p>
<p class=fondonegroletrasblancas>
Esto es un p&aacute;rrafo con estilo de fondo negro y las letras blancas. Es todo!</p>
</body>
</html>
```

Ver el ejemplo anterior

Estilo en los enlaces

Una técnica muy habitual, que se puede realizar utilizando las hojas de estilo en cascada y no se podía en HTML, es la definición de estilos en los enlaces, quitandoles el subrayado o hacer enlaces en la misma página con distintos colores.

Para aplicar estilo a los enlaces debemos definirlos para los distintos tipos de enlaces que existen (visitados, activos...). Utilizaremos la siguiente sintaxis, en la declaración de estilos global de la página (<STYLE>) o del sitio (Definición en un archivo externo):

Enlaces normales
A:link {atributos}

Enlaces visitados
A:visited {atributos}

Enlaces activos (Los enlaces están activos en el presiso momento en que se pincha sobre ellos)
A:active {atributos}

Enlaces hover (Cuando el ratón está encima de ellos, solo funciona en ieplorer)
A:hover {atributos}

El atributo para definir enlaces sin subrayado es **text-decoration:none**, y para darles color es color:tu_color.

También podemos definir el estilo de cada enlace en la propia etiqueta <A>, con el atributo style. De esta manera podemos hacer que determinados enlaces de la página se vean de manera distinta

Ejemplo de estilos en enlaces

```
<html>
<head>
 <title>Ejemplos de estilo en enlaces</title>
 <STYLE type="text/css">
  A:link {text-decoration:none;color:#0000cc;}
  A:visited {text-decoration:none;color:#ffcc33;}
  A:active {text-decoration:none;color:#ff0000;}
  A:hover {text-decoration:underline;color:#999999;font-weight:bold}
 </STYLE>
</head>
<body>
<a href="http://dominioinexistente.nofunciona.com">Enlace normal</a>
```

```
<br>
<br>
<a href="enlaces.html">Enlace visitado</a>
Pulsar este enlace para verlo activo,
poner el rat&oacute;n por encima para que cambie.
</body>
</html>
```

Ver el ejemplo anterior

URL como valor de un atributo:

Determinados atributos de estilos, como **background-image** necesitan una URL como valor, para indicarlas podemos definir tanto caminos relativos como absolutos. Así pues, podemos indicar la URL de la imagen de fondo de estas dos maneras:

background-image: url(fondo.gif) En caso de que la imagen esté en el mismo directorio que la página.
background-image: url(http://www.desarrolloweb.com/images/fondo.gif)

Ocultar estilos en navegadores antiguos

En caso de definir dentro de la etiqueta <STYLE> unas declaraciones de estilos debemos asegurarnos que estas no se imprimirán en la página web con navegadores antiguos. Pensar en un navegador que no reconozca la etiqueta <STYLE>, pensará que corresponde con algo que no entiende y se olvidará de la etiqueta. Lo siguiente que encuentra es texto normal y hará que este se vea en la página, como con cualquier otro texto.

Para evitarlo debemos ocultar con comentarios HTML (<!-- esto es un comentario -->) todo lo que hay dentro de la etiqueta <STYLE>. Se puede ver un ejemplo de esto a continuación:

De este modo hemos terminado la primera parte del manual de CSS, que espero pueda ayudar a hacer páginas mejores y más rápidamente. Ahora el manual continua explicando conceptos sobre capas y maquetación CSS, entre otros asuntos.

Quiero recordaros que siempre es útil ver como han hecho sus páginas otros progradores de Internet. Para ver una página definida enteramente con hojas de estilos podemos visitar Web Site Album, que está incluso maquetada con CSS. También podemos visitar la dirección www.guiarte.com, que está maquetada con tablas, pero todos los estilos se aplican con css.

Para ver otros manuales, artículos y enlaces a páginas que enseñan a utilizar las hojas de estilos visitar la sección correspondiente a CSS de nuestro directorio.

*Artículo por **Miguel Angel Alvarez***

Qué son las capas

Veamos una pequeña introducción a lo que son las capas, la etiqueta HTML <DIV> utilizada para construirla y los atributos CSS con los que podemos aplicar estilos.

Como ya hemos visto en nuestro manual de CSS, <SPAN> sirve para aplicarle estilo a una pequeña parte de una página HTML. Por ejemplo, con ella podríamos hacer que una parte de

un párrafo se coloree en rojo. Con <SPAN> no es habitual englobar un trozo muy grande de texto, por ejemplo el que comprenda a varios párrafos.

Con <DIV> también podemos aplicar estilo a partes de la página HTML.

La diferencia entre <SPAN> y <DIV> es que con esta última si que podemos aplicar estilo a una parte más amplia de la página, por ejemplo a tres párrafos. **Además que la etiqueta <DIV> tiene un uso adicional que es el de crear divisiones en la página a las que podremos aplicar una cantidad adicional de atributos para modificar sus comportamientos**. Por ejemplo, con el atributo align de HTML podemos alinear la división al centro, izquierda, derecha o justificado. Pero **su uso más destacado es el de convertir esa división en una capa**.

Una capa es una división, una parte de la página, **que tiene un comportamiento muy independiente** dentro de la ventana del navegador, ya que la podemos colocar en cualquier parte de la misma y la podremos mover por ella independientemente, por poner dos ejemplos. En el uso de capas se basan muchos de los efectos más comunes del DHTML.

Las etiquetas <LAYER> e <ILAYER> tienen como objetivo construir capas, pero estas no son compatibles más que con Netscape, por lo que es recomendable utilizar la etiqueta <DIV> para hacer capas preferentemente a las otras dos.

Los atributos que podemos aplicar a estas etiquetas, pero en concreto a las dos recomendadas <SPAN> y <DIV>, son principalmente de estilos CSS. Estos atributos nos permiten, como hemos podido ver en el manual de hojas de estilo en cascada de desarrolloweb, modificar de una manera muy exhaustiva la presentación de los contenidos en la página. Para aplicar estilos a estas etiquetas se utiliza el atributo de HTML style, de esta manera:

```
<SPAN style="text-decoration:underline;font-weight:bold">...</SPAN>

<DIV style="color:red;font-size:10px">...</DIV>
```

Como ya pudimos ver muchos ejemplos en el manual de CSS, nos referimos a él para ampliar esta información. Pero no habíamos visto todavía una serie de atributos que nos sirven para posicionar la división en la página como una capa. Estos atributos se pueden aplicar a la etiqueta <DIV> que es la que servía para crear capas compatibles con todos los navegadores.

Los atributos para que la división sea una capa son varios y se pueden ver a continuación.

```
<div id="c1" style="position:absolute; left: 200px; top: 100px;">
Hola!
</div>
```

El primero, **position**, indica que se posicione de manera absoluta en la página y los segundos, **left** y **top**, son la distancia desde el borde izquierdo de la página y el borde superior.

Hay otros atributos especiales para capas como **width** y **height** para indicar la anchura y altura de la capa, **Z-index** que sirve para indicar qué capas se ven encima de qué otras, **clip** que sirve para recortar una capa y hacer que partes de ella no sean visibles, o **visibility** para definir si la capa es visible o no. Estos y otros atributos los veremos en el siguiente capítulo, donde hablaremos del posicionamiento de capas.

*Artículo por **Miguel Angel Alvarez***

Atributos para capas

Hemos visto en el capítulo anterior qué son las capas y algunas pequeñas muestras sobre cómo crearlas y darle algún estilo. Ahora vamos a ver en detenimiento los atributos específicos para aplicar posicionamiento a una capa y otros estilos.

Antes que nada cabe decir que una capa puede tener cualquier atributo de estilos de los que hemos visto en el manual de CSS. Así, el atributo color indica el color del texto de la capa, el atributo font-size indica el tamaño del texto y así con todos los atributos ya vistos.

Ahora bien, existen una serie de atributos que sirven para indicar la forma, el tamaño de las capas, la visibilidad, etc, que no hemos visto en capítulos anteriores y que veremos a continuación.

Atributo position

Indica el tipo de posicionamiento de la capa. Puede tener dos valores, relative o absolute.

- relative indica que la posición de la capa es relativa a el lugar donde se estaba escribiendo en la página en el momento de escribir la capa con su etiqueta

- absolute indica que la posición de la capa se calcula con respecto al punto superior izquierdo de la página.

Atributo top

Indica la distancia en vertical donde se colocará la capa. Si el atributo position es absolute, top indica la distancia del borde superior de la capa con respecto al borde superior de la página. Si el atributo position era relative, top indica la distancia desde donde se estaba escribiendo en ese momento en la página hasta el borde superior de la capa.

Atributo left

Básicamente funciona igual que el atributo top, con la diferencia que el atributo left indica la distancia en horizontal a la que estará situada la capa.

Atributo height

Sirve para indicar el tamaño de la capa en vertical, es decir, su altura.

Atributo width

Indica la anchura de la capa

Atributo visibility

Sirve para indicar si la capa se puede ver en la página o permanece oculta al usuario. Este atributo puede tener tres valores.

- visible sirve para indicar que la capa se podrá ver.
- hidden indicará que la capa está oculta.
- inherit es el valor por defecto, que quiere decir que hereda la visibilidad de la capa donde

está metida la capa en cuestión. Si la capa no está metida dentro de ninguna otra se supone que está metida en la capa documento, que es toda la página y que siempre está visible.

Atributo z-index

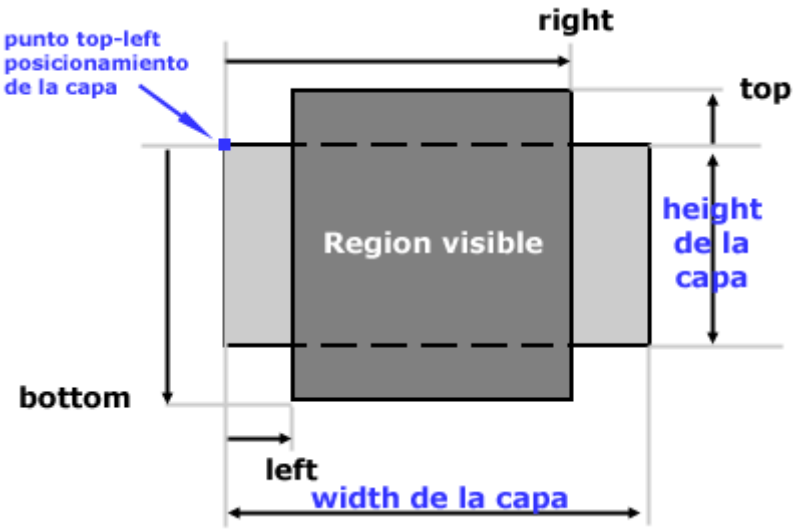
Sirve para indicar la posición sobre el eje z que tendrán las distintas capas de la página. Dicho de otra forma, con z-index podemos decir qué capas se verán encima o debajo de otras, en caso de que estén superpuestas. El atributo z-index toma valores numéricos y a mayor z-index, más arriba se verá la página.

Atributo clip

Es un atributo un poco difícil de explicar. En concreto sirve para recortar determinadas áreas de la capa y que no se puedan ver. Una capa que está visible se puede recortar para que se vea, pero que no se vea algún trozo de esta. El cliping se indica por medio de 4 valores, con esta sintaxis.

rect (<top>, <right>, <bottom>, <left>)

Los valores <top>, <right>, <bottom> y <left> indican distancias que se pueden apreciar en este esquema.



Este es un ejemplo de capa que utiliza todos los atributos que hemos visto en este artículo y alguno más para aplicar estilo a la capa.

```
<div style="clip: rect(0,158,148,15); height: 250px; width: 170px; left: 10px; top: 220px; position: absolute;
visibility: visible; z-index:10; font-size: 14pt; font-family: verdana; text-align: center; background-color: #bbbbbb">

Esta capa tiene un cliping, por eso se ve entrecortada.
<br>
<br>
Esto es una capa de prueba

</div>
```

Puede verse el ejemplo en una página web, donde también podrá apreciarse el efecto conseguido al realizar el cliping.

*Artículo por **Miguel Angel Alvarez***

Problema con el posicionamiento absoluto de capas

He recibido una consulta en mi correo sobre colocación de capas de manera absoluta, pero en la que no nos importe la definición de la pantalla del usuario y otros ir y venir de los elementos HTML. Nuestro compañero expresó su duda de la siguiente manera:

Si trabajamos con position:absolute dando un left y un top funciona si tienes tu página alineada a la izquierda. Mi página está alineada en el centro, entonces lo que sucede es que dependiendo de la resolución de pantalla que tengas (ancho de 800px,1024px,etc) me baila toda la página y no cuadra nada.

Primero que todo, debemos saber que si trabajamos con el position relative las capas se colocan en el lugar donde aparecen dentro del código HTML. De este modo, si colocamos una capa con position relative dentro de una celda de una tabla, dicha capa aparecería dentro de la celda donde la estamos colocando, independientemente del lugar donde se sitúe la celda al cambiar la definición de la pantalla.

El problema de esta solución es que la capa haría crecer la celda de la tabla donde queremos colocarla (al igual que cualquier otro elemento HTML que colocásemos dentro de la tabla) y es muy probable que nuestro diseño no nos permita este hecho. Seguramente ya habrías notado este problema y si no es así te invito a que crees la capa que intentas colocar con el atributo position a relative para ver si con eso tu problema ya está resuelto.

En casi todos los casos, la capa que intentamos colocar va a tener que tener el position absolute, porque con relative no arreglamos totalmente el problema. Entonces volvemos a el problema inicial, que era situar la capa con position absolute en el lugar exacto, independientemente de la definición de pantalla.

La solución final que propongo pasa por aplicar algún truquillo. De hecho, estuve hace unos días preguntándome sobre esa cuestión y al final encontré la solución, aunque no se me ocurrió a mi, sino que la extraje de www.cross-browser.com.

La idea es un poco compleja y para su puesta en marcha debemos realizar una serie de acciones que, sinceramente, considero excesivas para un problema inicialmente sencillo. Así pues, que no asuste lo que voy a soltar a continuación, que luego trataré de explicarlo un poco mejor.

Nuestro esquema de trabajo consistirá en una capa con posición relativa, que nos servirá de "ancla" y otra con la posición absoluta, donde colocaremos el contenido final a mostrar en la capa.

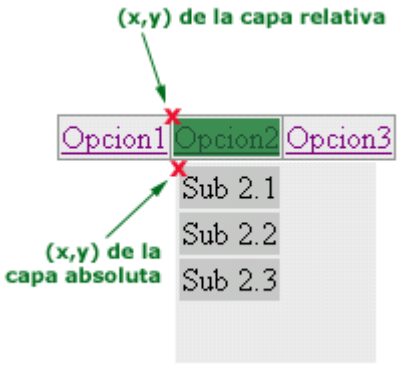
La capa relativa la colocaremos en el lugar aproximado donde queramos que aparezca la capa absoluta. La capa absoluta la posicionaremos, una vez cargada la página, en un lugar próximo a la capa relativa. Por supuesto, estas acciones las vamos a tener que realizar con Javascript, que es el lenguaje que nos permite actualizar las posiciones de las capas dinámicamente.

Detenidamente

Decíamos que habría que colocar una capa relativa cercana al lugar donde tiene que aparecer la capa con position absolute. Insisto en que las capas relativas se colocan en el lugar donde las metemos dentro del código HTML, por lo que será fácil colocar la capa relativa en el lugar exacto y que este lugar sea válido para cualquier definición.

La segunda capa, la que tiene el contenido final, la pondremos inicialmente en una posición cualquiera y escondida, de manera que no se vea que está mal colocada. Una vez terminada de cargar la página, podremos acceder a la posición de la capa relativa, extrayendo sus valores top y left y colocándolos en los correspondientes top y left de la capa con posición absoluta. Una vez marcada la posición de la capa absoluta podemos volverla visible.

A la vista de la imagen siguiente, la capa con posición relativa la hemos colocado en el enlace. En realidad habría tres capas con posición relativa para poder posicionar otras tantas capas con posición absoluta. La parte que vemos sombreada de verde corresponde al espacio que abarcaría la capa relativa.



Su posición sería la que está marcada por el aspa roja que aparece en su esquina superior izquierda. Dicha posición depende del lugar donde aparezcan los enlaces en la página.

Luego, con Javascript deberíamos asignar la posición de la capa absoluta de una manera parecida a esta.

left de la capa absoluta = left de la capa relativa
top de la capa absoluta = top de la capa relativa + altura de la capa relativa

Podemos sumarle algún píxel más a la posición de la capa, si es que queremos moverla un poco abajo y a la derecha, tal como hemos visto en la imagen.

No pretendo en este artículo, muy a mi pesar y por falta de espacio y tiempo, explicar cómo se hacen esas operaciones de Javascript. Advierto que si no se conoce nada de Javascript va a ser imposible ponerse con una tarea tan tediosa como el manejo de capas. Si por el contrario, ya hemos tenido contacto con Javascript y DHTML anteriormente, no debería ser un problema realizar esas acciones.

Referencias Javascript

En DesarrolloWeb tenemos un par de manuales de Javascript, que sería necesario estudiar para empezar a introducirse en el lenguaje.
- Programación en Javascript I
- Programación en Javascript II

En el Taller de Javascript tenemos algún artículo sobre tratamiento dinámico de capas.

En el momento en el que escribo estas líneas no tenemos un manual de DHTML y tratamiento de capas completo, por lo que sería recomendable estudiarlo en alguno de los enlaces recomendados de nuestra sección de DHTML del buscador.

*Artículo por **Miguel Angel Alvarez***

Maquetación CSS

En este artículo vamos a conocer la maquetación de paginas utilizando Hojas de estilos en cascada (CSS). Veremos cómo realizar este tipo de maquetación, junto con algunas ventajas e inconvenientes. Para muchos será todavía un campo por explorar. Aunque no vamos a entrar en grandes detalles, vamos a intentar dar a conocer la maquetación con CSS para que cubrir la posible laguna por parte del lector. En capítulos sucesivos ampliaremos la información y ofreceremos tutoriales más prácticos.

Como se ha podido aprender en el Manual de CSS, las hojas de estilo en cascada ayudan a separar el contenido de la forma, es decir, los elementos que componen una página de la forma con la que se muestran. Además, CSS ayuda en gran medida a la definición de estilos en la página, ya que permite ajustar de una manera mucho más precisa cualquier aspecto de cualquier elemento de la página.

La maquetación con CSS lleva la utilización de las hojas de estilo a su grado máximo, de manera que cualquier definición del aspecto de la página se realiza en la declaración CSS que enlazamos con el documento HTML. Para definir la situación de los elementos en la página se utilizan las capas, a las que se aplica un posicionamiento a través también de las hojas de estilo.

Para crear las capas se utilizan etiquetas <DIV>, en las que se introducen los elementos que queramos que aparezcan en la página. Los elementos dentro de los <DIV> también se pueden anidar, para heredar las propiedades y posicionamiento de las capas padre.

En la maquetación por capas se definen únicamente etiquetas <DIV> y las tablas sólo se utilizan para mostrar información tabulada, es decir, mostrada en filas y columnas. Cabe señalar que en la maquetación tradicional se utilizan las tablas para ajustar la posición de los elementos en la página. Seguro que muchos de los lectores están muy familiarizados con el uso de tablas para maquetar una web, pues se trata de una técnica bastante sencilla, aunque bastante poco práctica y que complica un tanto el código de las páginas web resultantes.

La maquetación por tablas es una forma de crear webs más evolucionada y que mejora en mucho a la maquetación tradicional, aunque también tiene sus inconvenientes.

Veamos primero algunas de las **ventajas** de la maquetación CSS:

- **La separación del contenido de la página y del estilo o aspecto con el que se deben mostrar**. Tener en cuenta que, cuanto más separemos estos dos elementos, más sencillo será el mantenimiento de las páginas y el procesamiento de la información. Con ello también podremos obtener páginas más limpias y claras.
- **Ahorro en la transferencia**. Si todos los estilos y posiciones de los elementos se introducen en un documento externo, liberaremos el código de la página y ocupará mucho menos. Como la declaración de estilos se almacena en la caché del navegador, sólo se transfiere en la primera página que se visita del sitio, con lo que la segunda y posteriores páginas que se soliciten se cargarán mucho más rápido.
- **Facilidad para alterar el aspecto de la página sin tocar el código HTML**. Como toda la información de los estilos y el posicionamiento de las capas se encuentra en un mismo archivo, si deseamos cambiar cualquier elemento de la página -ya sea su posición o su aspecto-, sólo tenemos que actualizar la hoja de estilos y los cambios se verán automáticamente en todo el web.

Como decimos, también hay algunas **desventajas**:

- **Compatibilidad con navegadores antiguos**. Se necesita que el visitante disponga de un navegador bastante avanzado. La mayoría de los visitantes disponen de navegadores que soportan características avanzadas de las CSS, pero todavía hay mucha gente que no ha actualizado sus equipos o que navega en sistemas de sólo texto. Los navegadores que no soportan hojas de estilos, por lo menos leerán el código de la página y lo mostrarán sin ningún posicionamiento. Ello puede resultar fastidioso, pero por lo menos visualizarán todos los datos de la página, aunque descolocados y sin estilo.
- **Diferencias entre navegadores**. Dependiendo del navegador también cambian las etiquetas de estilos soportadas, por lo que las páginas puede que no se vean exactamente igual en unos clientes que en otros. También, al igual que ocurre con HTML, hay atributos no estándar o que tienen valores por defecto diferentes. Cuando se empieza con la maquetación en CSS, puede resultar un tema bastante complicado y crearnos bastantes dolores de cabeza, no obstante, se trata de, poco a poco, ir aprendiendo todos los atributos y los navegadores donde se visualizan o no.
- **Dificultad**. Sin duda, si estamos acostumbrados al HTML, pasar a CSS resulta bastante más complicado y requiere un estudio más profundo. Sin embargo, este paso nos brindará un mayor control de los elementos de la página y ampliará nuestras fronteras a la hora de maquetar.

Ejemplo CssZenGarden

Existe un sitio muy interesante e ilustrador que nos puede ayudar a conocer rápidamente la potencia de las CSS y hacernos una idea de lo que puede significar su uso. Es un sitio donde se muestra un contenido y un diseño bastante logrado. Además, dispone de varios enlaces para poder ver el mismo sitio, con el mismo contenido, pero con distinto aspecto. De ese modo podemos ver cómo se puede llegar a alterar el diseño de una página con tan solo cambiar la hoja de estilos.

La URL del sitio es http://www.csszengarden.com. Es muy interesante que seleccionéis otros diseños para ver el cambio radical que puede tener las páginas web con distintas hojas de estilos.

Nosotros hemos explorado un poco las capacidades de las CSS y hemos realizado un ejemplo de diseño de CssZenGarden por nuestra cuenta. Podemos verlo en nuestro propio servidor en este enlace.

Por donde continuar

En DesarrolloWeb.com hemos publicado una línea de artículos para explicar el proceso de creación de una web, maquetada por completo en CSS. Además, podemos referiros a algunas páginas en inglés que ofrecen diversas documentaciones y ejemplos:

http://www.stylegala.com/
http://www.cssbeauty.com/
http://www.cssvault.com/
http://www.webstandardsawards.com/

*Artículo por **Miguel Angel Alvarez***

Formas de aplicar estilos en maquetación CSS

Vamos a ver otra vez distintos modos de aplicar estilos a las páginas. Es un tema que ya vimos en el manual de CSS, pero merece la pena refrescar conceptos y ampliar la información que se ofreció en su día.

Aplicación de estilo a etiquetas

Se puede asignar el estilo a una etiqueta concreta de HTML. Para ello, en la declaración de estilos escribimos la etiqueta y entre llaves, los atributos de estilo que deseemos.

```
body {
   background-color: #f0f0f0;
   color: #333366;
}
```

Podemos aplicar el mismo estilo en un conjunto de etiquetas. Para ello, indicamos las etiquetas seguidas por comas y luego, entre llaves, los atributos que queramos definir.

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

En este caso se define que los encabezados de nivel 1 y los párrafos, tengan letra roja.

Definición de clases

Podemos utilizar una clase si deseamos crear un estilo específico, para luego aplicarlo a distintos elementos de la página. Las clases en la declaración de estilos se declaran con un punto antes del nombre de la clase.

```
.miclase{
   color: blue;
}
```

Para asignar el estilo definido por una clase en un elemento HTML, simplemente se añade el atributo class a la etiqueta que queremos aplicar dicha clase. El atributo class se asigna al nombre de la clase a aplicar. Por ejemplo:

```
<p class="miclase">este párrafo tiene el estilo definido en la clase "miclase".</p>
```

El párrafo anterior se presentaría con color azul. La definición de clases y su utilización es sencilla, pero veamos un ejemplo más detallado:

Para la siguiente declaración de estilos:

```
body, td, p{
   background-color: #000000;
   color: #ffffff;
}

.inverso{
   background-color: #ffffff;
   color: #000000;
}
```

Se ha definido un fondo negro y color del texto blanco para el cuerpo de la página, así como las celdas y los párrafos. Luego se ha declarado una clase, de nombre "inverso", con los colores al revés, es decir, fondo blanco y texto negro.

```
<body>
<p>Hola esto es un parrafo normal</p>
<p class="inverso">Párrafo con los colores invertidos</p>
<table>
<tr>
   <td class="inverso">INVERSO</td>
   <td>NORMAL</td>
</tr>
</table>
</body>
```

Esta página tiene, generalmente, el fondo negro y el texto blanco. El primer párrafo, que es un párrafo normal, sigue esa definición general de estilos, pero el segundo párrafo, al que se ha aplicado la clase "inverso", tiene el fondo blanco y el texto en negro. Por lo que respecta a la tabla, en su primera celda se ha asignado la clase "inverso", por lo que se verá con fondo blanco y color de texto en negro. Mientras que la segunda celda, que no tiene asignada ninguna clase, se presentará como se definió en la regla general.

Para conocer los resultados obtenidos en el anterior ejemplo podemos verlo en una página aparte.

Estilos que sólo se utilizan una vez

También podemos tener un estilo específico para un único elemento, que no va a repetirse en ningún otro caso. Para ello tenemos los estilos asignados por identificador. Los identificadores se definen en HTML utilizando el atributo id en la etiqueta que deseamos identificar. El valor del atributo id será el que definamos nosotros.

```
<div id="capa1">
```

En la hoja de estilos, para definir el aspecto de ese elemento con id único, se escribe el carácter almohadilla, seguido del identificador indicado en la etiqueta y entre llaves los atributos css que deseemos.

```
#capa1{
   font-size: 12pt;
   font-family: arial;
}
```

En este caso se ha asignado fuente de tamaño 12 puntos y cuerpo arial.

Como se puede concluir en la lectura de estas líneas, generalmente se prefiere utilizar estilos definidos en clases a los definidos con identificadores, a no ser que estemos seguros que ese estilo no se va a repetir en todo el documento.

> **Referencia:** En nuestro taller de CSS hemos publicado varios artículos para mostrar el proceso de maquetación de una página en CSS.

*Artículo por **Miguel Angel Alvarez***

Notación de colores CSS

Con CSS se puede especificar colores para cada elemento HTML de la página, incluso hay elementos que podrían admitir varios colores, como el color de fondo o el color del borde. Pero bueno, vamos a ver ahora es las distintas maneras de escribir un color en una declaración CSS.

Porque lo más habitual es que especifiquemos un color con su valor RGB, de una manera similar a como aprendimos a definir colores en HTML. Pero en CSS tenemos otras maneras de declarar colores que pueden interesarnos, como mínimo para poder entender el código CSS cuando lo veamos escrito.

Notación hexadecimal RGB

Esta notación es la que ya conocemos. Se especifican los tres valores de color (rojo, verde y azul) con valores en hexadecimal entre 00 y FF.

```
background-color: #ff8800;
```

Notación hexadecimal abreviada

Esta notación es muy parecida a la anterior, pero permite abreviar un poco la declaración del color, indicando sólo un número para cada valor rojo, verde y azul. Por ejemplo, para especificar el color de antes (#ff8800) podríamos haber escrito:

```
background-color: #f80;
```

Nombre del color

También podemos definir un color por su nombre. Los nombres de colores son en inglés, los mismos que sirven para especificar colores con HTML.

```
color: red;
border-color: Lime;
```

Notación de color con porcentajes de RGB

Se puede definir un color por los distintos porcentajes de valores RGB. Si todos los valores están al 100% el color es blanco. Si todos están al 0% obtendríamos el negro y con combinaciones de distintos porcentajes de RGB obtendríamos cualquier matiz de color.

```
color: rgb(33%, 0%, 0%);
```

Notación por valores decimales de RGB, de 0 a 255

De una manera similar a la notación por porcentajes de RGB se puede definir un color directamente con valores decimales en un rango desde 0 a 255.

```
color: rgb(200,255,0);
```

De entre todas estas notaciones podemos utilizar la que más nos interese o con la que nos sintamos más a gusto. Nosotros en nuestros ejemplos venimos utilizando la notación hexadecimal RGB por habernos acostumbrado a ella en HTML.

Color transparente

Para finalizar, podemos comentar que también existe el color transparente, que no es ningún color, sino que específica que el elemento debe tener el mismo color que el fondo donde está. Este valor, transparent, sustituye al color. Podemos indicarlo en principio sólo para fondos de elementos, es decir, para el atributo background-color.

```
background-color: transparent;
```

*Artículo por **Miguel Angel Alvarez***

Definición de estilos CSS Shorthand

Shorthand:

Vamos a explicar cómo escribir de forma reducida nuestras reglas CSS para que nuestros archivos de estilo tengan menos peso y sean más entendibles a la hora de una actualización.

Según la W3C hay dos formas de escribir la misma regla de CSS: la estándar y la shorthand. Una es la larga y la otra es la reducida.

Propiedad Font (fuente)

font-style || font-variant || font-weight || font-size / line-height || familia de fuente

Ejemplo:

P {font: italic normal bold 12px/14pt Verdana, Tahoma, Arial}

Propiedad Background (fondo)

background-color || background-image || background-repeat || background-attachment || background-position

Ejemplo:

Body {background: #FFF url(../images/ejemplo.gif) no-repeat fixed center}

Margin (Margen)

longitud | porcentaje | auto

Ejemplo:

Body {margin: 5px} /* todos los márgenes a 5px */ P {margin: 2px 4px} /* márgenes superior e inferior a 2px, márgenes izquierdo y derecho a 4px */ DIV {margin: 1px 2px 3px 4px} /* margen superior a 1px, right margin a 2px, bottom margin a 3px, left margin a 4px */

Padding (Relleno)

longitud | porcentaje | auto

Ejemplo:

Body {padding: 2em 3em 4em 5em} /* Si definimos cuatro valores estamos aplicando el padding superior, derecho, inferior e izquierdo */ Body {padding: 2em 4em) /* Si definimos dos o tres valores, los valores faltantes se toman del lado opuesto: superior e inferior a 2em, derecho e izquierdo a 4em */ Body {padding: 5em} /* Si definimos un solo valor se aplican a todos los lados */

Border (Borde)

border-width || border-style || color

Ejemplo:

```
H3 {border: thick dotted blue}
```

*Artículo por **Federico Elgarte***

Pseudo-element en CSS

Los pseudo-element (pseudo-elementos, si se me permite la traducción al castellano) sirven para aplicar estilos a partes más específicas dentro de una etiqueta. Es decir, para el ejemplo concreto de la etiqueta párrafo, con los pseudo-elementos podemos definir el estilo para la primera letra del párrafo y para la primera línea.

Pseudo-element first-letter

Un efecto habitual de algunas publicaciones, por ejemplo las de cuentos para niños, es hacer más grande la primera letra de una página y decorarla de alguna manera. Con CSS podemos aplicar estilos específicos y hacer, por ejemplo, que esa primera letra sea más grande y tenga un color distinto del resto del párrafo.

Se utiliza de esta manera:

```
P:first-letter {
font-size: 200%;
color: #993333; font-weight: bold;
}
```

Así estamos asignando un tamaño de letra 200% más grande del propio del párrafo. También estamos cambiando el color de esa primera letra.

De entre todas las propiedades de estilos, sólo algunas se pueden aplicar a los pseudo-elementos first-letter. Son las siguientes, según la especificación del W3C: font properties, color properties, background properties, 'text-decoration', 'vertical-align' (sólo si 'float' está asignado a 'none'), 'text-transform', 'line-height', margin properties, padding properties, border properties, 'float', 'text-shadow' y 'clear'.

Se puede ver un ejemplo de aplicación de un estilo con first-letter.

Pseudo-element first-line

Como adelantaba, este pseudo-elemento, sirve para asignar un estilo propio a la primera línea del texto. Es posible que hayamos visto alguna revista o periódico que utilice este estilo para

remarcar las primeras líneas del párrafo. Un ejemplo de su uso sería el siguiente:

```
P:first-line {
  text-decoration: underline;
  font-weight: bold;
}
```

Las propiedades de estilos que se pueden aplicar al pseudo-element first-line son las siguientes: font properties, color properties, background properties, 'word-spacing', 'letter-spacing', 'text-decoration', 'vertical-align', 'text-transform', 'line-height', 'text-shadow' y 'clear'.

Se puede ver un ejemplo de aplicación de un estilo con first-line.

Uso de clases con los pseudo-elementos

En determinadas ocasiones podemos necesitar crear una clase (class) de CSS a la que asignar los pseudo-elementos, de modo que estos no se apliquen a todas las etiquetas de la página. Por ejemplo, podemos desear que solamente se modifique el estilo de la primera línea del texto en algunos párrafos y no en todos los de la página.

Una clase se define con el nombre de la etiqueta seguido de un punto y el nombre de la clase. Si además deseamos definir un pseudo-elemento, deberíamos indicarlo a continuación, con esta sintaxis:

```
P.nombreclase:first-line {
   font-weight: bold;
}
```

Pseudo-elementos en CSS2

Aparte de first-line y first-letter, en las especificaciones de CSS 2 existen otros pseudo elementos que voy a nombrar para conocimiento de los lectores, aunque profundizaré en su uso. Se tratan de before y after y sirven para insertar "contenidos generados" antes y después del elemento al que asignemos estos pseudo-element.

Un ejemplo de ello es:

```
P.nota:before {
   content: "Nota: "
}
```

Así se ha definido una clase de párrafo llamada "note" en la que se indica que antes de la propia nota se debe incluir el texto indicado, osea, "Nota: ".

Atención a la compatibilidad con CSS2, que, por lo menos para estos elementos, no está soportada en versiones 6 de Internet Explorer. Firefox, en cambio, sí que es compatible con estas características de CSS2.

*Artículo por **Miguel Angel Alvarez***

El Modelo de Caja en CSS

Una primera aproximación visual

Tarde o temprano, todo libro o taller práctico de CSS queda bajo la influencia del Modelo de Caja.
Es uno de los elementos básicos de las Hojas de Estilo en Cascada y por lo tanto su correcta interpretación resulta fundamental a la hora de lograr los resultados deseados en un diseño, por más simple que éste resulte.
Para entrar en tema, vamos a construir un sencillo ejemplo utilizando un único elemento al que aplicaremos un estilo. El resultado producido será analizado con la ayuda de una figura en la que hemos modelado el orden de apilado de los elementos del en una disposición tridimensional que nos ayudará a comprenderlo.
Supongamos el siguiente código (lo mostramos fuera de su contexto, restringiéndonos a lo significativo para el ejemplo):

El elemento

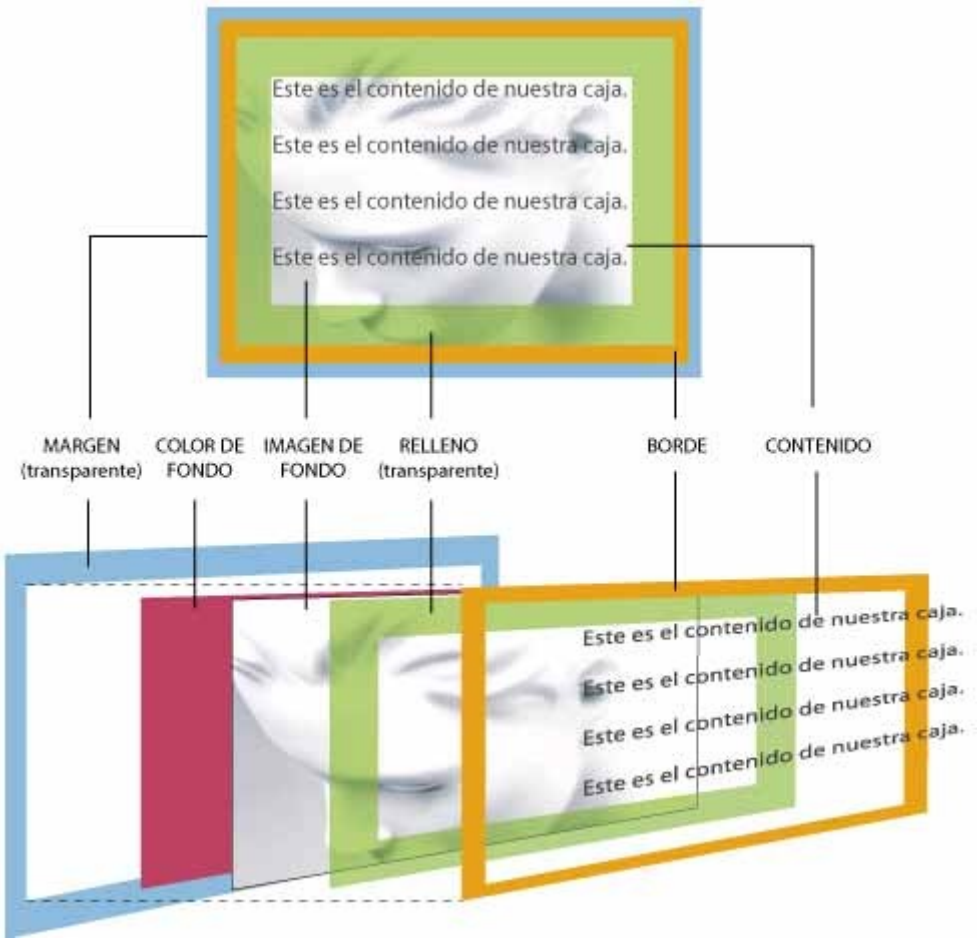
```
<div>
<p>Este es el contenido de nuestra caja.</p>
<p>Este es el contenido de nuestra caja.</p>
<p>Este es el contenido de nuestra caja.</p>
<p>Este es el contenido de nuestra caja.</p>
</div>
```

El estilo

```
div {
background-color: #be4061; /*color bordó para el fondo*/
background-image: url('cabeza.jpg');
border: 10px solid #e7a219; /*color naranja para el borde*/
margin: 10px;
padding: 20px;
}

p {
margin: 0 0 20px 0; /*margen inferior de 20 px para el párrafo*/
padding: 0;
}
```

El código anterior generará una caja como la que muestra la figura siguiente, en la que adicionalmente se ha dado color a los elementos transparentes (margen y relleno) solo para hacerlos visibles.
Un detalle interesante que puede apreciarse en la representación tridimensional en que la capa superior del apilamiento no es el borde, como podría suponerse intuitivamente.
La capa situada encima de todas las demás es la de Contenido.
Aunque el caso específico sea materia de otro artículo, comentaremos que esta disposición fue utilizada por el diseñador Douglas Bowman de Stopdesign para el rediseño del sitio de Blogger , logrando las armoniosas líneas curvas de sus páginas mediante CSS, ubicando imágenes en la capa de Contenidos de modo que oculten el borde anguloso de las cajas.



Áreas y propiedades

Cada caja posee, además de su área de Contenido, otras tres áreas opcionales:

* Área de Margen - Margin
* Área de Relleno - Padding
* Área de Borde - Border

Cada área, a su vez, puede dividirse en cuatro segmentos según su posición: izquierdo (left), derecho (right), superior (top) e inferior (bottom).
El tamaño de cada área o de sus segmentos está dado por el valor de las respectivas propiedades, definidas en forma global o discriminadas por segmento.
Por ejemplo, la siguiente sentencia asignará un margen homogéneo de 20 píxeles alrededor de la caja:

```
div { margin: 20px }
```

Si en cambio se desea asignar valores distintos a cada uno de los segmentos, pueden reflejarse los cuatro valores numéricos siguiendo el orden top - right - bottom - left.
El siguiente ejemplo asigna 10 píxeles arriba, 5 a la derecha, 20 abajo y nada a la izquierda:

```
div { margin: 10px 5px 20px 0 }
```

Pueden especificarse valores también con la siguiente notación, en la que ya no es necesario mantener el orden:

```
div {
margin-top: 10px ;
margin-right: 5px ;
margin-bottom: 20px ;
}
```

En cualquier caso puede obviarse el valor 0 ya que es el valor que toman las propiedades por defecto.
La lista completa de propiedades es la siguiente:

Propiedades del Margen
"margin-top", "margin-right", "margin-bottom", "margin-left" y "margin"

Propiedades del Relleno
"padding-top", "padding-right", "padding-bottom", "padding-left" y "padding"

Propiedades del Borde
1) Ancho (width)
"border-top-width", "border-right-width", "border-bottom-width", "border-left-width" y "border-width". Pueden ser valores específicos o los valores "thin" (fino), "medium" (medio) y "thick" (grueso)

2) Color (color)
"border-top-color", "border-right-color", "border-bottom-color", "border-left-color" y "border-color"

3) Estilo (style)
"border-top-style", "border-right-style", "border-bottom-style", "border-left-style" and "border-style". Toma una serie de posibles valores, tales como: none, hidden, dotted, dashed, solid, double, groove, ridge, inset y outset. Es una propiedad algo conflictiva ya que no todos los navegadores interpretan sus valores de la misma manera.

Como corolario de esta aproximación al modelo de caja resta analizar qué es lo que se verá en cada área cuando la página se muestre en un navegador:

- En el área de Contenido y en la de Relleno se verá aquello que se determine en la propiedad "background" del elemento (un color o una imagen, según el orden de apilado).
- En el área de Borde se verá aquello que se determine en las propiedades del Borde (ancho, color y estilo).
- El área de Margen es siempre transparente.

El Secreto

Hay un solo secreto para comprender cabalmente cada una de las propiedades y su utilización: probar, probar y probar.

*Artículo por **Fernando Campaña***

Utilizar porcentajes para tamaños de texto con CSS

El porcentaje es otra de las medidas o unidades que podemos utilizar en los atributos de hojas de estilo en cascada (CSS) para definir un tamaño. En este artículo veremos ejemplos acerca de modificar los tamaños de los textos por medio de porcentajes, para conseguir nuevos tamaños que sean relativos a los que se están utilizando.

Por ejemplo, podríamos definir un estilo para escribir con un texto el doble de grande del que se esté trabajando:

```
<span style=”font-size:200%”>Este texto es el doble de grande</span>
```

Esto quiere decir que el texto será el doble de grande, 2 por las unidades de texto que estemos trabajando. Por ejemplo, si estamos trabajando con tamaños de texto de 10pt, el texto dentro del anterior span sería 20pt. El del siguiente código ejemplifica este caso concreto:

```
<span style="font-size:10pt;">Hola amigos <span style="font-size:200%">Este texto es el doble de grande</span>
</span>
```

Lo mismo se puede hacer, pero para definir un texto menor, asignando porcentajes por debajo del 100%. Por ejemplo, si quisiéramos hacer un texto de la mitad del tamaño utilizaríamos la siguiente etiqueta:

```
<span style="font-size:50%">Este texto es la mitad del anterior<,/span>
```

Si estuviéramos trabajando con un tamaño de texto de 16pt, con la anterior etiqueta se escribiría con tamaño 8pt. El código sería el siguiente:

```
<span style="font-size:16pt;">Hola amigos
<span style="font-size:50%">Este texto es la mitad del anterior</span>
</span>
```

Ahora vamos a definir un par de clases para un texto mayor y menor, que podríamos utilizar para aumentar y reducir el texto respectivamente.

```
<style type="text/css">
  .mayor {font-size:150%}
  .menor {font-size:75%}
</style>
```

Este código indica que la clase mayor es un texto el 150%, es decir, la mitad más que el anterior, y la clase menor un texto del 75%, es decir tres cuartas partes del anterior. Podríamos utilizar estas clases con un código como este:

Este es un texto normal y este es mayor, este vuelve a ser normal y este es menor

Los distintos ejemplos de este artículo podemos verlos en una página aparte.

*Artículo por **Miguel Angel Alvarez***

Problemas del Modelo de Caja en Internet Explorer 5

Una primera aproximación visual

Partiremos de la base de que el lector posee las nociones básicas sobre el Modelo de Caja.

Básicamente, la "caja" en CSS puede ser asimilada a una tabla con una sola celda.
De forma obviamente rectangular, esa caja puede tener bordes (border), márgenes transparentes por fuera de los bordes (margin) y relleno transparente por dentro de los bordes (padding). Cualquiera de estos atributos puede ser asignado para cada uno de los cuatro lados de la caja separadamente.
El contenido de la caja se ubicará dentro del área de relleno.

En general, podemos hablar de dos tipos de diseño:

- Los diseños "líquidos" no asignan un valor específico al ancho (width) y alto (height) de la caja, por lo que resulta que esas dimensiones surgirán del contenedor de la caja y de la extensión del contenido. Concretamente, el ancho de esa caja será todo el permitido por la estructura que la contenga (utilizará todo el ancho disponible) y el alto será el necesario para mostrar todo el contenido ("crecerá" hacia abajo todo lo necesario).
- Los diseños "estáticos" surgen cuando se decide asignar a la caja un ancho específico mediante la asignación de un valor concreto a la propiedad "width".

Los problemas comienzan cuado trabajamos con diseños estáticos. Nuestra caja tendrá un ancho dado, pero no todos los navegadores interpretarán ese ancho de la misma manera.

Para el W3C el ancho de una caja se mide desde el límite interno del relleno izquierdo (left-padding) hasta el límite interno del relleno derecho (right-padding). En caso de no existir relleno se toman los límites internos izquierdo y derecho del borde (border-right y border-left).

Cuando Microsoft lanzó su Internet Explorer 5 para Windows (IE5/win) no respetó éste estándar, interpretando la propiedad width como el ancho comprendido entre los límites exteriores del borde (border-left y border-right).
Para verlo con más claridad supongamos un ejemplo sencillo: una caja de 100 pixeles de ancho, 10 de relleno, 5 de borde y 10 de márgen, todos ellos uniformes.

El elemento

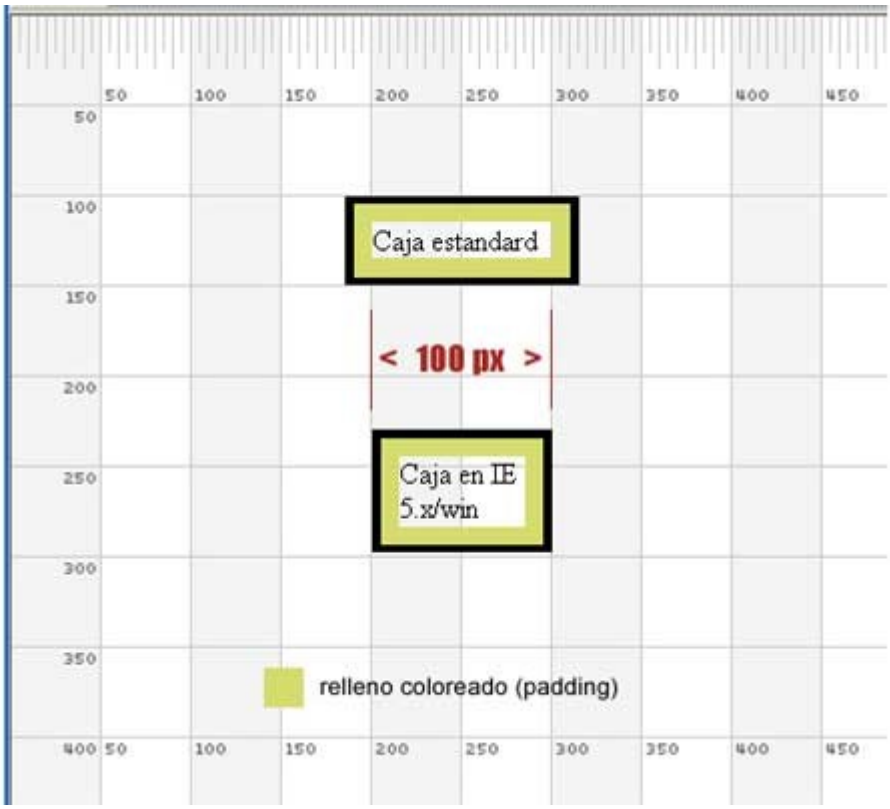
```
<body>
<div>Aquí el contenido de la caja</div>
</body>
```

Aplicando CSS ...

```
div {
width: 100px;
padding: 10px;
border: 5px solid black;
margin: 10px;
}
```

En la imagen siguiente (en la que se ha aplicado un color al padding únicamente con fin ilustrativo) puede verse una representación de la diferente interpretación entre el IE5/win y los

demás navegadores y anticipar las desastrosas consecuencias para el diseño final.



Tal como se aprecia, la caja se ve más pequeña cuando se la visualiza con IE5/win.
La versión del navegador para Mac (IE5/mac) no tiene ese inconveniente e interpreta la caja según el estándar W3C.
Afortunadamente, Microsoft remedió este problema en el Explorer 6, pero hay millones de usuarios que todavía utilizan IE5.x/win!

La solución en general

La solución para este tipo de problemas consiste en utilizar "trucos" que aprovechan limitaciones o errores de los navegadores en la interpretación de CSS.
Hay varios de estos recursos, denominados en general Box Model Hacks, que basan su acción en suministrarle al navegador conflictivo un ancho de caja tal que su representación resulte la esperada, pero sin que este valor afecte a los otros navegadores.
En el caso de nuestro ejemplo, para igualar las áreas de contenido, deberíamos decirle al IE5.x/win (y únicamente al IE5.x/win) que el ancho es un valor tal que incluya relleno y borde:

borde izquierdo + relleno izquierdo + contenido + relleno derecho + borde derecho

5 px + 10 px + 100 px + 10 px + 5 px
o sea …

width: 130 px;

De este modo, la caja se vería idéntica en todos los navegadores.
El problema con muchas de las soluciones (hacks) propuestas es que de un modo u otro terminaban afectando a otros navegadores menos populares. Por lo tanto, aún cuando la cosa mejoraba sustancialmente en términos de audiencia, faltaba dar todavía con un recurso que no generara efectos indeseados.

La mejor solución: Tan hack

La solución fue propuesta por Edwardson Tan y reune las inusuales características de ser simple y perfectamente eficaz.

Volviendo a nuestro ejemplo:

```
<body>
<div>Aquí el contenido de la caja</div>
</body>
```

y aplicando CSS con algunos cambios ...

```
div {
width: 100px;
padding: 10px;
border: 5px solid black;
margin: 10px;
}
/* Aquí está el truco, el Tan Hack */
* html div {
width: 130px;
w\idth: 100px;
}
```

Analicemos un poco el código del truco.

Observamos un asterisco (*) al comienzo. El asterisco en CSS es conocido como selector universal y alude a todos los elementos que están contenidos dentro de otro.
Luego encontramos html, el elemento raíz de toda página, y luego nuestro div.
La regla * html div { } se aplicará a todo elemento div contenido en un elemento html que a su vez esté contenido en otro. Suena raro, no?

Para cualquier navegador distinto del Explorer la regla será interpretada como errónea y por lo tanto ignorada, ya que no existe ningún elemento que contenga a html, que acabamos de decir que es el elemento raíz.

El Explorer (en todas las versiones) parece creer en la existencia de un misterioso elemento que engloba a html (obviamente se trata de un defecto de estos navegadores), por lo que la regla le resultará válida.

Hasta aquí logramos que únicamente los IE resulten afectados y redefinan el ancho a 130px. Solo resta lograr que esa redefinición del ancho (en el ejemplo a 130px) sea válida únicamente para el IE5.x/win, ya que habíamos dicho que tanto el IE5/mac como el IE6 interpretaban correctamente el modelo de caja y no necesitan ningún truco para respetar nuestro diseño. Para eso utilizamos la siguiente línea, de apariencia un poco extraña:

w\idth: 100px;

Tanto el IE5/mac como el IE6 son capaces de ignorar la barra invertida (escape) dentro del nombre de una propiedad, siempre que se cumplan determinadas condiciones que detallamos más adelante.
Los dos navegadores "corregirán" w\idth transformándolo en width y volverán a modificar el ancho a 100px, su valor correcto.

El IE5.x/win no puede manejar esa barra invertida dentro del nombre y por lo tanto considera a esa línea como errónea y la ignora. El resultado es que el ancho, para IE5.x/win y sólo para él, permanece en 130px.

Es exactamente lo que queríamos!

Un último comentario acerca de la barra invertida dentro del nombre: hay ciertas limitaciones en cuanto a su posición.

- Debe estar dentro del nombre
- No debe estar justo antes de cualquiera de las primeras seis letras del alfabeto (a, b, c, d, e o f) ya que se interpretaría como un carácter hexadecimal y echaría a perder el truco.

*Artículo por **Fernando Campaña***

Por qué diseñar con CSS

Las tablas existen y existieron desde el comienzo en HTML, pero no se crearon para diseñar un sitio, sino para la presentación de datos tabulares. La utilización del "border=0" y las imágenes transparentes hicieron posible crear una rejilla que permitió a los diseñadores organizar textos e imágenes, establecer tamaños y ubicar objetos. Pero ésto es sencillamente incorrecto. Las tablas no se crearon para maquetar y no deben utilizarse para eso, porque de esta forma se mezclan presentación y contenido.

La solución es clara: CSS+HTML/XHTML. Afortunadamente, cada vez son más las empresas que deciden dejar atrás las tediosas tablas y evolucionar desarrollando sus sitios respetando los estándares establecidos por la W3C (organización internacional que desde hace unos 12 años se dedica al desarrollo de pautas y estándares web), lo que facilita la accesibilidad y la correcta visualización de las páginas en los navegadores que respeten dichos estándares.

Algunas de las ventajas de la maquetación con CSS:

- Separación de forma y contenido. Generalmente CSS y HTML se encuentran en archivos separados, lo que facilita el trabajo en equipo porque diseñador y programador pueden trabajar independientemente. Por otro lado, permite el acceso a distintos navegadores y dispositivos.
- Tráfico en el servidor. Las páginas pueden reducir su tamaño entre un 40% y un 60%, y los navegadores guardan la hoja de estilos en la caché, ésto reduce los costos de envío de información.
- Tiempos de carga. Por la gran reducción en el peso de las páginas, mejora la experiencia del usuario, que valora de un sitio el menor tiempo en la descarga.
- Precisión. La utilización de CSS permite un control mucho mayor sobre el diseño, especificando exactamente la ubicación y tamaño de los elementos en la página.

También se pueden emplear medidas relativas o variables para que la pantalla o la caja contenedora se acomode a su contenido.
- Mantenimiento. Reduce notablemente el tiempo de mantenimiento cuando es necesario introducir un cambio porque se modifica un solo archivo, el de la presentación, sin tener que tocar las páginas que contienen la estructura con el contenido.
- Diseño unificado y flexibilidad. Es posible cambiar completa o parcialmente el aspecto de un sitio con sólo modificar la hoja de estilos. Por otro lado, el tener el estilo de una web en un solo archivo permite mantener la misma apariencia en todas las páginas.
- Posicionamiento. Las páginas diseñadas con CSS tienen un código más limpio porque no llevan diseño, sólo contenido. Esto es semánticamente más correcto y la página aparecerá mejor posicionada en los buscadores. Google navega obviando el diseño.

Recomiendo un sitio simpático y didáctico sobre el tema:
http://www.effectivetranslations.com/stupidtables/everything_es.html

*Artículo por **Serviweb***

Trucos CSS para no enloquecer

Todavía no tires el monitor contra la pared. Aquí los principales trucos CSS para hacer frente a los típicos problemas que se enfrentan los diseñadores web cuando maquetan con CSS. Podrán existir discrepancias entre los lectores, pero aclaro que estas son técnicas que a mi personalmente me han dado resultado, después de muchas pruebas e intentos aprendí esto...

Usa un contenedor global para todas las cajas (cuando las cosas se disparan)
De esta forma estas prefijando globalmente el orden de todas las demás cajas. En referencia a este contenedor ordena el resto de las cosas interiores. Es como si haces una cerca o valla para que nada es escape. Obviamente estamos hablando de sitios "fijos" no elásticos.
A veces es bueno usar un contenedor hasta el cuerpo del sitio, luego dejar el pié afuera.

Ejemplo para un contenedor de 900px centrado:
```
#contenedor {
margin-top: 0px;
margin-right: auto;
margin-bottom: 0px;
margin-left: auto;
width: 900px;
}
```

Que flote a la izquierda (cuando las cajas se superponen)
Esta es una muy buena forma de evitar incompatibilidades entre navegadores. El uso de hacks de CSS se debía en gran parte porque se trabajaba "centrando" las cajas. Si por ejemplo precisas poner tres cajas de 300px en un contenedor de 900px puedes hacer lo siguiente.

Ejemplo:
```
#caja {
float: left;
width: 300px;
}
```

Calcular bien los paddings o rellenos (cuando las cajas se van abajo)
Casi todos los dolores de cabeza y maldiciones hechadas sobre el CSS se deben al mal uso o a la mala interpretación que se hace del padding. Pero es más simple de lo que parece.
¿Para que sirven los paddings o rellenos? Bueno, lo que hace es generar un "relleno" de determinada medida para dar por ejemplo como un margen a los elementos, pero lo hace sobre el ancho en píxeles que esté prefijado. Por ejemplo: si tenemos una caja de 300px y le aplicamos un padding de 10px en la izquierda, ahora tendremos una caja de 310px. Esto hará desbordar al resto de las cajas y las desplazarán para abajo. Ahí es cuando el diseñador principiante se vuelve loco. El tema es que si hay una diferencia de hasta un 1px se producirán estos desbordes, sino fíjate cuando le incluyes bordes a tu caja, se producirán difrencias.
Lo que se debe hacer es simple, calcular bien y recordar cada ajuste que se haga de los rellenos. Ahora tendremos que hacer una caja de 290px con paddings de 10px a la izquierda.

Ejemplo:
```
#caja {
float: left;
width: 290px;
padding-left: 10px;
background-color: #FFE6DD;
}
```

El pié de página con ancho fijo (cuando el pié de página enloquece):
Para entender mejor como funciona el uso de cajas en CSS se puede pensar en un grupo de objetos de diferentes formas que luchan por adaptarse y ocupar el espacio que se ha prefijado. Sucede muchas veces que los pié de página son los más problemas traen cuando se maqueta un sitio. O se va para arriba, se alinea a la izquierda, o se desborda, etc. Muchos resolvíamos este tema prefijando valores fijos a las alturas de cajas, pero no tiene sentido. Lo que se debe hacer es de nuevo establecer un valor de ancho fijo. De esta forma el pié se va a hacer su lugar del resto e irá a parar donde tiene que ir.

Ejemplo:

```
#pie {
width: 900px;
background-color: #666666;
}
```

No todo es 1+1=2 en CSS (cuando los anchos no cierran)
Un problema común en css es pensar que todos los anchos entre cajas cierran perfectamente. A veces es necesario jugar con los valores de los contenedores, a veces contrario a la lógica hay que añadir algunos px a los contenedores.

OTROS TRUCOS RÁPIDOS:

Trucos sencillos, de los que no hace falta explicar mucho pero que son muy prácticos y te harán más fácil el trabajo y evitarán posibles errores.

- Usa colores diferentes para distinguir las cajas
- Pon una palabra descriptiva en cada caja
- Comenta el código fuente y señala los finales de los contenedores grandes
- No mezquines espacios entre los divs
- No seas un fundamentalista y no quieras escribir tu CSS con dos o tres líneas. Si no quieres

errores escribe lo necesario.
- Cuidado con el tamaño de las imágenes que insertas, estas cambian el ancho de los contenedores.
- Elige bien los nombres de cada div y trata de ser ordenado en el código.
- Si vas a trabajar con varias cajas, trata de agruparlas de a grupo, esto es muy importante. Por ejemplo un contenedor que agrupe tres o cuatro cajas.

CONCLUSIÓN:

Todas estos párrafos son simplemente algunas sugerencias o comentarios de lo que me ha dado resultado a mi. Existen otras muchas "ataduras" de este tipo, si tienes alguna no dudes en comentarlas en este mismo artículo.

Que pasa cuando no puedes resolver un problema con CSS o similar? A mi me ha dado resultado levantarme un rato, hacer cualquier otra cosa y luego volver e intentar de nuevo.

Dejar de renegar y no enloquecer con CSS dependerá de la cantidad de tiempo, trabajo y esfuerzo que le metas a tu trabajo. No lo dudes.

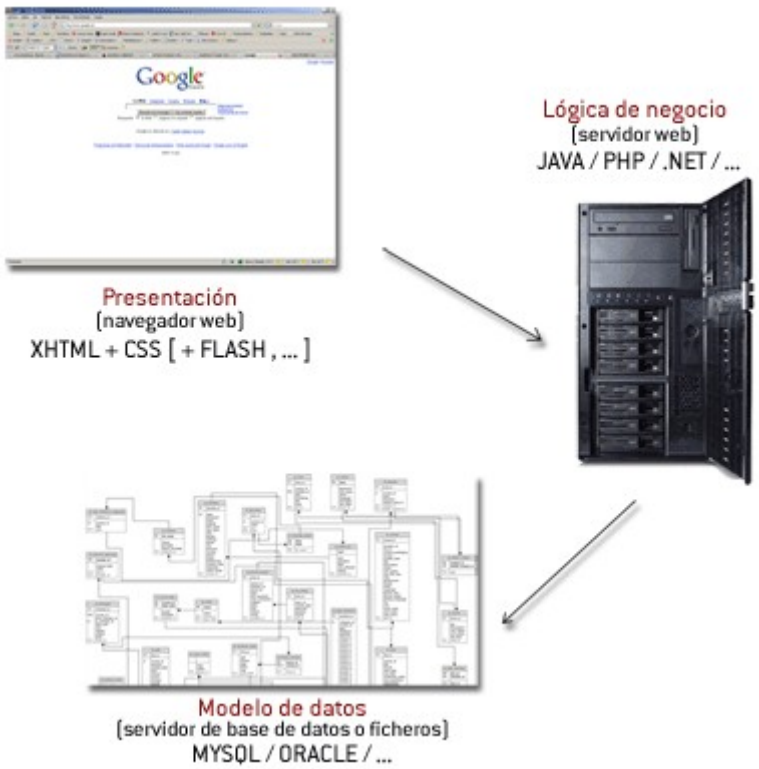
*Artículo por **Leonardo A. Correa***

CSS semánticas. Un nuevo enfoque

Lo que es difícil de encontrar es aquella solución que se adapte perfectamente a tus desarrollos, a tu entorno concreto. Quizá más difícil aun es adecuar algún enfoque similar al tuyo. Esta tarea es proclive a múltiples errores, que irán saliendo conforme se vaya utilizando y que llegado el momento, podemos comprobar que nos hemos equivocado de base, lo que exige una reestructuración desde el inicio.

Este es el contexto en el que nos encontramos actualmente. Tras una fuerte apuesta por la reestructuración y organización de CSS basadas en su semántica de uso se ve que si es quizá uno de los enfoques más acertados, deja bastantes puntos abiertos que es necesario concretar. Esa es la tarea que nos proponemos aquí.

Para los no iniciados, comentar que el enfoque semántico se basa en la idea de que la manera de estructurar la información relativa a la **capa de presentación** de nuestros proyectos web debe de seguir el criterio de qué es y el contexto donde se usa cada elemento.



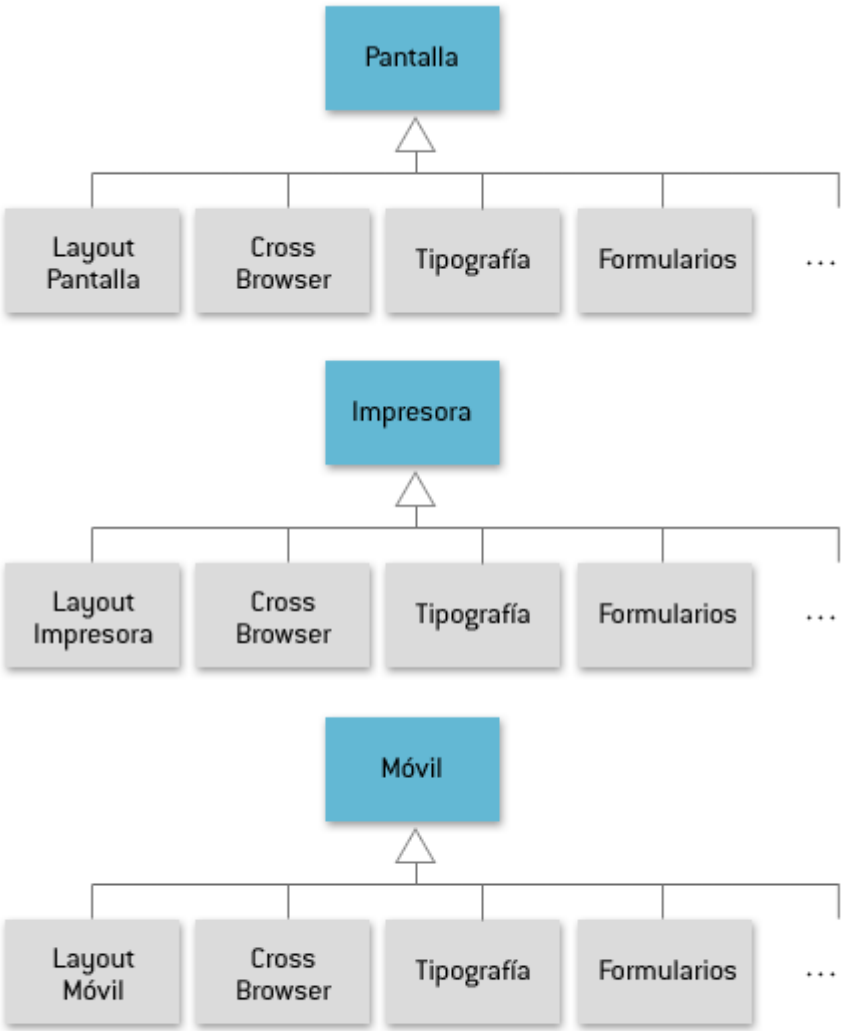
El entorno web tiene una característica fundamental que pocos otros tienen y es la capacidad y potencialidad de uso en múltiples tipos de dispositivos, lo cual nos abre aun más el abanico de puntos que debemos controlar a la hora de crear nuestras hojas de estilos, a la vez que multiplica la casuística y potenciales errores que es necesario controlar.

Aquí trataremos de dar una posible solución que se adecue a los principios de CSS semántica y que siga las pautas de accesibilidad y de facilidad de extensión a múltiples dispositivos.

Puntos a tener en cuenta

Estructura jerárquica de las CSS

Nuestras hojas de estilo seguirán una estructura jerárquica, cuyo elemento principal será el que incluya, para cada tipo de dispositivo, las hojas de estilo correspondientes.



Explicación de cada una de las hojas de estilo

- **Pantalla.css, Impresora.css y Movil.css.** Estas son CSS específicas para cada tipo de dispositivo. Su misión en la parte superior de la jerarquía es la importación de las CSS hijas (@import url("…css")). No incluyen estilos concretos.

- **LayoutPantalla.css, LayoutImpresora.css y LayoutMovil.css.** Incluyen la información relativa a las diversas capas que forman el layout de la página, es decir, información de maquetación de las distintas zonas del portal (Banner, menús, contenidos, créditos, …)
- **CrossBrowser.css.** Aglutina trucos, fixes y demás elementos para hacer que las páginas se vean igual en todos los navegadores (Internet Explorer, Mozilla, Firefox, Opera, etc.)
- **Tipografía.css.** Cualquier elemento relacionado con las forma en que se presenta la tipografía de la página queda recogida aquí.Comienza con una medida relativa de 62.5% en la etiqueta body, que se establece como medida equivalente a 1em. El resto de tamaños vendrán supeditados a este (mayores: 1.2em, 1.5em, etc…, y menores: 0.8em, 0.5em, etc.)
- **Formularios.css.** Definición de los estilos aplicables a todas las etiquetas relacionadas con los formularios genéricos.

Documentos asociados a este artículo

- CSS Semánticas. Incluye los distintos CSS descritos.
- Página ejemplo en XHTML con una maquetación tipo basada en capas

*Artículo por **José Juan Corpas Martos***

CSS semánticas. Un nuevo enfoque (II)

Toda la información está basada en el post citado, así como los ficheros relacionados.

Recomendaciones de uso y escritura de código

Usar las etiquetas HTML estrictamente para lo que se han concebido

Si se consigue crear un código XHTML lo más limpio y claro posible se verá reducido considerablemente el número de estilos propios que tendremos que crear y evitaremos la mayoría de los errores e inconsistencias comunes.

- Maquetar en base a capas (div), no usar tablas (table).
- Párrafos: (p)
- Encabezados: (h1, h2, … h6)
- Tablas: (table, thead/tr/th, tbody/tr/td)
- Listas: (ul/li, ol/li)
- Menús: basados en listas con el atributo display:inline

Uso de medidas relativas en los tamaños de las fuentes

Las medidas relativas son aquellas que no establecen un tamaño fijo en píxeles o puntos para un elemento. En concreto son el porcentaje (%) y el em (1em equivale a 100%, 1.2em a 120%…, 0.7em 70%, …)

Es muy importante este punto por varios aspectos:

- No todas las personas tienen el monitor a la misma resolución y nuestra tipografía le puede resultar muy pequeña o grande según el caso.

- Al establecer la medida de tipografía de manera relativa en la etiqueta body, se tomará el tamaño relativo al navegador y dispositivo que interprete la página. Esto es especialmente en dispositivos con pequeñas pantallas (móviles, pda, etc.).

Maquetación de layout fijo basado en capas

Esta es una antigua polémica, el maquetado de tamaño fijo de ancho o el maquetado líquido o de tamaño variable.

Los estudios de usabilidad web aconsejan el uso de tamaños fijos adecuados a la resolución más usada por los usuarios de internet, intentando que el porcentaje discriminado sea el menor posible. La explicación de este aspecto se justifica por el constante crecimiento en las ventas de monitores de gran tamaño (17 pulgadas o más) y resolución (1024×768 en adelante).

La expansión de las páginas sobre todos en resoluciones de monitores panorámicos dificulta enormemente la legibilidad de los párrafos de texto.

Crear estilos o redefinirlos solo en el caso estrictamente necesario

Si creamos nuestras páginas haciendo un uso adecuado de las etiquetas XHTML y seguimos las pautas descritas en este documento, se reducirá drásticamente la necesidad de creación de estilos innecesarios y redundantes, lo que conlleva enormes beneficios sobre todos relativos a la facilidad de mantenimiento y reducción de errores y comportamientos extraños.

*Artículo por **José Juan Corpas Martos***

10 errores comunes en los css

1.Uso innecesario del valor 0

El código siguiente no necesita la unidad especificada si el valor es cero.

```
padding:0px 0px 5px 0px;
```

En su lugar puede ser escrito de esta manera:

```
padding:0 0 5px 0;
```

De la misma manera es igual para otros estilos. Ej.:

```
margin:0;
```

No malgastes espacios agregando unidades tales como px, pt, em, etc, cuando el valor es cero. La única razón de hacer esto es si necesitas cambiar estos valores más tarde. Si no declarar estas unidades no tiene sentido. Los pixeles cero son iguales que los puntos cero.

Sin embargo,line-height puede no tener unidad.Por eso es válido lo siguiente:

```
line-height:1;
```

De cualquier manera puedes utilizar una unidad en concreto como em si lo deseas.

2.Los colores en formato hexadecimal necesitan una almohadilla

Esto está mal:

```
color: ea6bc2;
```

Debe ser:

```
color: #ea6bc2;
```

O esto otro:

```
color: rgb (234.107.194);
```

3.Valores duplicados en los códigos de colores

No escribir el código de esta manera:

```
color: #ffffff;
background-color:#000000;
border:1px solid #ee66aa;
```

Los valores duplicados pueden ser omitidos.Escribiendo los códigos de esta manera:

```
color:#fff;
background-color:#000;
border:1px solid #e6a;
```

¡Por supuesto esto no debes hacerlo con códigos como este!

```
color: #fe69b2;
```

4.Evitar repeticiones de código innecesaria

Evita usar varias líneas cuando lo puedes conseguir con una sola. Por ejemplo, al fijar los bordes, algunas veces se debe hacer por separado pero en casos como el siguiente no es necesario:

```
border-top:1px solid #00f;
border-right:1px solid #00f;
border-bottom:1px solid #00f;
border-left:1px solid #00f;
```

Podríamos resumirlo en una única línea esta:

```
border:1px solid #00f;
```

5.La duplicación es necesario con los estilos en cascada

En los estilos en cascada es aceptable repetir el mismo codigo para un elemento elemento dos veces, si significa evitar la repetición mencionada en el punto arriba. Por ejemplo, digámos que tenermos un elemento donde solamente es diferente el "border" izquierda. En vez de poner

cada "border" escrito usando cuatro líneas, uso sólo dos:

```
border:1px solid #00f;
border-left:1px solid #f00;
```

En este caso primero definimos todos los "borders" con el mismo color pero más tarde para ahorrarnos dos lineas de código redefinimos el "border" izquierda a otro color, de esta manera hemos ahorrado dos líneas de código.
El ejemplo malgastando espacio quedaría así:

```
border-top:1px solid #00f;
border-right:1px solid #00f;
border-bottom:1px solid #00f;
border-left:1px solid #f00;
```

Obviamente supuestamente este ahorro de carga supone un retraso en la carga de la página pues estamos definiendo el "border" izquierda dos veces, pero la carga de este proceso es insignificante.

6.Los estilos inválidos no hacen nada

Un ejemplo es suficiente para explicar este error:

```
padding:auto
```

Este estilo solo puede ser aplicado a width y height pero no a padding.

7.Código Específico para cada navegador

Obviamente este tipo de código solo funcionará en el navegador al que va destinado , pero es hay que pensar si es rentable puesto que solo algunos usuarios podrán apreciar esos cambio.

8.Espacio perdido

No estoy seguro del porqué pero muchos diseñadores estan empeñados en desaprovechar el espacio en su código, usando un montón de innecesarios saltos de línea. Recuerda que eso sólo lo verás tu y estas haciendo un uso excesivo de ancho de banda. Tambien tu código será más facil de leer puesto que tendrá menos "boquetes".

Por supuesto es sabio dejar un cierto espacio para mantenerlo legible, aunque a algunos les encanta condensar todo, no dejando ningún espacio.

9.Especificar los colores sin usar palabras

Definir los colores usando las palabras que lo definen no es una buena idea puesto que estaríamos confiando en el navegador para que el interprete que color y código debe aplicar.Las tonalidades para un mismo nombre de color cambian mucho de un navegador a otro.

Es una buena práctica especificar siempre el color por su código hexadecimal.
Ej.: utilizar "#fff" en lugar de blanco.

10.Agrupar estilos idénticos

Es común ver los estilo escritos una y otra vez con el mismo código, aún cuando el estilo es igual.

Sería conveniente agruparlos y asi optimizaríamos espacio:

```
h1, p, #footer, .intro {
font-family:Arial,Helvetica,sans-serif;
}
```

Tambien nos hara mucho más facil la tarea de actualizar el código.

*Artículo por **Manu Gutierrez***

Introducción a CSS3

Desde que CSS comenzó han pasado muchos años y ya vamos por la especificación de CSS3, que incorpora una serie de novedades que vamos a tratar de resumir en este artículo.

Qué es CSS

Si no sabes lo que es CSS probablemente te interesaría comenzar leyendo nuestro manual de CSS o la sección de CSS a fondo. No obstante, cabría decir que CSS es un lenguaje para definir el estilo o la apariencia de las páginas web, escritas con HTML o de los documentos XML. CSS se creó para separar el contenido de la forma, a la vez que permite a los diseñadores mantener un control mucho más preciso sobre la apariencia de las páginas.

Con CSS3, más control sobre la forma

El objetivo inicial de CSS, separar el contenido de la forma, se cumplió ya con las primeras especificaciones del lenguaje. Sin embargo, el objetivo de ofrecer un control total a los diseñadores sobre los elementos de la página ha sido más difícil de cubrir. Las especificaciones anteriores del lenguaje tenían muchas utilidades para aplicar estilos a las webs, pero los desarrolladores aun continúan usando trucos diversos para conseguir efectos tan comunes o tan deseados como los bordes redondeados o el sombreado de elementos en la página.

CSS 1 ya significó un avance considerable a la hora de diseñar páginas web, aportando mucho mayor control de los elementos de la página. Pero como todavía quedaron muchas otras cosas que los diseñadores deseaban hacer, pero que CSS no permitía especificar, éstos debían hacer uso de trucos para el diseño. Lo peor de esos trucos es que muchas veces implica alterar el contenido de la página para incorporar nuevas etiquetas HTML que permitan aplicar estilos de una manera más elaborada. Dada la necesidad de cambiar el contenido, para alterar al diseño y hacer cosas que CSS no permitía, se estaba dando al traste con alguno de los objetivos para los que CSS fue creado, que era el separar por completo el contenido de la forma.

CSS 2 incorporó algunas novedades interesantes, que hoy ya utilizamos habitualmente, pero CSS 3 todavía avanza un poco más en la dirección, de aportar más control sobre los elementos de la página.

Así pues, la novedad más importante que aporta CSS 3, de cara a los desarrolladores de webs, consiste en la incorporación de nuevos mecanismos para mantener un mayor control sobre el estilo con el que se muestran los elementos de las páginas, sin tener que recurrir a trucos o hacks, que a menudo complicaban el código de las web.

Propiedades nuevas en CSS3

He aquí una lista de las principales propiedades que son novedad en CSS3.

Bordes

- border-color
- border-image
- border-radius
- box-shadow

Fondos

- background-origin
- background-clip
- background-size
- hacer capas con múltiples imágenes de fondo

Color

- colores HSL
- colores HSLA
- colores RGBA
- Opacidad

Texto

- text-shadow
- text-overflow
- Rotura de palabras largas

Interfaz

- box-sizing
- resize
- outline
- nav-top, nav-right, nav-bottom, nav-left

Selectores

- Selectores por atributos

Modelo de caja básico

- overflow-x, overflow-y

Otros

- media queries
- creación de múltiples columnas de texto
- propiedades orientadas a discurso o lectura automática de páginas web
- Web Fonts

Este listado de nuevas propiedades de CSS3 lo he sacado de: http://www.css3.info/preview/. Es un sitio en inglés, pero puede estar bien visitar para ir conociendo más detalles sobre CSS3.

En futuros artículos daremos algunas claves y explicaciones sobre varias de estas propiedades, al menos las más interesantes, con especificaciones detalladas, así como ejemplos que sirvan para ir conociendo esta nueva especificación de CSS.

*Artículo por **Miguel Angel Alvarez***

Bordes redondeados en CSS 3

CSS 3 incorpora nuevas propiedades para el control de bordes de los elementos. Ahora se permiten bordes con las esquinas redondeadas, bordes con imágenes (incluso varias imágenes se pueden utilizar para definir el aspecto del borde, sombras, etc.

En este artículo vamos a explicar cómo realizar bordes redondeados con CSS3.

Tenemos la propiedad border-radius, que permite definir bordes redondeados en las esquinas, especificando las medidas del radio que deben darse a la curva de las esquinas

```
border-radius: 5px;
```

Definiría un radio de 5 píxeles en el redondeo de las esquinas del elemento. Por el momento Mozilla ha adoptado este atributo con un nombre especial, que es válido para productos como Firefox, mientras que las especificaciones de CSS3 no hayan alcanzado el estado "Candidate Recommendation", que es cuando se supone que los distintos navegadores deben implementarlas. El nombre del atributo por el momento es:

```
-moz-border-radius
```

Por lo que respecta a Internet Explorer hay que decir que todavía no soporta este atributo de CSS 3, pero esperemos que los chicos de Microsoft lo puedan implementar pronto en el navegador, o que las especificaciones de CSS3 pasen rápido al estado "Candidate Recommendation", que sería el llamamiento para que se implementen en la generalidad de los

navegadores.

Por el momento, para navegadores Mozilla, el atributo border-radius se utilizaría, por ejemplo, así:

```
DIV {
border: 1px solid #000000;
-moz-border-radius: 7x;
padding: 10px;
}
```

Con esto conseguimos que todos los div tengan un borde redondeado en las esquinas de radio de 7píxeles.

Podemos ver el ejemplo en una página aparte. Pero recordar que sólo se verá el efecto utilizando Firefox o navegadores basados en Mozilla.

Pero el atributo border-radius tiene otras posibles configuraciones, en la que se pueden definir los valores para el radio de las cuatro esquinas por separado. De esta manera:

```
-moz-border-radius: 7px 27px 100px 0px;
```

Así estaríamos definiendo un borde redondeado con radio de 7 pixel para la esquina superior izquierda, luego 27px para la esquina superior derecha, de 100px para la inferior derecha y 0px para la inferior izquierda. (Hay que explicar que un border-radius de 0px es un borde con esquina en ángulo recto)

Podemos ver este ejemplo en una página aparte.

Los bordes redondeados con CSS 3, como se podrá imaginar, sólo se ven si se tiene definido algún borde visible al elemento que se los asignamos, ya sea solid, dotted, etc. Eso es lo que definen las especificaciones de CSS3, aunque en el momento de escribir el artículo debo señalar que incluso Mozilla o Firefox (el único que por ahora soporta este atributo de CSS3) no funciona del todo correctamente con esto y sólo muestra los bordes redondeados con solid.

Otra cosa que definen las especificaciones de CSS y que por el momento no está funcionando del todo bien, es que las imágenes de fondo deben ajustarse a los bordes redondeados. Ocurre, por el momento, que las imágenes de fondo salen por fuera de los bordes redondeados. Confiamos en que en el futuro esto pueda ser revisado y optimizado, para que todo funcione correctamente. Debemos recordar que en el momento de escribir el artículo todavía se tienen que terminar de definir las especificaciones de CSS 3 y después, los navegadores web deberán actualizarse en un cierto espacio de tiempo para soportarlas completamente.

Nota: Ofrecimos una lista de las características principales de CSS 3 en el artículo Introducción a CSS3.

*Artículo por **Miguel Angel Alvarez***