Guía básica de hojas de estilo (CSS) y

de marcado semántico de páginas web (v1.)

**Geovanny Quilarque**

UNEWEB. Mayo 2012

**1.Hojas de estilo: una base para el marcado semántico**

Una hoja de estilo es un conjunto de especificaciones que declara cómo

deben mostrarse los componentes de una página web, es decir, cómo debe

interpretar el navegador, a efectos de presentación visual, los distintos

elementos presentes en una página web. El formato de cada elemento de

una página web se especifica mediante declaraciones de estilo

denominadas reglas.

**2.Reglas**

Una regla CSS se compone de tres partes:

1.Selector

2.Propiedad

3.Valor

Una regla simple organiza los tres componentes anteriores de acuerdo a la

siguiente estructura:

selector {propiedad: valor;}

Por ejemplo, la regla siguiente evita que un elemento marcado con h1

aparezca en negrita.:

h1{ font-weight: normal; }

Una regla puede contener más de una propiedad con sus respectivos

valores, separadas entre ellas por punto y coma (se considera una buena

práctica añadir también punto y coma al final). Ejemplo:

h1{ font-weight: normal; **color: red**;}

Una regla puede dar formato a más de un elemento a la vez, separando

estos mediante comas. Ejemplo:

**h1, h2, h3, h4**{ font-weight: normal; color: red; }

**3.Declaración**

Las reglas de las hojas de estilo se pueden declarar en tres sitios distintos:

1.En la página web

2.En un archivo externo

3.En un elemento

En los casos 1 y 2, las reglas de declaran con la misma sintaxis (que es la

que ya hemos visto). En el caso número 3, hay una sintaxis específica ya

que se declara dentro de un elemento (X)HTML y debe respetar sus

convenciones. Además, cada forma de declaración necesita el procedimiento

de vinculación específico que se muestra a continuación.

**3.1.En la página web**

La hoja de estilo se declara en la secciónhead, dentro del elementostyle.

Por ejemplo:

<head>

<title>Aquí va un título</title>

**<style type="text/css">**

h1, h2, h3 { font-weight: normal; color: blue; }

**</style>**

</head>

**3.2.En un archivo externo**

Las declaraciones se escriben en un archivo de texto con extensión**.css**sin

ningún tipo de preámbulo.

En la secciónhead de la página se debe indicar el enlace con el archivo que

contiene las declaraciones de la hoja de estilo mediante el elementolinky

los atributosrel, hrefytype:

<head>

<title>Aquí va un título</title>

**<link rel="stylesheet" href="estilo.css" type="text/css" />**

</head>

Como se puede ver, el atributorel(*relation*) advierte que se trata de una

hoja de estilo, hrefaporta el nombre (y el camino si fuera necesario) del

archivo ytypeindica que se trata de un archivo de texto que contiene una

hoja de estilo que sigue la norma CSS.

**3.3.En un elemento**

Los estilos se pueden aplicar a nivel de un elemento individual. En el

siguiente ejemplo, se aplica un estilo específico a un elementoacronym. En

condiciones normales,un texto marcado con acronymaparecería en letra

normal (redonda). En el siguiente ejemplo, se ha definido un estilo a nivel

de una ocurrencia este elemento para que la palabra marcada conacronym

en este caso aparezca en cursiva:

<p><acronym **style="font-style: italic"**>W3C</acronym>son las siglas

del World Wide Web Consortium</p>

Como se puede ver hay un cambio significativo en la sintaxis. Se utilizan las

mismas denominaciones para las propiedades y sus valores, pero, por

compatibilidad con el lenguaje XHTML, todo el conjunto propiedad/valor de

CSS se delimita mediante comillas y se presenta a su vez como un valor de

del atributo style de (X)HTML.

**4.Cajas**

Desde el punto de vista de CSS, todo lo que hay en una página web es una

caja. Lo que hace un selector es identificar la caja a la que se aplica el

formato.

Las cajas tienen:

**Contenido** (*content)*. El contenido de la caja: texto, imagen, etc.

**Relleno** (*padding*). La distancia entre el contenido y el borde.

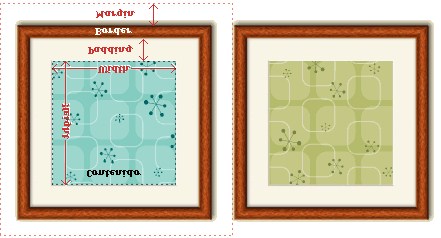
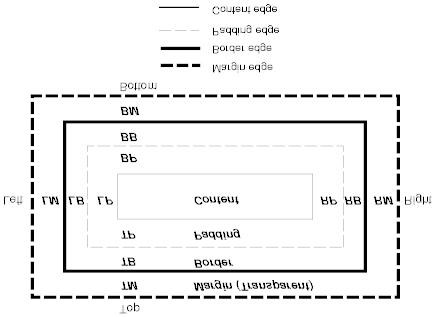
**Borde**(*border*). El borde de la caja.

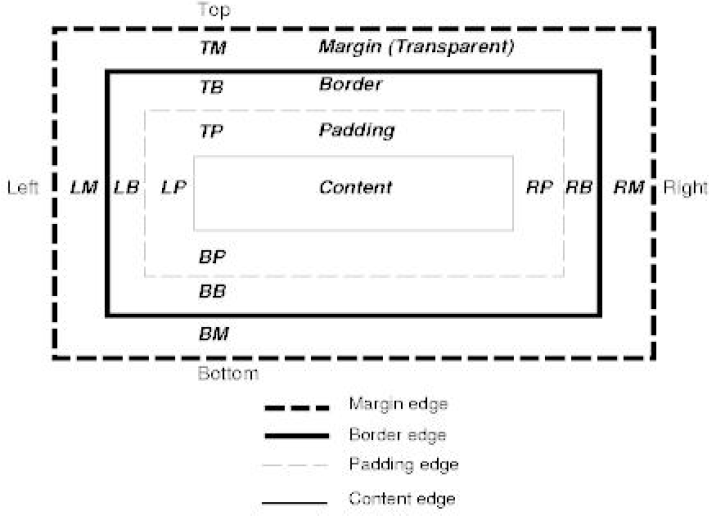
**Margen**(*margin*). La distancia entre el borde y el elemento

contenedor (p.e.body) u otro elemento adyacente.

El siguiente diagrama muestra modelo oficial de cajas de CSS según el

W3C:





Los componentes de la caja se pueden dividir en cuatro partes: *top*,*bottom*,

*right* y *left*. Por tanto, en la ilustración anterior, las siglas LM, RM, etc.,

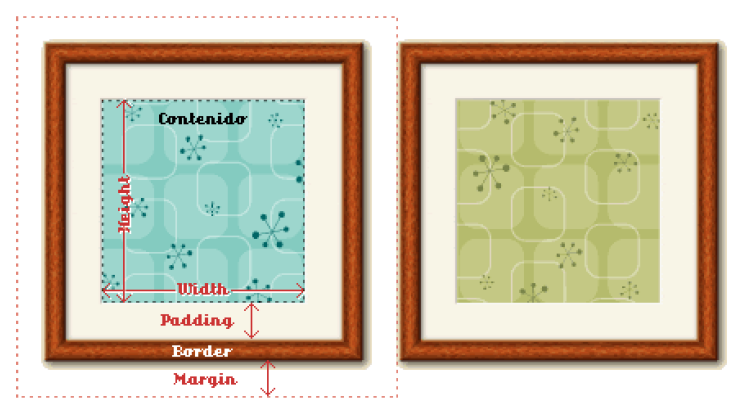
significan *LeftMargin*,*Right Margin*, etc.

La figura siguiente, debida a Kemie Guaida, ilustra estas ideas

comparando el modelo de cajas de CSS con un cuadro colgado en la pared y

de acuerdo con el cual *margin*sería la relación entre el cuadro y la pared,

*border*el grosor del marco, etc.:



Las hojas de estilo pueden aplicar formato (p.e. color, anchura, tipo de

línea, etc.) a cualquiera de los componentes de esta caja, así como permite

posicionar estas cajas en el conjunto de la página.

De este modo, si queremos que un elemento muestre el borde en forma de

puntos de color azul, podemos generar esta declaración:

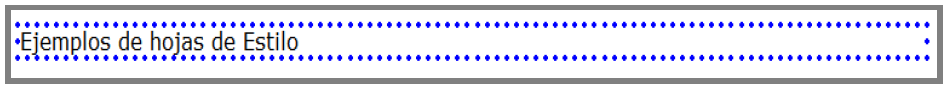
p { border-style: dotted; border-color: blue}



A partir de lo anterior, este código en la página:

<p>Ejemplos de hojas de Estilo</p>

Se mostrará así:



Si queremos ampliar el relleno entre el contenido y el borde así como tener

un tamaño de punto más pequeño podemos modificar la regla anterior

acudiendo a las propiedades de borde (border) y de margen (padding):

p{ border-style: dotted; border-width: 2px; border-color: blue;

padding: 15px; }

Ahora, el navegador lo mostrará así:



**5.Herencia**

Las cajas están contenidas dentro de otras cajas y heredan los estilos de la

caja contenedora si no hay una regla de nivel específico. Por ejemplo, si

declaramos un tipo de letra para el elementobody, todos los elementos que

están dentro, comoh1,h2, p, etc. heredarán este tipo de letra.

En el siguiente ejemplo declaramos un color de letra para bodyy otro

distinto para h2. De este modo, todos los elementos, exceptoh2, tendrán el

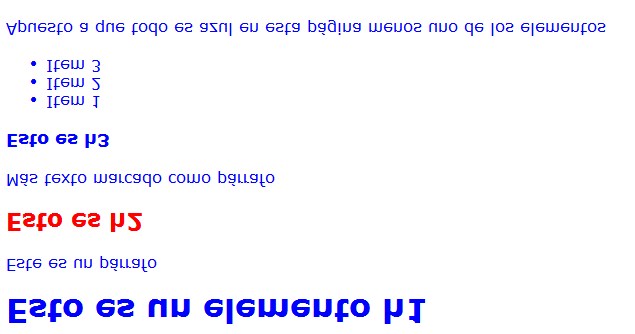
color declarado para body, mientras queh2tendrá su propio color.

Veamos. Primero la declaración de reglas CSS:

body { color: blue }

h2 { color: red }

Ahora el código html de la página:



<h1>Esto es un elemento h1</h1>

<p>Este es un párrafo</p>

<h2>Esto es h2</h2>

<p>Más texto marcado como párrafo</p>

<h3>Esto es h3</h3>

<ul>

<li>Item 1</li>

<li>Item 2</li>

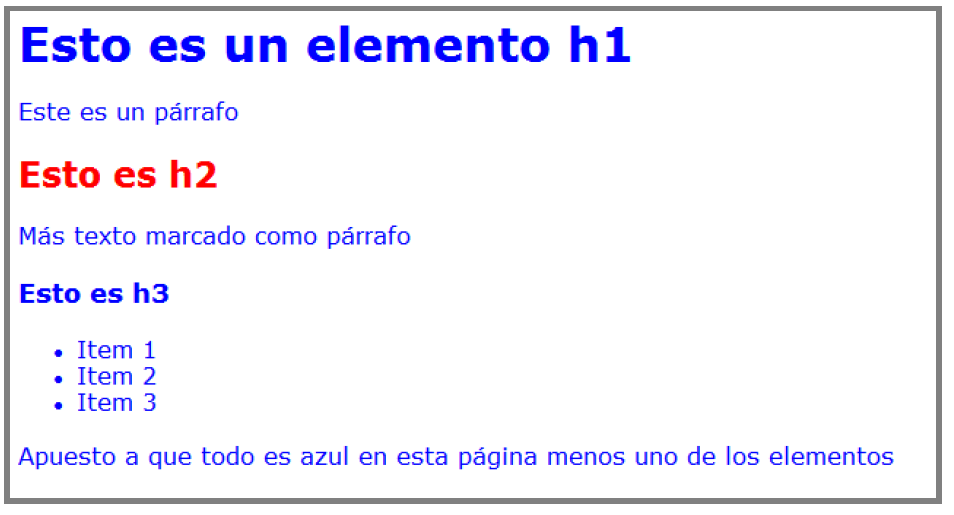
<li>Item 3</li>

</ul>

<p>Apuesto a que todo es azul en esta página menos uno de los

elementos</p>

Ahora, el resultado en el navegador:



**6.Tipos de selectores**

Existen dos grandes categorías de selectores:

*Elementos:* son los nombres correspondientes a elementos del

lenguaje (X)HTML como body,h1, p, table, etc.

*Nombres propios:* son nombres que puede crear el autor de la

hoja de estilo. De esta clase de selectores, existen, a su vez

dos tipo: de**clase**(*class*) y de **identidad**(*id*).

Por tanto, en realidad tenemos en total**tres**tipos de selectores que

examinaremos a continuación.

***6.1. Selectores de elemento***

Se declaran utilizando como selector un elemento (X)HTML, p.e.:

h1 { color: blue }

De este modo, el aspecto de todos y cada uno de los elementos que forman

el lenguaje (X)HTML puede ser modificado mediante un regla. Cabe

recordar que los navegadores disponen de una hoja de estilo interna que es

la que otorga formato a los elementos (X)HTML cuando no hay ninguna hoja

de estilo específica. Por ejemplo, aún sin hoja de estilo la mayor parte de

los navegadores aplican negrita a los elementos marcados comostrong, así

como un tipo de letra superior + negrita a los elementos marcados con h1,

etc.

***6.2. Selectores de clase***

Estos selectores se declaran mediante una palabra propia que asigna el

autor de la hoja de estilo. Esta palabra va precedida por un punto. Por

ejemplo: supongamos que se necesita una clase de selectores para

formatear títulos de películas (pongamos que se trata de una web sobre

cinema). El autor de la hoja de estilo puede crear el selector con el nombre

tituloFilm para formatear de la misma forma los títulos de los films. Para

ello, escribe esta regla:

.tituloFilm{ font-size: 1.5em; font-family: Times New Roman; color:

blue;}

Se aplica mediante el atributo classseguido por el nombre del selector (sin

el punto). Por ejemplo, en el siguiente código fuente solo a uno de los dos

elementos de párrafo (p) se le ha añadido esta clase de selector:

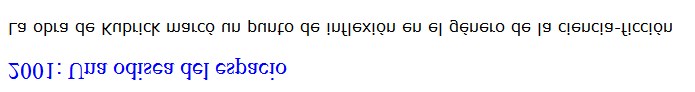
<p **class=”tituloFilm”**>2001: Una odisea del espacio</p>

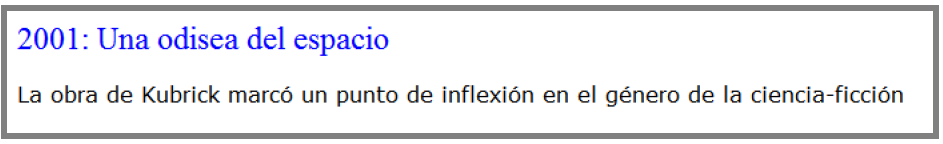
<p>La obra de Kubrick marcó un punto de inflexión en el género de la

ciencia-ficción</p>

La declaración solamente afectará a uno de ellos y el resultado es que el

primer párrafo tendrá un aspecto muy distinto de los demás:





***6.3. Selectores de identidad***

Los selectores de identidad se nombran mediante una palabra propia

precedida por el símbolo**#**(almohadilla). Por ejemplo:

#navegacion { background-color: LightGrey; }

Los selectores de identidad solo se pueden aplicar a un elemento en cada

página. Dicho al revés, y tomando el ejemplo anterior: en cada página

solamente puede haber un elemento#navegacion (en cambio, podemos

tener múltiples elementos de clase en una misma página).

En este caso, hemos supuesto que queremos dar un estilo propio a la zona

de navegación de cada página. Se supone que deseamos destacar la barra

de navegación mediante un fondo gris y queremos que la fuente del texto

en esa zona sea un poco más pequeña. La declaración sería la siguiente:

#navegacion { background-color: LightGrey; font-size: 0.8em }

El código fuente en la página sería el siguiente:

<div id="navegacion">

<ul>

<li><a href="item1.htm">Item 1</a></li>

<li><a href="item2.htm">Item 2</a></li>

<li><a href="item3.htm">Item 3</a></li>

</ul>

</div>

Se supone que Item 1, Item 2, etc., son opciones de menú. El resultado

sería este:



**7.Uso de div y span**

El lenguaje (X)HTML dispone de dos elementos con gran potencialidad

cuando se utilizan junto con las hojas de estilo, y cuya característica

principal, a diferencia de elementos como body,p, h1, etc., es que no

poseen ningún significado intrínseco. Son los siguientes:

div

span

Mientras el elementodiv es de bloque, el elemento spanes de línea.

Recordemos que los elementos de bloque tienen un salto de línea integrado.

El ejemplo más conocido esp. Un elemento de línea, por el contrario no

genera ningún espacio a su alrededor, sino que se mantienen en la misma

línea que lo contiene. Un buen ejemplo puede ser strong.

Tantodiv como span pueden utilizarse para mejorar la apariencia de una

página, pero además contribuyen a añadir valor semántico a la misma.

**7.1. div**

En concreto, div ayuda a crear la estructurar básica de la página. Por

ejemplo, podemos imaginar un sitio web cuyas páginas se estructuran en

tres grandes secciones, como muestra la figura siguiente:

Título

Navegación

Contenido

Pie

Con una estructura como la anterior, el código fuente de cada página puede

contener cuatro elementosdiv, cada uno de ellos identificado con un

atributoiddiferente, de este modo:

<body>

<div id="tituloPag">

</div>

<div id="navegacion">

</div>

<div id="contenido">

</div>

<div id="piePag">

</div>

</body>

La estructuración anterior permitirá definir estilos para las cuatro secciones

de la página sin perjuicio de aplicar estilos más específicos mediante

selectores de elemento, de clase, etc. Esta estructuración facilitará el

mantenimiento de los estilos del sitio y, en caso necesario, el

posicionamiento de los mismos en la página.

**7.2. span**

El elementospansuele utilizarse también combinado conido conclasspara

especificar con tanto nivel de detalle el aspecto de cualquier elemento de la

página. Por ejemplo, podemos desear marcar y tratar de forma distinta el

apellido de una lista de nombres, para lo cual podemos usarspande este

modo. Definimos un selector de clase apellido para el cual declaramos un

estilo:

.apellido { font-style: italic; font-weight: bold }

Aplicamos entonces spany classa los componentes (li) de la siguiente lista:

<h1>Realizadores de los **años 90</h1>**

**<ul>**

<li>Martin<span class="apellido">Escorsese</span></li>

<li>James<span class="apellido">Cameron</span></li>

<li>Pedro<span class="apellido">Almodóvar</span></li>

</ul>

El resultado en el navegador será el siguiente



**8.Posicionamiento**

Las hojas de estilo proporcionan propiedades que permiten posicionar de

forma fija, absoluta o relativa (*fixed, absolute, relative*) las cajas en una

página web. Estas propiedades, junto con los selectores de clase y de

identidad que permiten identificar y clasificar las cajas respectivamente,

proporcionan al diseñador un dominio total de la estructura de la página sin

necesidad de recurrir, por ejemplo, al uso de tablas.

A título ilustrativo, mostramos un ejemplo típico de código de estilo con

indicaciones de posicionamiento:

#navegacion { position: absolute; top: 0; left: 0; width: 30%; }

#contenido { width: 70%; margin-left: 30%; }

El código anterior generaría dos secciones como las que muestra la figura

siguiente:

Navegación

Contenido

Lo cierto es que el posicionamiento mediante CSS merece un capítulo

aparte y por ello escapa a los objetivos de esta presentación. Pero no

hemos querido dejar de señalar su importancia con el fin de animar a todo

aquel que, precisamente, quiera avanzar en estos temas.

**10. 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.

orque lo más habitual es que especifiquemos un color con su valor RGB, de una manera similar a como aprendimos a [definircolores en HTML](http://www.desarrolloweb.com/articulos/colores-html.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;**

**11. Propiedades de la notación CSS abreviada**

La especificación CSS permite crear estilos mediante una sintaxis abreviada conocida como *notación CSS abreviada*.

La notación CSS abreviada permite especificar los valores de diversas propiedades con una sola declaración. Por ejemplo, la propiedad font permite definir las propiedades font-style, font-variant, font-weight, font-size, line-height y font-family en una única línea.

Un aspecto esencial que debe tenerse en cuenta cuando se utiliza CSS en forma abreviada es que a los valores omitidos en una propiedad CSS en forma abreviada se les asignan sus valores predeterminados. Esto puede hacer que las páginas se muestren de forma incorrecta cuando hay dos o más reglas CSS asignadas a la misma etiqueta.

Por ejemplo, la regla h1 que se muestra a continuación utiliza la sintaxis CSS sin abreviar. Observe que a las propiedades font-variant, font-stretch, font-size-adjust y font-style se les han asignado sus valores predeterminados.

h1 {

font-weight: bold;

font-size: 16pt;

line-height: 18pt;

font-family: Arial;

font-variant: normal;

font-style: normal;

font-stretch: normal;

font-size-adjust: none

# }

# La misma etiqueta podría tener el aspecto siguiente si se especificara como única propiedad en forma abreviada:

# h1 { font: bold 16pt/18pt Arial }

Con la notación abreviada, se asignan los parámetros predeterminados a los valores omitidos. Así, en el ejemplo de notación abreviada anterior se omiten las etiquetas font-variant, font-style, font-stretch y font-sizeadjust.

Si tiene estilos definidos en más de una ubicación (por ejemplo, incluidos en una página HTML e importados de una hoja de estilos externa) y se ha utilizado tanto el formato abreviado como el formato no abreviado en la sintaxis CSS, tenga en cuenta que las propiedades omitidas en una regla de notación abreviada pueden tener prioridad (cascada) sobre las propiedades que están definidas explícitamente en otra regla.

Por este motivo, Dreamweaver utiliza la notación CSS no abreviada de forma predeterminada. Con ello se evitan posibles problemas causados por una regla de notación abreviada que tenga prioridad sobre una regla con formato no abreviado. Si abre una página Web codificada con notación CSS abreviada en Dreamweaver, tenga en cuenta que Dreamweaver creará las nuevas reglas CSS en formato no abreviado. Puede especificar el modo en que Dreamweaver crea y edita las reglas CSS mediante la modificación de las preferencias de edición de CSS en la categoría Estilos CSS del cuadro de diálogo Preferencias (Edición > Preferencias en Windows; Dreamweaver > Preferencias en Macintosh).

***Nota:*** *El panel Estilos CSS crea reglas empleando la notación no abreviada. Si crea una página o una hoja de estilos CSS mediante el panel Estilos CSS, tenga en cuenta que la codificación manual de reglas CSS mediante notación abreviada puede provocar que las propiedades en formato abreviado tengan prioridad sobre las creadas con la notación no abreviada. Por este motivo, utilice la notación CSS no abreviada para crear sus estilos.*

**12. Inspección de CSS en la Vista en vivo**

El modo de inspección funciona conjuntamente con la Vista en vivo para ayudarle a identificar rápidamente los elementos HTML y sus estilos CSS asociados. Con el modo de inspección activado, puede pasar el puntero del ratón por encima de elementos de la página para ver los atributos de modelo de cuadro CSS correspondientes a cualquier elemento de nivel de bloque.

Además de ver una representación visual del modelo de cuadro en el modo de inspección, también puede usar el panel Estilos CSS mientras pasa el puntero del ratón por encima de elementos de la ventana de documento. Cuando tenga abierto el panel Estilos CSS en el modo Actual y pase el puntero del ratón por encima de un elemento de la página, las reglas y propiedades del panel Estilos CSS se actualizarán automáticamente para mostrar las reglas y propiedades de dicho elemento. Asimismo, también se actualizará cualquier vista o panel relacionado con el elemento por el que está pasando el puntero del ratón (por ejemplo, la vista de Código, el selector de etiquetas, el inspector de etiquetas, etc.).

**1** Con el documento abierto en la ventana de documento, haga clic en el botón Inspeccionar (situado junto al botón Vista en vivo de la barra de herramientas Documento).

***Nota:*** *Si todavía no se encuentra en la Vista en vivo, el modo de inspección la activará automáticamente.*

**2** Pase el puntero del ratón por encima de elementos de la página para ver el modelo de cuadro CSS. El modo de inspección resalta diferentes colores para el borde, el margen, el relleno y el contenido.

**3** (Opcional) Presione la flecha izquierda del teclado de su equipo para resaltar el elemento padre del elemento actualmente resaltado. Presione la flecha derecha para regresar al resaltado del elemento hijo.

**4** (Opcional) Haga clic en un elemento para bloquear una selección resaltada.

# *Nota: Al hacer clic en un elemento para bloquear una selección resaltada, se desactiva el modo de inspección.*

# 13. Flujo HTML y atributos CSS

# Merece la pena detenerse a explicar lo que es el flujo HTML, pues es un concepto sencillo y básico para poder entender muchos asuntos acerca del posicionamiento web y en concreto el posicionamiento con CSS.

# El flujo de la página es algo así como el flujo de escritura de elementos dentro del lienzo que nos presenta el navegador.

Sabemos que las páginas web son codificadas en HTML y los elementos aparecen en el código en una posición dada. El navegador, en el momento que interpreta el código HTML de la página, va colocando en la página los elementos (definidos por medio de etiquetas HTML) según los va encontrando en el mismo código.

Por ejemplo, pensemos en una página que tiene un titular con H1, luego varios párrafos y alguna imagen. Pues si lo primero que aparece en el código HTML es el encabezamiento H1, pues ese encabezado aparecerá en la página también en primer lugar. Luego se colocarán los párrafos y si la imagen aparecía en el código por último, en la página también aparecerá al final. Es decir, los elementos aparecen colocados tal como estén ordenados en el código. A esto se le llama el flujo HTML, la colocación de los elementos en el lugar que corresponda según su aparición en el código.

Esto en general ocurre con cualquiera de los elementos de la página. Sin embargo, hay algunos atributos HTML que pueden marcar distintas propiedades en el flujo, como que una imagen se alinee a la derecha, con align="right", con el texto del párrafo que pueda haber a continuación rodeando la imagen. Pero con HTML, si por ejemplo, una imagen va antes que un párrafo, nunca vamos a poder intercambiar sus posiciones y colocar la imagen detrás del párrafo que le sigue en el código.

Esto no ocurre de igual manera cuando trabajamos con CSS, puesto que existen diversos atributos que pueden cambiar radicalmente la forma en la que se muestran en la página, por ejemplo el atributo position que puede definir valores como absolute, que rompe el flujo de la página, o mejor dicho, saca del flujo de la página al elemento que se le asigna.

## Comportamientos inline y block y cómo afectan al flujo de la página

Cuando tratamos con etiquetas, existen dos modos principales de de comportamiento. Etiquetas como una imagen, o una negrita, que funcionan en línea ("inline"), es decir, que se colocan en la línea donde se está escribiendo y donde los elementos siguientes, siempre que también sean "inline" se posicionan todo seguido en la misma línea. Tenemos por otra parte los elementos que funcionan como bloque ("block") que implican saltos de línea antes y después del elemento. Por ejemplo, los párrafos o encabezamientos son elementos con comportamiento predeterminado tipo "block".

Dos etiquetas muy utilizadas en la maquetación CSS son DIV y SPAN. Una de las diferencias principales es que DIV funciona con coportamiento "block" y SPAN funciona como "inline". En realidad este es el comportamiento por defecto, puesto que nosotros con CSS en cualquier momento podemos cambiarlo por medio del atributo display. Por ejemplo:

**<div style="display: inline;">  
Este elemento funcionará en línea  
</div>**

O bien:

**<span style="display: block;">  
Este span ahora funciona como bloque  
</span>**

Realmente ambas posibilidades funcionan dentro del flujo HTML normal, así que, tanto los elementos display inline como display block, se encuentran dentro del flujo HTML estándar, la única diferencia es que los bloques se escriben en líneas independientes, es decir, con saltos de línea antes y después del elemento, así como una cantidad de margen arriba y abajo que depende del tipo de elemento de que se trate.

## Atributo CSS Float y el flujo

Otro atributo que afecta al fluir de los elementos en la página es el atributo float de CSS, que se utiliza bastante en la maquetación web. Este atributo podemos utilizarlo sobre elementos de la página de tipo "block" y lo que hace es convertirlos, en "flotantes" que es un comportamiento parecido a lo que sería el mencionado anteriormente "inline". Con float podemos indicar tanto left como right y conseguiremos que los elementos se posicionen a la izquierda o la derecha, con el contenido que se coloque a continuación rodeando al elemento flotante. La diferencia es que los elementos continúan siendo tipo "block" y aceptan atributos como el margen (atributo CSS margin), para indicar que haya un espacio en blanco a los lados y arriba y abajo del elemento.

Por ejemplo, los elementos de las listas (etiqueta LI) son por defecto de tipo "block", por eso aparecen siempre uno abajo de otro, en líneas consecutivas. Pero nosotros podríamos cambiar ese comportamiento con:

i{  
   float: right;  
}

Así, una lista como esta:

<ul>  
<li>Elemento1</li>  
<li>Elemento2</li>  
<li>Elemento3</li>  
</ul>

Veríamos como el primer elemento aparece a la derecha del todo y los otros elementos van colocándose en la misma línea en el siguiente espacio libre que haya. Así, el segundo elemento se colocaría en la misma línea, todo a la derecha que se puede, conforme al espacio que se tenga en el contenedor donde estén colocados.

## Flujo y el atributo position

El atributo position de CSS sí que es capaz de cambiar radicalmente el flujo de los elementos de la página. Este atributo, que explicaremos con detalle más adelante en otros artículos de DesarrolloWeb.com, por defecto tiene el valor "static", que indica que el elemento forma parte del flujo HTML normal de la página.

Sin embargo, con el atributo CSS position, podemos indicar otros valores que hacen que los elementos salgan del flujo HTML y se posicionen en lugares fijos, que no tienen que ver con la posición en la que aparezcan en el código HTML. Por ejemplo:

<div style="position: absolute; top: 10px; left: 100px;">  
Este elemento tiene posicionamiento absoluto  
</div>

Hace que ese elemento quede fuera del flujo de elementos en la página y entonces aparecería en el lugar que se indica con los atributos top y left (top indica la distancia desde la parte de arriba y left la distancia desde el borde izquierdo). Los otros elementos que formen parte del flujo de la página no quedan afectados por los elementos con posicionamiento absoluto.

Otro valor para el atributo position que hace que los elementos queden posicionados fuera del fluir normal de elementos en la página es "fixed", cuyo comportamiento veremos más adelante en otros artículos. Recomendamos seguir la lectura, para las personas que quieran profundizar en este tema, a partir del artículo Posicionamiento CSS.

# 14. Posicionamiento CSS

## Atributos para posicionamiento CSS

Existen numerosos atributos para posicionar con CSS cualquier elemento de la página. Además, a medida que van siendo presentadas nuevas versiones de CSS, estos atributos y sus posibles valores van aumentando. En CSS 2 contamos con diversos atributos que veremos a continuación.

**Atributo position:**

Este atributo es, digamos, el principal para definir el tipo de posicionamiento de un elemento. Merece la pena verlo por separado y en detalle. Más adelante lo trataremos en el artículo Tipos de posicionamiento con el atributo position, pero adelantamos que va a permitir varios valores para establecer cómo se posicionará el elemento en la página y si formará parte del flujo normal de HTML. Susvaloresposibles son absolute, fixed, relative, static e inherit.

**Atributos top, left, right, bottom:**

Sirven para indicar la posición de un elemento, cuando éste tiene los valores de position "absolute", "relative" o "fixed" (en otros valores del atributo position estos atributos son ignorados). El atributo top indica la distancia desde el borde superior de la página y left desde el borde de la izquierda. También se puede indicar opcionalmente la posición con bottom, que es la distancia desde abajo y right, que es la distancia desde la derecha.

**Atributos float y clear:**

Float sirve para establecer que un elemento tiene que "flotar", colocándose los valores "right" o "left" para que floten a izquierda o derecha. Por si sirve de aclaración, que los elementos floten es algo así como lo que pasa cuando definimos el atributo HTML align="right" o align="left" en las imágenes o tablas. Con el atributo clear hacemos que el elemento se coloque en el primer área libre que tenga al lugar donde se indique. Por ejemplo el valor de clear "right" hace que el elemento se coloque en el primer lugar donde no tenga ningún elemento flotando a la derecha. El valor de clear "both" hace que el elemento se coloque donde no tenga elementos flotanto, tanto a la derecha como a la izquierda.

**Atributo clip:**

Establece un área de recorte de la porción visible de un elemento. Este área de recorte se establece con varios valores, como se detalla en el artículo atributos para capas.

**Atributo display:**

Especifica el tipo de caja que debe que tener un elemento, que puede ser de diversas formas. Este atributo también tiene bastante utilización y entre los valores más corrientes podríamos destacar: "none", que hace que esa caja o elemento no aparezca en la página ni se reserve espacio para ella. "block", que sirve para que la caja sea un bloque y se muestre en una línea o líneas independientes de otros elementos de la página. "inline", que indica que esa caja tiene que mostrarse en la misma línea que otros elementos escritos antes o después.

**Atributo overflow:**

Este atributo sirve para decir qué es lo que pasa con los elementos que no caben en una caja debido a las dimensiones de la misma y del contenido que tenga. Se explica con detalle en el artículo Overflow en CSS.

**Atributo visibility:**

Atributo para definir la visibilidad de un elemento. Con este atributo podemos decir que ciertos elementos de la página sean visibles o invisibles, pero atención, aunque un elemento sea invisible, continúa ocupando espacio en la página. Si queremos que no sea invisible y no se le reserve espacio en la página, hay que utilizar el atributo display con el valor "none". Los valores más corrientes de visibility son: "visible", que hace que el elemento se vea (valor por defecto) y "hidden", que hace que el elemento sea invisible, aunque continúe ocupando espacio.

**Atributo z-index:**

Este atributo tiene como valor cualquier número entero. Sirve para indicar qué capa se tiene que ver por encima o por debajo de otra u otras, en caso que varias capas estén superpuestas. A mayores valores de z-index, la capa se coloca más al frente, tapando otras capas que tengan valores menores de z-index.

# 15. EL atributo Overflow de CSS

# Overflow sirve en el modelado de cajas para indicar al navegador qué es lo que debe hacer con el contenido que no cabe dentro de una capa, según las dimensiones que se le han asignado.

Como sabemos, a las capas (elementos DIV) podemos asignarles un tamaño, en anchura y altura. Pero muchas veces el contenido que colocamos en la capa sobrepasa el espacio que hemos destinado a ella. Entonces lo que suele ocurrir es que la capa crece lo suficiente para que el contenido colocado dentro quepa. Habitualmente las capas crecen en altura, por lo que a más contenido más tamaño tendrá en altura. Este es un comportamiento que podemos alterar con el uso del atributo overflow.

Dicho de otro modo, overflow permite que se recorte el contenido de una capa, para mostrar únicamente el contenido que quepa, según sus dimensiones. Para acceder al contenido que no se muestra, porque no cabe en la capa, se puede configurar overflow para que aparezcan unas barras de desplazamiento.

Así pues, pasemos directamente a ver cuáles son los atributos posibles con el atributo overflow:

* **visible:** Este valor indica que se debe mostrar todo el contenido de la capa, aunque no quepa en tamaño con la que la hemos configurado. En Internet Explorer ocurre que capa crece en tamaño lo suficiente para que quepa todo el contenido que hemos colocado dentro. En Firefox ocurre que la capa tiene el tamaño marcado, pero el contenido se sigue viendo, aunque fuera del espacio donde de la capa, pudiendo superponerse a un texto o imagen que hubiera debajo. El contenido no se recorta en caso alguno, es decir, siempre estará visible.
* **hidden:** Este valor indica que los contenidos que, por el tamaño de la capa, no quepan en la misma, se deben recortar. Por ello, la capa tendrá siempre el tamaño configurado, pero los contenidos en ocasiones podrán no verse por completo.
* **scroll:** Este valor indica que la capa debe tener el tamaño que se haya configurado inicialmente y que además se deben mostrar unas barras de desplazamiento, para mover el contenido de la capa dentro del espacio de la misma. Las barras de desplazamiento siempre salen, se requieran o no.
* **auto:** Con este valor también se respetarán las dimensiones asignadas a una caja. El contenido será recortado, pero aparecerán las barras de desplazamiento para moverlo. Sin embargo, en este caso las barras de desplazamiento podrán salir o no, depende de si son necesarias o no para ver todo el contenido de la capa.

# Así pues, el atributo overflow nos permitirá tener un mayor control sobre los espacios que destinamos a cada caja de nuestro diseño. Es muy utilizado para mostrar textos largos, que se desean integrar dentro de otro texto o una interfaz donde no tenemos espacio disponible para colocarlos o no deseamos que crezcan más de la cuenta. Por ejemplo para mostrar código fuentedentro del texto de un artículo, como sigue:

<html>  
<head>  
   <title>Título</title>  
</head>  
  
<body>  
  
Cuerpo...  
  
</body>  
</html>

Este ejemplo, habremos podido apreciar la barra de desplazamiento vertical, se obtiene con un atributo overflow: auto;. El código utilizado es como sigue:

<div style="overflow: auto; width: 300px; height: 100px; background-color:#ededed; border: 1px solid #990000;">

CONTENIDO....  
</div>

Ahora veamos otro ejemplo, en el que simplemente se recorta el texto que no cabe en la capa. Hemos indicado overflow: hidden, por lo que el texto que sobra no se va a visualizar.

Esta capa tiene un contenido mayor del que cabe en el espacio que he asignado con

En este caso vemos como el texto aparece recortado, porque no cabe en el espacio asignado de la capa. El código sería como el que sigue:

<div style="overflow: hidden; width: 200px; height: 50px; border: 1px solid #990000;">  
CONTENIDO...  
</div>

**16. Tipos de posicionamiento con el atributo position de CSS**

**position: static**

Es el valor predeterminado del atributo y el posicionamiento normal de los elementos en la página. Quiere decir que los elementos se colocarán según el flujo normal del HTML, es decir, según estén escritos en el propio código HTML. Por decirlo de otra manera, static no provoca ningún posicionamiento especial de los elementos y por tanto, los atributos top, left, right y bottom no se tendrán en cuenta.

Podemos ver un ejemplo de posicionamiento static:

<div style="position: static; background-color: #ff9; padding: 10px; width: 300px;">Esto es una capa con posicionamiento estático</div>  
<div style="position: static; background-color: #f9f; padding: 10px; width: 500px;">posicionamiento static, predeterminado.</div>  
<h1>CSS</h1>

<div style="background-color: #9ff; padding: 10px; width: 400px;">Posicionamiento static, aunque en este caso no se indicó el atributo position static, pues no hace falta.</div>

**position: absolute**

El valor absolute en el atributo position permite posicionar elementos de manera absoluta, esto es de manera definida por valores de los atributos top, left, bottom y right, que indican la distancia con respecto a un punto. Las capas o elementos con posicionamiento absoluto quedan aparte del flujo normal del HTML, quiere decir esto que no se afectan por el lugar donde aparezcan en el código HTML y tampoco afectan ellas a otros elementos que sí que formen parte del flujo normal del HTML.

Los valores top, left, bottom y right se expresan con unidades CSS y son una distancia con respecto al primer elemento contenedor que tenga un valor de position distinto de static. Si todos los contenedores donde esté la capa posicionada con position absolute (todos sus padres hasta llegar a BODY) son static, simplemente se posiciona con respecto al lado superior de la página, para el caso de top, el inferior para bottom, del lado izquierdo para left o el derecho, en el caso de utilizar right.

<div style="position: absolute; width: 300px; height: 140px; top: 100px; left: 30px; background-color: #ff8800; color: #fff; padding: 15px;z-index: 2;">  
Esta capa tiene posicionamiento absoluto.  
<br>  
<br>  
Me permite especificar top y left para colocarla con respecto a la esquina superior izquierda.  
</div>  
  
<div style="position: absolute; width: 820px; height: 30px; padding: 10px; background-color: #ddf; top: 150px; left: 10px; z-index: 1;">Posicionamiento absoluto con z-index menor (la capa aparece por debajo de otras que se superponen con z-index mayor.</div>  
  
<div style="position: absolute; width: 100px; height: 20px; padding: 10px; background-color: #ddf; bottom: 10px; right: 10px;">Posicionamiento absoluto con atributos bottom y right</div>  
  
<h1>Posicionamiento CSS</h1>

La primera capa (llamamos así a los elementos DIV que tienen posicionamiento CSS), tiene como todas las del ejemplo, posicionamiento absoluto. Los atributos top: 100px y left: 30px quieren decir que se posiciona a 100 píxeles de la parte superior de la página y a 30 píxeles de la izquerda. En este caso las distancias top y left para ubicar la capa con position absolute son relativas a la esquina superior izquierda del área disponible del navegador, pues esta capa no está dentro de ninguna otra con posicionamiento distinto de static. Cabe llamar la atención en esta primera capa también sobre el atributo z-index: 2, que servirá para indicarle al navegador la posición de la capa, en la tercera dimensión, con respecto a otras que se puedan superponer, para que sepa cuál tiene que estar debajo y cuál arriba.

La segunda capa podemos ver que tiene un z-index:1. Eso quiere decir, que en caso se posicione en el mismo lugar se ocultará por la capa primera, que tiene un z-index mayor.

En la tercera capa hemos probado el posicionamiento utilizando los atributos bottom y right, así que la estamos posicionando con respecto a la esquina inferior derecha.

<div style="position: absolute; top: 100px; left: 200px; background-color: #ff9966; width: 400px; height: 100px;">  
<div style="position: absolute; top: 10px; left:10px;">  
Uno  
</div>  
<div style="position: absolute; top: 10px; left:100px;">  
Dos  
</div>  
<div style="position: absolute; top: 10px; left:200px;">  
Tres  
</div>  
</div>

En este caso la primera capa, que no está dentro de ninguna otra, se posiciona con top y left con respecto a la esquina superior izquierda del espacio disponible en el navegador para el cuerpo de la página. Las capas anidadas están también con position: absolute, pero al estar dentro de otra capa que tiene posicionamiento distinto de static, sus valores top y left son relativos a la esquina superior izquierda de la capa que las contiene.

**position: relative**

El valor relative en el atributo position indica que la capa sí forma parte del flujo normal de elementos de la página, por lo que su posición dependerá del lugar donde esté en el código y el flujo HTML. Además, las capas con posicionamiento relative, admiten los valores top y left para definir la distancia a la que se colocan con respecto al punto donde esté en ese momento el flujo normal del HTML. Como afectan al mencionado flujo del HTML, los elementos colocados después de las capas relative, tendrán en cuenta sus dimensiones para continuar el flujo y saber dónde colocarse. Sin embargo, no se tendrá en cuenta los top y left configurados.

<h1>Hola</h1>  
<div style="background-color: #606; color:#ffc; padding:10px; text-align: center; width: 300px;">Hola esto es una prueba</div>  
<div style="position: relative; width: 300px; padding: 10px; background-color: #066; color:#ffc; top:100px; left: 30px;">capa de posicionamiento relative<br>Se tiene en cuenta esta capa para posicionar las siguientes.</div>  
<h2>hola de nuevo!</h2>

Las etiquetas H1 y H2 respetan el flujo HTML y también tenemos un elemento DIV donde no se ha especificado nada en position, luego es static y por tanto también es afectada por el flujo. Hay una capa relative, en el segundo elemento DIV, que también se posiciona con respecto al flujo normal. Como tiene un top y left, aparece un poco desplazada del lugar que le tocaría con respecto al flujo.

El último H2 que aparece se coloca teniendo en cuenta al flujo y tiene en cuenta la capa relative, por eso deja un espacio en blanco arriba, pero no atiende a la posición real de ésta, que se marcó con los atributos top y left.

**position: fixed**

Este atributo sirve para posicionar una capa con posicionamiento absoluto, pero su posición final será siempre fija, es decir, aunque se desplace el documento con las barras de desplazamiento del navegador, siempre aparecerá en la misma posición.

El lugar donde se "anclará" la capa siempre es relativo al cuerpo (el espacio disponible del navegador para la página). Si utilizamos top y left, estaremos marcando su posición con respecto a la esquina superior izquierda y si utilizamos bottom y right su posición será relativa a la esquina inferior derecha.

<div style="position: fixed; width: 300px; height: 140px; top: 100px; left: 30px; background-color: #ff8800; color: #fff; padding: 15px;z-index: 1;">  
Esta capa tiene posicionamiento fixed.  
<br>  
<br>  
Me permite especificar top y left para colocarla con respecto a la esquina superior izquierda.  
</div>  
  
<div style="position: fixed; width: 700px; height: 30px; padding: 10px; background-color: #d0f; top: 150px; left: 10px; z-index: 2;">Posicionamiento fixed</div>  
<h1>Hola</h1>  
<div style="position: fixed; width: 100px; height: 30px; padding: 10px; background-color: #0df; bottom: 10px; right: 10px; z-index: 4;">Posicionamiento fixed</div>  
<br>  
<br>  
<br>  
<br>  
Pongo texto para que se vea!!  
<br>  
<br>  
<br>  
Esto hace desplazamiento, con tanto br  
<br>  
<br>  
...  
<br>

Se puede ver que hay varias capas con position: fixed y un montón de BR para que la página pueda tener un desplazamiento. Si vemos la página en marcha y hacemos scroll hacia abajo con la barra de desplazamiento, veremos que las capas fixed siempre mantienen la misma posición.

*Nota: El valor fixed en el atributo position funciona en todos los navegadores, pero en el caso de Internet Explorer sólo funciona en la versión 7 y superiores. Además, para que funcione en Explorer tiene que declararse un DOCTYPE!. Servirían varios tipos de DOCTYPE!, sin embargo debería declararse con el formato completo. Algo así como:*

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">

O por poner otro ejemplo:

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

**position: inherit**

El valor inherit indica que el valor de position tiene que heredarse del elemento padre. No funciona en Explorer, al menos hasta la versión 8. Tiene en verdad poca utilidad y además, como no funciona en el navegador más utilizado en la actualidad, tiene aun menos sentido usarlo. Por ello, no ponemos ejemplos.

**Conclusión sobre el atributo position de CSS**

Esperemos que con las anteriores explicaciones y ejemplos se hayan podido entender bien las distintas posibilidades del atributo position, que es sin duda clave para el posicionamiento CSS. Lo más común para la maquetación web es utilizar el posicionamiento static, pero el posicionamiento absoluto, junto con el posicionamiento fixed, e incluso relative, pueden ser muy útiles para diseños más complejos, donde se requiera una mayor precisión en la colocación de los distintos elementos o las capas.

Además, para hacer efectos Javascript y DHTML en general, se utilizan frecuentemente posicionamientos absolutos. Son muy útiles porque permite que los elementos dinámicos no formen parte del flujo normal del HTML, y por tanto, podemos situarlos en cualquier lugar el área disponible del navegador, e incluso moverlos dinámicamente al cambiar sus propiedades top y left mediante scripts en el lado del cliente.

**17. 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 <span class="mayor">y este es mayor</span>, este vuelve a ser normal <span class="menor">y este es menor</span>

**18. Declaración !important en CSS**

Es una declaración un tanto especial que podemos utilizar al definir reglas de estilos para una página web. Se trata de !important, una palabra que hará que determinadas propiedades tomen mayor importancia y, por tanto, se tengan más en cuenta que otras que puedan sobreescribirlas.

Otra cosa que veremos de paso es cómo el uso de !important nos proporcionará una sencilla y valida técnica para poder definir reglas de estilos distintas para navegadores antiguos, como Internet Explorer 6.

Además, important! se puede utilizar en las hojas de estilo de usuario, para que cada persona pueda definir para su propio navegador si lo desea, un estilo CSS por defecto que se tenga en cuenta en todas las web que visitemos.

**Uso y efecto de la declaración !important**

Para utilizar !important en una regla de estilo, siempre se coloca en la parte del valor del atributo, antes del punto y coma ";". Por ejemplo

body{  
font-family: verdana, arial !important;  
}

El efecto es que siempre se aplicará el estilo definido como !important, aunque luego se pueda sobrescribir con otro estilo más tarde en la misma declaración o en otra distinta. Veamos este ejemplo:

td{  
font-size: 16pt !important;  
font-size: 8px;  
}

Tenemos una declaración de estilos para los elementos TD, donde definimos dos veces el atributo font-size. En condiciones normales, se tendría en cuenta el valor definido en segundo lugar, porque lo sobrescribe. Sin embargo, que que el primer font-size está definido como !important, en realidad lo que ocurrirá es que se tenga en cuenta finalmente y el tamaño de letra por tanto será 16pt.

Este efecto lo podemos aplicar también a distintos tipos de selectores de CSS. De modo que podremos encontrarnos que para un elemento se indique un estilo y luego para una clase (class de CSS) se aplique otro y se tenga en cuenta el definido como !important. Veamos este ejemplo de CSS:

td{  
font-family: verdana, arial !important;  
}  
td.micelda{  
font-family: monospace;  
}

Que aplicado sobre el siguiente HTML:

<table>  
<tr>   
<td class="micelda">Hola</td>  
<td>23232</td>  
</tr>  
</table>

Daría como resultado, en condiciones normales, que la primera celda, de clase "micelda", tuviese la fuente font-family: monospace y la segunda celda, que no tiene ningún class, tuviera el estilo font-family: verdana, arial. Sin embargo, como el font-family definido en primer caso tiene la declaración !important, la fuente será siempre verdana, arial, para las dos celdas.

**Usar !important para definir estilos diferentes en navegadores antiguos**

La declaración !important no la entienden todos los navegadores, por tanto, algunos simplemente la ignorarán y otros no. El caso más representativo, por ser un navegador que todavía se utiliza habitualmente por los internautas, sería Internet Explorer 6.

Así pues, utilizando !important podemos conseguir definir estilos diferentes para Internet Explorer 6 y para la mayoría de los otros navegadores que pueden visitar nuestra web. Esto lo podemos conseguir de la siguiente manera.

div{  
   background-image: url(fondo-semitransparente.png) !important;  
   background-image: url(fondo.gif);  
}

Como Internet Explorer 6 ignora la directriz !important, ocurrirá que tendrá en cuenta el segundo valor de background-image, ya que está repetido y por tanto sobrescribe al primero. Por ello, en este caso IE6 mostrará como fondo el archivo llamado "fondo.gif".

Los otros navegadores, como entienden !important, mostrarán el estilo que había definido anteriormente y por tanto utilizarán como fondo el archivo "fondo-semitransparente.png".

***Nota:****Dicho sea de paso, como IE6 tiene problemas al mostrar fondos semitransparentes (con*[*canal alpha en el PNG*](http://www.desarrolloweb.com/articulos/formatos-imagen.html)*) esta sería una posible técnica para conseguir que en Explorer 6 se utilice un fondo de imagen distinto (por ejemplo en .gif) que el que se utiliza en otros navegadores que no tienen problema con el .png.*

**19. Conclusiones y buenas prácticas**

**19.1. Conclusiones**

El uso (adecuado) de hojas de estilo es una de las bases de la codificación

semántica, la cual descansa sobre el principio esencial de separar elcontenido de la presentación.

De este modo, se promueve la utilización de los elementos de (X)HTML con

valor semántico, y no para dar formato. Éste último es responsabilidad de

las reglas que se declaran en las hojas de estilo. Porejemplo, si no nos

gusta que el título principal de una página aparezca en negrita, la buena

práctica consistirá en modificar la apariencia de h1, y no en marcar el título

con un elemento que no le corresponda (o al revés, si nos interesa dotar a

una palabra o una frase de un tamaño mayor, la clave estará en definir su

presentación con una regla de estilo y no en asignarle, por ejemplo, un

marca de cabecera que no le corresponda).

**19.2. Buenas prácticas**

Para finalizar, mostraremos una lista de algunas buenas prácticas:

1.Declarar la hoja de estilo en un archivo separado (mejor que en la hoja de estilo, salvo necesidad específica).

2.Utilizar medidas relativas (tanto por ciento en lugar de píxeles, o unidades emen lugar de puntos, etc.).

3.Usar elementos semánticos (h1, h2, blockquote, etc.) paraorganizar los contenidos de la página, no para conseguir unadeterminada apariencia de la misma.

4.Modificar la apariencia de la página exclusivamente medianteestilos (y no con elementos o atributos desaconsejados como font,center, etc. o usando elementos fuera de lugar).

5.Limitar al máximo declarar estilos dentro de elementosindividuales (es decir, evitar el uso del atributostyle dentro deelementos).

6.Denominar los selectores de clase y de identidad con nombres defunción, p.e.:tituloFilm, menuGlobal, menuLocal, etc., (y no conel nombre de la apariencia resultante, p.e.,negrita,letraMenor,letraGrande, etc.). Por tanto: usar nombres *semánticos*y no de*apariencia*.

7.Organizar los elementos principales de la página, p.e. cabecera,navegación, contenido, etc., con secciones marcadas comoelementosdiv.

8.Utilizar elementos div(así como atributos y selectoresid) conpropiedades de posicionamiento para establecer la estructura de lapágina (*layout*) si fuera necesario en lugar de tablas o *frames.*