**Curso Básico de CSS2**

En la parte de XHTML hemos comentado muchas veces que podemos cambiar el aspecto de una página web mediante **CSS** (*Cascading Style Sheets*). Ahora es el momento de aprender cómo hacerlo.  
  
Me ha costado decidirme sobre cómo explicar las diferentes propiedades del CSS, pero creo que la mejor manera de aprender sin que se haga muy pesado es poniendo ejemplos de problemas comunes con los que nos topamos, junto con la manera de solucionarlo en CSS. Después de todo, esto no pretende ser una guía de referencia con todas las propiedades y todos sus posibles valores.  
  
*Googleando un poco, podrás encontrar muchas. Y siempre te queda consultar las especificaciones oficiales en la web del W3C. Es un PDF muy majo de 359 páginas..*  
  
**¿Dónde escribo el código CSS?**  
  
Hay dos opciones para insertar CSS en un documento XHTML. Lo más normal es hacerlo en un**archivo externo** (que se llama "hoja de estilos") y enlazarlo desde nuestro documento XHTML. Esto tiene una ventaja enorme, y es que podemos tener muchas páginas enlazando a la misma hoja de estilos. Si más adelante queremos cambiar algo, sólo tenemos que modificar un único archivo (la hoja de estilos) y afectará a todas las páginas. Podemos enlazar una o más hojas poniendo esto dentro de la cabecera (*<head>*):

**<link** **href**="hoja\_estilos.css" **rel**="stylesheet" **type**="text/css" **/>**

La otra opción es escribir la información referente a los estilos incrustada **en el mismo archivo XHTML**. Lo podemos hacer escribiendo entre las etiquetas *<style>* y *</style>*, que también deben ir en la cabecera.  
  
**Vale, ¿pero cómo funciona?**  
  
En una hoja de estilos utilizamos **reglas** que consisten en elegir **selectores** a los que asignamos una serie de **propiedades**. Por ejemplo, si queremos que nuestra página web tenga el fondo blanco, las letras grises, y una fuente Arial de 10 puntos de tamaño, escribimos esta regla:

body {  
    background-color: #fff;  
    color: #666;  
    font-family: Arial, sans-serif;  
    font-size: 10pt;  
}

Como ves, las líneas terminan en un **punto y coma**. Es muy importante que no se nos olvide. Ah, y si te lo estás preguntando, *#fff* representa al color blanco, y *#666* a un gris oscuro.  
  
**¿Selectores?**  
  
Los **selectores** los utilizamos para elegir a qué elementos se aplican las propiedades que escribimos. Hay diferentes tipos de selectores, los más importantes son los que veremos ahora.  
  
Si queremos elegir una **etiqueta**, simplemente escribimos su nombre. Por ejemplo, si queremos establecer las propiedades para los enlaces:

a {  
    ...  
}

También podemos elegir un **elemento único** utilizando su atributo *id*. Para ello, empleamos la almohadilla:

#menu {  
    ...  
}

Otra cosa que podemos hacer es definir una **clase** y hacer que muchos elementos la utilicen, escribiendo un punto antes del nombre. Por ejemplo, si queremos destacar algo:

.importante {  
   ...  
}

Luego podríamos emplear esa clase en los párrafos que queramos (o cualquier otro elemento), usando el atributo *class* de este modo:

**<p** **class**="importante"**>**Bla bla bla**</p>**

También podemos seleccionar ciertos elementos, pero sólo cuando estén contenidos dentro de otros. Por ejemplo, si queremos seleccionar los *<li>*, pero sólo de las listas sin ordenar:

ul li {  
   ...  
}

**¿Qué significa eso de "cascading"?***Cascading* significa cascada, y tiene que ver con la **herencia**. En CSS, los elementos hijos heredan todas las propiedades de sus padres. Por ejemplo, si establecemos una regla para el elemento *table*, sus hijos (*td* entre otros) también tendrán esas mismas reglas.  
  
Es por esto que si, por ejemplo, queremos establecer un tipo de fuente para todo el documento, debemos indicarlo en el elemento *body* para que se propague por toda nuestra página.¿Puedo poner comentarios?Claro, pero son distintos a los de XHTML, que van entre *<!--* y *-->*.

   /\* Esto es un comentario \*/

**¿Por qué #fff significa blanco?**  
  
En CSS hay varias maneras de indicar un color. Podemos hacerlo en inglés, por ejemplo. En lugar de *#fff* escribimos *white* y lo solucionamos. El problema es que así es muy incómodo porque tenemos que aprendernos los nombres de cada color, y puede que exista algún color que queramos poner y que no tenga nombre\ldots además de que supone una falta de precisión absoluta. Así que normalmente se usa la **notación en hexadecimal**.  
  
*¿Recuerdas esas discusiones en parvulario sobre si cierto color era lila, morado, o fucsia? ¿Quién te dice que tu concepto de morado sea el mismo que el de tu navegador?*  
Los colores en nuestro monitor está formado por tres haces de luz: rojo, verde y azul. Se llama **sistema RGB** (*Red Green Blue*). Si cogemos un programa de dibujo, vemos que podemos eleigr un color indicando el valor de sus componentes rojo, verde y azul por separado. Este valor puede variar de *0* a *255* (se usan 8 bits para representar este rango). Por ejemplo, si queremos amarillo puro, ponemos *255* para el rojo, *255* para el verde y *0* para el azul.  
  
¿Qué tiene que ver esto con el sistema hexadecimal? Pues que este rango de valores (*0..255*) puede ser representado con dos dígitos hexadecimales, que van desde el *00* hasta el *FF*. La forma de escribir un color completo es *#RRGGBB*, donde la almohadilla indica que se trata de un color hexadecimal, y *RR*, *GG* y *BB* son los dígitos correspondientes al rojo, verde y azul, respectivamente. Entonces el amarillo sería *#ffff00*.  
  
La mayoría de los colores que usaremos tendrán por cada componente los mismos dígitos. Por ejemplo: *#ff0000* (rojo), *#0000aa* (azul marino) o *#000000* (negro). Estos colores los podemos abreviar y dejarlos en tres cifras. Por ejemplo, *#f00* es completamente equivalente a *#ff0000*. Sin embargo, colores como *#a0a0a0* no pueden ser abreviados.  
  
Por tanto, para conseguir el color blanco, debemos poner poner todas las componentes a *255*, que en hexadecimal nos quedaría *#ffffff*, y abreviado en *#fff*.  
  
La mayoría de editores de código (X)HTML o de programas de dibujo nos mostrarán en la paleta el valor del color en hexadecimal. En cualquier caso, siempre podemos obtener el valor de sus componentes por separado y convertirlo a hexadecimal con una calculadora.  
  
**COLORES, FONDOS Y FUENTES**Para empezar a meter mano al CSS, conviene conocer algunas propiedades sencillas. ¡Empezamos!  
El color de primer planoLa propiedad *color* hace referencia al *foreground color*, es decir, al **color** que está por encima del fondo. Hablando en plata, viene a ser el color del texto. Si queremos que nuestra página tenga las letras de gris oscuro, lo conseguimos con esto:

body { color : #666; }

Muy fácil, ¿no? Que pase el siguiente.  
  
El fondoPodemos modificar el fondo de un elemento con la propiedad *background*, que tiene la siguiente sintaxis:

background : color | image | repeat | attachment | position ;

El primer parámetro corresponde al **color** de fondo, los siguientes son relativos a la **imagen** de fondo:  
  
**Image**: aquí indicamos la ruta a la imagen que pondremos. Por ejemplo, *url(fondo.gif)*.  
  
**repeat**: con esto establecemos si queremos que la imagen se repita o no, tanto horizontal como verticalmente. Con *repeat* se repite siempre en ambos sentidos (valor por defecto), mientras que con *no-repeat* no se repite nunca. Con *repeat-x* se repite sólo en horizontal, y con *repeat-y* sólo en vertical.  
  
**attachment**: sirve para indicar si el fondo se queda fijo en el sitio o se desplaza con scroll. Es un poco difícil de explicar, así que lo mejor es que lo probéis vosotros mismos: con *scroll*(valor por defecto) el fondo se desplaza, y con *fixed* se queda siempre en el mismo sitio.  
  
**position**: indica la posición del fondo. Indicamos tanto la posicion desde la izquierda como desde arriba (ya sea en píxeles, porcentajes, o incluso palabras).  
  
*En este caso, si podemos usar palabras, es preferible que lo hagamos. Disponemos de top (arriba), bottom (abajo), left (izquierda) y right (derecha).*  
Los valores por defecto son *0 % 0 %*, que sitúan al fondo en la esquina superior izquierda. Si lo quisiéramos en las coordenadas 20,30 (tomando como el origen a la esquina superior izquierda), escribiríamos *20px 30px*. Si queremos el fondo centrado, pues *50 % 50 %*.  
  
Podemos omitir alguna propiedad si queremos. Además, podemos establecer los valores de forma individual, usando las propiedades *background-color*, *background-repeat*, etc.  
  
Veamos algunos ejemplos para poner fondo a nuestra página:

/\* sólo color de fondo \*/  
body { background : #fff; }  
  
/\* color de fondo e imagen en mosaico \*/  
body { background : #fff url(fondo.gif); }  
  
/\* imagen fija , centrada y sin repetir \*/  
/\* a modo de marca de agua \*/  
body {  
    background-color: #fff;  
    background-image: url(fondo.gif);  
    background-attachment: **fixed**;  
    background-repeat: **no**-repeat;  
    background-position: 50% 50%;  
}

Hemos explicado de forma bastante detallada cómo funciona la propiedad *background*, pero no te malacostumbres. Ahora iremos mucho más deprisa, y seguramente te toque buscar más información por tu cuenta.  
  
*¿Recuerdas ese librito majo de 359 páginas?*  
  
**Fuente**  
  
Hay varias propiedades que nos permiten jugar con el aspecto del texto. Podemos englobar todas bajo *font*, pero primero veremos algunas subpropiedades.  
  
Antes de nada, debemos considerar que **no todos los ordenadores tienen las mismas fuentes instaladas**. Es decir, que mi wonderfulosa fuente llamada *chachi.ttf* no tiene por qué tenerla mi vecino. Es más, lo normal es que no la tenga. ¿Entonces qué hacemos? Usar sólo fuentes "estándar", que tengan la mayoría de ordenadores. Además, podemos especificar varias, de forma que si no se tiene la primera, se muestre la segunda, si no se tiene la segunda, pues la tercera, etc.  
  
*La Ley de Murphy, como en cualquier otro aspecto de la vida, es aplicable al diseño web también.*  
  
A lo que íbamos. La propiedad que sirve para cambiar la fuente es *font-family*. Como acabamos de ver, podemos indicar varias fuentes, por orden de preferencia. Si el nombre de una fuente tiene espacios en blanco, hay que ponerla entre comillas. Por ejemplo:  
  
*Haz esto y muere. La irrupción de la Comic Sans por doquier sólo es comparable a la aparición de Jar Jar Binks en Star Wars.*

font-family: "Comic Sans MS", Arial, sans-serif;

EL **tamaño** de la fuente lo controlamos con *font-size*. Podemos indicar medidas en píxeles (*px*) o en puntos (*pt*).  
  
*Mentira cochina. Podemos (y mucha gente opina que debemos) usar medidas relativas, pero el tema de la herencia es muy importante en estos casos y... Dejémoslo en que no es newbie-friendly.*  
Para **alinear** el texto tenemos a *text-align*, que puede tomar los valores *right* (derecha), *left*(izquierda), *center* (centrado) o *justify* (justificado).  
  
*A mucha gente le gusta montar Guerras Sagradas acerca de la conveniencia o no de usar texto justificado. Todavía no se han puesto de acuerdo, así que probablemente es un asunto de gustos.*Es muy normal tener una clase así:

.centrar { text-align: center; }

Si queremos *indentar* los párrafos, pues usamos *text-indent*:

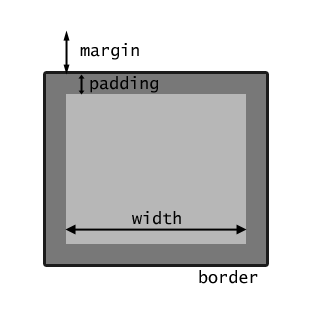
p { text-indent : 2em; }

¿Qué significa em? Es una **unidad relativa**. Lo más fácil es trabajar con píxeles o porcentajes, pero en algunos casos conviene utilizar otras unidades. *1 em* equivale al tamaño de la fuente.  
  
*Bueno, exactamente equivale a la anchura de la letra "M"*  
  
mayúscula.Así, si por herencia o por cualquier otra cosa el texto se muestra en un tamaño de fuente diferente, la proporción de indentado sería siempre la misma. Si pusiéramos *20px*, siempre sería esa, aunque el tamaño de la fuente fuera de *20* o de *32* píxeles.  
  
También podemos cambiar la **decoración** del texto mediante *text-decoration*. Puede tomar diversos valores, como *none* (sin adornos), *underline* (subrayado), *overline* (subrayado superior) o *line-through* (tachado).  
  
¡Cuidado! Los humanos tenemos la estúpida manía de pensar que cualquier texto subrayado es un enlace.Si queremos que nuestros links no tengan subrayado, pondremos:

a { text-decoration : none; }

Podemos **transformar** las mayúsculas y minúsculas con la propiedad *text-transform*. Si ponemos *lowercase*, todo se mostrará en minúsculas; con *uppercase*, en mayúsculas; y con*capitalize* pondremos una letra capital al principio de cada palabra.  
  
En cuanto al **espaciado**, para la distancia entre palabras usamos *word-spacing*; para el de las letras, *letter-spacing*; y para el **interlineado**, *line-height*.  
  
Un ejemplo con todo esto, para la etiqueta *<strong>*:

strong {  
    color: #000;  
    letter-spacing: 0.25 em;  
    text-trasform: uppercase;  
}

¿Demasiada información? Sí, pero ten en cuenta que no hay que memorizar todo esto. Ni de coña. Lo importante es saber que **existe** una propiedad que hace tal cosa, no cómo se llama ni qué parámetros tiene. Para eso ya están las guías de referencia rápida y los editores de **CSS**.  
  
**EL MODELO DE CAJA**  
Es hora de conocer uno de los fundamentos más importantes del CSS: **el modelo de caja** (*box model*). Es muy fácil, pero entenderlo bien es vital para poder realizar una buena maquetación de la web.  
  
**¿Cómo es el modelo de caja?**  
  
En realidad, todos los elementos de una web (párrafos, enlaces, imágenes, tablas, etc.) son**cajas** rectangulares. Los navegadores sitúan estas cajas de la forma que nosotros les hayamos indicado para maquetar la página.  
  
Hay dos tipos de cajas: **block** e **inline**. Los elementos **block** rompen el flujo de maquetación. Esto es, aparecen solos, insertando “saltos de línea”. Los elementos **inline** siguen el flujo, y están contenidos dentro de elementos de bloque.  
  
Por ejemplo, un párrafo sería un elemento **block** (no podemos tener un párrafo al lado del otro, sino que dos párrafos seguidos aparecerán uno **abajo del otro**. En cambio, un enlace es un elemento **inline**, ya que no “corta” el texto donde está metido.  
  
Estas dos diferencias son importantes, pero hay que tener en mente que ambos tipos comparten el modelo de caja, que es el que aparece en la figura 1  
.  
Figura 1 - Modelo de Caja  
  
Las propiedades más importantes de una caja son: **width** (ancho), **height** (alto), **padding**(relleno), **border** (borde) y **margin** (margen).  
  
**Ancho y alto**  
  
La propiedad **width** es un poco confusa, y durante mucho tiempo era horroroso trabajar con ella debido a que Ya-Sabes-Quién no la implementaba correctamente. Afortunadamente, desde la versión 6 del IE, **width** funciona como debería, así que es un quebradero de cabeza menos.  
  
*Lo que ha dado lugar a competiciones de a ver quién conseguía el chanchullo más bonito para sortear el bug. Si estás interesado, googlea buscando “box model hack”.*  
  
A lo que íbamos, **width** representa el **ancho de la caja**. Pero es el **ancho interior**, es decir, si bordes, márgenes, ni *padding*. Podemos indicar este ancho en medidas absolutas (normalmente píxeles) o relativas (normalmente %).Aunque los elementos **inline** tienen **width**, si la modificamos con CSS no veremos ningún resultado visual. Esto es porque el ancho de estos elementos se establece automáticamente para que se ajuste a las dimensiones del elemento inline. Por ejemplo, si tenemos un enlace que consiste en un texto de cinco caracteres, el ancho (**width**) de este elemento será lo que ocupen esos cinco caracteres.  
  
Sobre el **alto** de la caja, se controla con la propiedad **height**, y todo lo que hemos dicho antes sobre el ancho, también se aplica aquí.  
  
**Padding**  
  
Con **padding** establecemos la distancia de “**relleno**” entre el límite interior de la caja y el exterior (borde). Es una definición muy mala, pero se entiende a la perfección en el dibujito.  
Si queremos poner un padding de 20 píxeles para toda la caja, lo haríamos así:

padding : 20px;

Podemos establecer un padding distinto para cada lado, usando los sufijos **-top** (superior), **-bottom** (inferior), **left** (izquierda) y **right** (derecha):

padding-top: 10px;  
padding-bottom: 5px;  
padding-left: 30px;  
padding-right: 20px;

Podemos abreviar lo anterior en una sola línea, indicando primero el padding superior y luego siguiendo el orden de las agujas del reloj. Es decir, nos quedaría: arriba, derecha, abajo, izquierda. El ejemplo anterior se acortaría así:

padding: 10px 20px 5px 30px;

Otro atajo útil es especificar sólo dos medidas: una corresponderían al padding superior e inferior, y la otra al lateral. Si queremos que los paddings superior e inferior sean de 10 píxeles, y los laterales (izquierdo y derecho) de 20 píxeles, escribimos:

padding: 10px 20px;

**Bordes**  
Si queremos que nuestra caja tenga **bordes**, lo conseguimos con la propiedad border. Tiene la siguiente sintaxis:

border: width | style | color

Como habrás supuesto, width especifica el grosor del borde. Normalmente es una medida en píxeles, pero también podemos utilizar las palabras thin (fino), medium (normal) y thick (grueso). Por supuesto, cómo de gordo es thick queda a interpretación del navegador.  
  
En cuanto a style, es el **tipo** de borde. Hay bastantes, pero los más comunes son: **solid** (línea continua), **dashed** (línea discontinua), **dotted** (línea de puntos) y **double** (línea continua doble).  
  
Por último, color indica el **color** del borde  
  
*.¡Wow!*  
  
Podemos escoger un tipo de borde diferente para cada lado con los sufijos **-top**, **-bottom**, **-left**y **-right**. Por ejemplo, para poner que algo tenga el borde inferior de 1 píxel a puntitos rojos:

border-bottom: 1px dotted #f00;

Para **eliminar el borde**, simplemente ponemos que tiene de grosor 0 píxeles o que el estilo del borde es none. Esto es muy importante con las imágenes, ya que si tenemos una imagen enlazando a algo, los navegadores la ponen con un reborde muy feo. Así que esto se ha convertido ya en un fijo de las hojas de estilos:

img { border: none; }

**Márgenes**  
  
Los **márgenes** se controlan con la propiedad margin, y es la distancia entre el borde de la caja y los elementos que la rodean.  
  
En cuanto a la forma de usarla, es igual que con la propiedad padding, así que la forma de escribir y los atajos es exactamente la misma. Por ejemplo, si queremos márgenes superior e inferior de 20 píxeles, y laterales de 5 píxeles:

margin: 20px 5px;

¡Truco del almendruco! **Para centrar un elemento de bloque**, podemos hacer uso de auto:

margin: 0px **auto**;

**Capas**Vamos a hablar de una etiqueta XHTML que nos quedó por ver y está estrechamente ligada con CSS: <div>. Esta etiqueta se encarga de crear una **capa**, que es un elemento de **bloque**que sirve de contenedor a otros elementos de bloque e inline.  
  
¿Para qué nos sirven? Primero, para organizar **semánticamente** nuestra página. El atributo id tiene carga semántica, así que si queremos poner en la cabecera3 de nuestra web el título y el menú, haríamos esto:  
  
*¡Ojo! Cabecera en cuanto a maquetación. ¡No tiene nada que ver con la etiqueta head!*

**<div** **id**="header"**>**  
    **<h1>**Mi blog**</h1>**  
    **<ul** **id**="menu"**>**  
        **<li><a** **href**="..." **title**="..."**>**Principal**</a></li>**  
        **<li><a** **href**="..." **title**="..."**>**Acerca de**</a></li>**  
        **<li><a** **href**="..." **title**="..."**>**Enlaces**</a></li>**  
    **</ul>**  
**</div>**

El otro uso de las capas es el de **maquetar**. Por ejemplo, el layout típico de un blog tiene cuatro capas: la cabecera, la del contenido principal, la de la barra lateral y la del pie de página. Mediante CSS, podemos controlar la disposición de estas capas, y así conseguir el diseño que queramos.  
  
*Muy útil y socorrida para poner pijadas, como la wishlist de Amazon. Y si cuela, cuela.*  
  
**Floats**  
  
Los **floats** son probablemente una de las cosas que más cuesta dominar. En inglés los califican como “*tricky*”. Yo los califico como “pequeños cabroncetes”.  
  
Lo que hacen los **floats** **es alterar el flujo** de la página, “incrustando” en él un elemento de bloque. El ejemplo más típico es el de si queremos poner una imagen acompañando a un texto, y que el texto “envuelva” a la imagen. Esto lo conseguimos creando una clase como la siguiente:

.izquierda {  
    **float**: left;  
}

Podemos indicar tanto **left** (izquierda) como **right** (derecha). ¿Fácil, rápido y para toda la familia? No. Después de un float, suelen ocurrir **sucesos paranormales**. La mayoría de ellos suelen desaparecer mediante la propiedad clear, que se encarga de “anular” los floats. Los valores que admiten son **left**, **right** y **both** (que significa “ambos”).  
  
*Ya te darás cuenta, ya.*  
  
Volviendo al ejemplo de layouts de blogs, lo normal es poner el contenido y la barra lateral mediante floats. Lo que suele pasar es que una de estas dos columnas pasa por encima del pie de página, en lugar de quedar el pie al final de todo. Esto se suele arreglar así:

#footer {  
    clear: both;  
}

Buena suerte, y que la Fuerza te acompañe.  
  
**TRUCOS CON CSS**A partir de ahora, los capítulos de CSS serán totalmente prácticos. Este primero mostrará diferentes “trucos” para conseguir ciertas cosas. Asegúrate de probarlos y de comprender todos ellos antes de continuar.  
 **Links que cambian**  
  
Los enlaces tienen tres estados: **sin visitar** (“normales”), **hover** (cuando pasamos el cursor del ratón por encima), **activos** (cuando hacemos clic) y **visitados** (cuando ya hemos ido a esa dirección). Estos estados se corresponden a **pseudoelementos**, y podemos cambiar la apariencia de los enlaces con CSS. Sin embargo, algunos navegadores tienen un bug que, dependiendo del orden en el que escribamos las reglas, se mostrará el resultado correctamente o no. ¿Cómo lo solucionamos? Pues **LoVe/HAte.** Es un mnemotécnico que nos ayudará a recordar el orden en el que estas reglas funcionan bien en cualquier navegador. Quedaría así:

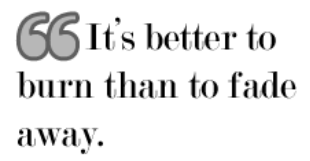
/\* links normales \*/  
a:link {  
    text-decoration: none;  
color: #00a;  
}  
/\* visitados \*/  
a:visited {  
color: #a00;  
}  
/\* hover \*/  
a:hover {  
    text-decoration: underline;  
}  
/\* activos \*/  
a:active {  
    font-weight: bold;  
}

Lo que hace el código anterior es poner los enlaces de color azul marino y sin subrayado. Cuando pasamos el cursor por encima, aparece el subrayado. Al hacer clic, el texto del enlace se pone en negrita. Los enlaces que ya hemos visitado aparecen de color granate.  
  
Fácil, ¿no?  
  
**Links con el subrayado de diferente color**  
  
Al usar la propiedad **text-decoration**, el color de la línea de subrayado es el mismo que el del texto del enlace. Podemos hacer un pequeño chanchullo para cambiar esto, y es **quitar el subrayado** y poner en su lugar un **borde:**

a {  
    color: #fff;  
    text-decoration: none;  
    border-bottom: 1px solid #f0c;  
}  
  
a:hover {  
    border: none;  
}

Esto haría que los enlaces fueran de color blanco, y que la línea de subrayado fuese fucsia. Al pasar el cursor por encima, se eliminaría el subrayado. El pseudoelemento **hover** funciona para cualquier otro elemento de la web. No dudes en usarlo en celdas de tablas o capas.  
  
*A no ser que uses una versión antigua de un navegador subdesarrollado. . . Sí, ese.*  
Campos de formulario chulos  
  
Un efecto muy Web 2.0 es hacer los **campos de formulario** con fuente gigante y que el fondo del campo cambie cuando el usuario está usándolo. Esto se consigue con el pseudoelemento focus:

input, textarea {  
    font-size: large;  
    border: 3px solid #70C332;  
    color: #666;  
    background: #fff;  
}  
  
input:focus, text-area: focus {  
    background: #eee;  
}

Lo anterior pondría los campos de formulario input y textarea con una fuente grande y gris oscura, borde gordito de color verde y el fondo blanco. Al hacer foco (es decir, cuando el visitante está situado en ese campo), el fondo cambia a un color gris clarito.  
  
Blockquotes 2.0Otro legado de la Web 2.0: **blockquotes** con texto gigante y unas comillas gigantes. Quizás es algo difícil de imaginar con esta descripción, pero si echas un vistazo a la figura 1, seguro que ya lo has visto en algún sitio antes.  
Figura 1: Blockquote molón  
  
Un ejemplo de código podría ser este:

blockquote {  
    background: url (comillas.gif) top left;  
    background-repeat: **no**-repeat;  
    text-indent: 30px;  
    text-align: left;  
    font-size: x- large;  
    padding: 0px;  
}

Tendrás que ajustar los valores según la fuente y la imagen en sí. En muchos casos, es más conveniente utilizar padding y quitar el indentado.  
  
**Cambiar texto por imagen**  
  
Muchas veces, especialmente en headings, queremos **cambiar** el texto y en su lugar que aparezca una **imagen** bonita. Un ejemplo del código sería este (suponiendo que nuestra imagen mide 300x100 píxeles):

<!--XHTML-->  
**[<h3](http://december.com/html/4/element/h3.html" \t "_blank)** **id**="enlaces"**>[<span>](http://december.com/html/4/element/span.html" \t "_blank)**Enlaces**</span></h3>**  
  
/\* CSS \*/  
h3# enlaces {  
  width: 300 px;  
  height: 100 px;  
  padding: 0px;  
  background: url(links.png) top left no-repeat;  
}  
h3 span {  
  visibility: hidden;  
}

La etiqueta span es nueva, y no significa nada (literalmente). Es una etiqueta “vacía”, y sólo la usamos para conseguir ciertos efectos CSS. Como semánticamente no tiene ningún valor, hemos de evitar su uso todo lo posible.  
  
Lo que hace el código anterior es hacer coincidir las dimensiones de <h1> con las de la imagen. Después, gracias al <span> nos encargamos de que el texto del heading sea **invisible.** No obstante, aunque el texto sea invisible, todavía existe. Por eso, si el título es muy largo, es posible que sea más grande que la imagen que usamos de fondo, y es posible que nos descoloque el layout. En este caso, puedes cambiar la fuente de <h1> y hacerla enana.  
  
**CREAR UN LAYOUT CON**En este capítulo veremos **cómo** crear un layout sin tablas a dos columnas, muy típico en los blogs. Es bastante sencillo de seguir, y es un ejemplo de que dejar de usar tablas para maquetar no tiene por qué ser un proceso doloroso.  
  
Características  
  
Esta maquetación tiene las siguientes propiedades:  
  
Anchura fija  
Centrado  
2 columnas (una de ellas de barra lateral)  
Cabecera (header)  
Pie de página (footer)  
  
  
Puedes echarle una ojeada en la figura 1.  
  
  
Figura 1: Layout de 2 columnas  
  
**Código XHTML**  
  
El esqueleto del código XHTML de nuestra página es el siguiente (iría dentro del body):

**<div** **id**="container"**>**  
    **<div** **id**="header"**>**  
        **<h1>**Título**</h1>**  
  
        **<h2>**Subtítulo**</h2>**  
    **</div>**  
    **<div** **id**="sidebar"**>**  
        **<h3>**Sección**</h3>**  
        **<p>**Bla bla ...**</p>**  
    **</div>**  
    **<div** **id**="main"**>**  
  
        **<h3>**Sección**</h3>**  
        **<p>**Contenido principal**</p>**  
    **</div>**  
    **<div** **id**="footer"**>**  
        **<p>**Pie de página**</p>**  
    **</div>**  
**</div>**

Como ves, tenemos cuatro capas. Ahora veremos el código CSS de cada una de ellas.  
  
**#container**  
  
Esta capa es un **contenedor** para el resto de la página. Establece la anchura de todo y es la que centra el contenido. El truco está en usar auto dentro de margin para lograr el centrado. Esto no funciona en el IE, así que tendremos que echar mano de *text-align: center* en el body. El CSS completo es este:

body {  
    text-align: center;  
}  
#container {  
    width: 700px;  
    margin: 0px **auto**;  
    text-align: left;  
}

#sidebarEsta es la barra lateral tan famosa en los blogs. En otras páginas web, podemos poner ahí menús, publi, o cualquier otra cosa. La clave en esta capa es usar *float: left*, que la sitúa a la izquierda y hace que todos los demás elementos la rodeen. Aquí va el código

#sidebar {  
    width: 200px;  
    padding: 10px;  
    **float**: left;  
}

El padding no es obligatorio, pero lo he puesto para después poder explicar bien la siguiente capa. También es necesario especificar el **ancho** en píxeles. Ah, con este método **la barra no se extenderá hasta abajo**, sino que se corta en su final (esto se ve bien en la figura 13.1). Si pones la barra de otro color, y quieres que llegue hasta abajo, puedes usar la técnica de **Faux Columns**1.  
  
*[http://www.alistapart.com/articles/fauxcolumns/](http://www.alistapart.com/articles/fauxcolumns/" \t "_blank) Es un chanchullo sencillo e ingenioso. Básicamente consiste en poner una imagen de fondo que simule la columna.*  
  
**#mainEsta**  
  
es la capa donde irá el **contenido** (en un blog, esto serían los posts). Lo importante en esta capa es indicar con margin-left la **distancia** desde el borde del #container hasta esta capa, pasando por encima de la barra lateral.  
  
Nuestra barra lateral tiene 200 píxeles de ancho y 10 píxeles de padding a la izquierda y a la derecha. Si hacemos memoria del **modelo de caja**, el margen izquierdo que tenemos que indicar serían **200 + 10 + 10 + X**, donde **X** es la cantidad que nosotros queremos dejar de separación entre la barra lateral y el contenido principal. Para el ejemplo, pondremos 5 píxeles:

#main {  
    margin-left: 225px;  
}

**#footer**  
  
La capa del **pie de página** viene bien para poner información de copyright/copyleft2 o cualquier otro tipo de datos misceláneos. Aquí únicamente hemos de tener cuidado de “anular” el **ﬂoat**que tiene la barra lateral mediante un clear:2   
  
*Recordad, chicos, las licencias Creative Commons son buenas para la salud y para el karma.*

#footer {  
    clear: both;  
}

¡Ya tenemos nuestro propio layout tablas que cumple los estándares del W3C!  
  
Una de las cosas muy útiles que se han puesto de moda, es hacer que la cabecera de una web sea a su vez un enlace hacia la sección principal de una web. El hecho de que si nos perdemos tengamos un enlace bien gordo arriba del todo, es un salvavidas.  
  
**Código XHTML**  
  
Los estándares nos dicen que el **título** principal de la página se pone con la etiqueta <h1>. No obstante, el texto de esta etiqueta lo cambiaremos por una imagen. Como queremos además meterle un link, también usaremos la etiqueta <a>. Dentro del link, metemos una etiqueta <span> para poder ocultar con CSS el texto y que sólo se vea la imagen que hará de cabecera. . . pero nos estamos adelantando. El código XHTML sería este:

**<div** **id**=" header "**>**  
 **<h1><a** **href** ="..." **title** ="..."**>**  
 **<span>**Título**</span></a></h1>**  
**</div>**

**#header**Primero tocaremos el CSS de la capa **#header**. Establecemos el **ancho** y el **alto** de la capa con las medidas que tenga la **imagen**(400x100 en este caso). Además, ponemos la imagen de fondo y quitamos el margin y el padding:

#header {  
    background: url(bg.png) top left **no**-repeat;  
    width: 400px;  
    height: 100px;  
    margin: 0px;  
    padding: 0px;  
}

**El heading**  
  
También quitamos los márgenes a la etiqueta <h1> (para que la capa mida exactamente lo que la imagen) y hacemos invisible el contenido de la etiqueta <span>, para que el texto de la cabecera sólo se muestre en navegadores de texto o aurales:

#header h1 {  
 margin: 0px;  
}  
  
#header a span {  
 visibility: hidden;  
}

**El enlace**  
  
Ahora sólo nos falta la etiqueta <a>. Lo que queremos conseguir es que se pueda hacer clic en**todo el área** de la cabecera, no sólo en lo que sería el texto (invisible) del enlace. Para eso, tenemos que transformar el enlace en un **elemento de bloque**, y darle las medidas de la imagen  
  
Además, hemos de quitar los márgenes y el padding, así como la decoración del texto, para que no salga una línea de subrayado:

#header a {  
    width: 400px;  
    height: 100px;  
    display: block;  
    padding: 0px;  
    margin: 0px;  
    text-decoration: none;  
}

*Et voilà!* Ya tenemos nuestra cabecera cuca con enlace incorporado.  
  
Ahora vamos a aprender cómo modificar las **listas desordenadas** para que tengan **viñetas personalizadas**. Para ello, necesitaremos una imagen pequeñita que haga de viñeta. En Internet hay muchísimas para descargar, aunque puedes dibujar las tuyas propias. Supondremos que la imagen se llama *bullet.png*.  
  
**Código XHTML**  
  
Extremadamente simple. Se trata de una lista sin ordenar, sin más misterio:

**<ul>**  
    **<li>**Sonata Arctica**</li>**  
    **<li>**Nightwish**</li>**  
    **<li>**HIM**</li>**  
**</ul>**

**La lista**  
  
Ahora la parte de CSS. Disponemos de una propiedad que se llama *list-style-image*, pero da problemas con ciertas medidas de viñetas. Así que tenemos que guarrear un poco el código. Para el elemento *ul* sería:

ul {  
    padding-left: 10px;  
    margin-left: 10px;  
    list-style-type: none;  
}

Lo más importante es el *list-style-type: none*, que se encarga de quitar esas viñetas feas que pone el navegador por defecto. El *padding* y el *margin* es para sangrar la lista (puedes poner los valores que quieras, o incluso quitarlos).  
  
**Los ítems**  
  
¡Ya sólo queda poner nuestra viñeta! Para ello, modificamos el estilo de los *li* que estén en listas desordenadas:

ul li {  
    padding-left: 12px;  
    background: url(bullet.png) 0em 0.5em **no**-repeat;  
    margin-bottom: 1em;  
}

Lo primero que vemos es el padding} por la izquierda. Esto **no es el sangrado**, es una distancia de relleno que ponemos para que el texto del ítem **no esté encima** de la viñeta. Este valor lo tienes que modificar dependiendo de las dimensiones de la imagen de tu viñeta.  
  
Después, nos encontramos con la propiedad background}. Después de establecer la imagen, debemos indicar en qué **posición** (recuerda: primero horizontal, luego vertical) se encuentra la viñeta. ¡Aquí hay truco! Como ves, trabajamos con *em* y no con medidas absolutas en píxeles. ¿Por qué? Porque así nos vale para cualquier tamaño del texto.  
  
*Recuerda que 1em equivale a la anchura de la letra M.*  
  
Lógicamente, es muy difícil dar con el valor correcto a la primera, así que habrá que probar varias veces hasta topar con el que mejor quede. Ah, no se nos puede olvidar el i para evitar el**mosaico**.  
  
Por último, *margin-bottom* se encarga de establecer la **separación** entre un ítem y otro de la lista.Muy sencillo, ¿no?  
  
**MENUS VERTICALES**¿Sabías que con listas se pueden hacer **menús verticales** chulos? ¿Que **por qué listas**? Porque un menú vertical es una serie de elementos relacionados, y lo más semántico que podemos hacer es meterlo en una lista.  
  
El principal problema es que las listas son feas. En el capítulo anterior vimos cómo hacerlas más bonitas, pero quizás queramos algo un poco más diferente para nuestro menú. CSS nos da la solución, así que ya no tendrás excusa para hacer un menú en Flash.  
  
*Que, por cierto, es de lo peor que se te podría ocurrir. Si alguien no tiene instalado el plugin de Flash (nunca des por supuesto que el vistante tenga instalado un plugin), ni siquiera podrá navegar por tu página.*  
  
En el menú que haremos ahora, hacemos apaños con los colores de fondo y los bordes para conseguir efectos cuando el ratón pase por encima. Puedes ver cómo quedaría en la **figura 1**. Como va con CSS, podríamos fácilmente incorporar imágenes, pero eso queda como ejercicio.  
  
*No tiene gracia si te lo dan todo mascadito.*  
  
  
Figura 1: Menú vertical  
  
**Código XHTML**Al grano, necesitamos una lista tal que así:

**<div** **id**="menu"**>**  
 **<ul>**  
  **<li><a** **href**="..." **title**="..."**>**Home**</a></li>**  
  **<li><a** **href**="..." **title**="..."**>**Archivos**</a></li>**  
  **<li><a** **href**="..." **title**="..."**>**Enlaces**</a></li>**  
  **<li><a** **href**="..." **title**="..."**>**Acerca de**</a></li>**  
 **</ul>**  
**</div>**

**La lista**  
  
¡A meterle mano al CSS! Empezaremos primero con darle la **anchura** que queramos a la lista (que será la anchura del menú), poner una *fuente* maja y **quitar las viñetas y márgenes** de la lista.

#menu ul {  
    list-style-type: none;  
    margin: 0px;  
    padding: 0px;  
    width: 200px;  
    font-family: Arial, sans-serif;  
    font-size: 11pt;  
}

Ponemos a continuación un **color de fondo** para los ítems de la lista (*<li>*). Lo normal sería poner aquí los efectos de *hover*, para que se activen cuando el ratón pase por encima de todo el bloque, no sólo del texto del enlace; pero *Quien-Tú-Sabes* en sus versiones 6 e inferiores no soporta hovers en cosas que no sean un enlace, así que vamos a tener que emplear una artimaña más adelante. Por ahora, el CSS para el elemento *li* sería así de sencillo:

#menu ul li {  
    background-color: #666;  
}

**Enlaces**  
  
Y aquí viene el chanchullo del que os hablaba antes para hacer el hover: poner el enlace como si fuera un **bloque**, y así ocupará todo el *li* y podremos manipular sus dimensiones.  
  
*Haz memoria: es lo mismo que empleamos en un capítulo anterior para poner un [enlace en la cabecera de la web](http://www.cristalab.com/curso-css/cabecera/" \t "_blank).*

#menu ul li a {  
    color: #ccc;  
    text-decoration: none;  
    text-transform: uppercase;  
    display: block;  
    padding: 10px 10px 10px 20px;  
}

Sólo nos queda hacer los cambios para el **hover**:

#menu ul li a:hover {  
    background: #000;  
    border-left: 10px solid #333;  
    color: #fff;  
}

Si lo pruebas, verás que las letras se desplazan al hacer el hover, debido a que aparece el borde izquierdo. Si no os gusta este efecto, podéis añadir la línea siguiente al link cuando está normal. Lo que hace es poner un borde del mismo color que el fondo de los *li*, y así parece que no existe:

border-left: 10px solid #666;

Merece la pena trastear con esta técnica, puede dar lugar a menús muy logrados.