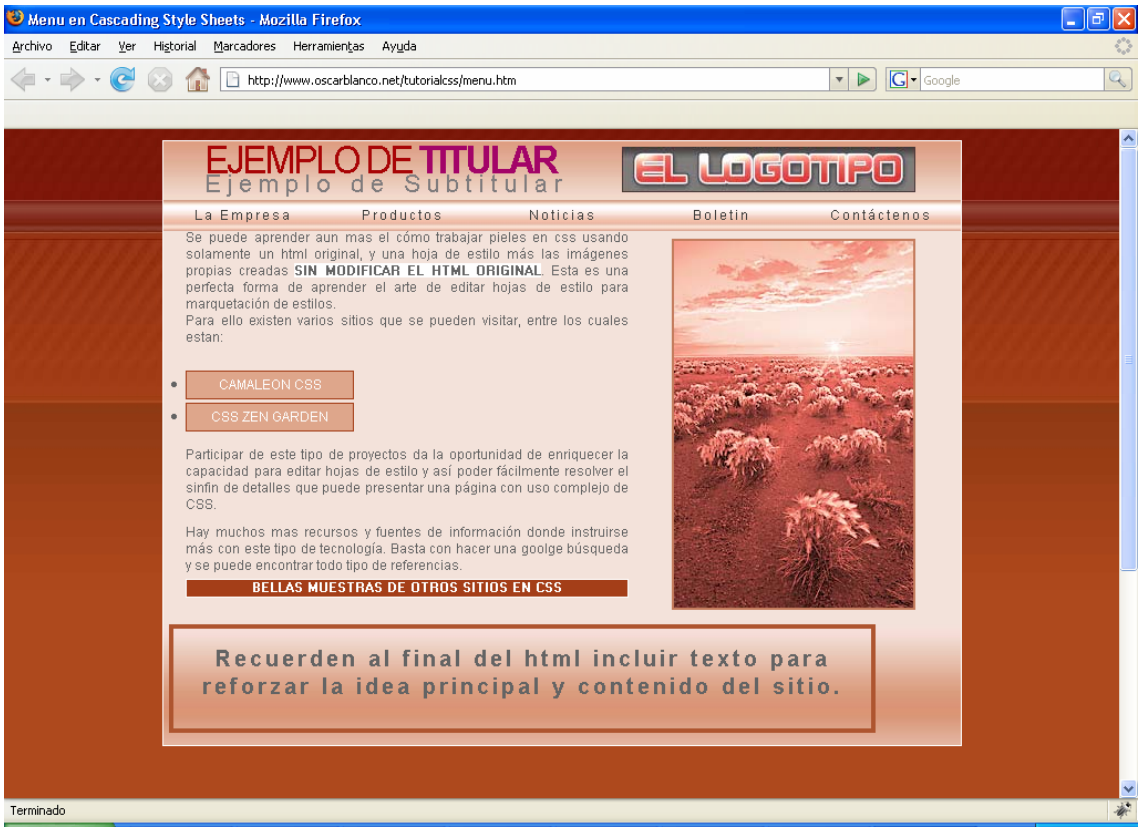
Tutorial de maquetación con CSS

Maquetación de un Sitio Web con CSS

Vamos a hacer esta página:
http://www.oscarblanco.net/tutorialcss/menu.htm



Vamos a echar un vistazo desnudando el html de su hoja de estilos usando el "botón" que les mostré como crear en su buscador en la primera parte de este tutorial.

Verán un contenido chorreado, constituido de h's, listas y texto en párrafos.

Verán el source de la página también.

Verán un source muy limpio y sencillo.

Bien, empezamos, lo primero es que obtengan el "kit" para trabajar este sitio.

Bájenlo AQUI

En el kit encontraran un html llamado "menu.htm" una hoja de estilos llamada "principal.css" y una carpeta llamada "images" que contiene todas las imágenes usadas para este tutorial.

Descompriman en alguna carpeta y estamos listos para empezar a configurar el css. Verán que "principal.css" en este momento solo contiene las clases e ID's ya establecidas pero sin parámetros.

Entonces el html, que ya tiene adjuntada esa hoja de estilos en realidad no despliega nada más que el texto desnudo. Tal y como lo va a leer cualquier buscador como google, altavista o yahoo.

De ahora en adelante no vamos a tocar más que la hoja de estilos, el html ya está con sus objetos definidos y las imágenes están recortadas y a tamaño. Sin embargo también les invito a que echen un vistazo a la carpeta de imágenes. Contiene dos imágenes verticales delgadas, que se repiten horizontalmente para fondos, una imagen grande que es meramente decorativa, un logo y un par de imágenes para el área de menú.

Viendo el source del html en detalle observarán que existe una estructura de jerarquías, igual que encontraríamos en una de tablas con sus td's y tr's.

Tenemos un DIV principal que se llama "alrededor", ése es solo un contenedor que nos ayuda a flotar todo el contenido centrado, por el tipo de diseño que escogí, pero casi siempre es bueno trabajar con contenedores principales y sus respectivos ***nested div's***, o div's encajados. En realidad este div llamado "alrededor" fue un agregado extra que metí a última hora tan solo para poder acomodar la foto grande que flota a la derecha del diseño y que Uds. pueden ver se encuentra al puro final del html, inclusive después de un texto que visualmente se encuentra luego de ella. Esto es simplemente pues siempre debemos buscar que el texto/contenido de nuestro html, sobre todo aquello cargado de palabras clave se encuentre antes que cualquier imagen, flash u otro elemento gráfico que cumple mera función decorativa.

Como decía, el DIV "alrededor" fue algo agregado a última hora por el capricho estético de poner esa imagen. El DIV llamado "principal" era el contenedor principal, y el que tenía todas las características para determinar el área principal de información. De una vez aclaro la diferencia entre un ID y un class.

ID's son usados más que todo para elementos únicos en la página, esto ayuda mucho si lo combinamos con programación, ya que los ID's son utilizados mucho en este medio precisamente porque son nombres únicos y así se puede dar comportamiento a un objeto específico. Las clases (class) en cambio son utilizadas más para elementos que se repiten varias veces en una página, por ejemplo, una clase para todos los anchors (links) de una página o sección determinada.

Para abrir la hoja de estilos notarán en la lista de clases a la izquierda que existen tres categorías principales, clases, ID's y otra llamada elementos. Ahí en este caso solo están "body" y un "*". Elementos son componentes o tags que por si solos ya existen como parte de la maquetación básica de un html. Ahí también caerían los h's, pero como notarán, los h's que he usado en este caso, están supeditados a ID's.

Qué es supeditado. Una clase, ID o elemento, pueden tener subclases supeditadas. Esta es una de las características que da tanta flexibilidad a la maquetación por hojas de estilos.

Por ejemplo, verán en la página que hay varios estilos de "hover" o sea, comportamiento de un hipervínculo al tener el puntero del ratón encima.

Si se fijan en la hoja de estilo, verán que hay varios "a" y "a:hover" supeditados a ID's (los estilos que empiezan con un símbolo de "#") en este caso.

Verán que el ID "arriba" o el ID "medio" se repiten varias veces con subclases a su lado: ul's li's p's e inclusive verán hasta un "a" "a:hover" y "strong" supeditados a su vez a una subclase.

O sea, hay un "strong" "supeditado" a un p del ID "medio".

Respiren hasta 3 veces...

¿Ok, listos? Seguimos...

Es importante tener mucho cuidado a la hora de trabajar estilos, en la medida de lo posible, eviten las mayúsculas, revisen siempre que los nombres tengan al principio su identificador si lo necesitan: "#" y "." para ID's y clases respectivamente.

Nunca olviden el punto y coma al final de cada parámetro. Nunca olviden cerrar llaves para cada estilo. Si es necesario comentar alguna línea o varias líneas abran el comentario con: /* y ciérrenlo con */ . Comentar es una buena forma de desactivar un parámetro sin tener que borrarlo.

Otra cosa que hay que tratar de hacer es evitar redundancias en nuestras hojas de estilo, o repeticiones innecesarias de parámetros en varias clases aplicados a DIV's "hermanos", pues cada línea de parámetros en nuestra hoja de estilos es una línea más de código que debe leer el buscador cuando indexa una página. Las hojas de estilos tienen la ventaja de centralizar despliegue de imágenes, para evitar que cada vez que se abra una página el explorador tenga que volver a cargarlas desde el html; sin embargo, las hojas de estilo también deben ser optimizadas, ya que muchas líneas de código hacen un documento lento de leer y pesado.

Una buena forma de evitar esto es tratar de descomponer el despliegue total en jerarquías y grupos. O sea, van a existir DIV's "padres" y estos tendrán sus propios DIV's "hijos". Esto lo que significa es que tendremos DIV's contenidos dentro de otros DIV's (como ya vimos en el html), y los DIV's que a nivel de jerarquía se encuentren al mismo nivel, serán "hermanos". Hablo en este momento de DIV's pero en realidad a lo que me refiero es contenedores, que también pueden ser SPAN's, H's, anchors, p's, li's, etc.

Una vez que hemos determinado esta jerarquía, entonces empezamos a analizar qué parámetros pueden heredarse de "padres" a "hijos".

Por ejemplo, en el BODY, que podríamos considerar el elemento "padre" para todos los contenedores de nuestra página, podemos colocar parámetros que vamos a heredar en todo el sitio, como por ejemplo la familia de fuentes, el tamaño default que queremos usar para fuentes, el alineamiento de texto, el color de texto, etc.

Hay parámetros que no se pueden heredar y otros que sí. Por ejemplo, padding y margin, no son heredables, pero existe un "hack" para también poder generalizar este tipo de parámetros.

Creamos un estilo llamado "*". Al usar un asterisco como nombre para un estilo, el explorador va a entender que por defecto a CUALQUIER estilo los parámetros que determinemos en la clase "asterisco" serán aplicados como suyos, esto se vuelve nulo al poner un parámetro específico en ese estilo único.

Veamos:

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

.vaca {
  color: #ff00ff;
  margin: 2px 3px 1px 2px;
}
```

En este caso, "asterisco" está diciendo que todos los estilos por defecto tendrán un margen de cero y un acolchonamiento de cero (esto es una buena práctica por cierto). Pero a su vez, "vaca" como estilo único que es, aparte de determinar un color único para textos dentro del contenedor que tengan la clase "vaca" aplicada, también tiene características únicas de

márgenes. Sin embargo, como no ha determinado sus parámetros de padding, va a tener por defecto los que ha predeterminado "asterisco" (margin = espacio alrededor de un contenedor, padding = espacio marginal de contenido a los bordes de su contenedor).

Otra cosa que aprovecho para mencionar, relacionado con optimización de estilos, es que en la medida de lo posible se use el "shorthand" (abreviación de parámetros en una sola línea).

Veamos:

```
.vaca {
  margin-top: 2px;
  margin-right: 4px;
  margin-bottom: 3px;
  margin-left: 5px;
  border-width: 2px;
  border-color: #BC6A47;
  border-style: solid;
}
```

podría optimizarse así:

```
.vaca {
  margin: 2px 4px 3px 5px;
  border: 2px #BC6A47 solid;
}
```

Al usar ***shorthand*** en el parámetro margin por ejemplo es importante saber que el orden en que los pongamos es esencial, el primero es el margen superior, le sigue el derecho, luego el de abajo y por último el izquierdo, o sea, empezando del de arriba se sigue a favor del reloj, así:

En el caso del borde, el orden en que determinemos grosor, color y estilo no es importante. Si quisiéremos un borde con grosores variables, entonces tendríamos que hacer un parámetro por aparte para "border-width" y podemos usar un ***shorthand*** igual al de "margin" siguiendo las manecillas del reloj, lo mismo va para el "padding".

Bien. Empezamos a llenar la hoja de estilos. A estilo asterisco, vamos a ponerle padding y margin de cero. Así:

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

Por cierto, "0" equivale a cero de cualquier tipo de medida, sean pixels u otro, por eso un shorthand se puede ver así: margin: 1px 0 1px 0;

Inclusive, si luego del primer valor todos son iguales, digamos, todos de cero, entonces se podría ver así: margin: 3px 0; Aquí estamos diciendo que los márgenes de derecha, abajo e izquierda todos tienen un valor de cero.

Bien, como ya vimos antes, al poner estos valores para margin y padding en "asterisco" hemos predeterminado que todos los estilos tendrán padding y margin de cero por defecto.

Un respiro, cuenten hasta 3 y vuelvan a leer.

¿Ok, listos? Seguimos.

En orden de jerarquía, vamos al estilo para el BODY.

```
body {
background-color: #AE491D;
background-image: url(images/fondo-principal.jpg);
background-position: top;
background-repeat: repeat-x;
color: #646464;
font-family: Arial, Helvetica, sans-serif;
font-size: 12px;
height: 100%;
text-align: center;
}
```

Veamos, traduciendo dice esto:

Color de fondo: #AE491D;
Imagen de fondo: fondo-principal.jpg (incluyendo su "path", que sea relativo);
La posición de la imagen de fondo es "arriba";
La imagen de fondo se repite en "x";
El color usado para el texto es: #646464;
La familia de tipografías es: Arial, con equivalencias a Helvetica y a san-serif (en el caso de que el tipo Arial no sea reconocido);
El tamaño de los tipos por defecto es de 12px;
El alto de este estilo es de 100% del área visible del buscador;
El alineamiento por defecto del texto será centrado (esto incluye objetos flotantes como DIV's).

Bien, asignen este estilo a su BODY, guarden y ahora refresquen la página en su buscador. Ya va tomando forma, aunque hay muchas cosas tiradas por todo lado.

Seguimos con el DIV padre de todos. El que tiene el ID llamado "alrededor".

Éste es el que nos va a permitir que el rectángulo de este diseño de página flote siempre al centro del área visible de nuestro buscador. Ah, un detalle importante. ¿**Cross-browser**... han escuchado esa fatídica palabra compuesta antes? Significa, mi página será vista por la mayor cantidad posible de buscadores (Internet Explorer, Firefox, Netscape, etc) y plataformas y los usuarios verán exactamente lo mismo o al menos casi la misma cosa. Para garantizarse esto, es bueno que se acostumbre a trabajar con el Firefox como buscador por defecto, y el Internet Explorer como secundario (esto es solo un consejo a nivel personal).

Existen "hacks" para solventar esto, pero para eso también los invito a usar google.

Bien, veamos el estilo "alrededor":

```
#alrededor {
height: 545px;
margin: 10px auto;
width: 720px;
float: left;
}
```

Traducción:

Altura del contenedor es de 545 pixels;

Márgenes alrededor del contenedor: 10 pixels arriba, y el resto automático. Aquí tenemos un "hack" pero es para el Firefox en este caso, si no ponemos el auto como segundo valor, nuestro div no le va a dar la gana de flotar centrado.

Ancho del contenedor es de 720 pixels.

Y por último este contendor va a flotar a la izquierda. O sea, si tuviésemos por ejemplo 10 contenedores de 100 pixels de ancho cada uno, todos con float: left, todos se acomodarían lado a lado flotando hacia la izquierda. Imagínense que inclinan el monitor hacia la izquierda y todos esos div's caen en esa dirección lado a lado.

Ahora vamos con el ID "principal":

```
#principal {
background-image: url(images/fondo-areachica.jpg);
background-position: top;
background-repeat: repeat-x;
border: 1px #FFFFFF solid;
height: 545px;
width: 720px;
}
```

Todos estos parámetros ya los hemos cubierto antes, solo cambian medidas y colores...

Sigamos para abajo en la jerarquía, existen tres div's hermanos: Arriba, Medio y Abajo. Los he nombrado así debido a que como cualquiera que había trabajado en su oscuro pasado con frames, me ha quedado esa costumbre de nombrar así para dividir páginas...

Veamos "arriba" en el html. Este DIV contiene un h1, un h2, un h3 y una lista ordenada (ul, li's). Aquí tenemos oportunidad de heredar si fuera el caso del div contenedor a todos sus sub-elementos.

Pero en este ejemplo no lo hice.

Observemos "arriba" con todas sus subclases de una vez:

```
#arriba h1 {
color: #A80506;
float: left;
font-size: 34px;
font-weight: normal;
height: 29px;
left: 37px;
letter-spacing: -3px;
line-height: 26px;
position: relative;
text-align: left;
text-transform: uppercase;
top: 3px;
width: 340px;
}

#arriba h1 strong {
color: #A30468;
}

#arriba h2 {
background-image: url(images/logotipo.jpg);
color: #494846;
float: left;
```

```
font-size: 4px;
height: 41px;
left: 74px;
position: relative;
top: 5px;
width: 265px;
}

#arriba h3 {
clear: both;
color: #807C7D;
float: left;
font-size: 22px;
font-weight: normal;
height: 22px;
left: 37px;
letter-spacing: 6px;
line-height: 18px;
margin: -12px 0;
position: relative;
text-align: left;
width: 340px;
}

#arriba li {
display: block;
float: left;
height: 25px;
width: 144px;
}

#arriba li a{
color: #403F3D;
display: block;
float: left;
height: 27px;
letter-spacing: 2px;
padding: 6px 0;
text-decoration: none;
width: 144px;
}

#arriba li a:hover {
background-image: url(images/menu-on.jpg);
background-repeat: repeat-x;
color: #FFFFFF;
display: block;
}

#arriba ul{
background-image: url(images/menu-off.jpg);
background-repeat: repeat-x;
clear: both;
display: block;
float: left;
height: 28px;
position: relative;
top: 12px;
width: 720px;
}
```

Traduzcamos los parámetros que aún no hemos visto.

En #arriba h1 vemos aparte de cosas obvias como tamaño de letra y estilo de letra (normal), un parámetro llamado "position" (position: relative;). Este parámetro permite que el contenedor que ya le dimos características de flotabilidad (float: left) además pueda posicionarse en un lugar específico. Posición relativa significa que estará relativa a los márgenes superior e izquierdo de su contenedor padre. O sea, h1 se posicionará relativo con las medidas de "left" y "top" (left: 37px; top: 3px;), definiendo a cuánto se encuentra h1 de los márgenes superior e izquierdo de "arriba".

Otros parámetros nuevos que tenemos son "text-transform: uppercase;" que en este caso convierten todo el texto en mayúsculas y "letter-spacing: -3px;" que en este caso aplica una distancia de -3 pixels entre cada letra (por eso se ven todas pegaditas).

Podemos ver que al STRONG del h1 (el estilo que sigue) le aplicamos otro color, por eso la palabra "titular" tiene otro tono. Fíjense en el html, ni h1 ni strong tienen clases ni ID's aplicados, esto es porque sus propios nombres de elemento son subclases del ID llamado "arriba".

Lo mismo sucede con los otros h's y con la lista ordenada.

Pueden bajar ya la hoja de estilos llena de AQUI.

Otros parámetros nuevos que pueden buscar ahí son "display" y "text-decoration".

El primero lo uso para cuando quiero que un link se trate como un objeto que puede tener alto y ancho.

Por ejemplo, los links dentro de los li's de la lista ordenada con “text-decoration: none;” es la forma de eliminar la rayita por debajo para un link.

Espero que todo esto les sirva.

Ejemplos de uso de display

La propiedad `display` es una de las propiedades CSS más infrautilizadas. Aunque todos los diseñadores conocen esta propiedad y utilizan sus valores `inline`, `block` y `none`, las posibilidades de `display` son mucho más avanzadas.

De hecho, la propiedad `display` es una de las más complejas de CSS 2.1, ya que establece el tipo de la caja que genera cada elemento. La propiedad `display` es tan compleja que casi ningún navegador es capaz de mostrar correctamente todos sus valores.

El valor más sencillo de `display` es `none` que hace que el elemento no genere ninguna caja. El resultado es que el elemento desaparece por completo de la página y no ocupa sitio, por lo que los elementos adyacentes ocupan su lugar. Si se utiliza la propiedad `display: none` sobre un elemento, todos sus descendientes también desaparecen por completo de la página.

Si se quiere hacer un elemento invisible, es decir, que no se vea pero que siga ocupando el mismo sitio, se debe utilizar la propiedad visibility. La propiedad `display: none` se utiliza habitualmente en aplicaciones web dinámicas creadas con JavaScript y que muestran/ocultan contenidos cuando el usuario realiza alguna acción como pulsar un botón o un enlace.

Los otros dos valores más utilizados son `block` e `inline` que hacen que la caja de un elemento sea de bloque o en línea respectivamente. El siguiente ejemplo muestra un párrafo y varios enlaces a los que se les ha añadido un borde para mostrar el espacio ocupado por cada caja:

Lorem ipsum dolor sit amet, consectetuer adipiscing elit.

Lorem ipsum Donec mollis nunc in leo Vivamus fermentum

Como el párrafo es por defecto un elemento de bloque (*"block element"*), ocupa todo el espacio disponible hasta el final de su línea, aunque sus contenidos no ocupen todo el sitio. Por su parte, los enlaces por defecto son elementos en línea (*"inline element"*), por lo que su caja sólo ocupa el espacio necesario para mostrar sus contenidos.

Si se aplica la propiedad `display: inline` al párrafo del ejemplo anterior, su caja se convierte en un elemento en línea y por tanto sólo ocupa el espacio necesario para mostrar sus contenidos:

[display: inline] Lorem ipsum dolor s, consectetuer adipiscing elit. Lorem ipsum Donec mollis nunc in leo Vivamus fermentum

Para visualizar más claramente el cambio en el tipo de caja, el siguiente ejemplo muestra un mismo párrafo largo con `display: block` y `display: inline`:

[display: block] Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Sed non sem quis tellus vulputate lobortis. Vivamus fermentum, tortor id ornare ultrices, ligula ipsum tincidunt pede, et blandit sem pede suscipit pede. Nulla cursus porta sem. Donec mollis nunc in leo.

[display: inline] Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Sed non sem quis tellus vulputate lobortis. Vivamus fermentum, tortor id ornare ultrices, ligula ipsum tincidunt pede, et blandit sem pede suscipit pede. Nulla cursus porta sem. Donec mollis nunc in leo.

De la misma forma, si en los enlaces del ejemplo anterior se emplea la propiedad `display: block` se transforman en elementos de bloque, por lo que siempre empiezan en una nueva línea y siempre ocupan todo el espacio disponible en la línea, aunque sus contenidos no ocupen todo el sitio:

Lorem ipsum dolor sit amet, consectetuer adipiscing elit.

[display: block] Lorem ipsum .

[display: block] Donec mollis nunc in leo .

[display: block] Vivamus fermentum .