Tutorial de CSS: Cascading Style Sheets

Contenidos

Presentación

Bienvenido a este tutorial de CSS, el lenguaje de hojas de estilo usado en páginas web. Este tutorial pretende hacer una introducción a este lenguaje, métodos para separar estilo de contenido, etc.
Se trata de introducir los conceptos, propiedades y estilos básicos de este lenguaje para que después cada uno haga sus propios diseños.

Normalmente la mayoría de manuales de creación y diseño de páginas web que hay en Internet, explican sistemas y métodos de HTML para dar formato tales como las etiquetas `<font>`, diseño con tablas, etc.

Con la llegada del CSS no sólo estos "antiguos" métodos están "desfasados", sino que son bastante más complicados de utilizar y mucho menos **accesibles** (como para navegadores en modo texto, reproductores orales, robots...).

Lo ideal sería que en los lenguajes de estructuración como HTML no incluyésemos nada acerca del estilo de los elementos, sino de su estructura.
Es decir, en el documento HTML en lugar de decir "esto lo quiero a tamaño 24 y en cursiva, esto en rojo" lo mejor es estructurar el documento: "esto es un encabezado, esto una lista ordenada, esto un párrafo" **sin incluír ningún elemento de diseño y presentación**.
Despues, creamos un documento CSS en el que **sí** "diseñamos" cada parte del documento, incluyendo colores, posición, bordes y demás adornos.

Las ventajas son evidentes: un navegador o sistema en modo texto o que no soporte CSS, se quedará con el documento HTML con el contenido estructurado.
Pero los que soporten CSS podrán ver el documento con todo sus estilos, adornos, etc.

También hay más ventajas, como el hecho de poder incluír la misma hoja de estilos en varias páginas HTML, lo que es muy cómodo y útil.
De esta manera podremos —por ejemplo— cambiar la fuente de los párrafos de dieciocho páginas con sólo editar un archivo; cosa que sería mucho más compleja si hubiésemos usado las viejas etiquetas como `<font>`.

Este manual asume que se tienen conocimientos básicos en cuanto a HTML, estructuración de contenidos, párrafos, etc.
De todas formas también veremos algunos enlaces interesantes sobre manuales de HTML, y más.

Antes de Empezar

Antes de empezar con CSS deberías tener por lo menos unos conocimientos básicos de HTML.

De entre las versiones varias de HTML yo te recomiendo el XHTML ya que es el "futuro" del HTML y el más claro.
Mira los siguientes enlaces, altamente recomendados.

- Tutorial de XHTML, por BenKo
- HTML correcto: cómo hacer buenas páginas web, por Daniel Clemente

Documentos Estructurados

Para poder aplicar cómodamente un estilo CSS a un documento HTML, éste —como hemos visto antes— debe estar bien estructurado; es decir, cabeceras, párrafos y demás.

Como en este manual se parte de la base de que ya se tienen conocimientos de HTML, sólo veremos un ejemplo:

```
<?xml version="1.0" encoding="utf-8" ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="es" lang="es">
<head>
<title>Mi documento</title>
</head>
<body>
<h1>Encabezado del documento</h1>
<h2>Segundo encabezado</h2>
<div id="contenido">
<p>Esto es un párrafo</p>
<p>Esto otro párrafo</p>
</div>
<div id="final">
<p>Final del documento</p>
</div>
</body>
</html>
```

Como se puede ver en la parte de color verde, hemos separado encabezados, párrafos y marcado las secciones "contenido" y "final".
Ésto nos servirá para diferenciarlas a la hora de aplicarle un CSS.

Si quieres puedes ver cómo quedaría. Como vés, el navegador le aplicará a la estructura una serie de estilos "predeterminados" que por supuesto podremos anular o modificar con CSS.

El estándar CSS y navegadores

CSS es un estándar-recomendación libre del W3C; con unas especificaciones que los navegadores que soporten CSS deberían de implementar.

Sin embargo algún navegador como Microsoft Internet Explorer se salta "a la torera" algunas de estas especificaciones y a veces hace lo que "le dá la gana" en vez de lo que debería hacer según el estándar.

Es por esto por lo que no es nada recomendable diseñar CSS basándose en el resultado que se obtiene en este navegador, ya que si te fías de él probablemente estés en realidad creando unos verdaderos churros.

Por supuesto sí es posible hacer CSSs que se vean bien tanto en el IE como en los navegadores que respetan los estándares, pero a veces hay que recurrir a "trucos" o cosas raras o no-estándares.

Aparte, el IE es un navegador obsoleto. Por ejemplo no soporta XHTML por lo que por ejemplo no podrás leer este tutorial desde el IE.

Cualquier navegador decente (Mozilla, Safari, Opera, etc.) te servirá perfectamente, aunque yo te recomiendo Mozilla Firefox por ser multiplataforma, ligero y soporta muy bien el CSS.

Qué NO es CSS

Si lo que pretendes es cambiar el color de la barra de desplazamiento y otras idioteces, que sepas que no son **para nada** estándar ni CSS, aparte de molestar y de dar mala imágen a la página.

Después de estos capítulos de teoría ya podemos empezar con la "práctica" :-)

Empecemos

Bien, vamos a ver la sintaxis básica de CSS y qué podemos hacer con él.

Antes de nada, decir que un archivo CSS es independiente del archivo HTML, y suele tener extensión `.css`.

Para incluír un archivo CSS a una página HTML o XHTML se incluye un código como este en `<head>`:

```
<link rel="stylesheet" title="Nombre del CSS" type="text/css" href="estilo.css">
```

Si estás usando XHTML recuerda cerrar la etiqueta `<link>`

Bien, ahora veremos cómo hacer los archivos .css:

Redefinir una etiqueta HTML

Si lo que queremos es dar formato o redefinir una etiqueta HTML, ésta es la sintaxis:

```
etiqueta {
<estilos CSS>
}
```

Como ves, los contenidos se agrupan entre llaves o corchetes :-) En "etiqueta" pondríamos el nombre de la etiqueta (por ejemplo "p", "div"...) pero sin los signos <> .

También podemos redefinir varias etiquetas a la vez, separándolas por comas:

```
etiquetauno,etiquetados,etiquetatres {
<estilos CSS>
}
```

Redefinir una etiqueta "hija" de otra etiqueta

Esto nos sirve para definir etiquetas que son "hijas" (es decir, que dependen de y están dentro de otras etiquetas como el caso de "li" que puede estar dentro de "ol" y "ul").

```
padre>hija {
<estilos CSS>
}
```

Claro; en este caso se aplicarían los estilos a las etiquetas "hija" sólo si son "hijas" de "padre".

Redefinir etiquetas "dentro" de otras etiquetas

Este caso muy parecido al anterior, serviría para aplicar estilos CSS a "etiqueta" sólo si está dentro de "contenedor". (como por ejemplo un `span` dentro de un `p`, etc.).

```
contenedor etiqueta {
<estilos CSS>
}
```

Identificadores y Clases

En el apartado anterior veíamos como redefinir etiquetas HTML, pero a veces tenemos varias etiquetas del mismo tipo pero queremos aplicar diferentes estilos según donde estén.

Para esto usamos los **identificadores** y las **clases**.

La principal diferencia entre ellos es que los IDs tienen que ser únicos en todo el documento HTML mientras que las clases pueden repetirse en varias etiquetas.
Los IDs se suelen usar mucho con etiquetas "neutras" como `div` y `span` para marcar las diferentes partes del documento y después aplicar diferentes estilos a cada una.

Incluír IDs y clases en un documento HTML

Se hace con los parámetros `id` y `class` respectivamente que se pueden aplicar a cualquier etiqueta:

```
<div id="capitulodos">
<p>Párrafo uno</p>
<p class="parrafoverde">Párrafo dos</p>
</div>
```

En este ejemplo "capitulodos" sería una sección **única** marcada en el documento en la cual podemos aplicar un estilo concreto; y el estilo de la clase "parrafoverde" se aplicaría a esa

etiqueta "p" en este caso pero podríamos aplicarlo a más etiquetas si queremos.

Aplicar estilos a estos IDs y clases

Para aplicar estilos CSS a **identificadores** esta es la sintaxis CSS:

```
#identificador {
<estilos CSS>
}
```

Como vemos, el nombre del identificador se precedería por una almohadilla (#) y los estilos CSS se aplicarían a la sección del documento con ese identificador.

Por supuesto podemos combinar esto con lo que hemos visto en el capítulo anterior :-) . Por ejemplo, para aplicar un estilo en concreto a las etiquetas "etiqueta" dentro del ID "identificador":

```
#identificador etiqueta {
<estilos CSS>
}
```

Para aplicar estilos a **clases** es parecido pero con un punto (.) en vez de almohadilla. Por ejemplo:

```
.clase {
<estilos CSS>
}
```

Con eso aplicaríamos los estilos a las etiquetas con clase "clase".

También podemos hacer como antes, combinar lo que hemos visto en el capítulo anterior.
Y además también podemos aplicar los estilos de la clase sólo a una determinada etiqueta:

```
etiqueta.clase {
<estilos CSS>
}
```

En este caso **sólo** se aplicaría el estilo a las etiquetas "etiqueta" con clase "clase". No a otras etiquetas aunque tengan la misma clase.

Empezando con Estilos

Hasta ahora hemos visto cómo redefinir e incluír estilos, pero nos hemos limitado a poner `<estilos CSS>` donde deberían ir las definiciones de los estilos.

Ahora veremos qué estilos podemos definir, empezando por los básicos :-)
Antes de nada, decir que la sintaxis para estos atríbutos es:

```
atributo: valor;
```

Los atributos siempre se separan por punto y coma, y despues del nombre se pone dos puntos (no igual, es un error típico al confundirse con el HTML).

Texto: fuente, familia, color, decoración, estilo...

Bien, empecemos por la **fuente**.
La forma básica de declarar un tipo de letra es:

```
font-family: <fuente>;
```

Bien, ahora veremos qué podemos poner en `<fuente>`. Antes de nada debes saber que las fuentes se dividen principalmente en tres clases: `serif` (las fuentes "con serifa" al estilo Times, Georgia...); `sans-serif` (las fuentes "sin serifa" al estilo Arial, Helvetica, Geneva...); y `mono` que son las de anchura fija como Courier o Monaco (ideales para código fuente por ejemplo).

A la hora de definir una o más fuentes con las que queremos que se vea un texto, siempre debemos dar como última alternativa uno de los tres grupos genéricos; para que en caso de que el visitante no tenga instalada la fuente (o fuentes) específicas que nos gustaría, su navegador escoja como alternativa una del mismo tipo que sí tenga instalada.

Bien, para definir las fuentes que queremos: si son varias fuentes se separan con comas; y si su nombre contiene espacios se pone entre comillas (esto es aplicable a todos los atributos). Ejemplo:

```
font-family: Georgia, "Book Antiqua", Palatino, Times, "Times New Roman", serif ;
```

En este ejemplo, el texto se vería con Georgia; si ésta no está disponible con Book Antiqua; si ésta no está disponible con Palatino, y así sucesivamente hasta llegar al caso en que ninguna fuente de las que hemos recomendado esté disponible. En ese caso se utilizaría una predeterminada del grupo "serif".

Bien, ahora veremos cómo definir el **color**.

```
color: <color>;
```

Los colores en CSS se pueden definir de varias formas:

- **Hexagesimal**: `#RRGGBB`. Se define con una almohadilla seguida de las cantidades de color para **rojo**, **verde** y **azul**.
  Las cantidades se expresan en hexagesimal (es decir, dos dígitos de 0-9 a A-F) . De esta manera negro es `#000000`, rojo es `#FF0000`, gris oscuro es `#333333` y blanco `#FFFFFF`.
  Existen programas que te calculan estas cantidades a partir de un color que tú le das.
- **Hexagesimal abreviado**: `#RGB`. Si en el caso anterior los dos dígitos para rojo, verde y azul son los mismos (por ejemplo DD, BB o 22) se puede abreviar dejando sólo uno.
  De esta manera negro es `#000`, rojo es `#F00`, gris oscuro es `#333` y blanco `#FFF`.
- **Combinaciones predefinidas en inglés**: Existen una serie de colores simples que ya vienen predefinidos y que podemos usar con sus nombres en inglés.
  De esta manera el negro es `black`; rojo es `red`, gris oscuro es `gray`, y blanco `white`.
- **Cantidades de color en RGB**: `rgb(ROJO, VERDE, AZUL)`. Con esta función podemos indicar el color directamente con sus cantidades de rojo, verde y azul; con números del 0 al 255 (máximo).
  De esta manera el negro es `rgb(0,0,0)`; rojo es `rgb(255,0,0)`, gris oscuro es `rgb(100,100,100)` y blanco `rgb(255,255,255)`.

Para poner un ejemplo de esto, el color granate:

```
color: #A00000;
```

Bien; ahora veamos como modificar el **tamaño** de la fuente:

```
font-size: <tamaño>;
```

CSS nos dá mucha libertad a la hora de especificar tamaños, ya que éstos los podemos expresar en muchas medidas: por ejemplo `px`, `pt`, `em`, `cm`, `mm`... :-)
Ejemplo:

```
font-size: 16px;
```

Bien, pasemos ahora a la **decoración** que le podemos dar a un texto.

El primer atributo que veremos para decorar nuestros textos es:

```
text-decoration: <decoración>;
```

Donde `<decoración>` puede valer lo siguiente:

- **`underline`**: Subraya nuestro texto. Ejemplo
- **`overline`**: Línea por encima de nuestro texto. Ejemplo
- **`line-through`**: Tacha nuestro texto. Ejemplo
- **`none`**: Modo normal, sin subrayar, sin línea por encima y sin tachar. Aunque este modo es el predeterminado en algunas etiquetas, en otras como `<a>` el modo predeterminado es `underline` por lo que podemos ponerlo a `none` si no queremos subrayar los enlaces. Ejemplo

El segundo atríbuto básicamente nos permite poner textos o en cursiva o en oblícuo:

```
font-style: <estilo>;
```

Donde `<estilo>` puede valer lo siguiente:

- **`italic`**: Pone el texto en cursiva. Ejemplo
- **`oblique`**: Pone el texto en oblícuo (casi idéntico a la cursiva). Ejemplo
- **`normal`**: Modo normal, no cursiva ni oblícuo. Ejemplo

Ahora pasemos al **grosor**.

```
font-weight: <grosor>;
```

Donde `<grosor>` puede valer lo siguiente:

- **`bold`**: La típica negrita. Ejemplo
- **`bolder`**: Más grueso que la típica negrita. Ejemplo
- **`lighter`**: Ligero. Ejemplo
- **Un número del 100 al 900**: Diferentes valores desde el mínimo (100) al máximo (900). Ejemplo (valor 100)
- **`normal`**: Grosor normal. Ejemplo

También podemos especificar una "variante" de la fuente. Aunque sólo hay una: las versales (aparte del normal).

```
font-variant: small-caps;
```

Ejemplo

Sigamos con el formateado de textos. Con CSS podemos especificar el **tamaño entre letras**.

```
letter-spacing: <tamaño>;
```

El tamaño (al igual que en todos los tamaños en CSS) se especifica de la misma forma que

como veíamos en `font-size`.

```
letter-spacing: 5px;
```

Ejemplo

También podemos especificar el tamaño entre palabras:

```
word-spacing: <tamaño>;
```

En HTML para "indentar" o espaciar un texto teníamos que tirar de cosas como ` `.

Con CSS podemos usar `text-indent`.

```
text-indent: <tamaño>;
```

Como siempre, los tamaños se especifican igual que antes. Además en este atributo también le podemos dar un valor en porcentaje con respecto al elemento contenedor.

```
text-indent: 3cm;
```

Ejemplo

Con CSS también podemos **transformar** las mayúsculas y minúsculas de los textos.

```
text-transform: <transformación>;
```

Donde `<transformación>` puede valer lo siguiente:

- **`uppercase`**: Todo a mayúsculas
- **`lowercase`**: Todo a minúsculas
- **`capitalize`**: La primera letra de cada palabra a mayúsculas
- **`none`**: Sin transformación (predeterminado)

Otro atributo que podemos especificar para los textos es el **interlineado**, es decir, el tamaño entre las líneas de un texto:

```
line-height: <tamaño>;
```

`<tamaño>` se expresa como hemos visto siempre :-)

Alineado de Textos

No hace falta usar `align="center"` ni cosas parecidas para nada. Con CSS tenemos mucho más control:

```
text-align: <alineado>;
```

Donde `<alineado>` puede valer `left`, `right`, `center` o `justify`.

Por ejemplo si aplicamos este estilo a un párrafo...

```
text-align: center;
```

... el texto del párrafo estará centrado :-)

El fondo de un elemento

Olvídate de atributos HTML tales como `bgcolor`: en CSS tenemos mucho más control sobre el fondo de los elementos:

Para cambiar el **color** de fondo:

```
background-color: <color>;
```

Los colores se especifican igual que cuando veíamos el atributo `color`. Ejemplo:

Párrafo con color de fondo rgb(200,200,255)

También podemos especificar el color `transparent`.

Con CSS podemos poner como fondo un gráfico:

```
background-image: <imágen>;
```

Las imágenes en CSS se expresan con la función URL:

```
url("imagen.jpg")
```

Por ejemplo:

```
background-image: url("/pic/fondo.png");
```

Recuerda que si la imágen es un PNG con trasnparencia alpha podrás conseguir efectos impresionantes :-)

Para conseguir que el gráfico de fondo se quede fijo en el sitio y no se mueva con el scroll no hace falta utilizar el infame `bgproperties="fixed"` propietario de Microsoft:

```
background-attachment: fixed;
```

Además con CSS tenemos más control: normalmente cuando ponemos un gráfico de fondo en un objeto; si este es más pequeño que el objeto se irá repitiendo en mosaico hasta llenar todo el objeto.

Con CSS podemos controlar esta repetición:

```
background-repeat: <modo>;
```

Donde `<modo>` puede ser uno de los siguientes:

- **`no-repeat`**: Simplemente cuando acabe el gráfico no lo repite, de manera que el resto del objeto queda sin fondo. También si especificamos un color de fondo aparte de la imágen, donde no cubra la imágen se verá el color.
- **`repeat-x`**: Se repite la imágen horizontalmente pero no verticalmente.
- **`repeat-y`**: Se repite la imágen verticalmente pero no horizontalmente.
- **`repeat`**: Se repite tanto horizontalmente como verticalmente.

También podemos especificar dónde queremos que empiece la imagen:

```
background-position: <posición>;
```

En posición podemos expresar dos medidas separadas por espacio. La primera es las coordenadas X y la segunda las coordenadas Y.

Podemos expresar las medidas en unidades (como hemos visto siempre), porcentajes con

respecto al contenedor; o palabras como `top`, `bottom` `left` y `right` que hacen referencia a las distintas esquinas de la pantalla.

Ejemplos:

```
background-position: 14px 29px;
background-position: top right;
```

Bordes

Con CSS podemos especificar **bordes** a los elementos de todo tipo.

En este apartado vamos a ver la sintaxis abreviada para incluír bordes. Con esta sintaxis tenemos cuatro atributos:

```
border: <tipo> <grosor> <color>;
border-top: <tipo> <grosor> <color>;
border-bottom: <tipo> <grosor> <color>;
border-left: <tipo> <grosor> <color>;
border-right: <tipo> <grosor> <color>;
```

El primer atríbuto hace referencia al borde general del objeto (los cuatro lados) y los siguientes hacen referencia a lados en concreto.

El orden de los valores no tiene porque ser ese, puede ser cualquier orden e incluso podemos omitir valores (estos tomarán el valor predeterminado).

En `<tipo>` ponemos el tipo de borde que queremos. Puede ser uno de los siguientes:

- **`solid`**: Un borde sólido, es decir, una línea.
- **`dashed`**: Un borde "rayado", con línea discontínua.
- **`dotted`**: Un borde hecho a partir de puntos.
- **`double`**: Dos líneas sólidas.
- **Bordes 3D**: Personalmente nada recomendados, pero son: `groove, ridge, inset, outset`. Experimenta con ellos si quieres pero...
- **`none`**

En cuanto a `<grosor>` y `<color>` se especifican como hemos visto hasta ahora: grosor en unidades (cm, px...) y color como siempre :-)

Veamos un ejemplo, aplicando algo de lo que hemos aprendido:

```
.miborde {
border-top: solid 2px #a00000;
border-bottom: outset 3px #a00000;
border-left: dotted 2px #a00000;
border-right: dashed 2px #a00000;
font-size: 20px;
color: #a00000;
font-variant: small-caps;
}
```

Ahora sólo tenemos que aplicar la clase a un objeto para ver el resultado:

Así Quedaría El Efecto

Bonito, ¿eh? :-)

Pseudoclases

En CSS existen unas clases especiales que se llaman **pseudoclases** que afectan a comportamientos especiales como pasar el ratón por encima, etc.

Para definir una pseudoclase:

```
etiqueta:pseudoclase {
<Formatos CSS>
}
```

Como vemos se ponen dos puntos y después el nombre de la pseudoclase predefinida.

- **`hover`**: Esta pseudoclase se activa mientras el ratón está por encima del objeto. Generalmente se aplica a enlaces y formularios.
- **`visited`**: Esta pseudoclase se activa en los enlaces que ya han sido visitados.
- **`link`**: Enlaces en estado normal (no visitados y el ratón no encima)
- **`active`**: Esta pseudoclase se define mientras el objeto está activo.
- **`target`**: Esta pseudoclase se activa cuando un elemento que hemos definido con un "id" es visitado a través de un enlace-ancla.

Hay más, pero estas suelen funcionar en todos los navegadores. Algunas como `after` y `before` las veremos en el siguiente capítulo :-)

Por ejemplo, al pasar por encima de los enlaces de este documento se activan unos efectos (cambio de color por ejemplo) que se pueden definir con `a:hover`.

Display. Bloques.

En HTML hay elementos de varios tipos: por ejemplo los `inline` que se visualizan en la misma línea (tales como `<a>`, `<span>`...) o los `block` que son bloques (como `<div>`, `<p>`...) .

Con CSS podemos modificar el tipo de elemento HTML que queramos, utilizando el atributo `display`.

```
display: <tipo>;
```

Donde `<tipo>` puede ser por ejemplo `inline`, `block`, `list-item` (como las etiquetas <li> ...)

Con esto podemos, por ejemplo, hacer listas que se visualicen en la misma línea; solapas, ¡lo que queramos!

Si un elemento es de tipo `block` (como un div) podemos definir nosotros su altura y anchura con los atributos:

```
width: <ancho>;
height: <alto>;
```

El ancho y el alto los podemos expresar con unidades pero también con porcentajes :-)
Ejemplo:

```
width: 55px;
height: 120px;
```

Márgenes y "padding"

Los **márgenes** nos sirven para regular el espacio que hay a continuación de un elemento en cualquiera de sus lados.

```
margin-top: <cantidad>;
margin-bottom: <cantidad>;
margin-left: <cantidad>;
margin-right: <cantidad>;
```

La cantidad se expresa como todas las medidas en CSS. Ejemplo:

```
margin-bottom: 7px;
```

Otro valor que podemos usar es `auto`, donde el navegador calculará automáticamente los márgenes que le hay que dar al objeto para, por ejemplo, centrarlo.

Por ejemplo:

```
div.centrado {
width: 500px;
margin-left: auto;
margin-right: auto;
text-align: right;
}
```

Con ese código el `div` con clase "centrado" será un div de 500px alineado en el centro de la pantalla, pero que el texto que contiene está alineado a la derecha :-)

Por su parte, el **padding** es el espacio entre el borde y el contenido de un objeto, y se expresa igual que los márgenes pero con "padding" en vez de "margin". Ejemplo:

```
padding-left: 2mm;
```

Elementos AFTER y BEFORE

Una de las cosas potentes de CSS son los pseudo-elementos AFTER y BEFORE, que junto con la propiedad "Content" nos permite insertar cosas antes y despues de un elemento (X)HTML.

Como ejemplo veremos cómo poner en un blockquote esas "comillas" que se ven muchas veces, que "envuelven" el contenido.

```
blockquote
{
color: #333333;
}

blockquote:before {
color: #BEBEBE;
content: url("blockquote.gif") " " attr(cite);
}

blockquote:after
{
content: url("blockquotefin.gif");
}
```

En verde están marcados los elementos AFTER y BEFORE. Está claro lo que hacen, el BEFORE inserta la imagen de las comillas antes del contenido y AFTER inserta las comillas despues.

Esto lo conseguimos gracias al atributo `content` .

Pero también vemos eso de `attr(cite)`. ¿Qué es eso? Pues eso nos "devuelve" el contenido del atríbuto "cite" que le pongamos al blockquote (p.e. `<blockquote`

`cite="http://www.barrapunto.com">`).

También conviene meter el contenido del Blockquote dentro de un `<div>` para que valide el XHTML.

Como imágenes podeis usar estas: imágen 1 e imágen 2.

Capas

Por fin, llegamos a una de las partes de CSS más potente.

Gracias a las capas tenemos todo el control que queramos sobre los elementos de nuestra página web.

Normalmente la posición de los elementos de una página es **relativa**, es decir, que depende de los demás elementos de una página. Por ejemplo, un párrafo estará más abajo si antes de él hay más párrafos o elementos.

Debido a esto, normalmente cuando se quería colocar elementos en un sitio concreto, se recurría a las **tablas invisibles** o **imágenes espaciadoras invisibles**, lo que es una chapuza y muy poco accesible como hemos visto en el capítulo uno.

Con CSS esto ha cambiado: ahora podemos colocar los elementos en posición **absoluta**, es decir, indicando el tamaño y coordenadas exactas al navegador para que las coloque :-)

Bien, empezaremos desde el principio :-) .

Antes de nada decir que debido a su naturaleza las capas se suelen usar con bloques `<div>`. Por ejemplo cada capa un bloque DIV con un identificador único que despues definimos en el archivo CSS :-) .

Bien, veremos ahora lo primero: cómo indicar que un elemento tiene posición **absoluta** y no **relativa**:

```
position: <posición>;
```

Y `<posición>` puede valer:

- **`absolute`**: La posición del elemento no depende de ninguna otra etiqueta. Normalmente lo que nos interesa para las capas
- **`fixed`**: Al igual que el anterior la posición es absoluta, pero el elemento se queda fijo en el sitio al hacer "scroll" con el documento. Como ejemplo, el menú de este mismo documento :-)
- **`relative`**: Posición relativa; normalmente lo predeterminado
- **`static`**: Al igual que el anterior la posición es relativa, pero no podemos redimensionar (por ejemplo) el objeto.

De manera que para las capas debemos poner la posición a `absolute` o `fixed` :-)

Especificando las coordenadas

Bien, ya tenemos nuestra capa con posición absoluta (o fija) pero... ¿cómo le situamos?

Fácil: utilizando los atributos `top`, `bottom`, `left` y `right`.

Normalmente se especifican sólo `top` y `left`:

```
top: <posición>;
left: <posición>;
```

`<posición>` se especifica como las medidas en CSS en unidades como por ejemplo `px`. También se admiten porcentajes :-)

De esta manera:

```
#micapa {
position: absolute;
top: 200px;
left: 150px;
width: 175px;
height: 175px;
border: dashed 1px maroon;
text-align: center;
color: maroon;
font-family: "Lucida Grande", Verdana, Arial, Helvetica, sans-serif;
font-size: 16px;
}
```

Despues hacemos un `<div id="micapa">` en un documento HTML de pruebas que estés haciendo :-) y dentro colocamos un fragmento de texto, cerramos el div y comprobamos el resultado :-)

La capa será un cuadrado de 175px de lado, en la posición 200x150, con un borde marrón discontínuo, tipo de letra Lucida Grande, color marrón, tamaño 16px y centrado :-)

El z-index

A veces tenemos varias capas unas por encima de otras y queremos especificar cuáles están por encima de cúales :-)

Para esto usamos el `z-index`:

```
z-index: <índice>;
```

Z-index es un número cualquiera, la capa con mayor z-index aparecerá por encima de la capa con z-index menor, etc.

Enlaces

Hasta aquí llega este tutorial del CSS, espero que te haya servido para algo :-)

Si aún no estás convencido de las posibilidades del CSS échale un vistazo a esto:

- **CSS Zen Garden**: Magnífico sitio donde puedes aplicar varios estilos CSSs a la página de ejemplo. Los diseños son realmente impresionantes, con técnicas y métodos muy interesantes que te pueden ser de utilidad. Altamente recomendable :-)

Y estos son blogs (bitácoras) en castellano donde se habla también de XHTML y CSS; trucos, estándares web, etc:

- **Minid**

- **Nordic Design**
- **Lechuga Hervida, mi propio blog :-)**

Y estos son también bitácoras sobre diseño web y CSS pero en inglés:

- **CSS information - inspiration**
- **A List Apart**
- **Anne Van Kesteren**

Para dudas, sugerencias, opiniones: epanelapse(arroba)gmail.com :-)