Tema 2: CSS.

1. Introducción.

Las **Hojas de Estilo en Cascada** (**CSS**, acrónimo de **C**ascading **S**tyle **S**heets) son una herramienta fantástica para añadir presentación a los sitios Web.

Las Hojas de Estilo nos va ahorrar mucho tiempo y nos van a permitir diseñar sitios Web de un modo distinto. **CSS** es imprescindible para todos aquellos que trabajen en el campo del diseño Web.

Para usar **CSS es necesario tener conocimientos de HTML,** un navegador y un editor de texto.

2. ¿Qué es CSS?

CSS son las siglas de Cascading Style Sheet que traducido significa Hojas de Estilo en cascada.

Las Hojas de Estilo es una tecnología que nos permite controlar la apariencia de una página Web. En un principio, los sitos Web se concentraban más en su contenido que en su presentación.

El lenguaje de las Hojas de Estilo está definido en las especificaciones del World Wide Web Consortium (W3C), es un estándar aceptado por toda la industria relacionada con la Web, o por lo menos, gran parte de navegadores (es verdad el IExplorer de Microsoft nos puede dar un dolor de cabeza). Podemos visitar W3C.

2.1. ¿Qué puedo hacer con CSS?

CSS es un lenguaje de estilo que define la presentación de los documentos **HTML**. Por ejemplo, **CSS** abarca cuestiones relativas a fuentes, colores, márgenes, líneas, altura, anchura, imágenes de fondo, posicionamiento avanzado y muchos otros temas. Es posible usar **HTML**, o incluso abusar del mismo, para añadir formato a los sitios Web. Sin embargo, **CSS** ofrece más opciones y es más preciso y sofisticado. **CSS** está soportado por todos los navegadores hoy día.

2.2. ¿Qué diferencia hay entre CSS y HTML?

HTML se usa para estructurar el contenido y **CSS** se usa para formatear el contenido previamente estructurado.

En un principio el lenguaje **HTML** sólo se usaba para añadir estructura al texto. Los autores podían marcar sus textos diciendo "esto en un título" o "esto es un párrafo", usando las etiquetas **HTML** <h1> y <p>, respectivamente.

A medida que la Web fue ganando popularidad, los diseñadores empezaron a buscar posibilidades para añadir formato a los documentos en línea. Para satisfacer esta reclamación, los fabricantes de los navegadores (en ese momento, Netscape y Microsoft) inventaron nuevas etiquetas **HTML**, entre las que se encontraban, por ejemplo, <font>, que se diferenciaba de las etiquetas originales **HTML** en que definían el formato... y no la estructura.

Esto también llevó a una situación en la que las etiquetas estructurales originales, por ejemplo, <table>, se usaban cada vez más de manera incorrecta para dar formato a las páginas en vez de para añadir estructura al texto. Muchas nuevas etiquetas que añadían formato, por ejemplo, <blink>, sólo las soportaban un tipo determinado de navegador. "Necesitas el navegador X para visualizar esta página" se convirtió en una declaración de descargo común en los sitios web.

CSS se inventó para remediar esta situación, proporcionando a los diseñadores Web con sofisticadas oportunidades de presentación soportadas por todos los navegadores. Al mismo tiempo, la separación de la presentación de los documentos del contenido de los mismos, hace que el mantenimiento del sitio sea mucho más fácil.

2.3. ¿Qué beneficios me ofrece CSS?

CSS fue toda una revolución en el mundo del diseño Web. Entre los beneficios concretos de **CSS** encontramos:

- ✓ Control de la presentación de muchos documentos desde una única hoja de estilo.
- ✓ Control más preciso de la presentación.

- ✓ Aplicación de diferentes presentaciones a diferentes tipos de medios (pantalla, impresión, etc.).
- ✓ Numerosas técnicas avanzadas y sofisticadas.

2.4. ¿Cómo funciona CSS?

Muchas de las propiedades que se usan en las **Hojas de Estilo en Cascada** (**CSS**) son parecidas a las de **HTML**. Así pues, si estás acostumbrado a usar **HTML** para cuestiones de presentación, lo más probable es que reconozcas gran parte del código usado. Examinemos un ejemplo concreto.

2.5. La sintaxis básica de CSS

Digamos que queremos un bonito color rojo como fondo de nuestra página web:

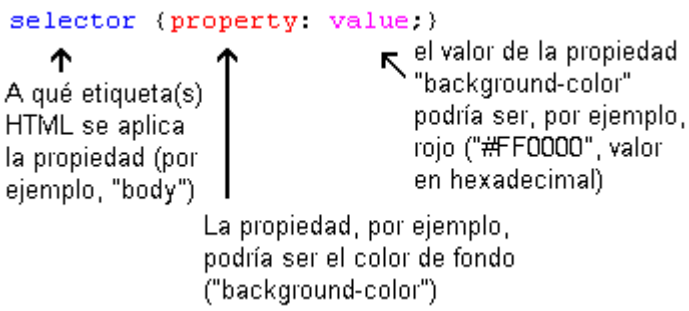
Usando **HTML** podríamos haberlo conseguido así:

```
<body bgcolor="#FF0000">
```

Con **CSS** el mismo resultado puede lograrse así:

```
body {background-color: #FF0000;}
```

Como verás, el código usado es más o menos idéntico para **HTML** y **CSS**. El ejemplo anterior te muestra además el modelo **CSS** fundamental:



Pero ¿dónde se sitúa el código **CSS**? Eso, precisamente, es lo que vamos a estudiar ahora mismo.

2.6. Aplicando CSS a un documento HTML

Podemos aplicar **CSS** a un documento **HTML** de tres maneras diferentes:

Método 1: En línea (el atributo style)

Un modo de aplicar **CSS** a **HTML** es usando el atributo de **HTML** style aplicado a una etiqueta **HTML**. Si ampliamos el ejemplo anterior sobre el color de fondo rojo, **CSS** se puede aplicar así:

```
<html>
  <head>
    <title>Ejemplo</title>
  </head>
  <body style="background-color: #FF0000;">
    <p>Esta es una página con fondo rojo</p>
  </body>
</html>
```

2.6.1. Método 2: Interno (la etiqueta style)

Otra forma es incluir el código **CSS** usando la etiqueta **HTML** <style>. Por ejemplo, así:

```
<html>
  <head>
    <title>Ejemplo</title>
    <style type="text/css">
      body {background-color: #FF0000;}
    </style>
  </head>
  <body>
    <p>Esta es una página con fondo rojo</p>
  </body>
</html>
```

2.6.2. Método 3: Externo (enlace a una hoja de estilo)

El método recomendado es enlazar con lo que se denomina hoja de estilo externa.

Una hoja de estilo externa es sencillamente un fichero de texto con la extensión **.css**. Como cualquier otro fichero, puedes colocar la hoja de estilo en el servidor web o en el disco duro.

Por ejemplo, digamos que tu hoja de estilo se llama **style.css** y está localizada en una carpeta que se llama **style**. Esta situación se puede ilustrar de la siguiente manera:

El truco consiste en crear un vínculo desde el documento **HTML** (por ejemplo, default.htm) con la hoja de estilo (style.css). Dicho vínculo se puede crear con una sencilla línea de código **HTML**:

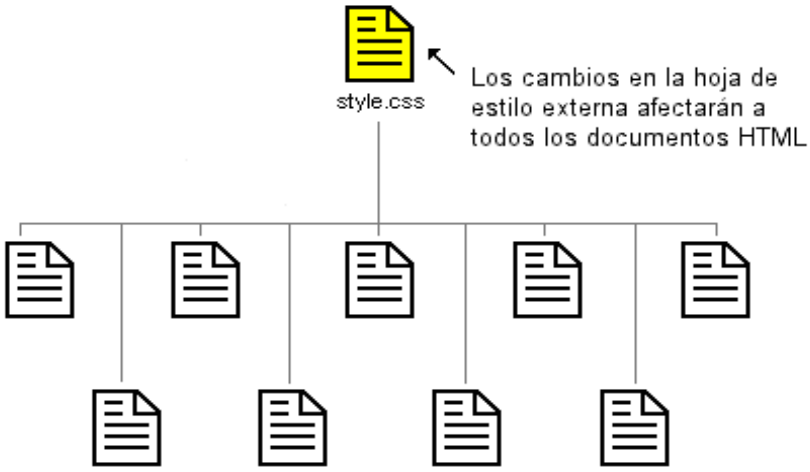
```
<link rel="stylesheet" type="text/css" href="style/style.css" />
```

Fíjate cómo la ruta a nuestra hoja de estilo aparece indicada por medio del atributo href.

La línea de código debe insertarse en la sección de encabezado del código **HTML**, es decir, entre la etiqueta <head> y </head>. De esta manera:

```
<html>
  <head>
    <title>Mi documento</title>
    <link rel="stylesheet" type="text/css" href="style/style.css" />
  </head>
  <body>
```

Este vínculo indica al navegador que debería usar la presentación del fichero **CSS** al mostrar el fichero **HTML**. Lo realmente bueno de este método es que se pueden vincular varios documentos **HTML** con la misma hoja de estilo. En otras palabras, se puede usar un único fichero **CSS** para controlar la presentación de los documentos **HTML**.



Documentos HTML que enlazan con la misma hoja de estilo

Esta técnica puede ahorrarte mucho trabajo. Si quisieras cambiar, por ejemplo, el color de fondo de un sitio Web compuesto por 100 páginas, un hoja de estilo puede ahorrarte el tener que cambiar de forma manual los 100 documentos **HTML**. Con **CSS**, el cambio se puede llevar a cabo en unos segundos modificando parte del código de la hoja de estilo principal.

Abre editor y crea dos ficheros, un fichero **HTML** y un fichero **CSS,** con el siguiente contenido:

Fichero default.htm

```
<html>
  <head>
    <title>Mi documento</title>
    <link rel="stylesheet" type="text/css" href="style.css" />
  </head>
  <body>
    <h1>Mi primera hoja de estilo</h1>
  </body>
</html>
```

Fichero estilo.css

```
body {
  background-color: #FF0000;
}
```

Ahora coloca los dos ficheros en la misma carpeta. Recuerda grabar los ficheros con las extensiones correctas ("**.htm**" y "**.css**", respectivamente).

Abre el fichero **default.htm** con el navegador y observa que la página tiene un color de fondo rojo

3. Agrupación de elementos. (span y div)

Los elementos y se usan para agrupar y estructurar un documento, y se usarán, a menudo, junto con los atributos class e id. Son elementos contenedores, es decir, pueden contener otros elementos.

Son elementos que podemos utilizar como capas dentro de una página web.

En este apartado revisaremos el uso de los elementos y , ya que estos dos elementos tienen una importancia clave en lo que se refiere a **CSS**.

3.1. Agrupación con

El elemento es lo que se podría denominar un elemento neutro que no añade nada al documento en sí. Pero con **CSS** se puede usar para añadir características visuales distintivas a partes específicas de texto en los documentos.

Un ejemplo de esto podría ser esta cita de Benjamin Franklin:

```
<p>El que pronto se acuesta y pronto se levanta,
es hombre saludable, rico y sabio.</p>
```

Digamos que queremos que lo que el señor Franklin considera como las ventajas de no pasarse todo el día durmiendo, aparezca enfatizado en rojo. Para este fin, podemos marcar dichas ventajas con el elemento . A cada elemento span se le añade el atributo class, que podemos definir así en nuestra hoja de estilo:

```
<p>El que pronto se acuesta y pronto se levanta,
es hombre <span class="ventaja">saludable</span>,
<span class="ventaja">rico</span>
y <span class="ventaja">sabio</span>.</p>
```

El código **CSS** necesario para producir este efecto es el siguiente:

```
span.benefit {
        color:red;
}
```

Por supuesto, se puede usar también el atributo id para añadir estilo a los elementos definidos con . Pero recuerda que tendrás que aplicar siempre un atributo id único para cada uno de los tres elementos , tal como aprendimos en la lección anterior.

3.2. Agrupación con el elemento

Mientras que se usa dentro de un elemento a nivel de bloque como vimos en el ejemplo anterior, se usa para agrupar uno o más elementos a nivel de bloque.

Aparte de esta diferencia, la agrupación con funciona más o menos igual.

Veamos un ejemplo con dos listas de presidentes de los EE.UU., divididas según su filiación política.

```
<div id="democrats">
<ul>
<li>Franklin D. Roosevelt</li>
<li>Harry S. Truman</li>
<li>John F. Kennedy</li>
<li>Lyndon B. Johnson</li>
<li>Jimmy Carter</li>
<li>Bill Clinton</li>
</ul>
</div>

<div id="republicans">
<ul>
<li>Dwight D. Eisenhower</li>
<li>Richard Nixon</li>
<li>Gerald Ford</li>
<li>Ronald Reagan</li>
<li>George Bush</li>
<li>George W. Bush</li>
</ul>
</div>
```

En nuestra hoja de estilo podemos utilizar la agrupación del mismo modo que antes:

```
#democrats {
	background:blue;
}

#republicans {
	background:red;
}
```

En los ejemplos anteriores, sólo hemos usado y con cosas muy sencillas como, por ejemplo, texto y colores de fondo. Ambos elementos tienen el potencial para realizar cosas más avanzadas.

4. Los Selectores.

Hasta ahora hemos visto la posibilidad de usar etiquetas html como selectores. De esta forma podemos diseñar estilos para cabeceras, enlaces, párrafos... pero no es una solución lo suficientemente flexible. Enseguida querremos, por ejemplo, establecer distintos estilos según los tipos de párrafos, por ejemplo para distinguir un menú del contenido de la página, o para distinguir el encabezado del resto de texto, por lo que el selector p se quedará corto.

Mediante las *clases* podremos definir estilos abstractos, que se podrán aplicar a cualquier etiqueta html. Mediante otros selectores aun mas específicos podemos alterar el estilo de partes muy concretas.

Variantes de selectores:

- ✓ etiquetas html: cada etiqueta es un posible selector. Por ejemplo un estilo para párrafos y otro para listas.
- ✓ clases: selectores abstractos aplicables a cualquier elemento, por ejemplo un estilo aplicable a algún párrafo y/o alguna lista
- ✓ selectores contextuales o descendentes, para elementos que estén dentro (desciendan) de otro elemento
- ✓ selectores hijos, para elementos que sean directamente hijos de otro elemento
- ✓ selectores adyacentes, para elementos que estén próximos a otros
- ✓ Selectores por identidad, para un elemento individual
- ✓ pseudoclases y pseudo elementos: links, primera letra, primera línea etc

4.1. class.

Mediante la definición de 'clases' se establecen estilos que pueden aplicarse a cualquier selector HTML o elemento de la página. El estilo definido en una clase no está vinculado a una etiqueta o elemento concreto sino a una 'clase', y esta clase se puede anudar a cualquier etiqueta HTML o grupo de ellas.

```
<html>
    <head>
        <title>Titulo</title>
        <style type="text/css">
            h1.miclase { color:blue; }
            .miotraclase {color:green;}
        </style>
    </head>
    <body>
        <h1 class="miclase">esto aparecería en azul</h1>
        <h1>esto aparecería como cabecera H1 normal</h1>
        <h1 class="miotraclase">esto aparecería en verde</h1>
        <p class="miotraclase">y este párrafo, tambien en verde</p>
    </body>
</html>
```

Pueden definirse clases para cualquier selector o etiqueta HTML:

```
h1.verde {color:green;}
```

o pueden asignarse clases independientemente de las etiquetas HTML a las que pueda afectar. En este último caso se omite en el selector el nombre de etiqueta:

```
.verde {color:green;}
```

Por ejemplo:

```
.Piepagina { font-size: small; }
```

La clase "piepagina" (definida en la cabecera de la pagina o en archivo externo) puede usarse con cualquier elemento, y visualizara con tipo de letra pequeña el texto al que afecte:

```
<p class="Piepagina">
<h1 class="Piepagina">
<div class="Piepagina">
<span class="Piepagina">
```

La clase así definida puede usarse en cualquier elemento o selector: en el primer caso, afectaría al texto dentro del párrafo ; en el segundo, a la cabecera h1; en el tercero, a todo el bloque div (cualquiera que fueran las etiquetas que incluya); y en el cuarto, al bloque span

Solo una clase se puede especificar a la vez para cada etiqueta HTML.

```
P.miclase.miotraclase
```

sería inválido.

Pero si pueden establecerse clases separadas. Una misma etiqueta HTML puede tener diferentes "clases", permitiendo que un mismo elemento ofrezca diferentes estilos:

```
h1.roja {font: 15pt/17pt; color: red;}
h1.verde {font: 15pt/17pt; color: green;}
h1.azul {font: 15pt/17pt;color: blue;}
```

y usar una clase u otra en la Página Web como sigue:

```
<h1 class="roja">Este es un encabezamiento rojo</h1>
...
<h1 class="azul">Este es azul </h1>
...
<h1 class="verde">Y este .... verde</h1>
```

Cuando aplicamos una sintaxis del estilo elementohtml.clase esa regla de estilo solo se aplicará a ese elemento html que tenga asignada esta clase.

Finalmente, tambien podemos asignar mas de una clase a una misma etiqueta:

```
<head>
    .mitexto { color: yellow }
    .mifondo { background-color: blue }
</head>
...
```

<h3 class="mitexto mifondo">titulo en amarillo, fondo azul</h3>

** color amarillo; fondo: el que herede de la pagina. **

4.2. Selector universal

El asterisco sirve para seleccionar cualquier elemento html

*** {estilos}**

identifica cualquier elemento html descendente de otro elemento con la clase "green".

Aquí tienes ejemplos *[http://www.ignside.net/man/css/ejemplos/ejemplo36.html]* de algunos selectores condicionales.

4.3. Selector ID

Mediante el atributo id podemos establecer una identidad única para un único elemento de la página. La sintaxis html sería por ejemplo , donde especificamos la identidad "menu" para ese único párrafo.

Al elemento html pueden le podemos asignar estilos a través del selector id:

#menu {estilos ...}

o bien

p#menu {estilos ...}

4.4. Selector universal

El asterisco sirve para seleccionar cualquier elemento html

*** {estilos}**

identifica cualquier elemento html descendente de otro elemento con la clase "green".

Aquí tienes ejemplos *[http://www.ignside.net/man/css/ejemplos/ejemplo36.html]* de algunos selectores condicionales.

4.5. Selectores contextuales

Las reglas de herencia de propiedades entre los distintos estilos definidos permiten crear estilos genéricos por defecto, y estilos por excepción, para elementos concretos.

Por ejemplo, supongamos que queremos dos estilos diferentes, uno para el elemento (enfatizado, cursiva) y otro para el elemento h1.

Ya hemos visto que se haría así:

h1 { color: blue; }

em { color: red; }

Y efectivamente, todo el texto enfatizado, dentro o fuera de un encabezamiento h1 se mostrará en rojo. Pero imaginemos que solo deseamos ver en rojo el texto enfatizado que aparezca dentro de una cabecera h1, y que el resto de texto enfatizado siga otro estilo.

Esto se puede definir así:

h1 em { color: red; }

em{color:green;}

Acabamos de definir un selector contextual. Las palabras enfatizadas se visualizaran en rojo *solamente si están en el contexto (dentro) de un selector h1*. En nuestro caso, el elemento em *dentro* o en el contexto de una cabecera. O dicho de otra forma, cuando el elemento **em** *desciende* jerárquicamente de un elemento h1. Por eso a estos selectores se les llama indistintamente *contextuales* o *descendentes*.

Un selector contextual consiste en varios selectores simples (h1 em) separados por un espacio en blanco (no por comas). El navegador, cuando tenga que visualizar texto em, investigará en las definiciones de estilo si existe una regla aislada para ese elemento o si dicho elemento es descendiente de otro, como ocurrirá, en este caso de ejemplo, si em esta dentro de h1.

Veamos otro ejemplo:

```
ul li { font-size: small; }

ul ul li { font-size: x-small; }
```

El primer selector asigna una letra pequeña a los elementos li que estén incluidos en al menos una etiqueta ul. El segundo selector asigna una letra extra-pequeña a los elementos li que dependan de al menos dos etiquetas ul (listas anidadas). El segundo selector, más específico que el primero, prevalece en este caso. Los ítems de las listas aparecerán en letra pequeña; y los ítems de las listas anidadas, en letra extra-pequeña.

Los selectores contextuales pueden definirse en relación con selectores ordinarios, como hemos visto, pero tambien con elementos CLASS, ID, o combinaciones:

```
div p { font: small "sans-serif"; }

.roja h1 { color: red; }

#x78y pre { background: blue; }

div.roja h1 { font-size: large; }
```

La primera definición determina que todos los párrafos (p) dentro de un bloque div /div se presentarán con letra sans-serif pequeña, pues dependen jerárquicamente (contextualmente) de dicho elemento div.

La segunda selección determina la presentación en rojo de todas las cabeceras h1 dentro de la clase .roja.

El tercer selector determina que todos los elementos con la etiqueta pre que desciendan de un elemento ID=x78y tendrán fondo azul.

El cuarto selector hará que todas las cabeceras h1 integradas en un bloque div con clase .roja tengan un tipo de letra grande.

Los selectores contextuales se pueden agrupar:

```
h1 b, h2 b, h1 em, h2 em { color: red; }
```

que equivale a:

```
h1 b { color: red; }

h2 b { color: red; }

h1 em { color: red; }

h2 em { color: red; }
```

4.6. Selectores hijos

En el ejemplo anterior h1 em { color: red; } define un estilo para todos los elementos *em* descendientes de una cabecera h1. Este estilo se aplicaría por ejemplo en estos dos casos:

```
<h1>Capitulo 1: El retorno del <em>Jedi</em></h1>

<h1>Capitulo 1: <strong>El retorno del <em>Jedi</em></strong></h1>
```

Pues en los dos casos *em* desciende de h1. Puede suceder sin embargo que solo quisiéramos aplicar el estilo al elemento cuando fuera *hijo directo* del otro, es decir, en el primer ejemplo y no en el segundo.

Esto lo conseguiríamos con esta sintaxis:

h1 > em

donde solo serian seleccionados elementos em directamente descendientes (hijos) de h1.

4.7. pseudo clases y pseudo elementos

Los pseudo-elementos son elementos de una página a quienes el lenguaje html no otorga identidad propia (y por tanto no pueden ser seleccionados) pero que si pueden ser identificados por las reglas de estilo.

Por ejemplo, no existe ninguna etiqueta html para marcar la primera letra de cada párrafo, o la primera línea. A través de los pseudo elementos podemos sin embargo acceder a ellas y dotarles de estilo.

Tambien son pseudo elementos aquellos que no existen en el código fuente de la página, pues son *generados* por las reglas de estilo.

Los pseudo-elementos disponibles en CSS2 son:

:first-letter

:first-line

:before

:after

Las pseudo-clases clasifican a los elementos basándose en el *estado* del elemento, atributos que no puede deducirse de la estructura del documento. Las pseudo-clases pueden ser dinámicas, en el sentido de que un elemento puede adquirir o perder una pseudo-clase a medida que el usuario interactúa con el documento, o como ':first-child', depender de la posición de un elemento en la estructura del documento.

Son pseudoclases :first-child :link :visited :hover, :active :focus y :lang

4.7.1. pseudo-elementos :first-letter :first-line

Estos pseudo elementos permiten asignar estilos a la primera letra de un documento, o a la primera línea.

```
P:first-letter {color: Green;font-size:x-large;}
```

Estos dos selectores solo pueden utilizarse en conjunción con elementos html formadores de bloque.

Asimismo, únicamente admiten un subconjunto de las propiedades CSS: con :first-letter podrás especificar las propiedades de las fuentes, color, background, text-decoration, vertical-align, text-transform, line-height, float y clear. Con :first-line propiedades de las fuentes, color, background, word-spacing, letter-spacing, text-decoration, vertical-align, text-transform, line-height, y clear.

4.7.2. La pseudo clase :link

Los navegadores normalmente visualizan de forma diferente los enlaces (links) visitados de los no visitados. En CSS, esta visualización se puede definir a través de pseudo clases en el elemento o selector a (ANCHOR), con las siguientes posibilidades:

a enlace.

a:link enlace que no ha sido explorado por el usuario.

a:visited se refiere a los enlaces ya visitados.

a:active enlace seleccionado con el ratón

a:hover enlace con el puntero del ratón encima, pero no seleccionado

a:focus enlace con el foco del sistema

```
a:link { color: red; } /* enlace no visitado*/
a:visited { color: blue; } /* enlace visitado */
a:active { color: lime; } /* enlace activo (pulsado) */
a:hover {color: red; } /* ratón en el enlace, sin pulsar. */
```

La sintaxis es semejante a la enunciación de clases (CLASS) si bien con dos puntos ":", en lugar de uno solo (a:Visited ... h1.miclase).

Estas definiciones afectarán a todos los elementos a que enlacen con otra página distinta, o con otra sección de la misma página. No afectan al elemento <a name>.

Las pseudo clases pueden ser usadas con selectores contextuales:

```
a:link img { border: solid blue;}
```

O combinadas con clases normales:

```
a.external:visited { color: blue; }
<a class="external" href="http://www.otrapagina.com/">link externo</a>
```

Se pueden especificar todo tipo de fuentes o formatos de texto para estos links (color, tamaño de fuente, grosor de la fuente, color de fondo), o a través de la propiedad "text-decoration" eliminar, por ejemplo, el subrayado de los links:

```
a:visited {color: blue; text-decoration: none; }
```

NOTA Si tienes problemas especificando estilos para los links, ten en cuenta el orden de precedencia de los estilos, que debe ser precisamente el indicado en la primera lista.

4.7.3. Extensión de las pseudoclases

El concepto de pseudo clases dinámicas se extiende con CSS2 a todos los elementos de la página, ya que básicamente cualquier elemento de la página puede tener el puntero del ratón encima (hover) estar seleccionado (active) o tener el foco (focus):

```
:hover
:active
:focus
```

4.7.4. pseudo-clase 'primer hijo'

La pseudo-clase :first-child equivale a un elemento que es el primer hijo de algún otro elemento. Considera este ejemplo:

```
<div class="capitulo">
    <p>primer párrafo del capitulo</p>
    <p>segundo párrafo del capitulo</p>
</div>
```

Si queremos dar estilo solo al primer párrafo, que es el primer hijo del elemento div, lo especificamos así:

```
p:first-child { estilo }
```

O así, si más exactamente, queremos señalar solo al párrafo *primer hijo* de cada capitulo, y no al párrafo primer hijo de otro elemento cualquiera:

```
div.capitulo > p:first-child { estilo }
```

4.8. Selectores adyacentes

Con estos selectores podemos aplicar estilos a elementos html *próximos* a otros.

En el ejemplo:

<h1>Capitulo 1: El retorno del Jedi de <cite> George Lucas</cite> </h1>

las etiquetas strong y cite son adyacentes. Podríamos aplicar un estilo a cite indicando:

strong + cite { estilo }

donde el estilo se aplicaría al elemento cite siempre que fuese adyacente, apareciera después de un elemento strong.

4.9. Seleccionando según el atributo

Podemos también seleccionar etiquetas html solo si contienen determinado atributo, o si este tiene determinado valor.

Veámoslo con el elemento img que como sabemos, puede contener los atributos alt y title.

img [alt] { estilo } /* selecciona imágenes con atributo alt */

img [alt="mis vacaciones"] { estilo } /* selecciona imágenes con atributo alt cuyo valor sea "mis vacaciones" */

Junto a estas dos variantes principales, [atributo ~="valor"] identifica la etiqueta que contenga "valor" como uno de sus valores de atributo separados por espacios; y la sintaxis [atributo|="valor"] identifica a la etiqueta que contenga el atributo indicado con el valor "valor" en una lista separada por guiones.

4.10. A tener en cuenta

Ninguno de estos dos selectores puede comenzar su denominación con un número. Asimismo, en la primera versión de CSS2, los guiones_bajos estaban prohibidos, y aunque posteriormente se admitieron, su soporte por algunos navegadores es errático, por lo que conviene evitarlos.

En la siguiente tabla se ve la sintaxis los selectores a modo de resumen.

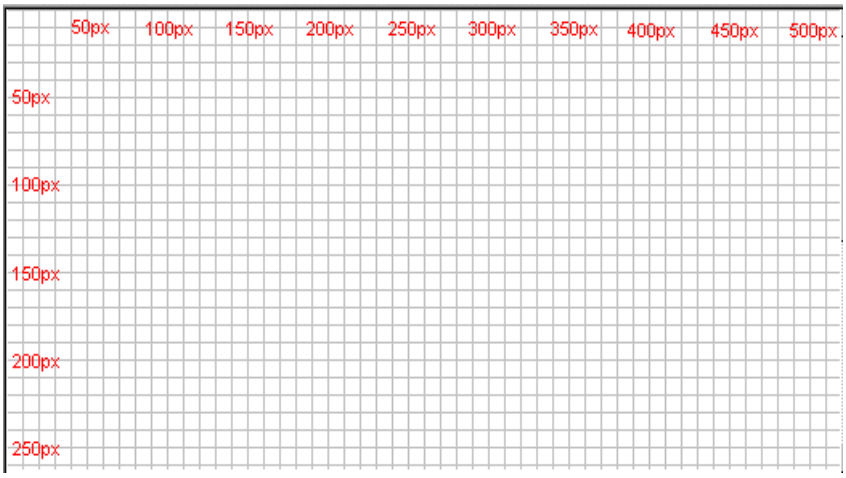
SELECTOR	A QUÉ TAGS AFECTA
CÓMO ESTÉ TAG SITUADO RESPECTO A OTROS TAGS.	
***	*Cualquier tag html.*
tag	*un tag específico.* *Por ejemplo, div equivaldría al tag *
tag1 tag2	*un tag2 anidado dentro de un tag1, en cualquier nivel de anidamiento.* *Por ejemplo, vale <tag1>...<otro>...<tag2>...</tag2>...</otro>...</tag1>* *Es decir, un tag2 que sea hijo, nieto, bisnieto... de tag1*
tag1 > tag2	*Un tag2 que sea hijo de tag1. No valen nietos.*
tag1 + tag2	*Un tag2 que va detrás y es adyacente a tag1.*
tag:first-child	*tag que sea el primer hijo de su padre.*
CON ALGÚN EVENTO DE USUARIO	
tag:link ***tag :visited***	*Para un tag que hace de hiperenlace. El más típico es <a>, aunque puede ser una <img>, etc. Afecta según no se haya visitado todavía (link) o sí se haya visitado (visited)*
tag:active ***tag :hover*** ***tag :focus***	*Un tag mientras se está haciendo click en él (active), cuando está el ratón sobre él (hover) o cuando tiene el foco (focus)*
TAGS CON ATRIBUTOS	
tag[atributo]	*Un tag que tenga el atributo, con cualquier valor*
tag[atributo="valor"]	*Un tag con el atributo asignado al valor.*
tag[atributo~="valor"]	*Un tag con un atributo cuyo valor son varios valores separados por comas y uno de ellos es valor.*
tag.clase	*Un tag con atributo class igual a clase. Es una forma abreviada que permite css de poner tag[class="~valor"]*
tag#id	*Un tag con atributo id igual a id. Es una forma abreviada que permite css de poner tag[id="~id"]*

5. Posicionamiento de elementos.

Con posicionamiento **CSS**, se puede colocar un elemento en el lugar exacto que se quiera de la página. Junto con las flotaciones [propiedad float], el posicionamiento proporciona muchas posibilidades de crear presentaciones avanzadas y precisas.

Principios **que rigen el posicionamiento CSS**

Imagina la ventana de un navegador como un sistema de coordenadas:



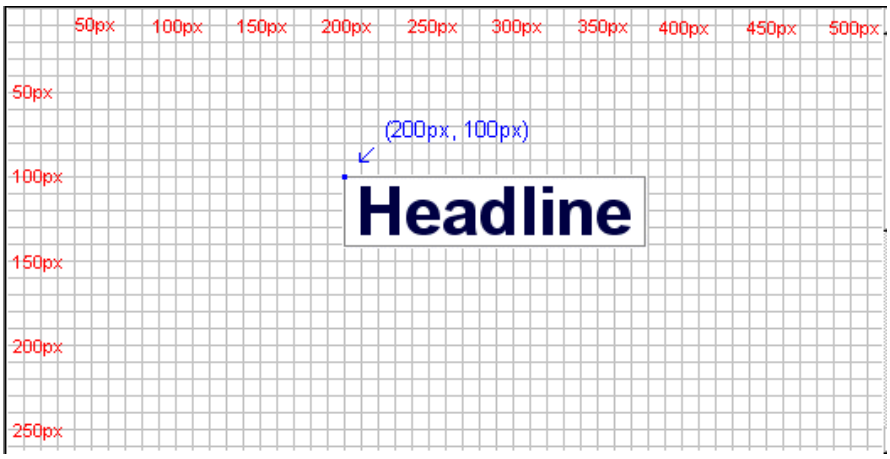
Los principios que rigen el posicionamiento **CSS** consisten en que se puede colocar cualquier caja en cualquier lugar del sistema de coordenadas.

Digamos que queremos posicionar un título. Usando el modelo de caja el título aparecerá así:

Si queremos posicionar la cabecera a 100 px del borde superior y a 200px del borde izquierdo del documento, tendríamos que escribir el siguiente código **CSS**:

```
h1 {
        position:absolute;
        top: 100px;
        left: 200px;
}
```

El resultado será el siguiente:



Como puedes observar, el posicionamiento con **CSS** es una técnica muy precisa a la hora de colocar elementos. Es mucho más sencillo que intentar usar tablas, imágenes transparentes o cualquier otra cosa.

5.1. Posicionamiento absoluto

El elemento que se posiciona de forma absoluta no ocupa espacio alguno en el documento. Esto significa que no deja un espacio vacío después de ser posicionado.

Para posicionar un elemento de forma absoluta, la propiedad position se establece como **absolute**. Posteriormente hay que usar las propiedades **left**, **right**, **top**, y **bottom** para colocar la caja y las propiedades **height** y **width** para dar tamaño a la caja.

Como ejemplo de posicionamiento absoluto, vamos a colocar 4 cajas en cada esquina del documento:

```
#box1 {
        position:absolute;
        top: 50px;
        left: 50px;
}

#box2 {
        position:absolute;
        top: 50px;
        right: 50px;
}

#box3 {
        position:absolute;
        bottom: 50px;
        right: 50px;
}

#box4 {
        position:absolute;
        bottom: 50px;
        left: 50px;
}
```

5.2. Posicionamiento relativo

Para posicionar un elemento de forma relativa, la propiedad position se establece como **relative**. La diferencia entre posicionamiento absoluto y relativo consiste en cómo se calcula la posición.

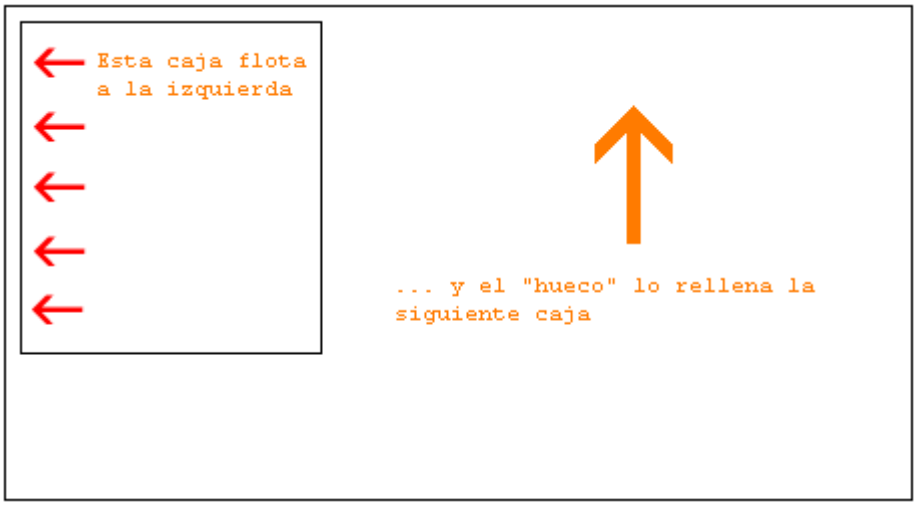
La posición para un elemento que se posiciona de forma relativa **se calcula desde la posición original** en **el documento.** Esto significa que se mueve el elemento hacia la derecha, la izquierda, arriba o abajo. De este modo, el elemento sigue ocupando espacio en el documento después de haberse posicionado.

Como ejemplo de posicionamiento relativo, podemos intentar posicionar tres imágenes de forma relativa respecto a su posición original en la página. Fíjate cómo las imágenes dejan espacios vacíos en sus posiciones originales en el documento:

```
#dog1 {
        position:relative;
        left: 350px;
        bottom: 150px;
}
#dog2 {
        position:relative;
        left: 150px;
        bottom: 500px;
}

#dog3 {
        position:relative;
        left: 50px;
        bottom: 700px;
}
```

5.3. Elementos flotantes (la propiedad float)

Los elementos se pueden hacer flotar a la derecha o a la izquierda usando la propiedad float. Es decir, que la caja con su contenido flota bien a la derecha o la izquierda de un documento (o de la caja contenedora). La siguiente imagen muestra este principio:



Por ejemplo, si quisiéramos texto con ajuste de línea alrededor de una imagen, el resultado sería el siguiente:

A floating image

Iste quidem veteres inter ponetur honeste, qui vel mense brevi vel toto est iunior anno. Utor permisso, caudaeque pilos ut equinae paulatim vello unum, demo etiam unum, dum cadat elusus Interdum volgus rectum videt, est ubi peccat. Si veteres ita miratur laudatque poetas, ut nihil anteferat, nihil illis comparet, errat. Si quaedam nimis antique, si peraque dure Interdum volgus rectum videt, est ubi peccat. Si veteres ita miratur laudatque poetas, ut nihil anteferat, nihil illis comparet, errat. Si quaedam nimis antique si peraque dure

¿Cómo se hace?

El código **HTML** del ejemplo anterior es el siguiente:

```
<div id="picture">
       <img src="bill.jpg" alt="Bill Gates">
</div>

<p>causas naturales et antecedentes,
idciro etiam nostrarum voluntatum...</p>
```

Para conseguir que la imagen flote a la izquierda y el texto se ajuste a su alrededor, sólo hay que definir el ancho de la caja que rodea la imagen y, después de eso, fijar la propiedad float con el valor left:

```
#picture {
       float:left;
       width: 100px;
}
```

Otro ejemplo: columnas

La propiedad float también se puede usar para crear columnas en un documento. Para crear dichas columnas tendrás que estructurar las columnas deseadas en el código **HTML** con la etiqueta , como se muestra a continuación:

```
<div id="column1">
       <p>Haec disserens qua de re agatur
       et in quo causa consistat non videt...</p>
</div>

<div id="column2">
       <p>causas naturales et antecedentes,
       idciro etiam nostrarum voluntatum...</p>
</div>

<div id="column3">
       <p>nam nihil esset in nostra
       potestate si res ita se haberet...</p>
</div>
```

Ahora, el ancho deseado de las columnas se fija, por ejemplo, en un porcentaje equivalente a un 33%, y luego simplemente se flota cada columna a la izquierda definiendo la propiedad float:

```
#column1 {
	float:left;
	width: 33%;
}

#column2 {
	float:left;
	width: 33%;
}

#column3 {
	float:left;
	width: 33%;
}
```

La propiedad float se puede establecer con los siguientes valores: **left** (izquierda), **right** (derecha) o **none** (ninguna).

5.4. La propiedad clear

La propiedad clear se usa para controlar cómo se comportarán los elementos que siguen a los elementos flotados de un documento.

Por defecto, los elementos siguientes se mueven hacia arriba para rellenar el espacio disponible que quedará libre al flotar una caja hacia un lado. Echa un vistazo al ejemplo anterior en el que el texto se desplaza de forma automática hacia arriba junto a la imagen de Bill Gates.

La propiedad clear puede tomar los valores: **left**, **right**, **both** o **none**. El principio consiste en que, si clear, por ejemplo, se fija en both para una caja, el borde del margen superior de esta caja siempre estará debajo del borde del margen inferior para las posibles cajas flotantes que vengan de arriba.

```
<div id="picture">
	<img src="bill.jpg" alt="Bill Gates">
</div>

<h1>Bill Gates</h1>

<p class="floatstop">causas naturales et antecedentes,
idciro etiam nostrarum voluntatum...</p>
```

Para evitar que el texto flote hacia arriba junto a la imagen, podemos añadir lo siguiente al código **CSS**:

```
#picture {
	float:left;
	width: 100px;
}

.floatstop {
	clear:both;
}
```

6. Colores y fondos.

En este apartado aprenderemos a aplicar colores y colores de fondo a tus sitios Web.

6.1. Unidades de color

Un valor de color es una palabra clave o una especificación numérica RGB.

Las 16 palabras clave se toman de la paleta Windows VGA: aguamarina, negro, azul, fucsia, gris, verde, verde lima, marrón, azul marino, olivo, morado, rojo, plata, turquesa, blanco, y amarillo.

Los colores RGB se dan en una de cuatro maneras:

- ✓ #rrggbb (por ej., #00cc00)
- ✓ #rgb (por ej., #0c0)
- ✓ rgb(x,x,x) donde x es un entero entre 0 y 255 inclusive (por ej., rgb(0,204.0))
- ✓ rgb(y%,y%,y%) donde y es un número entre 0.0 y 100.0 inclusive (por ej., rgb(0%,80%,0%))

Los ejemplos de arriba especifican el mismo color.

6.2. Color de primer plano: la propiedad 'color'

La propiedad color describe el color de primer plano de un elemento. Por ejemplo, imagina que queremos que todos los títulos de un documento aparezcan con color rojo oscuro. Todos los títulos están marcados con el elemento <h1>.

El código siguiente establece el color de los elementos <h1> como rojo.

```
h1 {
      color: #ff0000;
}
```

Los colores se pueden introducir como valores hexadecimales, como en el ejemplo anterior: #ff0000; o se pueden usar los nombres de los colores: "red" (rojo), o bien como valores rgb: (rgb(255,0,0)).

6.3. La propiedad 'background-color'

La propiedad background-color describe el color de fondo de los elementos. El elemento <body> contiene todo el contenido de un documento **HTML**. Así pues, para cambiar el color de fondo de una página, la propiedad background-color debería aplicarse al elemento <body>.

También se pueden aplicar colores de fondo a otros elementos, entre ellos, a los encabezados y al texto. En el ejemplo que sigue se aplicarán diferentes colores a los elementos <body> y <h1>.

```
body {
      background-color: #FFCC66;
}

h1 {
      color: #990000;
      background-color: #FC9804;
}
```

Fíjate cómo hemos aplicado dos propiedades a <h1> separándolas por medio de un punto y coma.

6.4. Imágenes de fondo [background-image]

La propiedad **CSS** background-image se usa para insertar una imagen de fondo. Para el ejemplo de la imagen de fondo, vamos a usar la mariposa que ves más abajo. Puedes descargar la imagen para usarla en tu propio ordenador (haz clic con el botón derecho sobre la imagen y elige "guardar imagen como..."), o bien puedes usar cualquier otra imagen.

Para insertar la imagen de la mariposa como imagen de fondo de una página web, aplica sencillamente la propiedad background-image al elemento <body> y especifica la localización de la imagen.

```
body {
        background-color: #FFCC66;
        background-image: url("butterfly.gif");
}

h1 {
        color: #990000;
        background-color: #FC9804;
}
```

NOTA: Fíjate cómo hemos especificado la localización de la imagen: **url ("butterfly.gif")**

Esto significa que la imagen está en la misma carpeta que la hoja de estilo. También puedes hacer referencia a imágenes en otras carpetas usando **url("../imagenes/butterfly.gif")** o incluso imágenes de internet si indicas la dirección completa del fichero: **url("http://www.html.net/butterfly.gif")**.

6.5. Repetir la imagen de fondo [background-repeat]

En el ejemplo anterior, ¿te fijaste en que, por defecto, la mariposa se repetía tanto en el eje horizontal como en el vertical para ocupar toda la pantalla? La propiedad background-repeat controla este comportamiento.

La tabla siguiente resume los cuatro valores diferentes para la propiedad background-repeat.

Valor	Descripción
Background-repeat: repeat-x	La imagen se repite en el eje horizontal
background-repeat: repeat-y	La imagen se repite en el eje vertical
background-repeat: repeat	La imagen se repite en el eje horizontal y vertical
background-repeat: no-repeat	La imagen no se repite

Por ejemplo, para evitar que se repita una imagen de fondo, el código que tendríamos que usar sería el siguiente:

```
body {
        background-color: #FFCC66;
        background-image: url("butterfly.gif");
        background-repeat: no-repeat;
}

h1 {
        color: #990000;
        background-color: #FC9804;
}
```

6.6. Fijar la imagen de fondo [background-attachment]

La propiedad background-attachment especifica si una imagen está fija o se desplaza con el elemento contenedor.

Una imagen de fondo fija no se moverá con el texto cuando el lector se desplace por la página, mientras que una imagen de fondo no fija se desplazará con el texto de la página Web.

La tabla siguiente resume los dos valores posibles para la propiedad backgroundattachment.

Haz clic en los ejemplos para ver la diferencia entre la imagen fija y la imagen que se desplaza.

Valor	Descripción
Background-attachment: scroll	La imagen se desplaza con la página - no está fija
Background-attachment: fixed	La imagen está fija

Por ejemplo, el siguiente código fijará la imagen de fondo.

```
body {
        background-color: #FFCC66;
        background-image: url("butterfly.gif");
        background-repeat: no-repeat;
        background-attachment: fixed;
}

h1 {
        color: #990000;
        background-color: #FC9804;
}
```

6.7. Ubicación de la imagen de fondo [background-position]

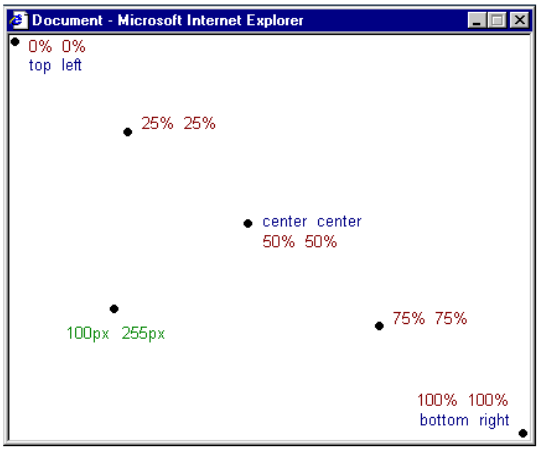
Por defecto, una imagen de fondo se posiciona en la esquina superior izquierda de la pantalla. La propiedad background-position te permitirá cambiar este valor por

defecto y posicionar la imagen de fondo en cualquier lugar de la pantalla que quieras.

Hay muchas formas diferentes de establecer los valores de la propiedad background-position. Sin embargo, todas ellas se formatean como un conjunto de coordenadas. Por ejemplo, el valor '100px 200px' posiciona la imagen de fondo a 100 píxeles del margen izquierdo y a 200 píxeles del margen superior de la ventana del navegador.

Las coordenadas se pueden indicar como porcentajes del ancho de la pantalla, como unidades fijas (píxeles, centímetros, etc.) o puedes usar las palabras "top" (superior), "bottom" (inferior), "center" (centro), "left" (izquierda) y "right" (derecha).

El modelo siguiente ilustra cómo funciona el sistema:



La tabla siguiente proporciona varios ejemplos.

Valor	Descripción
`background-position: 2cm 2cm`	La imagen se posiciona a 2 cm del margen izquierdo y a 2 cm del margen superior de la página
`background-position: 50% 25%`	La imagen se posiciona en el centro de la página y un 25 % del margen superior de la misma
`background-position: top right`	La imagen se posiciona en la esquina superior derecha de la página

El ejemplo de código siguiente posiciona la imagen de fondo en la esquina inferior derecha:

```
body {
        background-color: #FFCC66;
        background-image: url("butterfly.gif");
        background-repeat: no-repeat;
        background-attachment: fixed;
        background-position: right bottom;
}

h1 {
        color: #990000;
        background-color: #FC9804;
}
```

6.8. Combinación de propiedades [background]

La propiedad background es una forma abreviada de todas las propiedades de fondo listadas a lo largo de esta lección.

Con la propiedad background se pueden comprimir varias propiedades, y así escribir una hoja de estilo de forma más abreviada, lo que facilitará su lectura.

Por ejemplo, observa estas cinco líneas de código:

```
background-color: #FFCC66;
background-image: url("butterfly.gif");
background-repeat: no-repeat;
background-attachment: fixed;
background-position: right bottom;
```

Usando background se puede lograr el mismo resultado con una única línea de código:

```
background: #FFCC66 url("butterfly.gif") no-repeat fixed right bottom;
```

El orden en que deben aparecer las propiedades individuales es el siguiente:

[background-color] | [background-image] | [background-repeat]| [backgroundattachment] | [background-position]

Si se omite alguna propiedad, de forma automática ésta se establecerá con su valor por defecto. Por ejemplo, si se omiten las propiedades background-attachment y background-position del ejemplo anterior, quedando el código de la siguiente manera:

```
background: #FFCC66 url("butterfly.gif") no-repeat;
```

Estas dos propiedades que no se especifican se establecerían, sin más, con sus valores por defecto, que, como ya sabes, son scroll y top left.

7. Fuentes

En este apartado veremos nociones sobre fuentes y cómo se aplican usando **CSS**.

También veremos cómo solucionar el tema de que las fuentes específicas elegidas para un sitio Web sólo se pueden ver si están instaladas en el PC desde el que se accede a dicho sitio Web. Se describirán las siguientes propiedades **CSS**:

7.1. Unidades.

7.1.1. Unidades de longitud

Un valor de longitud se forma por un signo + o - opcional, seguido de un número y de una abreviación de dos letras que indica la unidad. No hay espacios en un valor de longitud; por ej., 1.3 em no es un valor de longitud válido, pero 1.3em si lo es. Una longitud de 0 no necesita las dos letras para identificar la unidad.

Tanto las unidades de longitud relativas y absolutas están soportadas en **CSS**1.

- ✓ Mediante valores relativos:
  - El valor en **em** para la fuente se referirá al valor que tiene por defecto o heredado indicando la proporción, en la relación 1:X, de forma que un valor **1em** no cambia, pero un **1.2em** lo aumenta un 20%, o un **0.8em** lo disminuye también en un 20%.
  - El valor en ex para la fuente se emplea del mismo modo que **em**, pero en lugar de referirse al tamaño, se refiere a la altura del carácter 'x' que tiene la fuente.
  - El valor en **px** para la fuente se especificará en píxeles referentes a la pantalla, dependiendo de la resolución que utilice el usuario, así será el tamaño para la fuente.
- ✓ Mediante valores absolutos:
  - **in**: medida en pulgadas (inches)
  - **cm**: medida en centímetros
  - **mm**: medida en milímetros
  - **pt**: medida en puntos
  - **pc**: medida en picas

Con respecto a estas medidas existen unas equivalencias métricas a tener en cuenta, para saber qué medida estamos indicando: 1 pulgada (in) equivale a 2.54 centímetros (cm) 1 punto (pt) equivale a 1/72 de pulgada (in) 1 pica (pc) equivale a 12 puntos (pt)

7.1.2. Unidades de porcentaje

Un valor de porcentaje se forma por un signo + o - opcional, seguido de un número y de %. No hay espacios en un valor de porcentaje.

Los valores de porcentaje son relativos a otros, tal como están definidos en cada propiedad. Generalmente, el valor de porcentaje es relativo al tamaño de fuente del elemento.

7.2. Familia de fuentes [font-family]

La propiedad font-family se usa para establecer una lista ordenada de fuentes que se usarán para mostrar un elemento determinado o una página Web. Si la primera fuente

de la lista no está instalada en el ordenador desde el que se accede al sitio, se seguirá probando con la siguiente fuente hasta encontrar una fuente apropiada.

Para clasificar las fuentes se usan dos tipos de nombres: nombres de una familia y familias genéricas. Estos dos términos se explican a continuación.

La diferencia se puede ilustrar así:

Al listar fuentes para el sitio Web, por supuesto se empieza por la preferida, seguida ésta de algunas fuentes alternativas. Se recomienda completar la lista con una familia de fuentes genérica. Así, al menos, la página se mostrará usando una fuente de la misma familia si ninguna de las especificadas está disponible.

Un ejemplo de lista de fuentes por orden de prioridad podría tener este aspecto:

```
h1 {font-family: arial, verdana, sans-serif;}
h2 {font-family: "Times New Roman", serif;}
```

Los encabezados marcados con la etiqueta <h1> se mostrarán usando la fuente "Arial". Si esta fuente no está instalada en el ordenador de usuario, se usará en su lugar la fuente "Verdana". Si ambas fuentes no están disponibles, se usará una fuente de la familia **sans-serif** para mostrar los encabezados.

Fíjate cómo el nombre de fuente "Times New Roman" contiene espacios y, por lo tanto, se lista usando comillas.

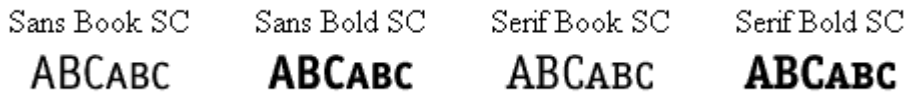
7.3. Estilo de la fuente [font-style]

La propiedad font-style define la fuente elegida bien con el valor **normal**, el valor **italic** o el valor **oblique**. En el ejemplo que sigue, todos los encabezados marcados con <h2> aparecerán en cursiva.

```
h1 {font-family: arial, verdana, sans-serif;}
h2 {font-family: "Times New Roman", serif; font-style: italic;}
```

7.4. Variante de fuente [font-variant]

La propiedad font-variant se usa para elegir entre las variantes **normales** o **small-caps** (versalita) de una fuente. La fuente a la que se aplica el valor **small-caps** es una fuente que usa letras en mayúscula inicial más pequeñas, en vez de letras en minúscula. ¿Confuso? Veamos los ejemplos siguientes:



Si la propiedad font-variant se establece con el valor **small-caps** y no hay disponible una fuente en versalita, el navegador probablemente mostrará el texto en mayúscula.

```
h1 {font-variant: small-caps;}
h2 {font-variant: normal;}
```

7.5. Peso de la fuente [font-weight]

La propiedad font-weight describe qué intensidad o "peso" en negrita debería tener la fuente. Toda fuente puede tener los valores **normal** o **bold**. Algunos navegadores soportan, incluso, el uso de números entre 100 y 900 (de cien en cien) para describir el peso de dicha fuente.

```
p {font-family: arial, verdana, sans-serif;}
td {font-family: arial, verdana, sans-serif; font-weight: bold;}
```

7.6. Tamaño de la fuente [font-size]

El tamaño de la fuente se establece por medio de la propiedad font-size.

A la hora de describir el tamaño de las fuentes, existen muchas unidades diferentes (por ejemplo, píxeles y porcentajes) entre las que elegir. En este tutorial nos centraremos en las unidades más comunes y adecuadas. Como ejemplo, podemos incluir:

```
h1 {font-size: 30px;}
h2 {font-size: 12pt;}
h3 {font-size: 120%;}
p {font-size: 1em;}
```

Existe una diferencia clave entre las cuatro unidades anteriores. Las unidades '**px**' y '**pt**' establecen el tamaño de la fuente de forma absoluta, mientras que '**%**' y '**em**' permiten al usuario ajustar el tamaño de la misma según considere oportuno. Muchos usuarios son discapacitados, mayores, o disponen de un monitor de mala calidad. **Para que tu sitio Web sea accesible** para todo el mundo, deberías usar unidades ajustables como, por ejemplo, '**%**' o '**em**'.

7.7. Combinación de propiedades [font]

Si usamos la propiedad abreviada font es posible incluir todas las propiedades diferentes relativas a fuentes en una única propiedad.

Por ejemplo, imagina estas cuatro líneas de código que usamos para describir las propiedades de fuente para la etiqueta <p>:

```
p {
        font-style: italic;
        font-weight: bold;
        font-size: 30px;
        font-family: arial, sans-serif;
}
```

Usando la propiedad abreviada, el código se puede simplificar así:

```
p {
        font: italic bold 30px arial, sans-serif;
}
```

El orden de los valores para la propiedad font es:

font-style | font-variant | font-weight | font-size | font-family

8. Texto.

Formatear y añadir estilo al texto es un tema clave para cualquier diseñador web.

En este apartado presentaremos las increíbles oportunidades que ofrece **CSS** a la hora de añadir presentación al texto

8.1. Sangría del texto [text-indent]

La propiedad text-indent permite añadir un toque de elegancia a los párrafos de texto al aplicar sangría a la primera línea de dicho párrafo. En el ejemplo siguiente se ha aplicado una sangría de **30px** a todos los párrafos de texto marcados con la etiqueta <p>:

```
p {
        text-indent: 30px;
}
```

8.2. Alineación del texto [text-align]

La propiedad **CSS** text-align es el equivalente al atributo align usado en versiones anteriores de **HTML**. Los valores posibles de esta propiedad son: **left** (texto alineado a la izquierda), **right** (texto alineado a la derecha) o **center** (texto con alineación centrada). Además, el valor **justify** (alineación justificada) alargará cada línea de forma que los márgenes izquierdo y derecho estén justificados. Esta tipo de presentación la habrás visto, por ejemplo, en periódicos y revistas.

En el ejemplo que sigue, el texto de los encabezados de la tabla, <th>, se ha alineado a la derecha, mientras que los datos de la tabla, <td>, aparecen centrados.

Además, los párrafos de texto normales están justificados:

```
th {
        text-align: right;
}

td {
        text-align: center;
}

p {
        text-align: justify;
}
```

8.3. Decoración del texto [text-decoration]

La propiedad **text**-decoration permite añadir diferentes "decoraciones" o "efectos" al texto. Por ejemplo, se puede subrayar el texto, tacharlo o ponerle un subrayado superior. En el ejemplo siguiente, el elemento <h1>aparecerá subrayado, el elemento <h2> aparecerá con un subrayado por encima del texto y el elemento <h3> tendrá el texto tachado.

```
h1 {
        text-decoration: underline;
}

h2 {
        text-decoration: overline;
}

h3 {
        text-decoration: line-through;
}
```

8.4. Espaciado entre caracteres [letter-spacing]

El espaciado entre los caracteres de texto se puede especificar usando la propiedad letter-spacing. El valor de esta propiedad corresponde, sencillamente, al ancho deseado. Por ejemplo, si queremos un espaciado de 3px entre los caracteres de un párrafo de texto <p> y6px entre los caracteres de los encabezados <h1>, usaríamos el siguiente código:

```
h1 {
        letter-spacing: 6px;
}

p {
        letter-spacing: 3px;
}
```

8.5. Transformación del texto [text-transform]

La propiedad text-transform controla la escritura en mayúsculas de un texto. Puedes elegir entre los valores **capitalize**, **uppercase** o **lowercase**, sin importar cómo aparece el texto original en el código **HTML**.

Un ejemplo podría ser la palabra "título" que se puede presentar al usuario como "TÍTULO" o "Título". A continuación ofrecemos una explicación de los valores de la propiedad text-transform mencionados en el párrafo anterior:

Capitalize: Pone en mayúscula la primera letra de cada palabra. Por ejemplo, "john doe" aparecerá como "John Doe".
Uppercase: Convierte todas las letras a mayúscula. Por ejemplo, "john doe" aparecerá como "JOHN DOE".
Lowercase: Convierte todas las letras a minúscula. Por ejemplo, "JOHN DOE" aparecerá como "john doe".
None: No se realiza transformación alguna; el texto se presenta tal como aparece en el código **HTML**.

Como ejemplo, usaremos una lista de nombres. Todos los nombres están marcados con la etiqueta <li> (de "list element", es decir, elemento de lista).

Supongamos que queremos que las iniciales de los nombres aparezcan en mayúscula y los títulos con todos los caracteres en mayúscula.

Échale un vistazo al código fuente del ejemplo y verás que el texto aparece realmente en minúscula.

```
h1 {
        text-transform: uppercase;
}

li {
        text-transform: capitalize;
}
```

8.6. Espaciado entre palabras [word spacing]

La propiedad word-spacing define el espacio entre palabras. El valor debe estar en formato de longitud; no se permiten valores negativos.

Ejemplos:

```
P EM   { word-spacing: 0.4em }
P.nota { word-spacing: -0.2em }
```

8.7. Alineación vertical [vertical alignment]

La propiedad vertical-align se utiliza para alterar la ubicación vertical de un elemento en línea, en relación a su elemento padre o a la línea del elemento. (Un elemento en línea es uno que no tiene salto de línea ni antes ni después de él; por ejemplo, EM, A y IMG en HTML.)

El valor puede ser un porcentaje relativo a la altura de línea (line-height) del elemento, que debería elevar la línea de base del elemento en la cantidad especificada por encima de la línea base del padre. No se permiten los valores negativos.

El valor también puede ser una palabra clave. Las siguientes palabras clave afectan la ubicación en relación al elemento padre:

baseline: alinea líneas bases del elemento y el padre

middle: alinea el punto medio vertical del elemento con la línea base más la mitad de x-height--la altura de la letra "x"--del padre

sub: subíndice

super: superíndice

text-top: alinea las partes superiores del elemento y la fuente del elemento padre.

text-bottom: alinea las partes inferiores del elemento y la fuente del elemento padre.

Las palabras clave que afectan la ubicación relativa a la línea del elemento son:

top: alinea la parte superior del elemento con el elemento más alto en la línea.

bottom: alinea la parte inferior del elemento con el elemento más bajo en la línea.

La propiedad vertical-align es particularmente útil para alinear imágenes. Algunos ejemplos:

```
img.centro { vertical-align: middle }
img      { vertical-align: 50% }
.exponente  { vertical-align: super }
```

8.8. Altura de línea [line height]

La propiedad line-height nos permite controlar el espacio entre líneas base de texto. Si el valor es un número, la altura de línea se calcula multiplicando el tamaño de fuente del elemento por el número. Los valores en porcentaje son relativos al tamaño de fuente del elemento. No se permiten los valores negativos.

La altura de línea también puede darse en la propiedad font junto con un tamaño de fuente.

La propiedad line-height podría usarse para texto a doble espacio:

```
p { line-height: 200% }
```

Microsoft Internet Explorer 3.x trata incorrectamente a los valores numéricos y a los valores con unidades em o ex como valores de pixel. Este error puede hacer ilegibles las páginas, por lo que los autores deberían evitar provocarlo; con frecuencia, las unidades de porcentaje son una buena opción.

9. Enlaces.

La novedad respecto a los enlaces es que **CSS** permite definir estas propiedades de forma diferente dependiendo del estado del mismo, es decir, si el enlace se ha visitado, no se ha visitado, si es el enlace activo o si el cursor está sobre dicho enlace. Esto permite añadir efectos divertidos y útiles a tu sitio Web. Para controlar estos efectos se usan lo que se ha denominado como pseudoclases.

9.1. ¿Qué es una pseudo-clase?

Una pseudo-clase te permite tener en cuenta diferentes condiciones o eventos al definir una propiedad para una etiqueta **HTML**.

Veamos un ejemplo. Como ya sabes, los enlaces se especifican en **HTML** con la etiqueta <a>. Por lo tanto, podemos usar a como selector en **CSS**:

```
a {
        color: blue;
}
```

Todo enlace tiene diferentes estados. Por ejemplo, visitado o no visitado. Puedes usar una pseudo-clase para asignar diferentes estilos a los enlaces visitados y no visitados.

```
a:link {
        color: blue;
}

a:visited {
        color: red;
}
```

Usa a:link y a:visited para enlaces visitados y no visitados, respectivamente.

A los enlaces activos se les aplica la pseudo-clase a:active, y a:hover cuando el cursor se coloca o pasa sobre el enlace.

Vamos a repasar ahora cada una de las cuatro pseudo-clases con ejemplos y más explicaciones.

9.2. Pseudo-clase a:link

La pseudo clase:link se usa para enlaces que dirigen a páginas que el usuario no ha visitado.

En el ejemplo de código que sigue, los enlaces no visitados tendrán un color azul claro.

```
a:link {
        color: #6699CC;
}
```

9.3. Pseudo-clase a:visited

La pseudo clase:visited se usa para enlaces que dirigen a páginas que el usuario ya ha visitado. Por ejemplo, el código siguiente hará que todos los enlaces visitados sean de color púrpura oscuro:

```
a:visited {
        color: #660099;
}
```

9.4. Pseudo-clase a:active:

La pseudo clase:active se usa para enlaces que están activos.

El código de este ejemplo hace que el color de fondo para los enlaces activos sea amarillo:

```
a:active {
        background-color: #FFFF00;
}
```

9.5. Pseudo-clase a:hover

La pseudo clase a:hover se usa cuando el puntero del ratón pasa por encima de un enlace.

Esta pseudo-clase se puede usar para crear efectos interesantes. Por ejemplo, si queremos que nuestros enlaces sean de color naranja y estén en cursiva cuando el cursor pase sobre ellos, el código **CSS** que utilizaremos será el siguiente:

```
a:hover {
        color: orange;
        font-style: italic;
}
```

Ejemplo 1: Efecto cuando el cursor está encima de un enlace

Es bastante popular crear diferentes efectos cuando el cursor está encima de un enlace. Por lo tanto, examinaremos unos cuantos ejemplos más relacionados con la pseudo clase:hover.

Ejemplo 1a: Espaciado entre letras

Como recordarás el espaciado entre los caracteres se puede ajustar usando la propiedad letter-spacing. Esta propiedad se puede aplicar a los enlaces para crear un efecto especial:

```
a:hover {
        letter-spacing: 10px;
        font-weight:bold;
        color:red;
}
```

Ejemplo 1b: MAYÚSCULAS y minúsculas

También examinamos la propiedad text-transform, que sirve para intercambiar entre letras en mayúscula y minúscula. Esto se puede usar también para crear un determinado efecto en los enlaces:

```
a:hover {
        text-transform: uppercase;
        font-weight:bold;
        color:blue;
        background-color:yellow;
}
```

Estos dos ejemplos dan una idea sobre las posibilidades casi infinitas al combinar diferentes propiedades. Ya puedes ir creando tus propios efectos... ¡inténtalo!

Ejemplo 2: Quitar el subrayado de los enlaces

Una pregunta muy recurrente es: ¿cómo quito el subrayado de los enlaces?

Deberías considerar cuidadosamente la necesidad de quitar el subrayado, pues esto podría disminuir la usabilidad de tu sitio Web de forma significativa. La gente está acostumbrada al subrayado azul de los enlaces en las páginas Web y saben que pueden hacer clic en ellos. Si cambias el subrayado y el color de los enlaces, existe la posibilidad de que los usuarios se confundan.

Dicho esto, es muy sencillo quitar el subrayado de los enlaces. La propiedad text-decoration se puede usar para determinar si el texto aparece subrayado o no. Para quitar el subrayado, establece el valor de **text-decoration** a **none** para que desaparezca y a **underline** para que aparezca.

```
a {
        text-decoration:none;
}
```

De forma alternativa, puedes establecer la propiedad text-decoration junto con otras propiedades para las cuatro pseudo clases.

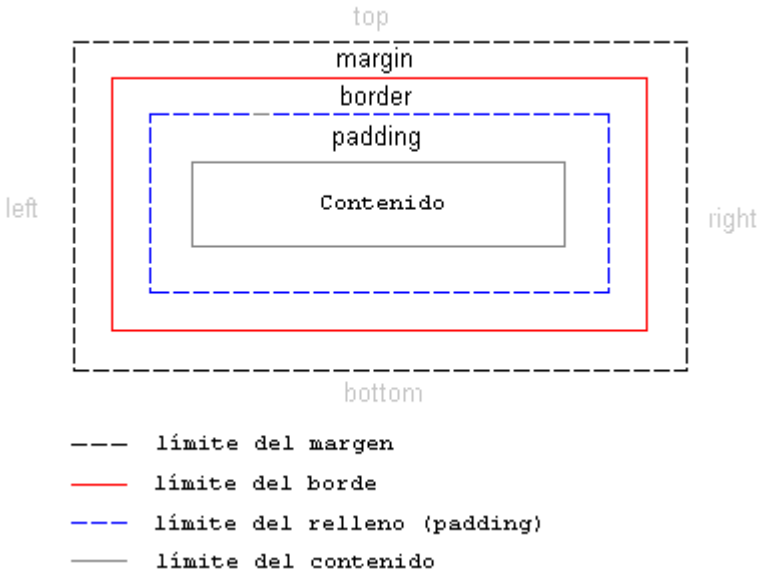
```
a:link {
        color: blue;
        text-decoration:none;
}

a:visited {
        color: purple;
        text-decoration:none;
}

a:active {
        background-color: yellow;
        text-decoration:none;
}

a:hover {
        color:red;
        text-decoration:none;
}
```

10. El modelo de caja.

El modelo de caja en **CSS** describe las cajas que se generan a partir de los elementos **HTML**. El modelo de caja también contiene opciones detalladas en lo referente al ajuste de márgenes, bordes, relleno (padding) y contenido de cada elemento.

La siguiente imagen muestra cómo se construye el modelo de caja:

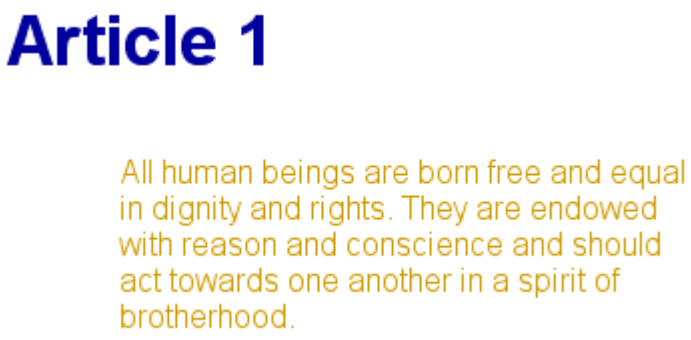
10.1. El modelo de caja en CSS



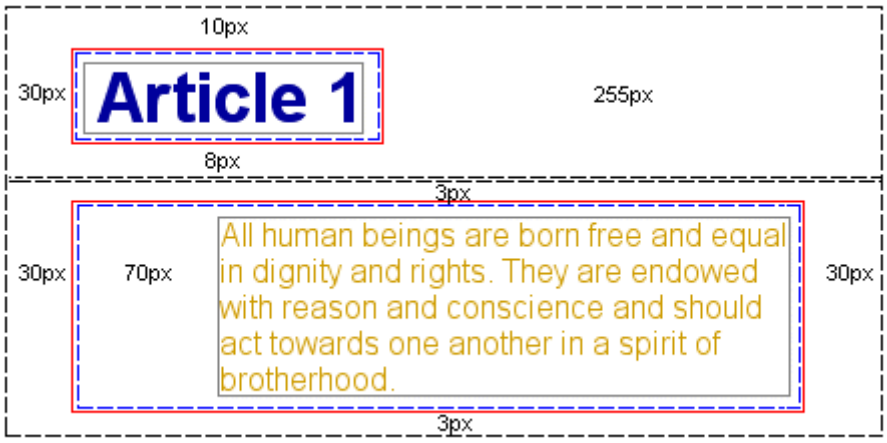
A primera vista, la imagen anterior puede parecer muy teórica, así que intentemos usar el modelo en un caso real con un encabezado y algo de texto. El código **HTML** de nuestro ejemplo es el siguiente (extraído de la Declaración Universal de los Derechos Humanos):

```
<h1>Artículo 1:</h1>

<p>Todos los hombres nacen libres
e iguales en diginidad y derechos. Están
dotados de razonamiento y consciencia y
deberían de comportarse entre sí con
espíritu de hermandad.</p>
```

si añadimos algo de color e información sobre la fuente, el ejemplo se podría presentar así:



El ejemplo contiene dos elementos: el elemento <h1> y el elemento <p>. El modelo de caja para los dos elementos se puede ilustrar como sigue:



Aunque puede parecer un poco complicado, la imagen muestra cómo cada elemento **HTML** está rodeado por cajas, cajas que se pueden ajustar usando **CSS**.

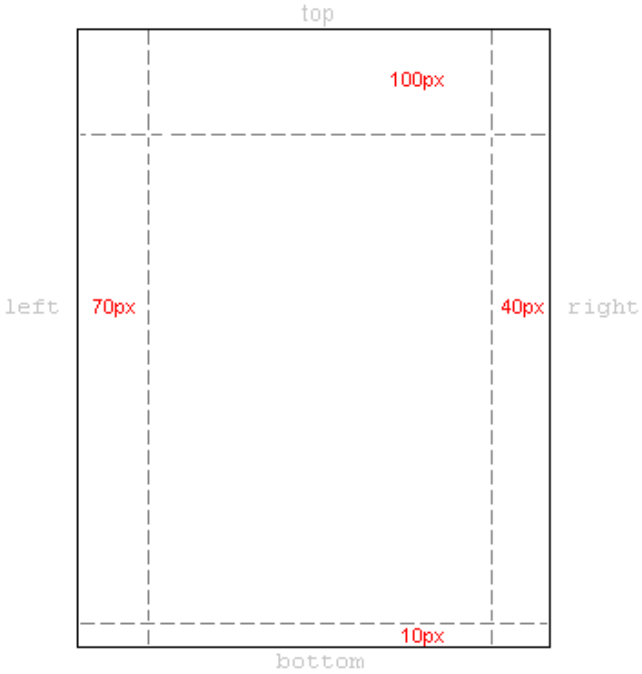
10.2. Margen y relleno (padding)

En el apartado anterior te presentamos el concepto de modelo de caja. En éste, examinaremos cómo cambiar la presentación de los elementos estableciendo las propiedades margin y padding.

10.2.1. Establecer el margen de un elemento

Todo elemento tiene cuatro lados: derecho, izquierdo, superior e inferior. La propiedad margin hace referencia a la distancia desde cada lado respecto al elemento colindante (o respecto a los bordes del documento).

En un primer ejemplo, veremos cómo definir los márgenes del documento en sí, es decir, del elemento <body>. La imagen siguiente muestra cómo queremos que sean los márgenes de nuestras páginas.



El código **CSS** necesario para esto es el siguiente:

```
body {
       margin-top: 100px;
       margin-right: 40px;
       margin-bottom: 10px;
       margin-left: 70px;
}
```

O podrías elegir usar la versión combinada de margin, que queda como más elegante:

```
body {
       margin: 100px 40px 10px 70px;
}
```

Se puede establecer los márgenes de casi todos los elementos del mismo modo.

Por ejemplo, podemos elegir definir márgenes para todos los párrafos de texto marcados con el elemento <p>:

```
body {
       margin: 100px 40px 10px 70px;
}

p {
       margin: 5px 50px 5px 50px;
}
```

10.2.2. Establecer el relleno de un elemento

La propiedad padding puede entenderse como "relleno". Esto tiene sentido puesto que el relleno (padding) no afecta a la distancia de un elemento respecto a otros elementos, sino que sólo define la distancia interior entre el borde y el contenido del elemento.

El uso de la propiedad padding se puede ilustrar viendo un sencillo ejemplo en el que todos los títulos tienen diferentes colores de fondo:

```
h1 {
       background: yellow;
}

h2 {
       background: orange;
}
```

Al definir el padding para los títulos, cambiamos la cantidad de "relleno" que habrá alrededor del texto en cada uno de ellos:

```
h1 {
       background: yellow;
       padding: 20px 20px 20px 80px;
}

h2 {
       background: orange;
       padding-left:120px;
}
```

10.3. Bordes

Los bordes se pueden usar para muchas cosas, por ejemplo, como elemento decorativo o para subrayar la separación entre dos cosas. **CSS** te ofrece opciones sin fin a la hora de usar bordes en tus páginas

10.3.1. Anchura del borde [border-width]

La anchura del borde se define por medio de la propiedad border-width, que dispone de los valores thin, medium y thick, o de un valor numérico indicado en píxeles. La siguiente imagen ilustra cómo funciona el sistema:

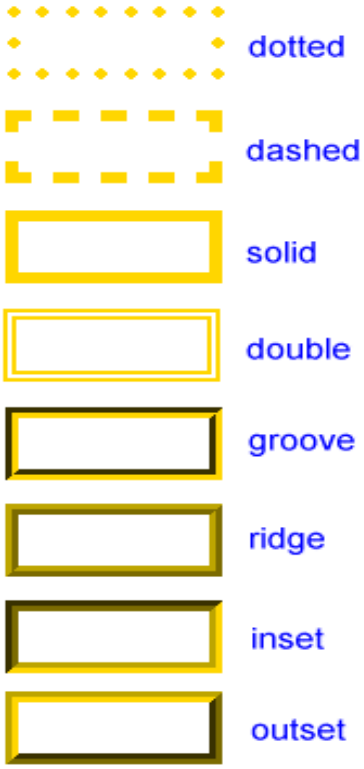
10.3.2. Color del borde [border-color]

La propiedad border-color define el color del borde. Los valores de esta propiedad son los valores de color normales, por ejemplo, "#123456" (en notación hexadecimal), "rgb(123,123,123)" (en notación RGB) o "yellow" (por nombre del color).

10.3.3. Estilo de borde [border-style]

Se puede elegir entre diferentes estilos de borde. Más abajo se muestran 8 estilos de borde según los interpreta Internet Explorer. Todos los ejemplos se muestran con el valor del color a "oro" y el valor de la anchura a "thick", pero se pueden mostrar, por supuesto, en otros colores y grosores.

Si no queremos mostrar ningún borde, se puede usar los valores none o hidden.



10.3.4. Ejemplos de definición de bordes

Las tres propiedades descritas anteriormente se pueden unir para cada elemento y así producir diferentes bordes. Para ilustrar esto, veremos un documento en el que definimos diferentes bordes para los elementos <h1>, <h2>, <ul> y <p>. El resultado puede que no sea demasiado bonito pero ilustra gráficamente algunas de las muchas posibilidades:

```
h1 {
        border-width: thick;
        border-style: dotted;
        border-color: gold;
}

h2 {
        border-width: 20px;
        border-style: outset;
        border-color: red;
}

p {
        border-width: 1px;
        border-style: dashed;
        border-color: blue;
}

ul {
        border-width: thin;
        border-style: solid;
        border-color: orange;
}
```

También es posible declarar propiedades especiales para el borde superior (top), inferior (bottom), derecho (right) e izquierdo (left). En el siguiente ejemplo vemos cómo:

```
h1 {
       border-top-width: thick;
       border-top-style: solid;
       border-top-color: red;

       border-bottom-width: thick;
       border-bottom-style: solid;
       border-bottom-color: blue;

       border-right-width: thick;
       border-right-style: solid;
       border-right-color: green;

       border-left-width: thick;
       border-left-style: solid;
       border-left-color: orange;
}
```

10.3.5. Combinación de propiedades [border]

Como ocurre con muchas otras propiedades, usando la propiedad border se pueden combinar otras muchas propiedades en una sola. Veamos un ejemplo:

```
p {
        border-width: 1px;
        border-style: solid;
        border-color: blue;
}
```

La declaración anterior se puede combinar así:

```
p {
        border: 1px solid blue;
}
```

10.3.6. Altura y anchura

Hasta ahora, no hemos prestado demasiada atención a las dimensiones de los elementos con los que hemos estado trabajando. En este apartado examinaremos lo fácil que es definir la altura y anchura de un elemento. Para lo cual usaremos las propiedades:

10.3.6.1. Estableciendo la propiedad width

Con la propiedad width se puede definir la anchura concreta de un elemento.

El sencillo ejemplo que sigue nos proporciona una caja en la que se puede introducir texto:

```
div.box {
        width: 200px;
        border: 1px solid black;
        background: orange;
}
```

10.3.7. Estableciendo la propiedad height

Fíjate cómo en el ejemplo anterior la altura de la caja queda establecida por el contenido de la misma. Se puede influir en la altura de un elemento con la propiedad height. Por ejemplo, probemos a fijar la altura de la caja en 500px:

```
div.box {
        height: 500px;
        width: 200px;
        border: 1px solid black;
        background: orange;
}
```

10.3.8. Resumen de propiedades:

Es probable que, hasta este momento, hayas estado usando tablas de **HTML** para crear la presentación de tus documentos; con **CSS** y el modelo de caja ya deberías de ser capaz de lograr elegantes presentaciones con mayor precisión y conformes con las recomendaciones del W3C.

- ✓ **Propiedades del margen (margin)**
  - *margin-top*
  - *margin-right*
  - *margin-bottom*
  - *margin-left*
  - *margin*
- ✓ **Propiedades del relleno (padding)**
  - *padding-top*
  - *padding-right*
  - *padding-bottom*
  - *padding-left*
  - *padding*
- ✓ **Propiedades del borde (border)**
  - ➢ **Ancho**
    - *border-top-width*
    - *border-right-width*
    - *border-bottom-width*
    - *border-left-width*
    - *border-width*
  - ➢ **Color (color)**
    - *border-top-color*
    - *border-right-color*
    - *border-bottom-color*
    - *border-left-color*
    - *border-color*
  - ➢ **Estilo (style)**
    - *border-top-style*
    - *border-right-style*
    - *border-bottom-style*
    - *border-left-style*
    - *border-style*
    - *border*

11. Propiedades de clasificación.

11.1. Display (Visualización).

La propiedad display se usa para definir la manera en la que se va a visualizar un elemento.

Puede tomar los siguientes valores:

- ✓ **block:** un salto de línea antes y después del elemento.
- ✓ **inline:** ningún salto de línea antes ni después del elemento.
- ✓ **list-item:** igual que block, salvo que se agrega un marcador de ítems de lista.
- ✓ **none:** sin visualización.

A cada elemento típicamente se le da un valor de display (visualización) por defecto para el navegador, basado en la interpretación sugerida en la especificación HTML.

La propiedad display puede ser peligrosa debido a su capacidad de mostrar elementos en lo que de otro modo sería un formato inapropiado. El uso del valor none desactivará la visualización del elemento al que está asignado, incluyendo cualquier elemento hijo.

11.2. Whitespace (Espacio en blanco).

La propiedad white-space determinará cómo se tratan los espacios dentro del elemento. Esta propiedad toma uno de tres valores:

- ✓ **normal:** contrae múltiples espacios en uno.
- ✓ **pre:** no contrae múltiples espacios.
- ✓ **nowrap:** no permite el ajuste de línea sin una etiqueta
.

11.3. List Style Type (Tipo de estilo de lista).

La propiedad list-style-type especifica el tipo de marcador de ítems de lista, y se usa si list-style-image es none o si la carga de imágenes está desactivada.

Puede tomar los siguientes valores: disc | circle | square | decimal | lower-roman | upper-roman | lower-alpha | upper-alpha | none

- ✓ **Ejemplos:**

```
li.cuadrado { list-style-type: square }
ul.simple { list-style-type: none }
ol      { list-style-type: upper-alpha }  /* a b c d e etc. */
ol ol    { list-style-type: decimal }     /* 1 2 3 4 5 etc. */
ol ol ol  { list-style-type: lower-roman }  /* i ii iii iv v etc. */
```

11.4. List Style Image (Imagen de estilo de lista).

La propiedad list-style-image especifica la imagen que se usará como marcador de ítems de lista cuando se active la carga de imágenes, remplazando al marcador especificado en la propiedad list-style-type.

- ✓ **Ejemplos:**

```
UL.marca { list-style-image: url(/LI-marcadores/visto.gif) }
UL LI.x { list-style-image: url(x.png) }
```

11.5. List Style Position (Posición de estilo de lista).

La propiedad list-style-position toma el valor inside o outside, donde outside es el valor por defecto. Esta propiedad determina donde se coloca el marcador en relación al ítem

de lista. Si se usa el valor inside, las líneas se ajustarán debajo del marcador en vez de estar con sangría. Ejemplo:

- ✓ Posición exterior (outside):
  * Item de lista 1
    segunda línea de ítems de la lista
- ✓ Posición interior (inside):
  * Item de lista 1
    segunda línea de ítems de la lista

11.6. List Style (Estilo de lista).

La propiedad list-style property es una forma rápida de las propiedades list-style-type, list-style-position y list-style-image.

- ✓ **Ejemplos:**

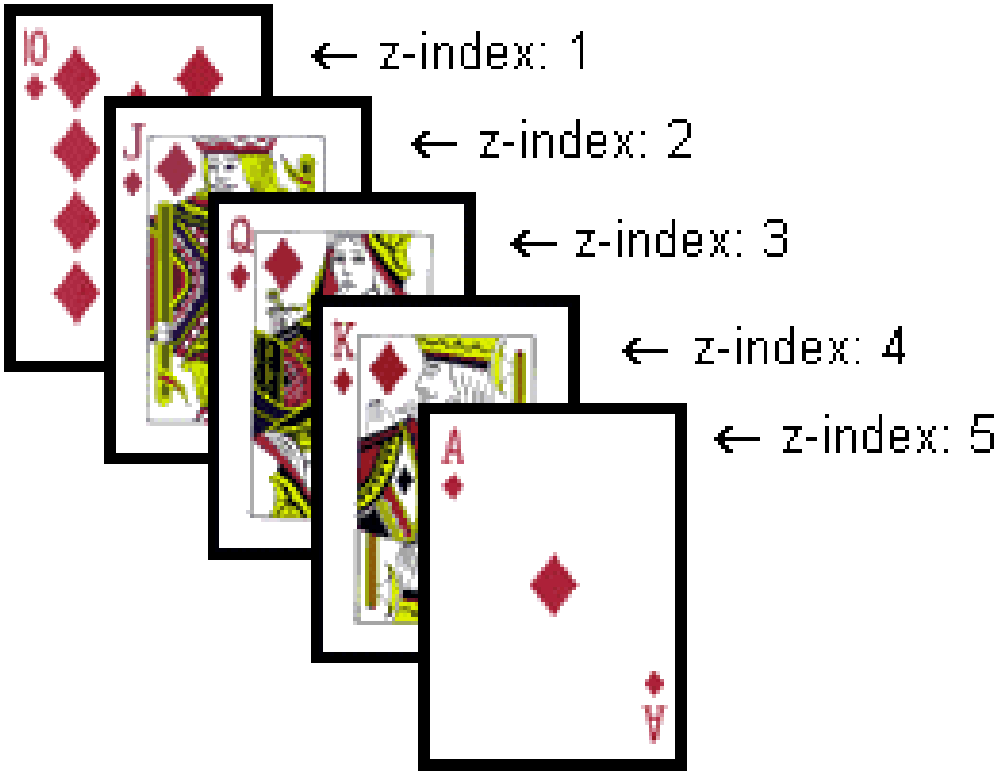
```
li.cuadrado { list-style: square inside }
ul.simple  { list-style: none }
ul.marca  { list-style: url(/li-marcadores/visto.gif) circle }
ol       { list-style: upper-alpha }
ol ol    { list-style: lower-roman inside }
```

12. Capa sobre capa con z-index. (Capas).

CSS funciona sobre tres dimensiones: altura, anchura y profundidad. En las lecciones anteriores hemos examinado las dos primeras dimensiones. En esta lección, aprenderemos cómo hacer que diferentes elementos se conviertan en capas. En pocas palabras, esto hace referencia al orden en que los elementos se superponen unos con respecto a otros.

Para tal propósito, se puede asignar a cada elemento un número por medio de la propiedad z-index. El sistema consiste en que el elemento con un número mayor se superpone al elemento con un número menor.

Supongamos que estamos jugando al póquer y tenemos una escalera de color. La mano se puede presentar de tal manera que cada carta tiene un número asignado por medio de z-index:



En este caso, los números son consecutivos (yendo del 1 al 5), aunque se puede lograr el mismo resultado usando cinco números diferentes. Lo importante es la secuencia cronológica de los números (el orden).

El código del ejemplo de las cartas quedaría así:

```
#diez_de_diamantes {
	position: absolute;
	left: 100px;
	top: 100px;
	z-index: 1;
}

#sota_de_diamantes {
	position: absolute;
	left: 115px;
	top: 115px;
	z-index: 2;
}

#reina_de_diamantes {
	position: absolute;
	left: 130px;
	top: 130px;
	z-index: 3;
}

#rey_de_diamantes {
	position: absolute;
	left: 145px;
	top: 145px;
	z-index: 4;
}

#as_de_diamantes {
	position: absolute;
	left: 160px;
	top: 160px;
	z-index: 5;
}
```

El método es relativamente sencillo pero las posibilidades que ofrece son múltiples. Es posible colocar imágenes sobre el texto, texto sobre imágenes, etc.

Resumen

Las capas se pueden usar en muchas situaciones diferentes. Por ejemplo, intenta usar la propiedad z-index para crear efectos en los títulos en lugar de crear gráficos.

Por un lado, es más rápido cargar texto, y por otro, proporciona una clasificación potencialmente mejor en los motores de búsqueda.