



ISSD

-Asc-

Analista de
Sistemas

CREANDO
ESTILOS...
¡AVANZADOS!

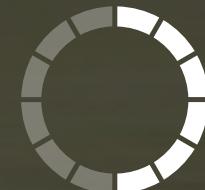
I2

Informática 2

Módulo didáctico - 2015



Unidad 2



Clase 6



(U2) CSS

| Trabajar con Esilos Avanzados aplicándolos en formato externo, es decir, a través de un archivo con extensión .css que enlazaremos a cada una de las páginas para que tomen el aspecto deseado.



| Profundizar sobre el uso de estilos realizando -junto con nosotros- ejercicios para que tus proyectos sean cada vez más profesionales.

Si te piden maquetar los siguientes sitios profesionales, ¿cómo los harías?





Creando estilos... ¡avanzados!

¡Bienvenido! El objetivo de hoy es trabajar con Estilos Avanzados, pero aplicándolos en formato externo, es decir, a través de un archivo con extensión .css que enlazaremos a cada una de las páginas para que tomen el aspecto deseado.

Cuando hayas finalizado la clase de hoy ya dominarás el uso de estilos para realizar efectos que llaman la atención del navegante y el manejo de las pseudoclases y pseudoelementos para manipular objetos completos o parte de ellos dentro de los estilos.

Terminaremos esta clase viendo un montón de ejercicios comentados y detallados paso a paso para incrementar tu habilidad y experiencia con HTML5 y CSS3.

¡Te deseo lo mejor para esta clase!



Desempeño de Exploración

Te proponemos que entre todos codifiquemos los ejercicios desarrollados en clase, lo que dará lugar a exponer nuestras inquietudes y dudas que seguramente nos surgirán a partir de la codificación y visualización de los resultados.



Desempeño de Investigación

Armaremos grupos de investigación para ampliar nuestros conocimientos. Entre todos los alumnos se mostrarán los trabajos realizados para que comentemos y demos nuestra opinión sobre los mismos. Además, se expondrán trabajos explicando los estilos utilizados.

Pseudo-clases

Las :**pseudo-clases** tienen muchas cosas en común con los selectores de atributos. Con ellas podremos apuntar a determinados elementos de nuestro código HTML sin la necesidad de emplear clases ni identificadores.

No debemos olvidar algunas diferencias entre :**pseudo-clases** y ::**pseudo-elementos**.

| Las **pseudo-clases**: Seleccionan elementos HTML que pueden tener añadidas diferentes clases. Se pueden utilizar varias en un mismo elemento.

| Los **pseudo-elementos**: Seleccionan partes del documento que no son elementos HTML, con lo cual no se les pueden aplicar clases. Únicamente admiten un pseudo-elemento por cada selector, éste debe ir colocado al final.

Las :pseudo-clases se escribirán con dos puntos delante ' : **delante ' del nombre**' y los pseudo-elementos con doble dos puntos ' :: **delante**' también del nombre. A continuación veremos algunas ¡prestá mucha atención!

:root

Selecciona el elemento HTML por ser la raíz del documento. Aplica los estilos definidos a todo el documento.

:root{

```
font-family:Arial, Helvetica, sans-serif;  
background-color:#666;  
color:#ccc;
```

```
}
```



:nth-child()

Con esta pseudo-clase podremos seleccionar mediante la posición, elementos hijos de una forma alterna. Con este ejemplo vamos a seleccionar del **elemento padre div**, el hijo situado en la **segunda posición** y aplicar un estilo.

```
#ejemplo :nth-child(2){ color:#0CF;}
```

:nth-last-child()

Con esta pseudo-clase podremos seleccionar mediante la posición, elementos hijos de una forma alterna. **Siendo la posición inicial el último elemento hijo.**

```
#ejemplo :nth-last-child(3){ color:#0CF;  
text-transform:uppercase;}
```

:first-child

Con esta pseudo-clase seleccionamos el primer elemento hijo.

:last-child

Con esta pseudo-clase seleccionamos el último elemento hijo.

Como en los primeros ejemplos, también podemos hacer referencia al último hijo de un determinado elemento.

No me acuerdo qué son los selectores de atributos...



¡Lo vimos la clase pasada! Los selectores de atributos nos permiten entregarle estilos a cualquier elemento que comparta algún atributo específico.



:only-child

Esta pseudo-clase selecciona cuándo hay un único hijo en ese elemento.

:nth-of-type()

Con esta pseudo-clase podremos seleccionar mediante la posición, elementos hijos de una forma alterna pero de un **determinado tipo**.

```
#ejemplo p:nth-of-type(2){  
    color:#CF0;  
}
```

:nth-last-of-type()

Con esta pseudo-clase podremos seleccionar mediante la posición elementos hijos de una forma alterna pero de un **determinado tipo**. Siendo la **posición inicial el último elemento hijo**.

```
#ejemplo p:nth-last-of-type(4){  
    font-weight:bold;  
    color:#FC0;  
}
```

:first-of-type

Con esta pseudo-clase seleccionamos al primer elemento hijo de un determinado tipo.

```
#ejemplo p:first-of-type{  
    text-transform:uppercase;  
    font-weight:bold;  
    color:#9FF;  
}
```

```
<h1>H1 es un título de nuestro  
documento</h1>  
<h2>H2 es otro título de nuestro  
documento</h2>  
<div>  
    <p>P es hijo de su padre DIV</p>  
</div>  
<div id="ejemplo">  
    <p>P es hijo de su padre DIV con  
    indentificado "ejemplo"</p>  
    <p>P es hijo de su padre DIV con  
    indentificado "ejemplo"</p>  
</div>
```

:root

:nth-child()

:nth-last-child()

:first-child

:last-child

):only-child

:nth-of-type()

:nth-last-of-type()

:first-of-type

:only-of-type

:empty



:only-of-type

Esta pseudo-clase selecciona cuándo hay un único hijo de ese tipo en ese elemento de un mismo tipo.

```
a:only-of-type{  
    text-decoration:none;  
    font-weight:bold;  
    color:#6FF;  
}  
  
<div>  
    <a href="">Enlace 1 del primer DIV</a>  
    <a href="">Enlace 2 del primer DIV</a>  
</div>  
  
<div>  
    <h1>Esto es un título para mi  
documento</h1>  
    <h1>Esto es otro título para mi  
documento</h1>  
    <a href="">Enlace único del DIV 2</a>  
</div>  
  
<div>  
    <a href="">Enlace 1 del DIV 3</a>  
    <a href="">Enlace 2 del DIV 3</a>  
</div>
```

:empty

Selecciona un elemento que no tiene hijos. Se considera como hijo también al texto. Es decir, que un elemento que contenga un espacio en blanco no está vacío, por lo tanto no se aplicarán los estilos declarados.

```
:empty{height:100px;    width:100px;  
        background-color:#CFF; }  
  
<div></div>  
<div> </div>  
<div>Este div contiene texto</div>
```

Estilos Externos

En este caso, todos los estilos CSS se incluyen en un archivo de tipo CSS que las páginas HTML enlazan mediante la etiqueta `<link>`. Un archivo de tipo CSS no es más que un archivo simple de texto cuya extensión es `.css`. Se pueden crear todos los archivos CSS que sean necesarios y cada página HTML puede enlazar tantos archivos CSS como necesite. Este archivo contendrá las reglas de estilo (igual como las hemos visto hasta ahora) pero estarán separadas del archivo HTML.

Un archivo de tipo CSS no es más que un archivo simple de texto cuya extensión es `.css`. Se pueden crear todos los archivos CSS que sean necesarios y cada página HTML puede enlazar tantos archivos CSS como necesite.



¿Cuáles son las ventajas de los estilos externos?

| La ventaja fundamental es que con esto podemos aplicar las mismas reglas de estilo a una parte o a todas las páginas del sitio web. Veremos que esto será muy provechoso cuando necesitemos hacer cambios de estilo (cambiando las reglas de estilo de este archivo estaremos cambiando la apariencia de múltiples páginas del sitio).

| También tiene como ventaja que al programador le resulta más ordenado tener lo referente a HTML en un archivo y las reglas de estilo en un archivo aparte.

| Otra ventaja es que cuando un navegador solicita una página, se le envía el archivo HTML y el archivo CSS, quedando guardado este último archivo en la caché de la computadora, con lo cual, en las sucesivas páginas que requieran el mismo archivo de estilos, ese mismo archivo se rescata de la caché y no requiere que el servidor web se lo reenvíe (ahorriendo tiempo de transferencia).

Si se quieren incluir los estilos del ejemplo anterior en un archivo CSS externo, se deben seguir los siguientes pasos:

- 1 | Se crea un archivo de texto y se le añade solamente los estilos que antes definíamos en la zona de <style>
- 2 | Se guarda el archivo de texto con el nombre estilos.css por ejemplo. Se debe poner especial atención a que el archivo tenga extensión .css y no .txt
- 3 | En la página HTML se enlaza el archivo CSS externo mediante la etiqueta <link>:

```
<!DOCTYPE html>
<head>
<link rel="stylesheet" type="text/css" href="/css/estilos.css" media="screen" />
</head>
<body>
<p>Aquí se escribe el texto al que queremos aplicarle el estilo por ejemplo</p>
</body>
</html>
```

Normalmente, la etiqueta <link> incluye cuatro atributos cuando enlaza un archivo CSS:

¿Qué ventajas tiene usar estilos externos?



La ventaja principal radica en que podemos aplicar las mismas reglas de estilo a una parte o a todas las páginas del sitio web.



| rel: indica el tipo de relación que existe entre el recurso enlazado (en este caso, el archivo CSS) y la página HTML. Para los archivos CSS, siempre se utiliza el valor stylesheet.

| type: indica el tipo de recurso enlazado. Sus valores están estandarizados y para los archivos CSS su valor siempre es text/css.

| href: indica la URL del archivo CSS que contiene los estilos. La URL indicada puede ser relativa o absoluta y puede apuntar a un recurso interno o externo al sitio web.

| media: indica el medio en el que se van a aplicar los estilos del archivo CSS. Más adelante se explican en detalle los medios CSS y su funcionamiento.

Observá la imagen 1 para esclarecer este punto.

The diagram illustrates the CSS linking process. At the top, the word "enlace" (link) is written above a code snippet showing a `<link href="css/miestilo.css" rel="stylesheet" type="text/css" />`. Below this, two red arrows point downwards from the "enlace" text to two separate boxes. The left box is labeled "HTML" and contains the following code:

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<link href="css/miestilo.css" rel="stylesheet" type="text/css" />
<title>Documento sin titulo</title>
</head>

<body>
<div id="contenedor">
<p class="texto"> aquí va el contenido</p>
</div>
</body>
</html>
```

The right box is labeled "CSS" and contains the following CSS code:

```
#contenedor{
    width:500px;
    height:200px;
    background:orange;
}

.texto{
    font: bold 12px Arial;
    color: #ffff;
}
```

At the bottom, the word "resultado" (result) is written above a yellow rectangular box containing the text "aqui va el contenido".

Reglas de importancia en los estilos

Los estilos se heredan de una etiqueta a otra, como se indicó anteriormente. Por ejemplo, si tenemos declarado en el <BODY> unos estilos, por lo general, estas declaraciones también afectarán a etiquetas que estén dentro de esta etiqueta, o lo que es lo mismo, dentro de todo el cuerpo.

En muchas ocasiones, más de una declaración de estilos afecta a la misma porción de la página. Siempre se tiene en cuenta la declaración más particular. Pero las declaraciones de estilos se pueden realizar de múltiples modos y con varias etiquetas, también entre estos modos hay una jerarquía de importancia para resolver conflictos entre varias declaraciones de estilos distintas para una misma porción de página. Se puede ver a continuación esta jerarquía, primero ponemos las formas de declaración más generales, y por tanto menos respetadas en caso de conflicto:

- | Declaración de estilos con fichero externo (para todo un sitio web).
 - | Declaración de estilos para toda la página (con la etiqueta <**STYLE**> en la cabecera de la página).
 - | Estilos definidos en una parte de la página (con la etiqueta <**DIV**>).
 - | Definidos en una etiqueta en concreto (utilizando el atributo style en la etiqueta en cuestión).
 - | Declaración de estilo para una porción pequeña del documento. (Con la etiqueta <**SPAN**>).
- El atributo opcional **MEDIO**, especifica el medio o medios en que debería aplicarse la hoja de estilo. Entre los valores posibles podemos encontrar los que observamos en el cuadro.

Descriptor de tipo de medios	Aplicar reglas de hojas de estilo a
"all"	Todos los dispositivos
"aural"	Sintetizadores de voz
"Braille"	Dispositivos táctiles y de braille
"embossed"	Impresoras de páginas de braille
"handheld"	Dispositivos de mano (como PDA)
"print"	Medios de páginas (como una página impresa) o páginas que se muestran en pantalla (como una "presentación preliminar").
"Projection"	Proyectores y dispositivos de impresión en diapositivas y transparencias
"screen"	Pantallas en color de computadoras
"tty"	Dispositivos con cuadrículas de caracteres de punto fijo, como teletipos
"tv"	Dispositivos del estilo de la televisión (baja resolución, color, desplazamiento limitado, dispone de sonido).

Archivos Externos

También podemos incluir estilos en un archivo externo con un código con la sintaxis @import url("estilo.css"). Este se utiliza para definir estilos comunes cuando hay también definición de estilos específicos. Ahora veamos otra manera de importar una declaración externa de estilos CSS: @import url("archivo_a_importar.css"), que se utiliza para importar unas declaraciones de estilos guardadas en la ruta que se indica entre paréntesis. Las comillas son opcionales, pero los paréntesis son obligatorios, por lo menos, en Explorer.

Se debe incluir en la declaración de estilos global a una página, es decir entre las etiquetas `<style type="text/css">` y `</style>`, que se colocan en la cabecera del documento. Es importante señalar que la sentencia de importación del archivo CSS se debe escribir en la primera línea de la declaración de estilos, algo parecido al código siguiente.

```
<style type="text/css">  
@import url ("estilo.css");  
body{  
    background-color:#ffffcc;  
}  
</style>
```

El funcionamiento es el mismo que si escribiésemos todo el fichero a importar dentro de las etiquetas de los estilos, con la salvedad de que, si redefinimos dentro del código HTML (entre las etiquetas `</style>`) estilos que habían quedado definidos en el archivo externo, los que se aplicarán serán los que hayamos redefinido. Así, en el ejemplo anterior, aunque hubiésemos definido en estilo.css un color de fondo para la página, el color que prevalecería sería el definido a continuación de la importación: #ffffcc. La diferencia entre este tipo de importación del tipo y la que hemos visto anteriormente:

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

Acordate que podés incluir estilos en un archivo externo con un código con la sintaxis `@import url ("estilo.css")`. Este se utiliza para definir estilos comunes cuando hay también definición de estilos específicos. En otras palabras, el código con sintaxis `@import url ("estilo.css")` se suele utilizar cuando hay unas pautas básicas en el trabajo con los estilos y unos estilos específicos para cada página.

Es que @import url ("estilo.css") se suele utilizar cuando hay unas pautas básicas en el trabajo con los estilos (que se definen en un archivo a importar) y unos estilos específicos para cada página, que se definen a continuación, dentro del código HTML entre las etiquetas </style>, como es el caso del ejemplo visto anteriormente.

Agregando Contenido a elementos

La propiedad **content** se emplea para generar nuevo contenido de forma dinámica e insertarlo en la página HTML. Como CSS es un lenguaje de hojas de estilos cuyo único propósito es controlar el aspecto o presentación de los contenidos, algunos diseñadores defienden que no es correcto generar nuevos contenidos mediante CSS. El siguiente ejemplo muestra cómo añadir la palabra Capítulo delante del contenido de cada título de sección <h1>:

```
h1:before { content: "Capítulo ";
```

Los **pseudo-elementos** :before y :after se pueden utilizar sobre cualquier elemento de la página. El siguiente ejemplo añade la palabra Nota: delante de cada párrafo cuya clase CSS sea nota:

```
p.nota:before { content: "Nota: ";
```

Los contenidos insertados dinámicamente en un elemento son a todos los efectos, parte de ese mismo elemento, por lo que heredan el valor de todas sus propiedades CSS. Los dos valores más sencillos de la propiedad **content** son **none** y **normal**. En la práctica, estos dos valores tienen el mismo efecto ya que hacen que el **pseudo-elemento** no se genere.

```
#ultimo:after { content: " Fin de los 'contenidos' de la página." ;}
```

Las cadenas de texto sólo permiten incluir texto básico. Si se incluye alguna etiqueta HTML en la cadena de texto, el navegador muestra la etiqueta tal y como está escrita, ya que no las interpreta. Para incluir un salto de línea en el contenido generado, se utiliza el carácter especial \A

La propiedad **content** se emplea para generar nuevo contenido de forma dinámica e insertarlo en la página HTML.

Los dos valores más sencillos de la propiedad **content** son **none** y **normal**. En la práctica, estos dos valores tienen el mismo efecto ya que hacen que el **pseudo-elemento** no se genere.

El siguiente valor aceptado por la propiedad content es una URL, que suele utilizarse para indicar la URL de una imagen que se quiere añadir de forma dinámica al contenido. La sintaxis es idéntica al resto de URL que se pueden indicar en otras propiedades CSS:

```
span.especial:after { content:  
url("imagenes/imagen.png");}
```

Otros valores que se pueden indicar en la propiedad content son **open-quote**, **close-quote**, **no-open-quote** y **no-close-quote**.

Los dos primeros indican que se debe mostrar una comilla de apertura o de cierre respectivamente. Las comillas utilizadas se establecen mediante la propiedad **quotes**:

```
blockquote { quotes: "«" "»" "‘" "’" }  
blockquote:before { content: open-quote; }  
blockquote:after { content: close-quote; }
```

Los valores **no-open-quote** y **no-close-quote** se utilizan para no mostrar ninguna comilla en ese elemento, pero incrementando el nivel de anidamiento de las comillas. De esta forma se puede evitar mostrar una comilla en un determinado elemento mientras se mantiene

la jerarquía de comillas establecida por la propiedad quotes.

Uno de los valores más avanzados de la propiedad content es **attr()**, que permite obtener el valor de un atributo del elemento sobre el que se utiliza la propiedad content. En el siguiente ejemplo, se modifican los elementos `<abbr>` y `<acronym>` para que muestren entre paréntesis el valor de sus atributos title:

```
abbr:after, acronym:after { content: " (" attr(title) ")"}
```

El valor de la propiedad content anterior en realidad es la combinación de tres valores:

- | Cadena de texto "(", que es el paréntesis de apertura tras el cual se muestra el valor del atributo title.
 - | Atributo title del elemento obtenido mediante attr(title)
 - | Cadena de texto ")", que es el paréntesis de cierre que se muestra detrás del valor del atributo title.

Otros valores que se pueden indicar en la propiedad content son: open-quote, close-quote, no-open-quote y no-close-quote. Los dos primeros indican que se debe mostrar una comilla de apertura o de cierre respectivamente.

Los dos segundos se utilizan para no mostrar ninguna comilla en ese elemento, pero incrementando el nivel de anidamiento de las comillas.



Si el elemento no dispone del atributo solicitado, la función attr(nombre_del_atributo) devuelve una cadena de texto vacía. Utilizando attr(), solamente se puede obtener el valor de los atributos del elemento al que se aplica la propiedad content. La función attr() es muy útil, por ejemplo, para mostrar la dirección a la que apuntan los enlaces de la página:

```
a:after { content: " (" attr(href) ")"; }
```

Los últimos valores que se pueden indicar en la propiedad content son los contadores creados con las propiedades counter-increment y counter-reset. Los contadores más sencillos se muestran con la función counter(nombre_del_contador). El siguiente ejemplo crea dos contadores llamado capítulo y sección para utilizarlos con los elementos <h1> y <h2>:

```
body { counter-reset: capitulo; }
h1 { counter-reset: seccion; }
h1:before { content: "Capítulo "
counter(capitulo) ". ";
counter-increment: capitulo; }
```

```
h2:before { content: counter(capitulo) ". "
counter(seccion) "";
counter-increment: seccion; }
```

En el ejemplo anterior, se crea e inicializa su valor a 0 un contador llamado capítulo cuando se encuentre el elemento <body>, es decir, al comienzo de la página. Además, se crea e inicializa su valor a 0 otro contador llamado sección cada vez que se encuentra un elemento <h1> en la página.

Posteriormente, se añade de forma dinámica a los elementos <h1> y <h2> el contenido generado mediante los contadores. Los elementos <h1> utilizan el contador capítulo y lo incrementan en una unidad cada vez que lo utilizan. Los elementos <h2> utilizan los dos contadores para generar un contenido que muestra su numeración completa. Además, los elementos <h2> actualizan el valor del contador sección.

Cuando un mismo elemento inicializa/actualiza un contador y lo utiliza en la propiedad content, en primer lugar se inicializa/actualiza y después, el valor

Uno de los valores más avanzados de la propiedad content, es attr(), que permite obtener el valor de un atributo del elemento sobre el que se utiliza la propiedad content.



actualizado es el que se utiliza mediante counter(). Además de mostrar el valor de un contador básico, la función counter() permite indicar el estilo con el que se muestra el valor del contador. La lista de estilos permitidos son los mismos que los de la propiedad list-style-type.

El siguiente ejemplo modifica el anterior para mostrar el valor de los contadores en números romanos:

```
body { counter-reset: capitulo; }
h1 { counter-reset: seccion; }
h1:before { content: "Capítulo " 
counter(capitulo, upper-roman) ". ";
counter-increment: capitulo; }
h2:before {
content: counter(capitulo, upper-roman) ". "
counter(seccion, upper-roman) " ";
counter-increment: seccion; }
```

Los estilos de los contadores también se pueden emplear para no mostrar el valor de los contadores en algunos elementos:

```
p { counter-increment: parrafos; }
p:before { content: counter(parrafos); }
#especial p:before { content:
counter(parrafos, none); }
```

Aunque el valor de los contadores siempre es numérico, también se pueden emplear estilos gráficos como **square**, **disc** o **circle**:

```
h2 { counter-increment: seccion; }
h2:before { content: counter(seccion, disc); }
```

La función counter() solamente muestra el valor de un contador. Por su parte, la función counters() se utiliza para mostrar de forma simultánea el valor de todos los contadores asociados con el elemento. Como se explica en la descripción de las propiedades counter-increment y counter-reset, los contadores se pueden anidar y un mismo elemento puede tener asociados varios contadores diferentes con el mismo nombre.

El siguiente ejemplo muestra unas reglas CSS que crean un contador para los elementos de una lista :

```
ol { counter-reset: elemento; list-
style-type: none; }
li:before { content: counter(elemento)
". "; counter-increment: elemento; }
```

Si se considera el siguiente código HTML:

```
<ol>
<li>Elemento</li>
<li>Elemento</li>
<li>Elemento
<ol>
<li>Elemento</li>
<li>Elemento
<ol>
<li>Elemento</li>
<li>Elemento</li>
<li>Elemento</li>
</ol>
</li>
<li>Elemento</li>
</ol>
</li>
<li>Elemento</li>
</ol>
```



Si tuvieras que elegir una propiedad ¿cuál sería la más interesante que ofrece CSS3?

Tal vez la capacidad de crear y configurar el efecto de sombra... Con CSS3 podés elegir el color, tamaño, desenfoque y desplazamiento de la sombra.

Si se aplican las reglas CSS al código HTML anterior, se crean tres contadores diferentes con el mismo nombre (elemento). Sin embargo, si se utiliza la función counters() en las reglas CSS anteriores:

```
ol { counter-reset: elemento; list-style-type: none;}
```

```
li:before { content: counters(elemento, '. ') ".";
counter-increment: elemento; }
```

En el ejemplo anterior, cada vez que se encuentra un elemento , se crea un contador llamado elemento. Por este motivo, los elementos anidados se ven afectados

por varios contadores llamados elemento. La función counter() sólo muestra el valor del contador que afecta más directamente al elemento, mientras que la función counters() muestra todos los contadores empezando desde el más externo hasta llegar al más interno.

El segundo argumento de la función counters() es una cadena de texto que se emplea para separar los valores de los diferentes contadores.

Por último, la función counters() también permite indicar el estilo con el que se muestra el valor de los contadores. De esta forma, el siguiente ejemplo modifica el anterior para mostrar el valor de todos los contadores en números romanos:

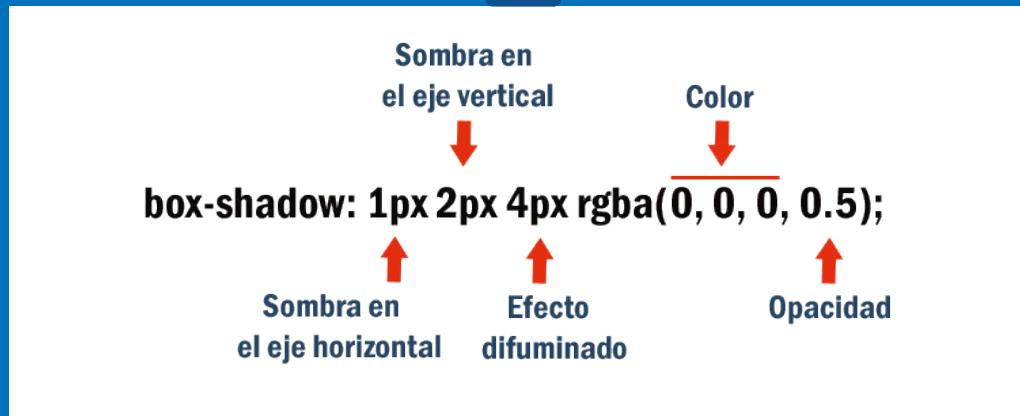
```
ol { counter-reset: elemento; list-style-type: none;}
```

```
li:before { content: counters(elemento, ' ', upper-roman) ". ";
counter-increment: elemento; }
```

Sombras

Una de las propiedades interesantes que nos ofrece CSS3 es la capacidad de crear y configurar el efecto de sombra. Las sombras se pueden aplicar tanto a bloques, como a texto. Entre otras cosas, podremos elegir el color, tamaño, desenfoque y desplazamiento de la sombra (drop shadow).

La sombra paralela más simple y más



frecuente probablemente sea una sombra gris que se extiende algunos píxeles a partir del cuadro, con algo de desenfoque.

| **box-shadow:** nos permite crear un sombreado todos los elementos HTML (elementos soportados como bloques "Divs").

| **text-shadow:** nos permite crearle sombras a los textos a los que apliquemos esta propiedad.

Por ejemplo si agregamos una sombra con desplazamiento vertical y horizontal de solo cinco píxeles, además de un ligero desenfoque de 5 píxeles sin difusión de color gris claro codificaremos la siguiente línea:

box-shadow: 5px 5px 5px lightgray;

Donde tendremos la sintaxis que te mostramos en la imagen 1

Por orden de aparición, los valores que se indican en **box-shadow** son:

| **Desplazamiento horizontal de la sombra:** La sombra de un elemento suele estar un poco desplazada con respecto al elemento que la produce y su posición será en función del ángulo con el que llegue la luz. Si quisieramos que la sombra apareciera un poco hacia la izquierda del elemento original que la produce, pondríamos un valor negativo a este atributo. Cuanto más desplazamiento tenga una sombra, el elemento que la produce parecerá que está más separado del lienzo de la página.

| **Desplazamiento vertical de la sombra:** El segundo valor que colocamos en el atributo **box-shadow** es el desplazamiento vertical de la sombra con respecto a la posición del elemento que la produce. Este valor es similar al desplazamiento horizontal. Valores positivos indican que la sombra aparecerá hacia abajo del elemento y valores negativos harán que la sombra aparezca desplazada un poco hacia arriba.

| **Difuminado:** El tercer valor indica cuánto queremos que esté difuminado el borde de la sombra. Si el difuminado fuera cero, querría decir que la sombra no tiene ningún difuminado y aparece totalmente definida.

Creando efecto de sombra a div caja 1

Box Shadow

Desplazamiento horizontal eje X Desplazamiento vertical eje Y Desenfoque Radial Color
box-shadow: 5px 5px 5px #888;
-webkit-box-shadow: 5px 5px 5px #888;
-moz-box-shadow: 5px 5px 5px #888;

text-shadow: 0px 0px 0px #FFF;

| Color de la sombra: El último atributo que se indica en el atributo box-shadow es el color de la sombra. Generalmente las sombras en el mundo real tienen un color negro o grisáceo, pero con CSS3 podremos indicar cualquier gama de color para hacer la sombra, lo que nos dará bastante más versatilidad a los diseños gracias a la posible utilización de sombras en distintos colores, que puedan combinar mejor con nuestra paleta.

Veamos un ejemplo,

```
#caja1{-Webkit-box-shadow: 5px 5px 5px #888;  

-Moz-box-shadow: 5px 5px 5px # 888;  

box-shadow: 5px 5px 5px # 888; }
```

Lo que dà como resultado lo que observamos en la imagen 1

Este ejemplo siguiente es para una sombra un poco menor, también desplazada hacia abajo y a la derecha y con un difuminado de 2 píxeles. Además hemos indicado un color amarillo claro para la sombra, por lo que, para verla bien, tendríamos que colocar este elemento sobre un fondo oscuro.

Texto con sombra

CSS tiene una propiedad que permite añadir sombra al texto. Lleva cuatro argumentos: el color de la sombra, el desplazamiento horizontal (donde positivo significa hacia la derecha), el desplazamiento vertical (donde positivo significa hacia abajo) y el desenfoque (0 significa una sombra nítida).

Por ejemplo:

```
h3 { text-shadow: red 0.2em 0.3em 0.2em }
```

```
.elemento { box-shadow: 2px 2px 5px #999; }
```

Observá también la imagen 2

CSS tiene una propiedad que permite añadir sombra al texto. Ésta propiedad lleva cuatro argumentos: el color de la sombra, el desplazamiento horizontal (donde positivo significa hacia la derecha), el desplazamiento vertical (donde positivo significa hacia abajo) y el desenfoque (0 significa una sombra nítida).

| La primera medida es obligatoria e indica el desplazamiento horizontal de la sombra. Si el valor es positivo, la sombra se desplaza hacia la derecha y si es negativo, se desplaza hacia la izquierda.

| La segunda medida también es obligatoria e indica el desplazamiento vertical de la sombra. Si el valor es positivo, la sombra se desplaza hacia abajo y si es negativo, se desplaza hacia arriba.

| La tercera medida es opcional e indica el radio utilizado para difuminar la sombra. Cuanto más grande sea su valor, más borrosa aparece la sombra. Si se utiliza el valor 0, la sombra se muestra como un color sólido.

| La cuarta medida también es opcional e indica el radio con el que se expande la sombra. Si se establece un valor positivo, la sombra se expande en todas direcciones. Si se utiliza un valor negativo, la sombra se comprime.

| El color indicado es directamente el color de la sombra que se muestra.

Las opciones del **filtro shadow** son mucho más limitadas que las de la propiedad **box-shadow**. Su sintaxis es la habitual de los filtros de Internet Explorer y las opciones son:

| **Color**: establecido mediante el formato hexadecimal (ejemplo: #CC0000).

| **Direction**: dirección hacia la que se desplaza la sombra. Su valor se indica en grados y sólo se permiten los valores 0, 45, 90, 135, 180, 225, 270 y 315.

| **Strength**: distancia en píxeles hasta la que se extiende la sombra.

A continuación se modifica la regla CSS anterior para incluir el filtro de Internet Explorer que muestra una sombra similar:

```
.elemento { -webkit-box-shadow: 2px 2px  
5px #999;  
-moz-box-shadow: 2px 2px 5px #999;  
filter: shadow(color=#999999,  
direction=135, strength=2);}
```

Texto con sombra borrosa

La forma más sencilla de la propiedad “**text-shadow**” tiene dos partes: un color y un desplazamiento. El resultado es una sombra nítida con el desplazamiento indicado. Pero también puede hacerse que el desplazamiento sea impreciso, lo que da por resultado una sombra más o menos borrosa. La cantidad de imprecisión se da como otro desplazamiento. Aquí se muestran dos líneas, una con un poco de imprecisión (0.05em) y la otra con mucha imprecisión (0.2em):

```
h3.a {text-shadow: 0.1em 0.1em 0.05em  
#333}  
h3.b {text-shadow: 0.1em 0.1em 0.2em black}
```

Texto blanco legible

El uso de sombras puede hacer más legible el texto cuando hay poco contraste entre el primer plano y el fondo. Lo que sigue es un ejemplo de texto blanco contra fondo azul pálido, primero sin sombra y luego con ella:

```
h3 {color: white}  
h3.a {color: white; text-shadow: black 0.1em  
0.1em 0.2em}
```

Varias sombras

También es posible poner más de una sombra. En general, el resultado es un tanto extraño:

```
h3 {text-shadow: 0.2em 0.5em 0.1em #600,  
-0.3em 0.1em 0.1em #060, 0.4em -0.3em  
0.1em #006}
```

Pero colocando acertadamente dos sombras, una oscura y la otra clara, el efecto puede ser útil:

```
h3.a {text-shadow: -1px -1px white, 1px 1px  
#333}  
h3.b {text-shadow: 1px 1px white, -1px -1px  
#333}
```

Esto es un poco peligroso, como podrá apreciar si su navegador no admite la propiedad “text-shadow”. De hecho, en este ejemplo los colores del fondo y del texto son casi iguales (#CCCCCC y #D1D1D1), de modo que sin las sombras apenas existe algún contraste.

Un gran beneficio del uso de sombras es que puede hacer más legible el texto cuando hay poco contraste entre el primer plano y el fondo.



Texto contorneado

Este texto tiene un pixelart

Trazar letras como contornos

El ejemplo con dos sombras de la versión previa se puede llevar todavía más allá. Con cuatro sombras, es posible dar a las letras un contorno (1).

```
h3 {text-shadow: -1px 0 black, 0 1px black,
1px 0 black, 0 -1px black}
```

Brillo de neón

Si se pone una sombra borrosa justo detrás del texto, es decir, con desplazamiento igual a cero, el efecto es un resplandor alrededor de las letras. Si el resplandor de una sola sombra no es suficientemente intenso, basta repetir la misma sombra unas pocas veces:

```
h3.a {text-shadow: 0 0 0.2em #8F7}
h3.b {text-shadow: 0 0 0.2em #F87, 0 0
0.2em #F87}
h3.c {text-shadow: 0 0 0.2em #87F, 0 0
0.2em #87F, 0 0 0.2em #87F}
```

Efecto Pixelart

Con un poco más de imaginación podemos conseguir efectos de lo más diversos. En este caso hemos hecho una prueba que da un resultado de diseño "pixelart", de aquellos gráficos creados píxel a píxel de los juegos de antaño (2).

```
h1.pixelart{
  text-shadow: 1px 1px #666, 2px 2px
```

```
#86D6D3, 3px 3px #666, 4px 4px #86D6D3;
color: #ccc;}
```

Efecto de fuego

Si usamos varias sombras de colores anaranjados podemos conseguir un efecto de fuego. Nos toca hacer un poco de prueba y ensayo para conseguir un resultado realista, pero se puede conseguir algo interesante.

```
h2.fuego{
  text-shadow: 0 0 20px #fefcc9, 2px -2px
3px #feec85, -4px -4px 5px #ffae34, 5px
-10px 6px #ec760c, -5px -12px 8px #cd4606,
0 -15px 20px #973716, 2px -15px 20px
#451b0e; color: #666;}
```

Linear - gradient



Orientación X de donde partira el degradado Orientación Y
 background-image: linear-gradient(left top , #11324b, #f4faff 450px);
 color 1 color 2
 Desplazamiento del degradado

Radial - gradient



Forma del degradado
 ubicación Y ubicación X
 Tamaño
 Color Inicial Color Final
 background: radial-gradient(center top , circle , #ccc, #555 184px);

Degradado

Los degradados implementan un gradiente de color, que pasa de un estado a otro a lo largo del fondo de los elementos HTML, ya sea capas, elementos de listas, botones, etc. Dichos degradados se obtendrán por medio de la especificación de una serie de características, como la posición inicial, la dirección hacia donde se realizará, si es circular o lineal, y los colores que se incorporarán en cada uno de los pasos del gradiente. Podemos definir un degradado así:

```
background-image: linear-gradient(izquierda
arriba , de negro, a blanco , la mitad)
```

Degradado lineal

Son aquellos en los que se crea un degradado que va de un color a otro de manera lineal. Puede ser de arriba a abajo, de izquierda a derecha y viceversa. Incluso se puede conseguir un degradado lineal con cualquier ángulo. Mirá el código y luego la imagen 1.

elemento hacia fuera, de manera circular, que puede tener el mismo valor de radio (para hacer degradados en círculos perfectos) o con valores de radio variables (lo que generaría elipses). El valor que asignamos a background en este caso será por medio del atributo radial-gradient, además de toda la serie de parámetros necesarios para definir el degradado según nuestras intenciones. Observá el código y luego la imagen 2 para tener un ejemplo.

Degradado circular

En ellos se implementa un degradado que se distribuye radialmente, desde un punto del

Los degradados implementan un gradiente de color, que pasa de un estado a otro a lo largo del fondo de los elementos HTML, ya sea capas, elementos de listas, botones, etc. Existen los degradados lineales y los circulares. Los primeros son aquellos en los que se crea un degradado que va de un color a otro de manera lineal (por ejemplo, de arriba a abajo). En los segundos se implementa un degradado que se distribuye radialmente, desde un punto del elemento hacia fuera, de manera circular.

Para asegurarse que un degradado funciona en la mayoría de las plataformas, de momento estamos obligados a escribir las reglas de estilos con etiquetas propietarias para cada navegador. O sea, para definir un degradado deberíamos escribir todos estos estilos:

```
background: -webkit-linear-gradient(orange, pink);
background: -moz-linear-gradient(orange, pink);
background: -o-linear-gradient(orange, pink);
background: linear-gradient(orange, pink);
```

Transparencia

El valor de la propiedad **opacity** se establece mediante un número decimal comprendido entre 0.0 y 1.0. La interpretación del valor numérico es tal que el valor 0.0 es la máxima transparencia (el elemento es invisible) y el valor 1.0 se corresponde con la máxima opacidad (el elemento es completamente visible). De esta forma, el valor 0.5 corresponde a un elemento semitransparente y así sucesivamente.

En el siguiente ejemplo, se establece la propiedad opacity con un valor de 0.5 para conseguir una transparencia del 50% sobre dos de los elementos <div>:

```
#segundo, #tercero { opacity: 0.5; }
```

```
#primero { background-color: blue; }
```

```
#segundo { background-color: red; }
```

```
#tercero { background-color: green; }
```

Los **filtros** permiten aplicar operaciones complejas a los elementos de la página. Los filtros de Internet Explorer se dividen en estáticos y de transición. Los primeros se utilizan básicamente para crear efectos gráficos sobre los elementos, normalmente imágenes. Entre ellos se encuentra el filtro alpha, que permite controlar la opacidad de un elemento de la página. La sintaxis completa del filtro **alpha** es muy compleja porque sus



¡Me perdí! ¿Para qué sirven los filtros?

Los filtros permiten aplicar operaciones complejas a los elementos de la página. Los filtros de Internet Explorer se dividen en estáticos y de transición.



posibilidades son numerosas. No obstante, la sintaxis necesaria para establecer solamente el nivel de transparencia de un elemento es muy sencilla:

```
#segundo, #tercero { filter:  
alpha(opacity=50);}
```

El valor de la opción **opacity** del filtro **alpha** se establece mediante un número entero comprendido entre 0 (el elemento es invisible) y 100 (el elemento es completamente opaco). El valor 50 del ejemplo anterior hace que el elemento sea semitransparente. A continuación se muestra la solución compatible con todos los navegadores para que un elemento de la página sea semitransparente:

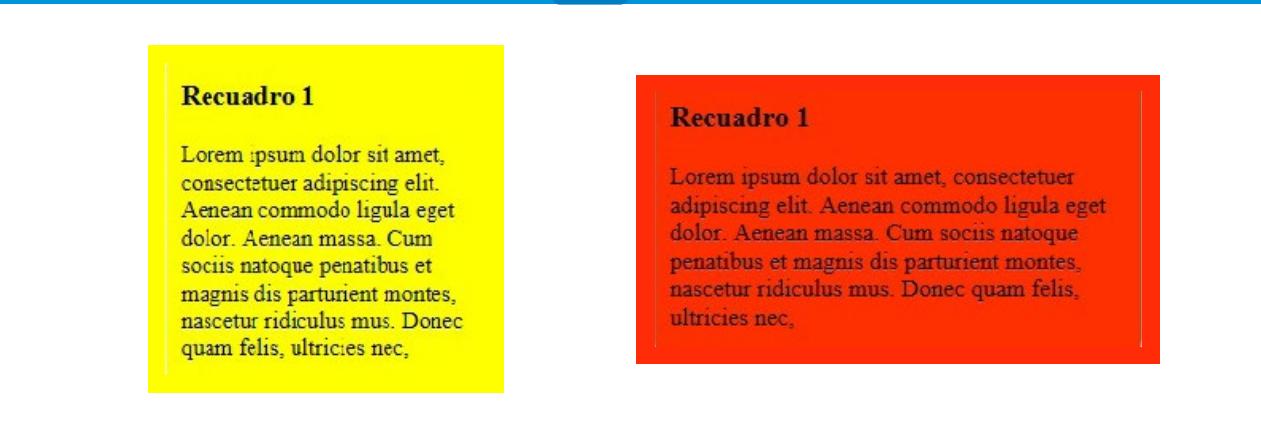
```
selector { opacity: 0.5; filter:  
alpha(opacity=50);}
```

Transición

Una transición es un cambio progresivo de un elemento. Aplicado a los estilos CSS significa que dentro de una propiedad, para pasar de un valor a otro, esto se puede hacer de manera progresiva, pasando por los valores intermedios. Esto la mayoría de las veces provoca un efecto de animación, sobre todo en propiedades que tienen que ver con el tamaño, la posición o el color. Las transiciones sólo se dan en las propiedades CSS cuyo valor tenga un componente numérico (número, medidas, colores, tiempo).

Las transiciones en CSS3 son creadas por una serie de propiedades que controlarán los elementos, las propiedades CSS y el tiempo, entre otras cosas. También podemos hacer transiciones de múltiples propiedades, para ello indicamos cada transición separada por coma:

```
Elemento {  
transition: [nombre de propiedad]  
[duración de la transición],  
[nombre de propiedad] [duración  
de la transición],  
[nombre de propiedad] [duración  
de la transición] ; }
```



Por ejemplo si queremos que el recuadro modifique su ancho y cambie de color luego, debemos codificar lo siguiente:

Como vemos en la propiedad **transition** indicamos la propiedad **width** con una duración de un segundo y la propiedad **background-color** con un valor de 8 segundos. Esto significa que cuando dispongamos la flecha del mouse dentro del div, se lanzarán ambas transiciones que tienen duraciones distintas (cambiará de 200 píxeles a 300 píxeles en el lapso de un segundo y también cambiará del color amarillo al rojo en forma gradual en un lapso de 8 segundos). Un ejemplo es el que vemos en la imagen 1

El ejemplo completo sería así:

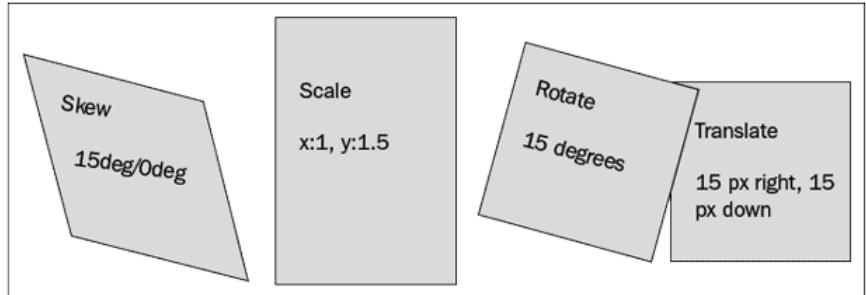
Las transiciones de CSS3 pueden tomar estos valores:

- | **transition-property.**
- | **transition-duration.**
- | **transition-delay.**
- | **transition-timing-function.**

`transform:rotate(45deg)`



Transformaciones básicas con CSS



Transformaciones

De las transformaciones de CSS3 en 2D, las más usadas son:

| **Rotate**: nos permite rotar un elemento dándole un ángulo de giro en grados.

| **Scale**: nos permite escalar un elemento, toma valores positivos y negativos y se le pueden poner decimales.

| **Translate**: nos permite trasladar un elemento a la vez en el eje de las X y de las Y, dándole las coordenadas iniciales y finales.

Podés verlas en la imagen 1

A continuación detallaremos cada una de ellas.

Rotar

Para la rotación de un objeto se puede utilizar **transform:rotate(grados)** como lo podés ver en la imagen 2, también nos permite rotar solo un eje cuando se trata de un objeto 3D utilizando **rotateX(grados)**, **rotateY(grados)** o **rotateZ(grados)**.

```
#rotacion{ -webkit-transition: all 1s ease-in-out;
-moz-transition: all 1s ease-in-out;
-o-transition: all 1s ease-in-out;
transition: all 1s ease-in-out; }
```

```
#rotacion:hover{-webkit-transform: rotate(360deg);
-moz-transform: rotate(360deg);
-o-transform: rotate(360deg);
-ms-transform: rotate(360deg);
transform: rotate(360deg); }
```

Ejemplo

```
{ -webkit-transform: rotate (45deg);
-moz-transform: rotate (45deg);
-o-transform: rotate (45deg);
-ms-transform: rotate (45deg);
transform: rotate (45deg); }
```

Fíjate que los grados se marcan en positivo si el elemento se rota en el sentido de las agujas del reloj y en negativo si es al revés.

transform: skew(15deg, 4deg);

15° en X 4° en Y

transformaciones en los angulos
15 en X y 4 en Y

transform: skew(15deg, 4deg);
-webkit-transform: skew(15deg, 4deg);
-moz-transform: skew(15deg, 4deg);
-ms-transform: skew(15deg, 4deg);
-o-transform: skew(15deg, 4deg);

Tambien utilizaremos los Prefijos
de cada navegador

```
div {  
background: #ccc;  
box-shadow: 2px 2px 3px #777;  
margin: 0 auto;  
transform: skew(15deg, 4deg);  
-webkit-transform: skew(15deg, 4deg);  
-moz-transform: skew(15deg, 4deg);  
-ms-transform: skew(15deg, 4deg);  
-o-transform: skew(15deg, 4deg);  
width: 350px;  
}
```

4deg 0 4°
X

Transform - Skew

15deg 0 15° Y

Sesgar

Esta función nos va a permitir inclinar o aplicar un tipo de perspectiva a un elemento en el documento HTML. Como en los demás, tenemos que especificar las coordenadas de los ángulos para su inclinación, (X, Y) el primero X que se aplica horizontalmente e Y que se aplica verticalmente.

Para sesgar un objeto debés utilizar **transform: skew(angulo x, anguloy)** o al igual que las anteriores también permite sesgar solo un eje utilizando **skewX(angulo)** o **skewY(angulo)**. Mirá los códigos y luego las imágenes 1 y 2 que corresponden a cada uno.

```
#sesgo{-webkit-transition: all 1s ease-in-out;  
-moz-transition: all 1s ease-in-out;  
-o-transition: all 1s ease-in-out;  
transition: all 1s ease-in-out;}
```

```
#sesgo:hover{-webkit-transform: skew(25deg,180deg);  
-moz-transform: skew(25deg,180deg);  
-o-transform: skew(25deg,180deg);  
-ms-transform: skew(25deg,180deg);  
transform: skew(25deg,180deg);}
```

Ejemplo 2

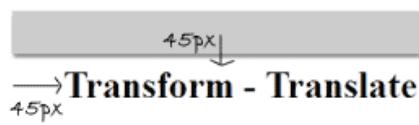
```
div { background: #ccc; box-shadow: 2px 2px 3px #777;  
margin: 0 auto; transform: skew(15deg, 4deg);  
-webkit-transform: skew(15deg, 4deg);  
-moz-transform: skew(15deg, 4deg);  
-ms-transform: skew(15deg, 4deg);  
-o-transform: skew(15deg, 4deg);  
width: 350px; }
```

```

background: #ccc;
box-shadow: 2px 2px 3px #777;
width: 350px;
}

h1{
-webkit-transform: translate(45px,45px);
-moz-transform: translate(45px,45px);
-ms-transform: translate(45px,45px);
-o-transform: translate(45px,45px);
transform: translate(45px,45px);
}

```



Desplazamiento

Para mover un elemento es necesario utilizar **transform: translate(desplazamiento-x, desplazamiento-y)** o si solo querés desplazar el objeto por un eje, podes trabajar con **translateX()** o **translateY()**. Observá el siguiente código y luego la imagen 1

```
#desplazar{-webkit-transition: all 1s ease-in-out;  
-moz-transition: all 1s ease-in-out;  
-o-transition: all 1s ease-in-out;  
transition: all 1s ease-in-out;}
```

```
#desplazar:hover{-webkit-transform:  
translate(250px, 20px);  
-moz-transform: translate(250px, 20px)}
```

```
div {  
    transform: scale(1.5, 0.75);  
}  
  
        ← escala en x  
        ↓ escala en y
```

Tambien funcionan con los
Prefijos de los Navegadores

-webkit-transform:
-moz-transform:
-ms-transform:
-o-transform:

Si deseas aplicar varias transformaciones a la vez solo tienes que encadenarlas, el orden en el que les apliques las transformaciones puede hacer que el efecto cambie, veremos un ejemplo a continuación:

Escalar

Otra función de la propiedad transform es **scale**, esta función nos va a permitir escalar un elemento tanto horizontal como verticalmente en torno a su origen y lo vamos a hacer por medio de su (x,y) . Otra vez, mirá el código y luego la imagen **2**

```
h1{ transform: scale(1.9, 1);  
-webkit-transform: scale(1.9, 1);  
-moz-transform: scale(1.9, 1);  
-ms-transform: scale(1.9, 1);  
-o-transform: scale(1.9, 1); }
```



Ejemplos y ejercicios comentados

De ahora es más
expondremos ejercicios
resueltos, comentados
y detallados en forma
completa aplicando todo lo
visto de CSS3 y HTML5.

Te aconsejo que practiques
y codifiques los ejemplos ya
que esto es un buen método
para aprender el código.



Listas con CSS

En este ejercicio utilizaremos una **lista, pseudo-clases y pseudo-elementos**.

Debemos recordar que los pseudo-elementos van con doble dos puntos delante de la palabra y las pseudo-clases con dos puntos únicamente. Veamos la imagen 1.

Para comenzar con el ejemplo crearemos una lista.

```
<ul>
  <li></li>
  <li></li>
  <li></li>
  <li></li>
  <li></li>
  <li></li>
</ul>
```

Ahora pasaremos a codificar los estilos para cambiar el aspecto de la lista. Primero aplicaremos algunos estilos al **body**:

```
body{ background-image:url(bg.png);
  background-color: hsla(0, 0%, 7%, 1);
  font-family:Verdana, Geneva, sans-serif;
  font-size:13px; color:#666;
  counter-reset:circulo; }
```

Ahora voy a crear los círculos y el contador aplicando algunos estilos a la **etiqueta li** directamente y otros mediante el pseudo-elemento **:before**

```
li{list-style-type: none; margin-top:25px;
  counter-increment:circulo; }
li:before{ padding:5px; padding-left:9px;
  padding-right:9px;
  margin-right:20px; background-color:#CFF;
  border-radius: 20px 20px 20px 20px;
  content: counter(circulo);
  text-shadow:0 0 6px #000; box-shadow:2px
  2px 5px #000; color:#000;}
```

De estos estilos destacamos **counter-increment: circulo;** que será nuestro contador incrementando en 1 cada uno de los **elementos li** de nuestra lista desordenada.

- 1 Ejemplo de lista de elementos
- 2 Ejemplo de lista de elementos
- 3 Ejemplo de lista de elementos
- 4 Ejemplo de lista de elementos
- 5 Ejemplo de lista de elementos
- 6 Ejemplo de lista de elementos

¡No te olvides que los pseudo-elementos van con doble dos puntos delante de la palabra y las pseudo-clases con dos puntos únicamente!

Título de la tabla			
...	Columna 1	Columna 2	Columna 3
Fila 1	celda 11	celda 12	celda 13
Fila 2	celda 21	celda 22	celda 23
Fila 3	celda 31	celda 32	celda 33

Indice de Precios de Consumo						
	2012M03	2012M02	2012M01	2012M03	2012M02	2012M01
General	101,055	100,38	100,275	0,7	0,1	-1,1
Alimentos y bebidas	101,574	101,472	101,357	0,1	0,1	0,3
Alcohólicas y tabaco	102,658	102,606	102,285	0,1	0,3	0,2
Vestido y calzado	95,131	91,604	93,232	3,9	-1,7	-14,4
Vivienda	102,154	101,908	101,729	0,2	0,2	0,5
Menaje	100,33	100,059	100,118	0,3	-0,1	-0,9
Medicina	97,704	97,709	97,648	0	0,1	-0,1

Después de esto nos queda únicamente añadir el texto para cada uno de los elementos de la lista y algunos estilos para que se muestren más elegantes en pantalla.

En este caso una sombra para el texto y un efecto :hover para que el texto cambie de color una vez que pongamos nuestro ratón encima. No debemos olvidar que el valor de esta lista se suele dar mediante código HTML.

```
li:after{content:"Ejemplo de lista de
elementos";
text-shadow:2px 2px 6px #000;}

li:hover{color:#fff;}
```

Aplicando CSS a tablas

Ejemplo 01

Con el desarrollo de este ejemplo podremos profundizar el cambio de aspecto de una simple tabla para poderla convertir en un objeto más estético. A continuación se detalla el código html generando la estructura básica de la tabla. Veremos cómo aplicar los estilos a los distintos elementos de la tabla.

La resultante será lo que vemos en la imagen 1

Selecciona tu Plan	Selecciona tu Plan	Selecciona tu Plan	Selecciona tu Plan
PLAN 1	PLAN 2	PLAN 3	PLAN 4
\$10 / mes	\$20 / mes	\$40 / mes	\$50 / mes
• 100MB de Capacidad	• 300MB de Capacidad	• 600MB de Capacidad	• 1GB de Capacidad
• 1GB de Transferencia	• 3GB de Transferencia	• 6GB de Transferencia	• 12GB de Transferencia
• Configuración gratuita	• Configuración gratuita	• Configuración gratuita	• Configuración gratuita
• 1 Cuenta FTP	• 6 Cuenta FTP	• 12 Cuenta FTP	• 20 Cuenta FTP
• 1 Cuenta Email	• 6 Cuenta Email	• 12 Cuenta Email	• 20 Cuenta Email
• 1 Base de Datos	• 6 Base de Datos	• 12 Base de Datos	• 20 Base de Datos
• Dominio	• Dominio	• Dominio	• Dominio
<button>Seleccionar</button>	<button>Seleccionar</button>	<button>Seleccionar</button>	<button>Seleccionar</button>

Ejemplo 02

Con este código repasaremos el uso de bordes redondeados y pseudo-elementos.

Ejemplo 03

Desarrollo de la hoja de Estilo.

Código que se vé como lo muestra la figura 1

Mi Sitio Web

[Blog](#) [Quiénes somos](#) [Contacto](#)

Tené en cuenta que para que las sombras puedan verse, el color de fondo no puede ser negro.

Menú brillante

El cuerpo de este ejemplo es muy negro, codificaremos lo siguiente.

```
<body>
  <h1>Mi Sitio Web</h1>
  <a href="" class="opcion_iluminada">Blog</a>
  <a href="" class="opcion_iluminada">Quiénes somos</a>
  <a href="" class="opcion_iluminada">Contacto</a>
</body>
```

Los estilos creados son los siguientes:

```
body { background: #444; }

h1 { font-size: 38px; font-style: italic;
  color: #333; padding: 10px;
  text-shadow: -1px -1px 0px #101010, 1px
  1px 0px #505050;
}

.opcion_iluminada { font-size: 20px;
color:#999;
text-decoration:none; padding: 10px; }

.opcion_iluminada:hover { color:#fff;
  text-shadow: 0px 0px 15px #fff; }

</style>
```

Un dato importante, para que las sombras puedan verse, el color de fondo, en este caso de la página (elemento body, propiedad background), no puede ser negro porque entonces no se vería la sombra. Nosotros lo hemos establecido a #444, un gris muy oscuro, pero sin llegar a ser negro. Andá mirando la imagen 1 así entendés mejor nuestra descripción.

El efecto de bajo relieve del título (h1) lo conseguimos dándole un color más oscuro que el fondo (#333) y aplicando dos sombras, separadas por una coma, en la definición de la propiedad text-shadow:

Sombra más oscura en la parte superior e izquierda: -1px -1px 0px #101010
Sombra más clara en la parte inferior y derecha: 1px 1px 0px #505050

Respecto al menú, cada opción tiene asignado el estilo opcion_iluminada y cuando el puntero del ratón se sitúa encima se aplica el estilo opcion_iluminada:hover. Este es un ejemplo muy sencillo, el menú se puede construir de otras formas.

En el estilo de opcion_iluminada (cuando el puntero del mouse no está sobre la opción), simplemente hemos establecido el color del texto a #999, un gris. De esta forma, cuando el puntero del mouse se sitúe encima de la opción, se aplicará el estilo de opcion_iluminada:hover y el color del texto será más intenso, concretamente blanco (#fff). Esto ya dará la sensación de que la opción se ilumina, pero el resultado no estará muy logrado. Lo vamos a mejorar añadiendo una sombra blanca para crear el efecto de resplandor mediante la propiedad text-shadow y el valor:

0px 0px 15px #fff

Los valores deberán ajustarse según el tamaño del texto y los enlaces del menú.

Como alternativa de este ejercicio y menú, pasaremos a desarrollar otro ejemplo a continuación. Comenzaremos a codificar el menú con una lista adentro de un contenedor, podés desarrollar el mismo en la zona de <nav>

```
<div id="menu">
  <ul>
    <li><a href="#">Inicio</a></li>
    <li><a href="#">Conecta</a></li>
    <li><a href="#">Descubre</a></li>
    <li><a href="#">Cuenta</a></li>
  </ul>
</div>
```

Los estilos están formados de la siguiente manera:

```
#menu{      background-color: #fff;
            height: 41px; overflow: hidden;
            padding-left: 100px; }

#menu ul{ padding: 0; margin: 0; }

#menu ul li{ border-bottom: solid 10px orange;
              display: inline-block; list-style: none;
              padding: 10px 0 20px 0;
              -webkit-transition: all 0.3s;
              -moz-transition: all 0.3s;
              transition: all 0.3s; }

#menu ul li:hover{ padding: 10px 0 10px 0; }

#menu ul li a{ color: black;
                padding: 10px 25px;
                text-decoration: none; }
```

Menú Multinivel

Trabajaremos con una lista con clase de "nav", que almacenara los ítems del menú:

Comenzaremos a definir los estilos para este menú, recuerda de enlazarlo en forma externa como se ve al comienzo de esta clase. Empezaremos con un reset básico y algo de decoración para el menú:

Para empezar, vamos a definir los estilos para este menú, recuerda de enlazarlo en forma externa como se ve al comienzo de esta clase. Iniciaremos con un reset básico y algo de decoración para el menú:

Los submenús se desplegaran cuando el usuario pase el cursor, estos irán como listas dentro de listas (ul > li > ul). Este ejemplo funcionara con cualquier cantidad de submenús, eso quiere decir que puedes incluir los niveles de menús que quieras, en este caso tendremos 3 niveles:

Se especifica que solo los **li** que sean descendientes directos del primer **ul** tengan **float: left**, esto es para que solo el menú principal sea horizontal y los submenús se mantengan en vertical, tal como se muestra en la imagen 1

Home	Servicios	Acerca	Contacto
Diseno grafico	Historia		
Diseno web	Mision		
Submenu 1	Vision		
Submenu 2			
Submenu 3			
Submenu 4			
Submenu 5			
Marketing			

Al comienzo esto es
¡muy complicado!
pero la única forma de
aprender es haciendo
juntos los ejercicios.



Por defecto todos los submenús no serán visibles, los ocultaremos usando **display: none**.

```
.nav li ul { display:none; position:absolute; min-width:140px; }
```

Todos los submenús tendrán un ancho mínimo de 140px para que no se vean desiguales y llevenan **position: absolute** para que no afecten el ancho del menú principal. Proseguimos con la parte que hará que se muestre el submenú oculto:

```
.nav li:hover > ul{display: block; }
```

En este código le estamos indicando que, al igual que en la imagen 1, cuando el cursor pase sobre cualquier **li** su descendiente **ul** se muestre (**display: block**).

El problema ahora es, como podés ver en la imagen 2, que los submenús de segundo nivel en adelante se están mostrando pero no como deberían.

Lo que tenemos que hacer es que estos

se muestren a la derecha de su respectivo ancestro **li**:

```
.nav li ul li {position:relative; }
```

```
.nav li ul li ul { right:-140px; top:0; }
```

Los submenús de segundo nivel tendrán **right: -140px**, para empujarlos hacia la derecha, es importante notar que este valor es el mismo que el que definimos anteriormente como ancho mínimo, y además tendrá **top: 0** esto es para que se posicione al mismo nivel que su ancestro **li** que tiene **position: relative**.

Como podés observar en la imagen 3, esto afectara todos los submenús de segundo nivel en adelante.

Añadir un indicador a los elementos con submenú

Lo primero será añadir el elemento que hará de indicador al lado de cada enlace del menú, en este caso estamos usando una flechita (si prefieres usar una imagen también funcionara) su código HTML es ▼



2



3



38
ISSD

Ahora lo que haremos será definir que, por defecto, esta flecha este oculta en todos los elementos del menú:

```
.nav li .flecha{font-size: 9px;  
padding-left: 6px; display: none; }
```

También le reduce el tamaño de la fuente y lo alejé un poco del texto para que se vea bien. La parte esencial de esto es la siguiente: vamos a usar la psuedo-clase CSS3 :not en combinación con :last-child para excluir a todos los elementos del menú que no contengan submenús (todos los <a> que no

sean :last-child) y solo afectar a los **span**.

Mirá el código y luego la imagen 1 para que te quede más claro.

```
.nav li a:not(:last-child) .flecha {  
display: inline;}
```

Menú vertical

Siguiendo las mismas técnicas para la construcción de un menú, te presentamos este ejercicio donde se muestra un menú vertical con efecto en cada una de sus opciones.

Home	Servicios ▾	Acerca ▾	Contacto
	Diseno grafico		
	Diseno web ▾		
Marketing ▾	Submenu 1		
SEO	Submenu 2		
	Submenu 3 ▾	Submenu 1	
		Submenu 2	
		Submenu 3	
		Submenu 4	



Efecto en imágenes

Ejemplo 01

Comenzaremos con el código html

El código HTML es básico y fácil de entender, tenemos una lista `` asociada a dos clases, dentro tenemos los respectivos ``, dentro tenemos la etiqueta `<figure>` de html5 que es la encargada de contener todo el contenido a mostrar, después tenemos un `<div>` y adentro mostramos la imagen ``, aparte tenemos la etiqueta `<figcaption>` donde mostraremos la información adicional, es decir, el caption, dentro tenemos un título en `<h3>` un `` para mostrar el texto y un enlace `<a>`. Definimos los estilos utilizados:

```
.contenedor { padding: 20px 20px  
100px 20px;  
max-width: 1200px; margin: 0 auto;  
list-style: none; text-align: center; }  
  
.contenedor li {  
display: inline-block; width: 363px;  
margin: 0; padding: 20px; text-align:  
left;  
position: relative; }
```

A los `` y el/los ``, le damos las proporciones y dimensiones en las que se van a mostrar, los `` al final imagen la mostramos con **display: inline-block** por si tenemos muchas se ubicaran una al lado de la otra, con un ancho de 363px para que encajen de a tres por líneas en caso de tener varias.

En este ejercicio -siguiendo las mismas técnicas que para la construcción de un menú- te vamos a mostrar un menú vertical con efecto en cada una de sus opciones.



Aplicamos los estilos de cada elemento que compone el ítem, es decir, la etiqueta **figure**, la imagen, la etiqueta **figcaption**, el texto que está dentro de un span, el h3 y el enlace. Estos son estilos para que se vean bien y para posicionarlos, hasta ahora esto es lo que vemos.

Aplicamos un poco de perspectiva con la propiedad **perspective** y **perspective-origin**, con esto determinamos la intensidad del efecto 3d y cambiar la posición de origen interior del elemento que tendrá efecto 3d.

```
.caption li { -webkit-perspective:  
1800px;  
-moz-perspective: 1800px; perspective:  
1800px;  
-webkit-perspective-origin: 0 50%;  
-moz-perspective-origin: 0 0%;  
perspective-origin: 0 0%;}
```

Básicamente le decimos al navegador que estamos trabajando con características 3d y por ahora ocultamos el div que tenemos dentro de figure

```
.caption figure {  
-webkit-transform-style: preserve-3d;  
-moz-transform-style: preserve-3d;  
transform-style: preserve-3d;  
}  
.caption figure > div {  
overflow: hidden;  
}
```

Aplicamos animación a la imagen dentro del ítem, le decimos que se mueva 23% de su ubicación en el eje x con la propiedad **translateX**, esta se moverá de izquierda a derecha y tendrá una transición que aplicamos con la propiedad transition.

Las animaciones CSS3 permiten hacer muchas de las cosas que antes teníamos reservadas sólo al uso de tecnologías supletorias, que no hacían más que incrementar la dificultad del desarrollo, limitar su compatibilidad entre distintos tipos de usuarios y plataformas, así como los requisitos de conocimientos del desarrollador para poder incorporarlas.



Ajustamos la altura al 100% de nuestro caption y lo dejamos invisible con la propiedad **opacity** en 0, con **backface-visibility: hidden** nos aseguramos que al momento de girar el elemento en efecto 3d la otra cara del elemento quede oculta ya que lo estamos rotando -90 grados y en su punto de origen lo dejamos en 0 0. Luego en el estado **hover**, es decir cuando sitúe el mouse sobre él lo hacemos totalmente visible cambiando su opacidad, cambiamos la rotación y aplicamos la transición a todos los cambios para dar la sensación de movimiento.

Y finalmente aplicamos los estilos al enlace que está dentro del **figcaption** que al final será un botón que tendrá un enlace, lo posicionamos absolutamente y aplicamos estilos de **background** y **transicion** en el estado hover.

```
.caption figcaption a {  
position: absolute;  
bottom: 20px; right: 20px; }
```

```
.btn_hover:hover {  
background: #3fc7e8;  
-webkit-transition: all 0.2s ease-in;  
-moz-transition: all 0.2s ease-in;  
-o-transition: all 0.2s ease-in;  
}
```

Y eso es todos, tenemos listo nuestro efecto caption con css3 para utilizar en proyectos, para mostrar nuestro portafolio personal, mostrar información de productos etc, tal como lo vemos en la imagen 1.

Transiciones CSS3

Las transiciones con CSS son efectos o animaciones que se aplican a los elementos al pasar de un estilo a otro. Con estas transiciones podemos suavizar el cambio de estilos sin la necesidad de utilizar javascript o flash. Antes de nada, es necesario indicar que para aplicar transiciones, debemos tener en cuenta varias cosas, la primera es la propiedad: **transition-property**.

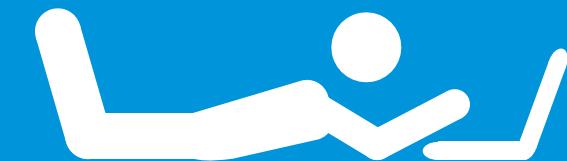
Comenzaremos a construir los estilos, para este ejemplo:

Ahora crearemos un marco para introducir unos botones:

```
.marco{  
    background-color:hsla(250, 1%, 22%,  
1);  
    background-image:url(bg-marco.  
png);  
    box-shadow: -1px 1px 1px hsla(180,  
1%, 30%, 1);  
    height:25px; width:400px;  
    padding:20px;  
    border: hsla(0, 0%, 0%, 1) 2px solid;  
  
    border-radius:5px 5px 5px 5px;  
    margin-top:20px;  margin-  
bottom:20px; }
```

Ahora crearemos un botón al que le aplicaremos las transiciones:

Recordá que las transiciones con CSS son efectos o animaciones que se aplican a los elementos al pasar de un estilo a otro. Con estas transiciones podemos suavizar el cambio de estilos sin la necesidad de utilizar javascript o flash.



Tenemos que tener en cuenta que muchas de estas propiedades todavía no funcionan de una forma correcta en todos los navegadores, por eso debemos emplear los prefijos para los navegadores CSS3. Definiremos el estado hover del botón:

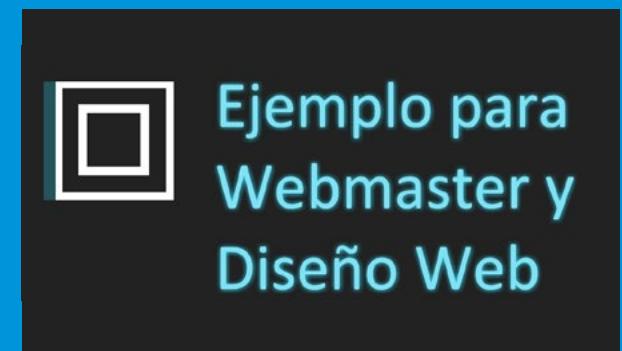
Después de tener ya todo creado, puedes incrementar el número de botones en el marco para generar un lindo menú.

Efecto Luz de Neón

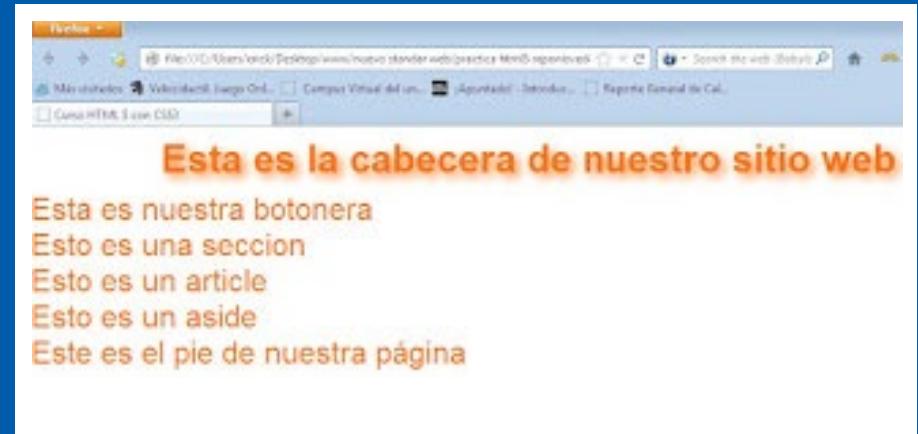
Con el siguiente ejemplo trabajaremos las sombras, transiciones y fondos de manera tal que de por resultado el famoso efecto luz de neón. El código se muestra a continuación y podés ver el resultado en la imagen [1](#)

Formatear zonas de Html5

Comenzaremos a desarrollar el siguiente código en HTML5.



Siempre tené en cuenta que muchas de las propiedades que estamos aplicando todavía no funcionan de manera correcta en todos los navegadores. Para evitar problemas debés usar los prefijos para los navegadores CSS3.



A continuación definiremos los estilos seteando la zona de trabajo con márgenes:

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

Ahora estableceremos la tipografía de nuestro sitio web. Para eso tenemos que utilizar el atributo de CSS3 que es **@font-face** y lo que le vamos a establecer a este atributo son 2 mandatos: el tipo de letras y el estilo normal. Esta es otra alternativa para trabajar con fuentes. Mirá el código y luego la imagen 1.

```
@font-face {font-weight:normal; font-style:normal;}
```

```
body{ background: #FFF; color:#F60;  
font-family:Helvetica, Verdana;  
font-size: 2em; }
```

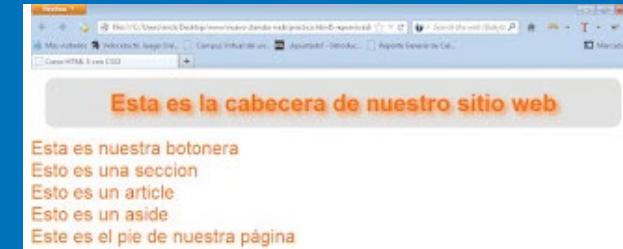
```
h1{ color:#F60; margin:0.25em auto;  
text-align:center;  
text-shadow: 5px 5px 10px  
rgba(255,255,255,0.5); }
```

En este código lo primero que establecimos fue el color de las letras del `<h1>`, en este caso (`color:#F60;`) que corresponde al naranja. Luego el margin es el espacio que va a tener respecto de los demás elementos. También establecimos el `text-align:center`; esto es

para alinear el texto al centro y algo muy importante es el atributo (`text-shadow:`). Observá la imagen 2 para comprender bien todas las propiedades que venimos aplicando.

Nuestro código para el `<header>` es el siguiente:

```
header { background:#DEDEDE;  
border-radius:0.5em; margin: 0.5em auto;  
max-width:960px; padding:0.25em; text-align:center; }
```



Lo que dá como resultado lo que se observa en la imagen 1.

Otro ejemplo es el del siguiente código, del cual podemos observar su aspecto en la imagen 2.

```
header ,nav,footer{ background:#DEDEDE;
border-radius:0.5em; margin: 0.5em auto;
max-width:960px; padding:0.25em; text-align:center; }
```

Por otro lado, el código CSS3 para el section contenedor sería el siguiente:

```
section#contenedor{ margin:0.2em auto;
max-width:960px;
padding:0; text-align:center; }
```

Ahora seguimos con el **section#principal**. A este le decimos que tenga un background o color de fondo, un **border-radius** para que tenga bordes curvos, y otro atributo que también podemos añadir con CSS3 es **display:inline-block**; éste le indica a nuestro section que todo lo que venga dentro de él será visto como bloques. También les añadimos el atributo **vertical-align:top** -que

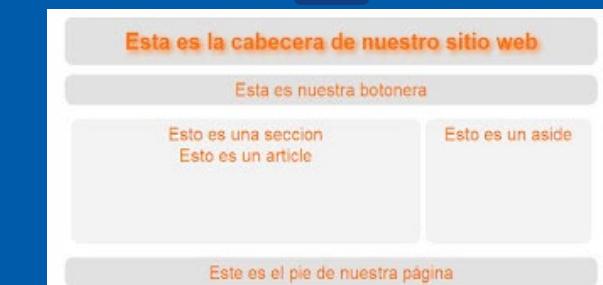
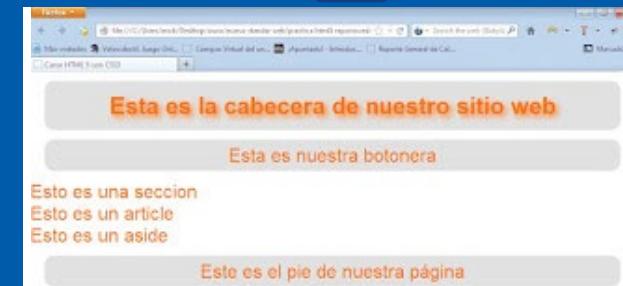
viene a sustituir al float del css anterior- y le indica a nuestro section que alinee todo verticalmente y que sea en la parte superior. Luego les indicamos que tome un tamaño de un 65% del section. Mirá el código y luego las imágenes 3 y 4 para aclarar todo lo dicho.

```
section#principal, aside{
background:#F1F1F1; border-radius:0.5em;
display: inline-block; margin:0.25em auto;
max-width:960px;
min-height:200px; padding:0.25em; text-align:center;
vertical-align:top; width:65%; }
```

Luego,

```
aside{ width:30%; }
```

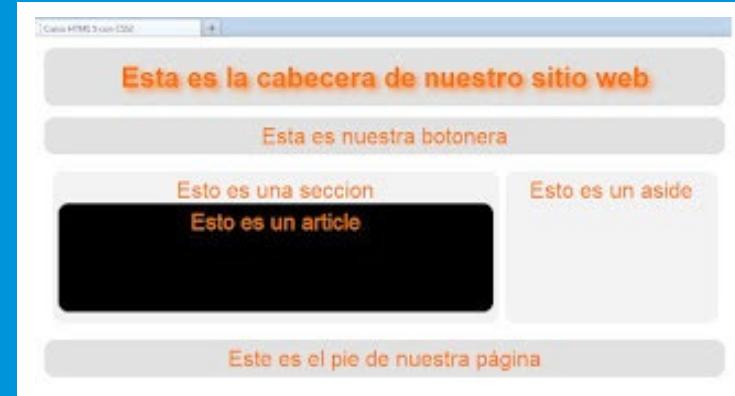
Ahora sólo nos falta darle estilo al **article**, este será sencillo: solo tenemos que ponerle un color de fondo distinto a fondo actual, para ello le ponemos un **background:#000**, que corresponde al negro, luego un **border-radius** para las esquinas curvas y también le agregamos un alto mínimo con el atributo **min-height:** de 140px; y un **padding:** de 0.25em. Con esto quedaría listo nuestro



article, tal como se observa en la imagen 1

```
article{ background:#011E30; border-radius:  
0.5em;  
min-height:140px; padding:0.25em; }
```

Si te fijas en el navegador reduciendo el tamaño de la ventana verás como todo se ajusta a su tamaño, lo cual sucede porque trabajamos con los tamaños en (em) y no en (px) también por los atributos **max-width** y **min-height**. Nuestro código completo sería:



Muchas veces, la barrera más grande que impide que algunas personas se involucren con la programación es que piensan que se necesita tener un perfil “matemático”, o piensan que es muy difícil. Luego de haber realizado con nosotros todos estos ejercicios comentados tal vez ya habrás notado que la programación es una herramienta para sacar las ideas que tenés en tu mente al mundo exterior, es decir, es un proceso absolutamente creativo.



Desempeños

Para que te queden bien claros los contenidos de esta clase, codificá los ejemplos que vimos a lo largo de ella. Esta será tu tarea para este día
¡Mucha suerte y buen trabajo!



Autoevaluación

Ahora respondé a estas preguntas:

- 1 |** ¿Cuál fue el primer navegador que introdujo CSS?
- 2 |** ¿Las hojas de estilo son sensibles a las mayúsculas y a las minúsculas?
- 3 |** ¿Es mejor usar la medida en píxel (px) o en puntos (pt)?
- 4 |** Investigá y aplicá las siguientes propiedades: transition-property, transition-duration, transition-delay, transition-timing-function.
- 5 |** ¿Cuáles son las diferencias entre Box-shadow y drop-shadow?



¡Felicitaciones, llegaste al final de una nueva clase! Los contenidos de hoy fueron eminentemente prácticos con el objetivo de que se vayan fijando los conocimientos que aprendemos. Los ejercicios comentados te van a ser de mucha utilidad ya que permiten practicar e ir chequeando si todo va bien. Esperamos que esta clase te haya resultado entretenida y útil.

Hasta la próxima!