Curso de HTML5

El futuro de la web

Versión 1.0

2012

Prólogo

Este curso está orientado a la introducción de los nuevos tags (etiquetas) y atributos de la última actualización de HTML y CSS, con la intención de que el lector gradualmente vaya conociendo el gran avance tecnológico que ésta implica.

Poco a poco se ha ido gestionando lo que es el desarrollo de una nueva experiencia de la programación web y estamos seguros de que esto nunca se detendrá y seguirá en constante movimiento.

Sin embargo ha sido un proceso lento y sometido a exhaustivos estudios y pruebas que aún siguen ejecutándose tanto por parte de los desarrolladores del lenguaje como los desarrolladores de los navegadores en los cuales este último debe ejecutarse. Al parecer los grandes desarrolladores de navegadores no se han puesto de acuerdo acerca de los códec que el navegador debería soportar nativamente para interpretar el contenido de las etiquetas, por lo tanto si queremos probar la potencia de HTML5 Y CSS3, debemos tomar las medidas necesarias, actualizar los navegadores que tengamos instalados en nuestras computadoras e instalar algunos otros.

Por otra parte garantizando el aprendizaje inmediato de nuestros lectores, se comenzó una intensiva investigación acerca del correcto funcionamiento del lenguaje en cuestión, consultando importantes webs que de manera gratuita han abierto el portal del conocimiento a los fanáticos de la innovación.

Esperamos que esta primera versión del **Curso de HTML5 y CSS3** sea el primer paso para la continuidad del estudio de tan importantes e innovadoras herramientas.

Índice

Introducción a HTML5

¿Qué es HTML?

HTML (*HyperText Markup Language*) es un lenguaje muy sencillo que permite describir hipertexto, es decir, texto presentado de forma estructurada y agradable, con *enlaces* (*hyperlinks*) que conducen a otros documentos o fuentes de información relacionadas, y con *inserciones* multimedia (gráficos, sonido...) La descripción se basa en especificar en el texto la estructura lógica del contenido (títulos, párrafos de texto normal, enumeraciones, definiciones, citas, etc) así como los diferentes efectos que se quieren dar (especificar los lugares del documento donde se debe poner cursiva, negrita, o un gráfico determinado) y dejar que luego la presentación final de dicho hipertexto se realice por programas especializados conocidos como Browsers o navegadores web (Firefox, Opera, Chrome, etc.)

Estructura Básica de un documento HTML

Un documento escrito en HTML contendría básicamente las siguientes etiquetas:

`<HTML>` Indica el inicio del documento.
`<HEAD>` Inicio de la cabecera.
`<TITLE>` Inicio del título del documento.
`</TITLE>` Final del título del documento.
`</HEAD>` Final de la cabecera del documento.
`<BODY>` Inicio del cuerpo del documento.
`</BODY>` Final del cuerpo del documento.
`</HTML>` Final del documento.

¿Qué es HTML5?

Es la quinta revisión importante del estándar que mueve internet, HTML y que ha sido impulsado por el aumento de las necesidades de mejorar la funcionalidad y la interactividad de usuarios y aplicaciones. **HTML 5** presenta una amplia gama de mejoras en controles para formularios, **APIs**, multimedia y también en la **estructura** y la **semántica**. El trabajo en **HTML 5** comenzó en el 2004 y es un esfuerzo conjunto entre el **W3C HTML WC** y el **WHATWG**. Muchos aportan su participación, como Apple, Mozilla, Opera, Microsoft y un sinnúmero de personas y empresas.

La aparición de estas revisiones implicará cambios importantes en el desarrollo de páginas web y algunos se aventuran incluso a prever la desaparición de algunas tecnologías de gran calado en la red, como puede ser Adobe Flash. Cierto o no, la realidad es que estamos presenciando una evolución radical de los lenguajes principales de la web que

aportará grandes mejoras, pero que al mismo tiempo implicará un periodo de adaptación y aprendizaje por parte de programadores y diseñadores.

Novedades de la estructura:

Empezamos con decir que el **DOCTYPE** es el encargado de indicarle al navegador web que el documento que está abriendo es un documento HTML, además de indicar que es un documento HTML también le indica la versión del HTML del mismo con el fin de renderizar la pagina de manera correcta para mostrarla de la mejor manera posible.

Este es un ejemplo del doctype de la versión anterior.-

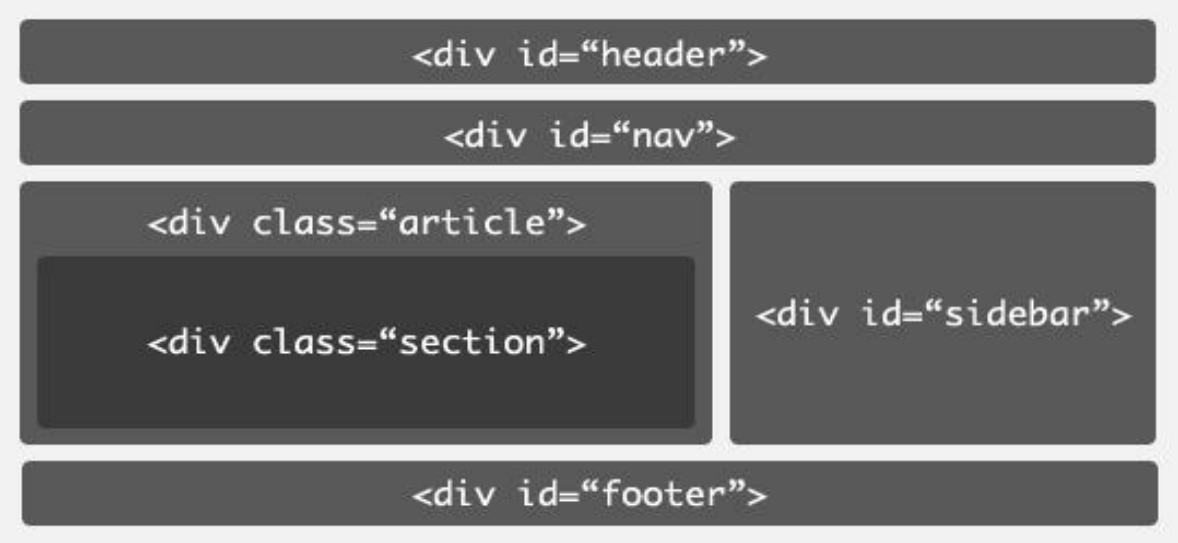
```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
```

Ahora con la llegada de HTML5, el doctype es bastante sencillo y compatible con las versiones anteriores de HTML y xHTML, esta es su sintaxis.-

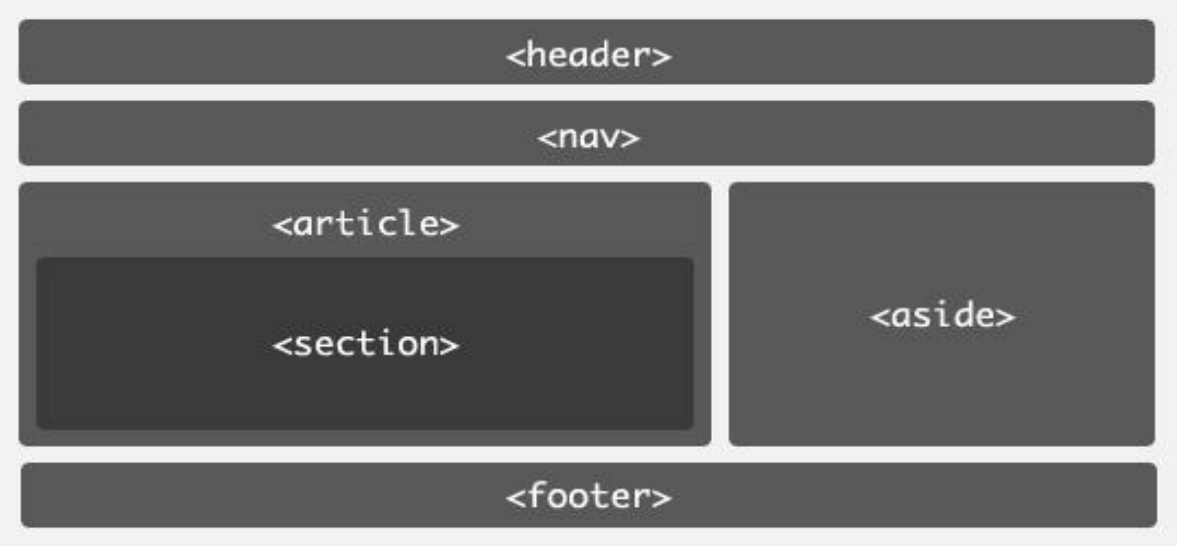
```
<!DOCTYPE html>
```

Para crear una página web, normalmente se incluyen estructuras comunes como headers, footers y columnas y es común usar divs para darles un id descriptiva o clase ya que las actuales versiones de HTML4 carecen de la semántica necesaria para describir estas partes de manera específica.

Por ejemplo:



El HTML5 soluciona esto incluyendo nuevos elementos que representan cada una de las diferentes secciones de una página web.



Nuevos elementos:

Los tiempos modernos requieren nuevos elementos para proporcionar una web más semántica, completa y homogenea. Para ello se han añadido una buena serie de elementos que nos permitirán encapsular más nuestro contenido.

- `<article />`: elemento que nos permite declarar un trozo del contenido como artículo. Ideal para blogs o periódicos.
- `<aside />`: representa un trozo de contenido que se relaciona muy levemente con el resto del contenido.

Dialog:

Este elemento nos permite representar conversaciones:

```
<dialog>
    <dt> Costello
    <dd> Look, you gotta first baseman?
    <dt> Abbott
    <dd> Certainly.
    <dt> Costello
    <dd> Who's playing first?
    <dt> Abbott
    <dd> That's right.
    <dt> Costello
    <dd> When you pay off the first baseman every month, who gets the   money?
    <dt> Abbott
    <dd> Every dollar of it.
</dialog>
```

Figure:

Podrá ser usado para asociar con un caption un contenido incrustado, como por ejemplo gráficos o vídeo:

```
<figure>
   <video src=ogg>…</video>
   <legend>Example</legend>
</figure>
```

- `<footer />` Sección para colocar información sobre el autor, copyright, etc.
- `<header />` Representa a la sección de cabecera.
- `<nav />` Representa la sección de la página orientada a la navegación.
- `<section />` Elemento que indica que se trata de una sección genérica.

Audio:

La etiqueta `<audio>` es muy sencilla porque solo implica un reproductor de audio. Para implementarla es muy sencillo:

```
<audio width="300" height="32" src="micancion.mp3" controls="controls"
autoplay="autoplay" preload="">

</audio>
```

Expliquemos los atributos de la etiqueta:

- **SRC:** Nos enlaza el archivo de audio que queremos reproducir.
- **CONTROLS**: Nos permite implementar los controles del reproductor por defecto del navegador como, botón play-pause, seek y volumen.
- **AUTOPLAY:** Nos permite reproducir el archivo de audio desde que se carga la página.
- **PRELOAD:** Nos carga un poco el archivo de audio antes de iniciar la reproducción en el buffer para que no se trabe por reproducir mas de lo que carga.

Como pueden ver, es demasiado sencillo para empezar a jugar con esta etiqueta. Pero aquí aun no termina todo, como es algo nuevo de implementar, los navegadores luchan por cual formato de audio debe liderar la web, entonces hay que atender a todos.

Implementación de formatos de audio

Tenemos muchos navegadores con diferente compatibilidad, pero lo que más nos interesa son los **Motores de Renderizado**, estos se encargan de renderizar el **código** de nuestra pagina web e implementar ahora el contenido multimedia.

Veamos cuales motores corresponden a cada navegador:

- Google Chrome: WebKit
- Safari: WebKit
- Mozilla FireFox: Gecko
- Internet Explorer: Trident
- Opera: Presto

Estos son los Navegadores mas importantes hasta ahora, y ahora veamos la compatibilidad con los archivos de audio.

	WebKit	Gecko	Trident (IE9)	Presto
Mp3	*		*	
Ogg	* (Safari no)	*		*
Wav	* (Safari si)	*	*	*

Visto que motores o navegadores aceptan un formato, pasamos a hacer Audio compatible con todos ellos.

```
<audio controls autoplay preload>
   <source src="cancion.ogg" type="audio/ogg" />
   <source src="cancion.mp3" type="audio/mpeg" />
   <source src="cancion.wav" type="audio/wav" />
   Tu navegador no soporta esta caracteristica
</audio>
```

Ahora quitamos el **SRC** y solo dejamos las demas propiedades pero ahora dentro de las etiquetas de apertura y cierre ponemos una etiqueta **SOURCE** para decirle que archivo va enlazar y qué tipo de audio es.

- **SRC:** Nos enlaza el archivo de audio que queremos reproducir.
- **Type:** Le decimos que tipo de audio o codec va reproducir

Video:

La manera más importante de implementarla es de esta manera:

```
<video src="video.mp4" width="640" height="360" controls autoplay preload>

</video>
```

Expliquemos los atributos de la etiqueta:

- **SRC:** Nos enlaza el archivo de video que queremos reproducir.
- **WIDTH:** Nos define el ancho del video en pixeles.

- **HEIGHT:** Nos define la altura del video en pixeles.
- **CONTROLS**: Nos permite implementar los controles del reproductor por defecto del navegador como, botón play-pause, seek y volumen.
- **AUTOPLAY:** Nos permite reproducir el archivo de video desde que se carga la pagina.
- **PRELOAD:** Nos carga un poco el archivo de video antes de iniciar la reproducción en el buffer para que no se trabe por reproducir mas de lo que carga.

Implementación de formatos de video

Al igual que en la etiqueta **AUDIO** tenemos muchos formatos de video que debemos implementar ya que cada motor de renderizado de los navegadores tiene soporte para un codec de video diferente.

Veamos la tabla de compatibilidad

- **Gecko:** OGG
- **WebKit:** OGG, H.264, WebM
- **Safari e Internet Explorer 9 (WebKit – Trident):** H.264
- **Presto:** OGG, H.264

Ya que sabemos que tipos de codecs acepta cada navegador, podemos pasar a la siguiente parte de este tutorial. Hay que hacer que todos los navegadores acepten en la etiqueta video todos los formatos de video.

Ahora tiene un poco de complejidad igual que en **AUDIO** solo hay que poner los diferentes **SOURCE** para cada formato de video.

```
<video width="640" height="360" controls autoplay preload>
  <source src="mivideo.mp4" type='video/mp4; codecs="avc1,mp4a"' />
  <source src="mivideo.ogv" type='video/ogg; codecs="theora,vorbis"' />
  <source src="mivideo.webm" type='video/webm; codecs="vp8,vorbis"' />
  Tu navegador no soporta esta caracteristica
</video>
```

Como vemos, la diferencia es poca, ahora solo quitamos el atributo **SRC** y agregamos los **SOURCE** de los diferentes archivos de video.

- **SRC:** Nos enlaza el archivo de video que queremos reproducir.
- **Type:** Le decimos que tipo de video o codec va reproducir

Es bastante fácil de usar esta etiqueta y podemos jugar mucho con los estilos de **CSS** para darle estilo, incluso podemos crear un reproductor mas personalizado.

En **Internet Explorer 9** podemos usar el formato **WEBM** con un **Plugin** de **Google** para los que quieran tener soporte de este formato. (http://tools.google.com/dlpage/webmmf).

Y algunas herramientas para convertir videos a otros formatos

- Miro Video Converter
- Handbrake
- PandaStream
- Zencoder
- Encoding

Embed

Es un elemento dedicado para contenido de plugins. Por ejemplo

```
<embed src="helloworld.swf" />
```

Meter

El elemento `meter` puede ser usado para *marcar* medidas definiendo que tales medidas **son parte** de una escala con unos valores máximos y mínimos. Existe también un atributo `min` así como `high`, `low` y `optimum` con los que se puede jugar. También puedes ocultar el valor actual:

```
<meter low="-10" high="100" min="12" max="50" optimum="26" value="25">
    <span> 1/5 </span>
</meter>

<meter min="0" max="120" value="40">
      <span>1/3</span>
</meter>
```

Time:

Uno de los microformatos más populares es hCalendar porque soluciona un problema muy común; permitir a los usuarios añadir eventos a sus calendarios. El único problema con hCalendar es que sirve para describir fechas y horas de forma que pueda ser entendido por una **máquina**. Sin embargo, los humanos tenemos la fea costumbre de describir las fechas con cosas como *17 de Julio del 2011* o *el próximo domingo* pero los parseadores esperan recibir una fecha en formato ISO: *YYYY-MM-DDThh:mm:ss* por ejemplo.

La comunidad desarrolló algunas elegantes soluciones para esto como por ejemplo la de usar la etiqueta `abbr`:

```
<abbr class="dtstart" title="2011-07-17">
```

```
    17 de Enero del 2011
</abbr>
```

En HTML5 este problema lo solventamos con la nueva etiqueta `time`:

```
<time class="dtstart" datetime="2011-07-17">
    17 de Enero del 2011
</time>
```

El elemento `time` puede ser usado para fechas, horas o una combinación de ambas:

```
<time datetime="21:00">9pm</time>
<time datetime="2011-09-12">12 de Septiembre del 2011</time>
<time datetime="2011-09-12T13:30">12 de Septiembre del 2011 a las 1:30pm</time>
```

Canvas

Es usado para mostrar gráficos renderizados en tiempo real, por ejemplo gráficos, juegos, etc.

```
<!DOCTYPE html>
<html>
<body>

<canvas id="myCanvas">Your browser does not support the canvas tag.</canvas>

<script type="text/javascript">
  var canvas=document.getElementById('myCanvas');
  var ctx=canvas.getContext('2d');
  ctx.fillStyle='#FF0000';
  ctx.fillRect(0,0,80,100);
</script>

</body>
</html>
```

Imagen del canvas creado

Datalist:

Junto con el nuevo atributo list para los <input /> puede ser usado para crear comboboxes:

```
<input list=browsers>
<datalist id=browsers>
  <option value="Safari">
  <option value="Internet Explorer">
  <option value="Opera">
  <option value="Firefox">
</datalist>
```

- `<event-sources />` puede ser usado para capturar eventos enviados desde servidor.
- `<output />` nos indica que tipo de salida vamos producir con nuestra página.

Progress:

Este elemento puede ser usado (en conjunto con javascript) para mostrar el progreso de una tarea o de un proceso que esté ocurriendo en la página web, como por ejemplo un archivo o recurso que se está subiendo o descargando. Su estructura es la siguiente:

```
<progress value="60" max="100">
   <span id="descargando">60</span>%
</progress>
```

Por los momentos este elemento es soportado únicamente por los navegadores: Firefox 6.0 (en adelante), Chrome 10.0 (en adelante) y Opera 11. Para el ejemplo anterior, el navegador Firefox 6.0 mostraría el siguiente resultado:

Para los demás navegadores, el contenido que se encuentra dentro de las etiquetas <progress> </progress> se utiliza como "fallback" o soporte de compatibilidad. El ejemplo anterior se mostraría de la siguiente forma:

60%

- **Detalles del elemento**

El elemento **progress** posee los siguientes atributos:

- **max:** es un número que indica el total que necesita la tarea para ser completada.
- **value**: es un número que indica qué tanto ha sido completada la tarea. Debe ser menor o igual que el valor del atributo **max**

Adicionalmente, el elemento **progress** puede tener un estado "indeterminado" cuando no se le coloca el atributo **value**. Este comportamiento se utiliza cuando el estado de la

completación de la tarea es indeterminado o desconocido. En el navegador Firefox 6.0 se vería de la siguiente manera:

Nuevas propiedades para controles de entrada Input:

Los elementos de entrada `<input />` dispondrán de una serie de tipos (type) nuevos para indicar los diferentes tipos de elementos de entrada posibles.

Dentro de la gran cantidad de novedades que nos ofrece el HTML5, una muy interesante son **los nuevos valores para el atributo *type* del elemento *input***.

Estos nuevos tipos de campos hacen que los navegadores adopten distintos comportamientos que, sin dudas, nos van a hacer la vida más fácil a los desarrolladores de sitios web.

Así como actualmente tenemos el conocido *password* que oculta la contraseña con asteriscos o círculos (dependiendo del navegador); ahora contamos con el nuevo *search* que presenta una pequeña cruz para poder borrar su contenido; o también el nuevo campo numérico *number* que muestra dos flechas (hacia arriba y hacia abajo) para aumentar o disminuir el valor del número.

Algo muy importante a tener en cuenta es que si bien estos elementos todavía no son soportados por todos los navegadores modernos, **el uso de los mismos no afectará de ningún modo en los navegadores que no los soporten**, actuarán simplemente como si fuesen del tipo *text*.

Solo se puede ver los nuevos comportamientos si se esta utilizando algún navegador moderno (últimas versiones de Chrome, Firefox, Safari, Opera o IE).

Search:

```
<input type="search" name="busqueda">
```

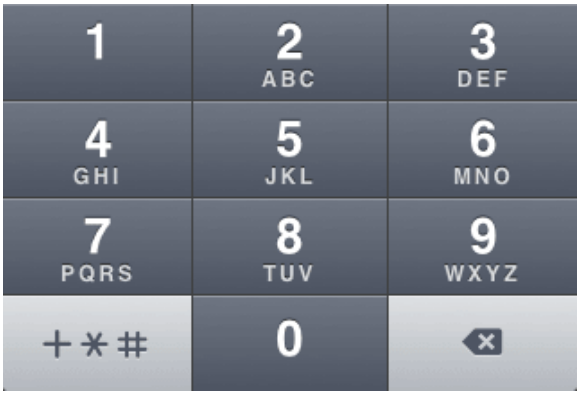
Al ingresar texto en el campo, el navegador muestra una cruz a la derecha para borrar todo lo que hemos escrito.

Vista en Chrome.

Tel:

```
<input type="tel" name="telefono">
```

A la hora de completar un *input* de tipo *tel*, un smartphone como el iPhone convierte su teclado a números de teléfono.



Url:

```
<input type="url" name="url">
```

En este campo, el teclado del iPhone es qwerty pero en modo "url", ya que ofrece teclas fundamentales para escribir una dirección web como son el punto, la barra "/" o la tecla ".com".

Teclado iPhone para completer url

Email:

```
<input type="email" name="correo">
```

Esta vez, el teclado del smartphone es qwerty pero también tenemos la tecla "@".

Teclado para introducir una dirección de correo

Datatime:

```
<input type="datetime" name="fechahora">
```

Con la última versión de Opera, al cliquear en el campo fechahora se verá un calendario muy completo que el navegador dispone de forma totalmente nativa.

Campo fecha y hora en Opera

Date:

```
<input type="date" name="fecha">
```

Si estamos usando Opera, el calendario es el mismo que el de la imagen anterior.

Month:

```
<input type="month" name="mes">
```

El calendario aquí permite seleccionar el número de mes.

Week:

```
<input type="week" name="semana">
```

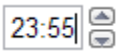
El calendario que nos muestra Opera para el campo de semana, nos permite elegir el número de semana del año.

Input week en Opera

Time:

```
<input type="time" name="hora">
```

Aquí el input está con el formato de hora, con los dos puntos ":" correspondientes y las flechas para subir o bajar el horario.



Number:

```
<input type="number" name="num" min="0" max="50">
```

Para este *input* tenemos los atributos *min* y *max* para establecer el máximo y el mínimo que acepta el campo.

En un smartphone vemos el teclado numérico:

Teclado numérico del iPhone

Range:

```
<input type="range" name="rango" min="0" max="50">
```

El input de tipo *range* se presenta como un control para arrastrar con el mouse (o con el dedo en un móvil con pantalla táctil). Este campo también acepta los atributos *min* y *max*

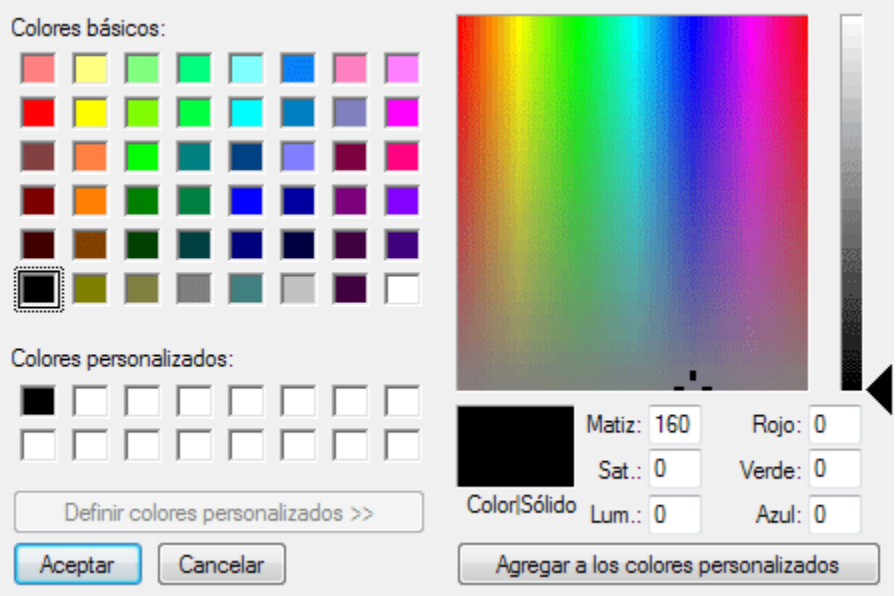
Input tipo range

Color:

```
<input type="color" name="color">
```

Con la última versión de Opera el navegador presenta de forma **nativa** un **selector de color**… otra funcionalidad que comúnmente tendríamos que hacer con javascript.

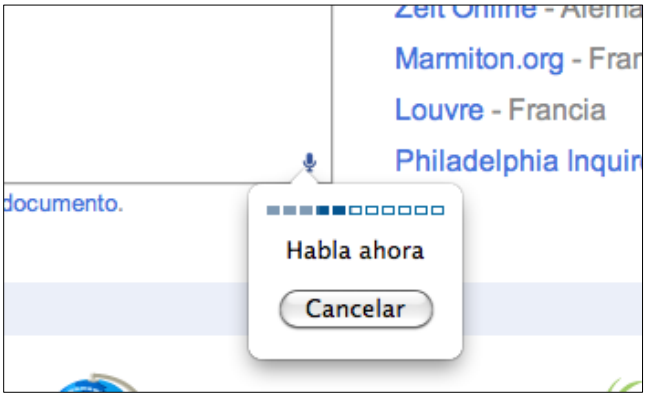
Al dar clic sobre "otros" se muestra una ventana con un **selector de color mucho más avanzado** como el de la siguiente captura de pantalla:



Captura de Voz:

```
<input type="text" x-webkit-speech="">
```

Este nuevo **campo** permite **capturar la voz** del usuario y escribir directamente en el campo de texto lo que se hablo o también procesar directamente la voz, ya sean comandos u otras cosas, pero para explicar mejor esta nueva capacidad podemos poner el ejemplo de **Google Translate**, si son usuarios de **Google Chrome** en la versión 11 se puede ver un micrófono en la parte esquina inferior derecha donde les permitirá hablar lo que querían traducir, como en la siguiente imagen.



Esta capacidad es compatible con todos los tipos de caja de texto de **HTML5** menos el **textarea**, quizá por motivos de rendimiento o seguridad, también tiene poca compatibilidad con navegadores, son pocos los nuevos navegadores que ya lo implementan.

La idea es que estos tipos sean proporcionados por el agente de usuario (navegador) en su interface que suministran el formato definido al servidor.

Nuevos atributos:

- media: Para conseguir una mayor consistencia con el link en los elementos `<a />`
- ping: Especificaremos una lista separada por espacios donde produciremos un ping cuando se siga el enlace, para los elementos `<area />` y `<a />`
- target: Disponible para mejorar la consistencia con el elemento `<a />`.
- autofocus: Destinado para indicar el elemento `<input />` (no hidden), `<button/>` `<select />, <textarea />` o que ha de coger el foco al cargar la página.
- form: Atributo para `<input />, <output />, <select />, <textarea />, <button />` y `<fieldset />` que permite que se asocien con un formulario.
- replace: atributo para `<input />, <button />` y `<form />` que le afectará cuando el contenido del elemento sofra algún cambio.
- data: Para `<form />, <select />` y `<datalist />`.
- required: Para elementos `<input />` (Excepto hidden e image) y `<textarea />`, indica que el campo es obligatorio.
- inputmode: Atributo para `<input />` y `<textarea />.`
- disabled: Para `<fieldset />,` permite desactivar el fieldset completo.
- autocomplete, min, max, pattern, step: Para elementos `<input />` permite delimitar las posibilidades de nuestros elementos de entrada.
- list: Para elementos `<datalist />` y `<select />.`
- template: Para `<input />` y `<button />` podrá usarse para repetir templates.
- scoped: Para elemento `<style />,` permitirá usar hojas de estilo “scoped”.
- async: Para el elemento `<script />` el ajax hecho atributo.

Elementos Cambiados

Estos elementos de HTML5 son incompatibles con HTML4.

- El elemento `<a />` sin href ahora creará un enlace al sitio.
- El elemento `<address />` es ahora un nuevo concepto de sección.
- El elemento `<b />` ahora representa un trozo de texto a ser estilizado sin ninguna importancia.
- Para elementos `<label />` el navegador no debe mover el foco desde la etiqueta al control a menos que el comportamiento sea estándar para el interfaz utilizado en la plataforma.
- `<menu />` ha sido redefinido para ser usado con los actuales menús.
- El elemento `<small />` ahora representa una impresión pequeña.
- El elemento `<strong />` definitivamente representa el énfasis puesto en trozo de nuestro texto.

Elementos eliminados

En la nueva versión, algunos de los elementos anteriormente desaprobados pasan a ser eliminados definitivamente.

- acronym
- applet
- basefont
- big
- center
- dir
- font
- frame
- frameset
- isindex
- noframes
- noscript (solo en XHTML5)
- s

- strike
- tt
- u

Atributos eliminados

Al igual que los elementos los atributos también pasarán a mejor vida.

- rev y charset en `<link />` y `<a />`
- target en `<link />`
- nohref en `<area />`
- profile en `<head />`
- version en `<html />`
- name en `<map />`
- scheme en `<meta />`
- archive, classid, codetype, declare y standby en `<object />`
- valuetype en `<param />`
- charset en `<script />`
- summary en `<table />`
- header, axis y abbr en `<td />` y `<th />`

Las nuevas propiedades de CSS3

La W3C permite efectuar nuevas acciones sobre los elementos de HTML, mediante nuevas propiedades, veamos qué cosas nuevas se pueden hacer con estas propiedades:

Nota: Aquí utilizamos el prefijo -moz- para Mozilla Firefox. Existen otros prefijos para Chrome (-webkit), Safari (-webkit) y Opera (-o), basta con sustituir el prefijo para que funcione con el navegador respectivo.

- **Borde con colores diferentes**

La propiedad -moz-border-colors: permite crear varios bordes de colores diferentes. Esta propiedad puede ser utilizada también como:

```
-moz-borders-top-color: (adición de bordes, top, bottom, left, right)
```

Ej:

```
#midiv {
    border: 8px solid #000000;
    -moz-border-colors : #CC0000 #BB0000 #AA0000 #990000 #880000 #770000 #660000
#550000;
    padding: 5px ;
}
```

Un ejemplo de esta propiedad es el siguiente:

- **Imágenes como bordes**

CSS3 permite el uso de imágenes como bordes de los elementos de la página. Las dos propiedades (y sus derivadas) son:

border-image:
border-top-image
border-right-image
border-bottom-image
border-left-image
border-corner-image:
border-top-left-image
border-top-right-image
border-bottom-left-image
border-bottom-right-image

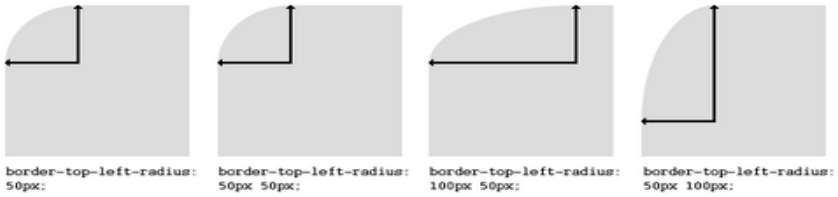
Ej:

```
#mi-div {
    border-image: url(border.png) 27 27 27 27 round round;
}
```

- **Bordes redondeados en las esquinas**

La propiedad **border-radius** de CSS3 permite a los desarrolladores web definir bordes redondeados en las esquinas, sin necesidad de imágenes ni necesidad de recurrir al uso de etiquetas div múltiples.

```
#contenedor {
   -moz-border-radius: 15px; /* Prefijo Moz para Mozilla  (no valido para la
W3C)*/
   border-radius: 15px; /* Ningún prefijo para los navegadores que incorporan la
propiedad sin prefijo (valido para la W3C) */
}
```



- **Crear sombras en CSS3**

Una nueva funcionalidad de CSS3 implementada a partir de la versión 3.1 de Firefox, es la posibilidad de crear sombras de colores: esta es la propiedad **box-shadow**.

Esta propiedad requiere de algunos parámetros para definir las características de la sombra:

1. **Desplazamiento horizontal de la sombra:** un valor positivo significa que la sombra aparece desde la derecha, un desplazamiento negativo hará que la sombra aparezca desde la izquierda.

2. **Desplazamiento vertical de la sombra:** un valor negativo significa que el box-shadow aparecerá desde arriba, un valor negativo hará aparecer la sombra desde abajo.

3. **En cuanto al difuminado**, cuanto más cerca de cero esté este valor, la sombra será más definida. En cambio, cuanto más se acerque de 1, la sombra estará más difuminada.

Ejemplos:

```
.sombra {
    box-shadow: 10px 10px 5px #888;
    padding: 5px 5px 5px 15px;
}
```

- **Imágenes de fondo de elementos en CSS3**

1. Propiedades background-clip y background-origin

La propiedad **background-origin** de CSS3 permite determinar la manera en que la imagen de fondo se posicionará en un elemento, y la propiedad background-clip permite seleccionar una porción de la imagen

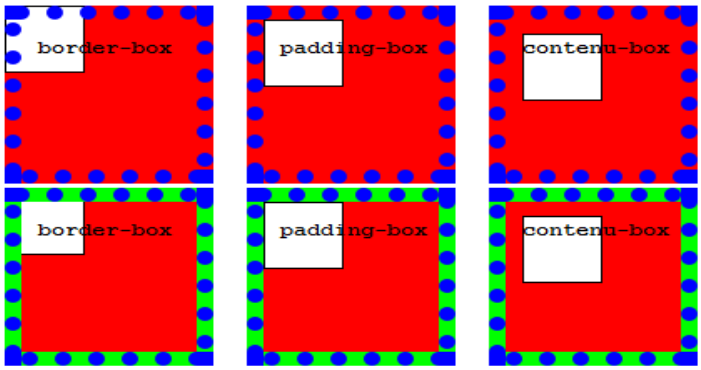
Ambas propiedades pueden tomar alguno de los 3 valores: **border-box, padding-box** y **content-box**.

Las implementaciones experimentales tienen como propiedades:

```
-webkit-background-origin: padding-box;
-moz-background-origin: padding-box

-webkit-background-clip: border-box;
-moz-background-clip: border-box
```

Las implementaciones estables (oficiales):

```
background-origin: content-box;
background-clip: content-box;
```



2. Propiedad background-size

CSS3 permite especificar un tamaño a las imágenes de fondo. Este tamaño puede ser especificado en pixeles, (height y width), o en porcentaje. Si se especifica un tamaño en porcentaje, el tamaño es relativo al ancho o altura de la zona a la que se ha atribuido la propiedad **background-origin.**

La primera imagen contiene un fondo que abarca todo el contenedor, mientras que la segunda especifica el tamaño del fondo sobre el contenedor.

3. Imágenes de fondo múltiples en CSS3

Para colocar fondos múltiples colocamos los "backgrounds" separados por coma y les asignamos las propiedades respectivas a cada uno. Por ejemplo:

```
body {
  background: url(paisaje.png) top left no-repeat, url(oveja.jpg) bottom
center 100px no-repeat;
}
```

- **Degrades**

Con la nueva propiedad **gradient** podemos crear degrades de varios colores sin necesidad de utilizar imágenes.

La sintaxis de la propiedad depende de los navegadores, he aquí un ejemplo:

```
.gradient {
   background: linear-gradient(1 225deg, 2 red, 3 green 20%, 4
yellow 70%, 5 white);
}
```

- **Animaciones y Transformaciones con CSS3**

1. **Transiciones**

Su característica es cambiar una propiedad (o un grupo de ellas) en un período de tiempo determinado. Una ventaja sobre **Javascript** es su degradación, ya que si esta propiedad no es soportada por el navegador, la animación simplemente no es mostrada.

El siguiente código muestra el aumento del ancho de un `<div>` en un segundo cuando el *mouse* se posiciona sobre él. Luego, cuando el mouse sale del `<div>` su ancho vuelve a la posición inicial en 2 segundos:

```
.prueba1 {
   margin:10px;
   width:200px;
   height:200px;
   background:#f00;
   border:1px solid #666;
   -webkit-transition: all 1s ease; /* Safari-Chrome */
}

.prueba1:hover{
   width:400px;
   -webkit-transition: all 2s ease; /* Safari-Chrome */
}
```

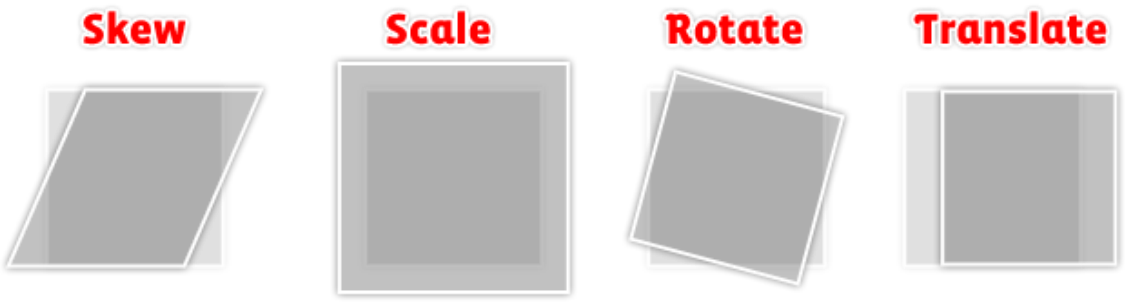
Propiedades de transition:

- **transition-property**: Determina la(s) propiedad(es) a ser animadas; puede ser cualquier propiedad CSS.
- **transition-duration:** Indica la duración de la animación del inicio al fin en segundos. Por defecto el valor es 0.
- **transition-timing-function:** Define cual sera el efecto de la animación: ease (por defecto), linear, ease-in, ease-out y ease-in-out.
- **transition-delay:** Tiempo en que la animación debe pausarse antes de comenzar.

2. **Transformaciones:**

Existen 4 tipos de valores para transformar elementos HTML mediante CSS3:

- **Skew:** desplazamiento de los ejes horizontales.
- **Scale:** modificación de la escala del elemento.
- **Rotate:** cambio de rotación del mismo definido en grados.
- **Translate:** desplazamiento del elemento desde su posición original.



Definiciones de CSS3

En la presente sección definiremos las propiedades y los selectores más importantes de CSS3 que estaremos utilizando en la sección de práctica. Como complemento también se presentarán otras propiedades de CSS2 y CSS1 cuyo uso es frecuente en las prácticas de este curso.

Propiedades nuevas de CSS3

Las propiedades que posee CSS3 se agrupan en categorías, dependiendo de su uso y sus capacidades. De esta forma tenemos las siguientes categorías: de texto, de bordes, de fondo, de transformaciones, de transiciones, entre otras.

No todas estas propiedades son soportadas de forma oficial por los navegadores modernos (Mozilla Firefox, Google Chrome, Opera, Internet Explorer 9.0). Sin embargo, cada navegador las implementa en una versión propia y para diferenciarlas de las propiedades oficiales se les antepone un prefijo. Estos prefijos son:

-moz-	Para el navegador Mozilla Firefox
-webkit-	Para los navegadores Google Chrome y Safari
-o-	Para el navegador Opera
-ms-	Para el navegador Internet Explorer versión 8, en adelante

Para ejemplificar su uso consideremos el caso en que queremos un rectángulo con los bordes redondeados, haciendo uso de la propiedad border-radius (su definición la veremos más adelante en esta sección):

```
.rectanguloRedondeado {
    -moz-border-radius:  4px 4px 4px;
    -webkit-border-radius:  4px 4px 4px;
    -o-border-radius:  4px 4px 4px;
    -ms-border-radius:  4px 4px 4px;
    border-radius:  4px 4px 4px 4px;
}
```

Como podemos ver, se define la misma propiedad anteponiéndole el conjunto de prefijos y al final la propiedad oficial. Cada navegador reconocerá su propio prefijo, y en caso de tener soporte oficial también reconocerá la propiedad sin el prefijo. Es una buena práctica colocar esta propiedad oficial debajo de todas las versiones propias para que sobrescriba a alguna propiedad no oficial que haya sido identificada por el navegador.

Ahora veamos la lista de propiedades nuevas, organizadas por sus categorías e identificando su soporte para los distintos navegadores:

1. Propiedades de texto:

text-shadow:

Aplica una sombra a un texto. Acepta 4 valores en su sintaxis:

```
text-shadow: sombra-horizontal sombra-vertical difuminar color
```

Explicación:
sombra-horizontal: tamaño en pixeles (u otra unidad) de la sombra horizontal.
sombra-vertical: tamaño en pixeles (u otra unidad) de la sombra vertical.
difuminar: distancia en pixeles (u otra unidad) del nivel de difuminación de las sombras.
color: color de la sombra.

- Soporte de los navegadores: todos los navegadores modernos excepto Internet Explorer.

- Ejemplo:

```
p {text-shadow: 1px 1px 1px black;}      /*Aplica una sombra negra a todos los textos de los tags p*/
```

2. Propiedades de bordes

box-shadow:

Aplica una sombra a un elemento contenedor (por ejemplo el `div`). Su sintaxis es:

```
box-shadow: sombra-horizontal sombra-vertical difuminar extensión color inset
```

Explicación:
sombra-horizontal: tamaño en pixeles (u otra unidad) de la sombra horizontal.
sombra-vertical: tamaño en pixeles (u otra unidad) de la sombra vertical.
difuminar: distancia en pixeles (u otra unidad) del nivel de difuminación de las sombras.
extensión: tamaño en pixeles (u otra unidad) de la sombra.
color: color de la sombra.
inset: palabra reservada que si aparece en la propiedad coloca la sombra en el interior del contenedor.

- Soporte de los navegadores: esta propiedad es soportada para los navegadores: Firefox 4 en adelante, Google Chrome, Opera 10 en adelante, Safari 5.1.1 e Internet

Explorer 9 en adelante. Para todas las demás versiones de los navegadores debe utilizarse el prefijo correspondiente en la propiedad.

- Ejemplos:

```
/* Aplica una sombra negra exterior a todos los tags div */
div {
  box-shadow: 1px 1px 1px black;
}

/* Aplica una sombra negra interior a todos los tags div */
div {
   box-shadow: 1px 1px 1px black inset;
}
```

border-radius:

Establece un borde redondeado para cada esquina de un contenedor (div). Su sintaxis es:

```
border-radius:  esq-sup-izq  esq-sup-der  esq-inf-der  esq-inf-izq
```

Explicación:

esq-sup-izq: tamaño en pixeles (u otra unidad) del radio de curvatura de la esquina superior izquierda.

esq-sup-der: tamaño en pixeles (u otra unidad) del radio de curvatura de la esquina superior derecha.

esq-inf-der: tamaño en pixeles (u otra unidad) del radio de curvatura de la esquina inferior derecha.

esq-inf-izq: tamaño en pixeles (u otra unidad) del radio de curvatura de la esquina inferior izquierda.

- Soporte de los navegadores: esta propiedad es soportada por los navegadores: Firefox 4 en adelante, Google Chrome, Opera 10 en adelante, Safari 5 en adelante, e Internet Explorer 9 en adelante. Para todas las demás versiones de los navegadores debe utilizarse el prefijo correspondiente en la propiedad.

- Ejemplo:

```
div {border-radius: 5px 5px 5px 5px;}     /* Aplica un borde redondeado de 5px a las 4 esquinas
                                             de todos los tags div*/
```

3. Propiedades de fondo:

background-image:

Define la imagen de fondo para un elemento. En CSS2 esta imagen podía ser un color o una imagen. Para CSS3 puede definirse además un degrade de colores. Su sintaxis es:

```
background-image: none | url('URL') | linear-gradient(direccion, color1, color2);
```

Nota: En esta sintaxis el operador "|" se lee "o", y quiere decir que puede aplicarse "este valor **o** este valor, pero no los 2 a la vez".

Explicación:

none: palabra reservada que indica que no debe mostrarse imagen de fondo.

url('URL'): permite indicar la url (o ruta) de una imagen que quiere utilizarse como imagen de fondo.

linear-gradient(direccion, colorIni, colorFin): establece un degrade lineal de dos colores indicando la dirección, el color inicial (colorIni) y el color final (colorFin) del degrade.

- Soporte de los navegadores: el uso de una imagen propia con `url()` como imagen de fondo es soportado por todos los navegadores. Por su parte, la función `linear-gradient` no posee soporte oficial por parte de ningún navegador, por lo que deben utilizarse las versiones propias de cada navegador con sus prefijos correspondientes.

- Ejemplos:

```
div {background-image: url('img/example.jpg';}    /* Aplica una imagen de fondo ubicada
                                                     en la ruta 'img/example.jpg' */


/* Fondo en degradado de colores  para los tags div. El degrade tiene la dirección desde arriba hacia abajo, iniciando en
un color blanco y terminando en un color gris claro, utilizando los distintos soportes de cada navegador
Nota: la línea que dice -webkit-gradient es la sintaxis de la propiedad para las primeras versiones de Chrome */
div {
   background:  #FFF;            /*Color de fondo sino existe soporte alguno*/
   background-image: -webkit-gradient(linear, left top, left bottom, from(#fff),
to(#eee));
   background-image: -webkit-linear-gradient(top, #fff, #eee);
   background-image:    -moz-linear-gradient(top, #fff, #eee);
   background-image:     -ms-linear-gradient(top, #fff, #eee);
   background-image:      -o-linear-gradient(top, #fff, #eee);
   background-image:         linear-gradient(top, #fff, #eee);
}
```

background-size:

Establece el tamaño de una imagen de fondo (`background-image`). Su sintaxis es:

```
background-size: tamaño | porcentaje | cover | contain
```

Nota: En esta sintaxis el operador "|" se lee "o", y quiere decir que puede aplicarse "este valor **o** este valor, pero no los 2 a la vez".

Explicación:

tamaño: establece el tamaño en pixeles (u otra unidad) de la imagen de fondo. Debe tener dos valores, el primero indicando el ancho de la imagen y el segundo indicando el alto de la misma.

porcentaje: establece el tamaño en porcentaje de la imagen de fondo, tomando como referencia el tamaño del elemento padre (o elemento que lo contiene). Debe tener dos valores, el primero indicando el ancho de la imagen y el segundo indicando el alto de la misma.

cover: palabra reservada que indica que la imagen debe escalarse al mínimo tamaño necesario para que tanto el ancho como el alto puedan caber en el área de su elemento padre o elemento contenedor.

contain: palabra reservada que indica que la imagen debe escalarse al máximo tamaño necesario para que tanto el ancho como el alto puedan caber en el área de su elemento padre o elemento contenedor.

- Soporte de los navegadores: esta propiedad es soportada por los navegadores: Firefox 4 en adelante, Google Chrome, Opera 10 en adelante, Safari 5 en adelante, e Internet Explorer 9 en adelante. Para todas las demás versiones de los navegadores debe utilizarse el prefijo correspondiente en la propiedad.

- Ejemplos:

```
div {background-size: 200px 150px;}    /* Establece la imagen de fondo en un tamaño de 200px
                                           de ancho por 150px de alto*/

div {background-size: 50% 60%;}    /* Establece la imagen de fondo en un ancho igual al 50% del ancho
                                      de su contenedor y un alto igual al 60% del alto de su contenedor*/

div {background-size: cover;}          /* Establece la imagen de fondo en el tamaño mínimo
                                          necesario para que pueda caber en su contenedor */
```

4. Propiedades de multi-columna

column-count:

Especifica el número de columnas en las que se debe dividir un elemento (generalmente un div). Su sintaxis es:

```
column-count: número | auto
```

Nota: En esta sintaxis el operador "|" se lee "o", y quiere decir que puede aplicarse "este valor **o** este valor, pero no los 2 a la vez".

Explicación:

número: número de columnas en las que se desea dividir el elemento.

auto: el número de columnas es determinado automáticamente.

- Soporte de los navegadores: esta propiedad es soportada de forma oficial únicamente por Opera 10 en adelante. Para utilizarla en los demás navegadores (Firefox, Google Chrome, Safari), es necesario colocar el prefijo correspondiente en la propiedad. Internet Explorer no ofrece soporte de ninguna forma para esta propiedad.

- Ejemplo:

```
div {
   -moz-column-count: 3;              /* Establece que el contenido de los tags div se separe en 3
   -webkit-column-count: 3;              columnas, utilizando el soporte para cada navegador */
   column-count: 3;
}
```

column-gap:

Especifica el espacio (gap) entre las columnas que hayan sido definidas. Su sintaxis es:

```
column-gap: tamaño | normal
```

Nota: En esta sintaxis el operador "|" se lee "o", y quiere decir que puede aplicarse "este valor **o** este valor, pero no los 2 a la vez".

Explicación:

tamaño: tamaño en pixeles (u otra unidad) que indica el espacio entre las columnas.

normal: establece un tamaño normal definido por la W3C (1em).

- Soporte de los navegadores: esta propiedad es soportada de forma oficial únicamente por Opera 10 en adelante. Para utilizarla en los demás navegadores (Firefox, Google Chrome, Safari), es necesario colocar el prefijo correspondiente en la propiedad. Internet Explorer no ofrece soporte de ninguna forma para esta propiedad.

- Ejemplo:

```
div {
   -moz-column-count: 3;          /* Establece un espacio de 10px entre cada una de las 3
   -webkit-column-count: 3;          columnas definidas para los tags div, utilizando el soporte para
   column-count: 3;                  cada navegador */
   -moz-column-gap: 10px;
   -webkit-column-gap: 10px;
   column-gap: 10px;
}
```

5. Propiedades de transiciones

transition:

Aplica un efecto de transición a una propiedad declarada de CSS. Su sintaxis es:
`transition: propiedad duración efecto-de-transición tiempo-de-retraso`

Explicación:
`propiedad`: propiedad de CSS a la cual se le aplica la transición.
`duración`: tiempo de duración de la transición (en segundos).
`efecto-de-transición`: nombre del efecto de transición.
`tiempo-de-retraso`: tiempo que debe retrasarse el comienzo de la transición.

Los efectos de transición pueden ser: `linear, ease, ease-in, ease-out, ease-in-out, cubic`, y reciben sus nombres debido a funciones matemáticas de movimiento.

- Soporte de los navegadores: esta propiedad no es soportada de forma oficial en ningún navegador. Para utilizarla en los demás navegadores (Firefox, Google Chrome, Safari, Opera), es necesario colocar el prefijo correspondiente en la propiedad. Internet Explorer no ofrece soporte de ninguna forma para esta propiedad.

- Ejemplo:

```
/* Establece una transición sobre el ancho (width) de los tags div, pasándolo de 100px a 300px cuando se pasa el mouse
   encima del elemento y empleando el efecto "ease". Utiliza el soporte para cada navegador */
div {
   width:  100px;
   -moz-transition: width 2s ease 0.5s;
   -webkit-transition:  width 2s ease 0.5s;
   -o-transition:  width 2s ease 0.5s;
   transition:  width 2s ease 0.5s;
}

div:hover {width:300px;}
```

6. Propiedades de transformación 2D/3D

transform:

Aplica una transformación 2D o 3D a un elemento. Esta propiedad acepta como valores una serie de funciones. Su sintaxis es:

`transform: lista-de-funciones`

Lista de funciones:

`none`: define que no se aplicará ninguna transformación.
`translate(x,y)`: define una translación en 2D sobre las coordenadas (x, y).
`translate3d(x,y,z)`: define una translación en 3D en las coordenadas (x, y, z).
`scale(x,y)`: define un redimensionamiento en 2D sobre las coordenadas (x, y).
`scale3d(x,y,z)`: define una redimensionamiento en 3D sobre las coordenadas (x, y, z).
`rotate(x,y)`: define una rotación en 2D sobre las coordenadas (x, y).
`rotate3d(x,y,z)`: define una rotación en 3D sobre las coordenadas (x, y, z).
`skew(x,y)`: define un desplazamiento horizontal en 2D sobre las coordenadas (x, y).
`skew3d(x,y,z)`: define un desplazamiento horizontal en 3D sobre las coordenadas (x,y,z).

- Soporte de los navegadores: esta propiedad no es soportada de forma oficial en ningún navegador. Para utilizarla en los demás navegadores (Firefox, Google Chrome, Safari, Opera), es necesario colocar el prefijo correspondiente en la propiedad. Internet Explorer no ofrece soporte de ninguna forma para esta propiedad.

- Ejemplo:

```
/* Indica una rotación de 5 grados en el eje X para los tags div, utilizando el soporte para cada navegador */
div {
  -o-transform: rotate(5deg);
  -webkit-transform: rotate(5deg);
  -moz-transform: rotate(5deg);
  transform: rotate(5deg);
}
```

Propiedades de CSS2 y CSS1

Para complementar esta guía se presentarán las propiedades más importantes de CSS2 y CSS1 utilizadas en los ejercicios de las prácticas. Las mismas serán organizadas por categorías y se mostrará su soporte para cada navegador.

1. Propiedades de texto:

color:

Define el color de un texto. Pertenece a la versión 1 de CSS. Su sintaxis es:

```
color: nombre-color | código-hexadecimal
```

Nota: En esta sintaxis el operador "|" se lee "o", y quiere decir que puede aplicarse "este valor **o** este valor, pero no los 2 a la vez".

Explicación:

nombre-color: nombre en inglés del color.

código-hexadecimal: código en formato hexadecimal que identifica a un color.

Los nombres de colores están definidos de forma oficial por la W3C, los más utilizados son: `red, green, blue, yellow, green, black, white, purple, brown, pink`, etc.

Los códigos hexadecimales son otra forma de representar los colores y son equivalentes a usar los nombres de los colores. Por ejemplo el código #0000FF es equivalente a usar el nombre de color `blue` (azul). No todos los colores tienen definido un nombre, pero todos los colores si tienen su representación en hexadecimal.

- Soporte de los navegadores: esta propiedad es soportada de forma oficial por todos los navegadores.

- Ejemplo:

```
p {color: green;}          /* #00FF00 es el código hexadecimal del color verde, por lo que ambas
p {color: #00FF00;}           reglas son equivalentes y establecen el color verde a todas las etiquetas p */
```

text-align:

Especifica el alineamiento horizontal de un texto. Pertenece a la versión 1 de CSS. Su sintaxis es:

```
text-align: right | left | center | justify | inherit
```

Nota: En esta sintaxis el operador "|" se lee "o", y quiere decir que puede aplicarse "este valor **o** este valor, pero no los 2 a la vez".

Explicación:

right: alinea el texto a la derecha.

left: alinea el texto a la izquierda.

center: centra el texto.

justify: alinea el texto de forma justificada.

inherit: alinea el texto de acuerdo al alineamiento de su elemento padre o elemento contenedor.

- Soporte de los navegadores: esta propiedad es soportada de forma oficial por todos los navegadores.

- Ejemplo:

```
/* Alinea a la derecha todas las etiquetas p*/
p {
  text-align: right;
}
```

2. Propiedades de fondo:

background:

Define todas las propiedades de fondo de un elemento. Pertenece a la versión 1 de CSS. Su sintaxis es:

```
background: color url('URL') posición repetición;
```

Explicación:

color: nombre en inglés del color. El color debe estar definido en la lista de colores de CSS o puede ser un código hexadecimal. Se utiliza el color si no se especifica o falla el uso de la imagen de fondo.

url(´URL´): url o ruta de una imagen que se quiere usar como fondo.

posición: indica la posición de inicio de la imagen de fondo.

repetición: indica si la imagen de fondo debe repetirse.

El valor de *posición* tiene dos valores, los cuales deber resultar de la combinación de las palabras reservadas: `left, top, center, right y bottom` (izquierda, arriba, centro, derecha y abajo, respectivamente). Los valores de la *posición* también pueden especificarse en función del eje X y del eje Y, en cuyo caso se utilizan pixeles u otra unidad de CSS.

Los posibles valores de *repetición* son: `no-repeat, repeat-x y repeat-y` (no repetir, repetir en el eje X y repetir en el eje Y).

- Soporte de los navegadores: esta propiedad es soportada de forma oficial por todos los navegadores.

- Ejemplo:

```
/* Establece una imagen de fondo posicionada en la parte superior izquierda y se indica su repetición en el eje X*/
div {
  background: green url(´img/fondo.png´) top left repeat-x;
}

/* Establece una imagen de fondo posicionada a 0px para el eje X y 10px para el eje Y y se indica que no se repita */
div {
  background: green url(´img/fondo.png´) 0px 10px no-repeat;
}
```

3. Propiedades de dimensión:

width:

Define el ancho de un elemento. Pertenece a la versión 1 de CSS. Su sintaxis es:

```
width: auto | tamaño | porcentaje
```

Nota: En esta sintaxis el operador "|" se lee "o", y quiere decir que puede aplicarse "este valor **o** este valor, pero no los 2 a la vez".

Explicación:

auto: indica que al ancho del elemento debe ser calculado automáticamente por el navegador.

tamaño: indica el ancho del elemento en pixeles (u otra unidad).

posición: indica que el ancho del elemento debe ser un porcentaje del ancho del elemento padre o elemento contenedor.

- Soporte de los navegadores: esta propiedad es soportada de forma oficial por todos los navegadores.

- Ejemplo:

```
div {                          /* Indica que el ancho de los tags div debe ser de 100px  */
  width: 100px;
}


p {                            /* Indica que el ancho de los tags p debe ser de 50% del ancho de su elemento
  width: 50%;                     contenedor o elemento padre  */
}
```

height:

Define el alto de un elemento. Pertenece a la versión 1 de CSS. Su sintaxis es:

```
height: auto | tamaño | porcentaje
```

Nota: En esta sintaxis el operador "|" se lee "o", y quiere decir que puede aplicarse "este valor **o** este valor, pero no los 2 a la vez".

Explicación:

auto: indica que al alto del elemento debe ser calculado automáticamente por el navegador.

tamaño: indica el alto del elemento en pixeles (u otra unidad).

posición: indica que el alto del elemento debe ser un porcentaje del alto del elemento padre o elemento contenedor.

- Soporte de los navegadores: esta propiedad es soportada de forma oficial por todos los navegadores.

- Ejemplo:

```
div {                          /* Indica que el alto de los tags div debe ser de 100px  */
  height: 100px;
}


p {                            /* Indica que el  alto de los tags p debe ser de 50% del alto de su elemento
  width: 50%;                     contenedor o elemento padre  */
}
```

4. Propiedades de fuente:

font-family:

Define la fuente de un texto. Pertenece a la versión 1 de CSS. Su sintaxis es:

```
font-family: familia-de-fuentes
```

Explicación:
familia-de-fuentes: la familia de fuentes es una lista de nombres de fuentes separadas por coma. La lista establece una prioridad para aquellas que se encuentren de primero en la lista y si alguna de ellas no tiene soporte en el navegador, se utiliza la que le sigue.

- Soporte de los navegadores: esta propiedad es soportada de forma oficial por todos los navegadores.

- Ejemplo:

```
/* Indica una lista de fuentes organizadas por prioridad  */
p {
  font-family: Arial, Helvetica, sans-serif;
}
```

font-size:

Define el tamaño la fuente de un texto. Pertenece a la versión 1 de CSS. Su sintaxis es:

```
font-size: tamaño
```

Explicación:

tamaño: indica el tamaño de la fuente en pixeles, porcentaje o cualquier otra unidad.

- Soporte de los navegadores: esta propiedad es soportada de forma oficial por todos los navegadores.

- Ejemplo:

```
p {font-size: 14px;}                /* Establece el tamaño de la fuente en 14px */
```

font-weight:

Define el nivel de grosor de una fuente. Pertenece a la versión 1 de CSS. Su sintaxis es:

```
font-size: normal  |  bold  | bolder | lighter
```

Nota: En esta sintaxis el operador "|" se lee "o", y quiere decir que puede aplicarse "este valor **o** este valor, pero no los 2 a la vez".

Explicación:

normal: palabra reservada que indica que la fuente debe visualizarse de forma normal.

bold: palabra reservada que indica que la fuente debe visualizarse en negrita (más gruesa que normal).

bolder: palabra reservada que indica que la fuente debe visualizarse en un nivel de mayor grosor que negrita (bold).

ligther: palabra reservada que indica que la fuente debe visualizarse en un nivel de menor grosor que normal.

- Soporte de los navegadores: esta propiedad es soportada de forma oficial por todos los navegadores.

- Ejemplo:

```
p {font-weight: bold;}              /* Establece la fuente en negrita */
```

5. Propiedades de posicionamiento:

margin:

Establece los márgenes exteriores de un elemento. Pertenece a la versión 1 de CSS. Su sintaxis es:

```
margin: marg-superior  marg-derecho  marg-inferior  marg-izquierdo
```

Explicación:

marg-superior: indica el margen superior del elemento en pixeles, porcentaje o cualquier otra unidad de CSS.

marg-derecho: indica el margen derecho del elemento en pixeles, porcentaje o cualquier otra unidad de CSS.

marg-inferior: indica el margen inferior del elemento en pixeles, porcentaje o cualquier otra unidad de CSS.

marg-izquierdo: indica el margen izquierdo del elemento en pixeles, porcentaje o cualquier otra unidad de CSS.

Si se indica un solo valor en la propiedad, se aplica el mismo valor a cada margen.

- Soporte de los navegadores: esta propiedad es soportada de forma oficial por todos los navegadores.

- Ejemplo:

```
div {                                  /* Indica  un margen superior de 10px,  margen derecho de 5px, margen
  margin: 10px 5px 20px 15px;             inferior de 20px y margen izquierdo de 15px  para los tags div  */
}


p {                                    /* Indica que todos los márgenes deben ser de 25px  */
  margin: 25px;
}
```

padding:

Establece el relleno interior de un elemento. Pertenece a la versión 1 de CSS. Su sintaxis es:

```
padding: pad-superior  pad-derecho  pad-inferior  pad-izquierdo
```

Explicación:

pad-superior: indica el relleno superior del elemento en pixeles, porcentaje o cualquier otra unidad de CSS.

pad-derecho: indica el relleno derecho del elemento en pixeles, porcentaje o cualquier otra unidad de CSS.

pad-inferior: indica el relleno inferior del elemento en pixeles, porcentaje o cualquier otra unidad de CSS.

pad-izquierdo: indica el relleno izquierdo del elemento en pixeles, porcentaje o cualquier otra unidad de CSS.

Si se indica un solo valor en la propiedad, se aplica el mismo valor a cada relleno.

- Soporte de los navegadores: esta propiedad es soportada de forma oficial por todos los navegadores.

- Ejemplo:

```
div {                                /* Indica un relleno superior de 10px, relleno derecho de 5px, relleno
 padding: 10px 5px 20px 15px;          inferior de 20px y relleno izquierdo de 15px  para los tags div */
}


p {                                  /* Indica que todos los rellenos deben ser de 25px */
  margin: 25px;
}
```

display:

Especifica cómo deben visualizarse algunos elementos de HTML. Pertenece a la versión 2 de CSS. Su sintaxis es:

```
display: none | block | inline | inline-block | table
```

Nota: En esta sintaxis el operador "|" se lee "o", y quiere decir que puede aplicarse "este valor **o** este valor".

Explicación:

none: indica que el elemento no debe ser visualizado de ninguna forma.

block: indica que el elemento debe ser visualizado como un bloque. Un bloque es un elemento que tiene un poco de espacio arriba y debajo del mismo y no permite ningún otro elemento a su lado.

inline: indica que el elemento debe ser visualizado como un elemento lineal. Un elemento lineal es el opuesto de un bloque: no posee espacios arriba o debajo del mismo y permite elementos a su lado.

inline-block: indica que el elemento debe ser visualizado como un elemento lineal (permite elementos a su lado), pero se comporta como un bloque (con espacios arriba y debajo del mismo).

table: indica que el elemento debe ser visualizado como una tabla.

- Soporte de los navegadores: esta propiedad es soportada de forma oficial por todos los navegadores, excepto para su valor "table", que no es soportada por Internet Explorer 8 y versiones anteriores.

- Ejemplo:

```
div {display: block;}                /* Indica que los tags div deben ser tratados como bloques */
```

z-index:

Especifica el orden de visualización de un elemento, es decir, indica si un elemento debe visualizarse por debajo o por encima de los demás. Pertenece a la versión 2 de CSS. Su sintaxis es:

```
z-index: auto | número
```

Nota: En esta sintaxis el operador "|" se lee "o", y quiere decir que puede aplicarse "este valor **o** este valor, pero no los 2 a la vez".

Explicación:

auto: indica que el orden de visualización debe ser igual al de su elemento contenedor.

número: indica el orden de visualización del elemento con respecto a otros elementos. Se permiten números negativos.

- Soporte de los navegadores: esta propiedad es soportada de forma oficial por todos los navegadores, excepto para su valor "table", que no es soportada por Internet Explorer 8 y versiones anteriores.

- Ejemplo:

```
div {z-index: 150;}          /* Indica el orden de visualización en 100 para los los tags div */
```

Selectores de CSS

Los selectores de CSS, son patrones utilizados para seleccionar los elementos de HTML a los cuales les queremos colocar estilos.

En esta sección definiremos los selectores que utilizaremos en la sección de práctica, indicando la versión de CSS a la que pertenecen y su soporte con los navegadores.

.class:

Permite seleccionar aquellos elementos que tengan su atributo `class` igual al del selector.

Pertenece al CSS1 y es soportado por todos los navegadores.

- Ejemplo:

/* En este ejemplo se muestra el código CSS en un archivo separado del código HTML. El selector **.cabecera** selecciona todos los elementos con su atributo class igual a **"cabecera**". En este ejemplo, se selecciona el único div presente y se le aplica un color de texto azul*/

estilo.css

```
.cabecera {
  color: blue;
}
```

ejemplo.html

```
<div class='cabecera'>
   Titulo
</div>

<p class='intro'>
   Texto
</p>
```

#id:

Permite seleccionar el elemento que contenga su atributo `id` igual al del selector.

Pertenece al CSS1 y es soportado por todos los navegadores.

- Ejemplo:

/* En este ejemplo se muestra el código CSS en un archivo separado del código HTML. El selector **#intro** selecciona aquel elemento con su atributo `id` igual a **"intro**". En este ejemplo, se selecciona el div presente y se le aplica un color de texto rojo */

estilo.css

```
#intro {
  color: red;
}
```

ejemplo.html

```
<div id='intro'>
   Titulo
</div>

<p class='cabecera'>
   Texto
</p>
```

***:**

Permite seleccionar todos los elementos.

Pertenece al CSS2 y es soportado por todos los navegadores.

- Ejemplo:

/* En este ejemplo se muestra el código CSS en un archivo separado del código HTML. Para este ejemplo, el selector ***** selecciona todos los elementos presentes y les aplica un color de texto verde */

estilo.css

```
* {
  color: green;
}
```

ejemplo.html

```
<div id='intro'>
   Titulo
</div>

<p class='cabecera'>
   Texto
</p>
```

elemento:

Permite seleccionar todos los elementos cuyo nombre del tag sea igual al del selector.

Pertenece al CSS1 y es soportado por todos los navegadores.

- Ejemplo:

/* En este ejemplo se muestra el código CSS en un archivo separado del código HTML. Para este ejemplo, el selector **p** selecciona todos los elementos **p** y les aplica un color de texto verde */

estilo.css

```
p {
  color: green;
}
```

ejemplo.html

```
<div id='intro'>
   <p>Titulo</p>
```

```
</div>

<p>
   Texto
</p>
```

elemento1, elemento2:

Este selector es una lista de selectores y permite seleccionar todos los elementos cuyo nombre sea igual al de alguno de los selectores de la lista.

Pertenece al CSS1 y es soportado por todos los navegadores.

- Ejemplo:

```
/* En este ejemplo se muestra el código CSS en un archivo separado del código HTML. El selector
p, a selecciona todos los elementos p y todos los elementos a y les aplica un color de texto verde
*/

estilo.css
p, a {
  color: green;
}


ejemplo.html
<div id='intro'>
   <a href=''>Titulo</a>
</div>

<p>
   Texto
</p>
```

elemento1 elemento2:

Este selector establece una relación padre-hijo, también conocida como ascendiente-descediente. Se dice que un elemento es hijo (descendiente) de otro elemento si el primero está contenido dentro del segundo. De esta forma, al elemento contenedor se le conoce como elemento padre (ascendiente). En este caso, el selector selecciona todos los elementos **elemento2** (hijos) contenidos dentro de elementos **elemento1** (padres), sin importar el nivel de profundidad.

Pertenece al CSS1 y es soportado por todos los navegadores.

- Ejemplo:

/* En este ejemplo se muestra el código CSS en un archivo separado del código HTML. El selector **div a** selecciona todos los elementos **a** que se encuentran dentro de los elementos **div** (sin importar el nivel de profundidad) y les aplica un color de texto verde. Para este ejemplo, los dos tags **a** son seleccionados */

estilo.css

```
div a {
  color: green;
}
```

ejemplo.html

```
<div id=’intro’>
   <a href=’’>Yo sere seleccionado</a>

   <p>
     <a href=’’>Yo sere seleccionado </a>
   </p>
</div>

<p>
   Texto
</p>
```

elemento1 > elemento2:

Este selector establece una relación padre-hijo directo. Es igual al selector anterior, pero solo selecciona los elementos hijos directos, o hijos que se encuentran en el primer nivel de profundidad.

Pertenece al CSS2 y es soportado por todos los navegadores.

- Ejemplo:

/* En este ejemplo se muestra el código CSS en un archivo separado del código HTML. El selector **.intro a** selecciona los elementos **a** que se encuentran en el primer nivel de profundidad dentro de los elementos con su atributo class igual a **intro,** y les aplica un color de texto verde. Para este ejemplo, solo el primer tag **a** es seleccionado */

estilo.css

```
.intro a {
  color: green;
}
```

ejemplo.html

```
<div id=’intro’>
   <a href=’’>Yo sere seleccionado</a>

   <p>
     <a href=’’>Yo NO sere seleccionado</a>
   </p>
</div>

<p>
   Texto
</p>
```

elemento1 + elemento2:

Selecciona aquellos elementos **elemento2** que se encuentran ubicados inmediatamente después de los elementos **elemento1**.

Pertenece al CSS2 y es soportado por todos los navegadores.

- Ejemplo:

/* En este ejemplo se muestra el código CSS en un archivo separado del código HTML. El selector **div + p** selecciona los elementos **p** que se encuentran ubicados inmediatamente después de los elementos **div**, y les aplica un color de texto verde. Para este ejemplo, solo el primer tag **p** es seleccionado */

estilo.css

```
div + p {
  color: green;
}
```

ejemplo.html

```
<div id=’intro’>
   <a href=’’>Texto</a>
</div>

<p>Yo sere seleccionado</p>

<p>Yo NO sere seleccionado</p>
```

[atributo=’valor’]:

Selecciona aquellos elementos que tengan un atributo indicado y su valor sea igual al valor especificado.

Pertenece al CSS2 y es soportado por todos los navegadores.

- Ejemplo:

/* En este ejemplo se muestra el código CSS en un archivo separado del código HTML. El selector **[type='text']** selecciona los elementos que tengan el atributo **type** y su valor sea igual a **text,** y les aplica un color de texto verde. Para este ejemplo, solo el primer input es seleccionado */

estilo.css

```
[type='text'] {
  color: green;
}
```

ejemplo.html

```
<div id='intro'>
   <a href=''>Texto</a>
   <input type='text' value='Yo sere seleccionado' />
</div>

<input type='submit' value='Yo NO sere seleccionado' />
```

:hover

Establece un conjunto de propiedades para un elemento cuando el mouse se pasa encima del mismo. Cuando el mouse es retirado del elemento, vuelve a colocársele el CSS que tenía anterior al evento.

Pertenece al CSS1 y es soportado por todos los navegadores.

- Ejemplo:

/* En este ejemplo se muestra el código CSS en un archivo separado del código HTML. El selector **div:hover** establece el color de texto verde a los **div** cuando el mouse pasa encima de alguno de dichos elementos. Para este ejemplo, el **div** tendrá inicialmente su texto en color rojo, y cuando se posicione el mouse encima del mismo, su color cambiara a verde */

estilo.css

```
div {
  color: red;
}
```

```
div:hover {
  color: green;
}
```

ejemplo.html

```
<div>
   Mi color cambiará
</div>
```

:focus

Establece un conjunto de propiedades para un elemento cuando el elemento tiene el foco del cursor. Cuando el foco es retirado del elemento, vuelve a colocársele el CSS que tenía anterior al evento.

Pertenece al CSS2 y es soportado por todos los navegadores.

- Ejemplo:

/* En este ejemplo se muestra el código CSS en un archivo separado del código HTML. El selector **input:focus** establece el color de texto verde a los **input** cuando estos ganan el foco del cursor. Para este ejemplo, el **input** tendrá inicialmente su texto en color rojo, y cuando gane el foco del cursor, su color cambiara a verde */

estilo.css

```
input {
  color: red;
}

input:focus {
  color: green;
}
```

ejemplo.html

```
<input type='text' value='Mi color cambiará' />
```

:before

Se utiliza para agregar contenido antes del elemento seleccionado sin tener que colocarlo en el código HTML. Normalmente se utiliza para agregar un estilo particular que no puede agregarse fácilmente de otra manera o para no recargar el código HTML.

Pertenece al CSS2 y es soportado por todos los navegadores.

- Ejemplo:

/* En este ejemplo se muestra el código CSS en un archivo separado del código HTML. El selector **p:before** agrega un contenido antes de cualquier elemento **p**. Para este ejemplo, al tag **p** se le antepondrá un texto en color rojo que diga "Contenido extra -" */

estilo.css

```
p:before {
  content: "Contenido extra - ";
  color: red;
}
```

ejemplo.html

```
<p>Texto<p/>
```

:after

Se utiliza para agregar contenido despúes del elemento seleccionado sin tener que colocarlo en el código HTML. Normalmente se utiliza para agregar un estilo particular que no puede agregarse fácilmente de otra manera o para no recargar el código HTML.

Pertenece al CSS2 y es soportado por todos los navegadores.

- Ejemplo:

/* En este ejemplo se muestra el código CSS en un archivo separado del código HTML. El selector **p:after** agrega un contenido después de cualquier elemento **p**. Para este ejemplo, al tag **p** se le agregará después del mismo un texto en color rojo que diga "- Contenido extra " */

estilo.css

```
p:before {
  content: "- Contenido extra";
  color: red;
}
```

ejemplo.html

```
<p>Texto<p/>
```

:target

Los urls que contengan un # seguido de un texto, están apuntando a un elemento con id igual a dicho texto. Este elemento se conoce como **target** y el texto se conoce como **ancla.**

Por ejemplo, para el url: `http://www.ejemplo.com/index.html#intro` el ancla de la misma es **`intro`** y el elemento **target** tiene id igual a **`intro`**

Este selector permite agregar estilos al target de un url.

Pertenece al CSS3 y es soportado por todos los navegadores excepto Internet Explorer 8.0 y sus versiones anteriores.

- Ejemplo:

/* En este ejemplo se muestra el código CSS en un archivo separado del código HTML. El selector **:target** agrega un estilo al elemento referenciado por el **target** o **ancla** del url. Para este ejemplo el ancla del url es **intro** y el selector coloca el texto en rojo al target, que es un elemento p con id igual a **intro** */

url: http://www.ejemplo.com/index.html#intro

estilo.css

```
:target {
  color: red;
}
```

ejemplo.html

```
<p id=’intro’>Texto<p/>
```

:first-child

Selecciona el primer elemento hijo de un elemento padre o elemento contenedor.

Pertenece al CSS2 y es soportado por todos los navegadores.

- Ejemplo:

/* En este ejemplo se muestra el código CSS en un archivo separado del código HTML. El selector **div:first-child p** selecciona al primer hijo **p** que se encuentra dentro de los elementos **div** y les aplica un color de texto rojo. Para este ejemplo, solo se selecciona el primer **p** que se encuentra dentro del **div** */

estilo.css

```
div p:first-child {
  color: red;
}
```

ejemplo.html

```
<div>
  <p>Yo seré seleccionado<p/>

  <p>Yo NO seré seleccionado<p/>

  <a href=’’>Texto<a/>
</div>
```

:last-child

Selecciona el último elemento hijo de un elemento padre o elemento contenedor.

Pertenece al CSS3 y es soportado por todos los navegadores.

- Ejemplo:

/* En este ejemplo se muestra el código CSS en un archivo separado del código HTML. El selector **div:last-child p** selecciona al último hijo **p** que se encuentra dentro de los elementos **div** y les aplica un color de texto rojo. Para este ejemplo, solo se selecciona el último **p** que se encuentra dentro del **div** */

estilo.css

```
div p:last-child {
  color: red;
}
```

ejemplo.html

```
<div>
  <p>Yo NO seré seleccionado<p/>

  <p>Yo seré seleccionado<p/>

  <a href=’’>Texto<a/>
</div>
```

:only-child

Selecciona aquellos elementos que sean los únicos elementos hijos de su elemento padre o elemento contenedor.

Pertenece al CSS3 y es soportado por todos los navegadores excepto Internet Explorer 8.0 y sus versiones anteriores.

- Ejemplo:

/* En este ejemplo se muestra el código CSS en un archivo separado del código HTML. El selector **div:only-child p** selecciona a los **p** que sean hijos únicos del elemento **div** y les aplica un color de texto rojo. Para este ejemplo, se selecciona el tag **p** que se encuentra dentro del primer **div,** ya que el segundo **div** posee dos hijos (p y a) */

estilo.css

```
div p:only-child {
  color: red;
}
```

ejemplo.html

```
<div>
  <p>Yo seré seleccionado<p/>
</div>

<div>
  <p>Yo NO seré seleccionado<p/>

  <a href=’’>Texto<a/>
</div>
```

:nth-child():

Selecciona todos los elementos que sean el *n*-esimo hijo de su elemento padre o elemento contenedor, es decir, selecciona los elementos hijo que ocupan la posición *n* entre el total de hijos, donde *n* es un numero.

Esta propiedad acepta múltiples valores, los cuales definiremos a continuación en su sintaxis:

```
:nth-child(valor)
```

donde `valor` puede ser:

posición: número que indica la posición del elemento hijo que quiere seleccionarse.
`odd` : selecciona aquellos hijos cuya posición sea impar.
`even` : selecciona aquellos hijos cuya posición sea par.
`an + b` : formula en donde “a” es un numero que indica cada cuantas posiciones debe seleccionarse un elemento, “n” es un contador que empieza en 0 y b indica el número donde debe comenzar a contarse. Este caso lo veremos bien explicado en los ejemplos.

Pertenece al CSS3 y es soportado por todos los navegadores excepto Internet Explorer 8.0 y sus versiones anteriores.

- Ejemplos:

/* En estos ejemplos se muestra el código CSS en un un archivo separado del código HTML */

/***Ejemplo 1**: El selector **div p:nth-child(2)** selecciona los elementos hijo **p** que se encuentran en la segunda posición dentro de los elementos **div** y les aplica un color de texto verde. Para este ejemplo, solo el primer tag **p** que está dentro del **div** es seleccionado */

estilo.css

```
div p:nth-child(2) {
  color: green;
}
```

ejemplo.html

```
<div>
   <a href=’’>Yo NO seré seleccionado</a>
   <p>Yo seré seleccionado</p>
   <a href=’’>Yo NO seré seleccionado</a>
   <p>Yo NO seré seleccionado</p>
</div>
```

/***Ejemplo 2**: El selector **div :nth-child(odd)** selecciona TODOS los elementos hijo que se encuentran en las posiciones impares dentro de los elementos **div** y les aplica un color de texto verde. Para este ejemplo, son seleccionados los elementos en las posiciones 1 y 3 */

estilo.css

```
div :nth-child(odd) {
  color: green;
}
```

ejemplo.html

```
<div>
   <a href=’’>Yo seré seleccionado</a>
   <p>Yo NO seré seleccionado</p>
   <a href=’’>Yo seré seleccionado</a>
   <p>Yo NO seré seleccionado</p>
</div>
```

/***Ejemplo 3**: El selector **div a:nth-child(even)** selecciona los elementos hijo **a** que se encuentran en las posiciones pares dentro de los elementos **div** y les aplica un color de texto verde. Para este ejemplo, es seleccionado el elemento **a** en la posición 2 */

estilo.css

```
div a:nth-child(even) {
  color: green;
}
```

ejemplo.html

```
<div>
   <a href=’’>Yo NO seré seleccionado</a>
   <a href=’’>Yo seré seleccionado</a>
   <p>Yo NO seré seleccionado</p>
   <p>Yo NO seré seleccionado</p>
</div>
```

/***Ejemplo 4**: El selector **div :nth-child(2n+1)** indica con el **2n** que debe seleccionar cada segundo elemento hijo que se encuentra dentro de los elementos **div**, es decir, cuenta cada dos elementos y selecciona el segundo, luego reinicia la cuenta y continua seleccionando cada dos elementos, y se le indica con el **+1** que empiece a contar a partir del elemento 1. Luego les aplica un color de texto verde. Para este ejemplo, son seleccionados los elemento en las posiciones 2 y 3 */

estilo.css

```
div :nth-child(2n+1) {
  color: green;
}
```

ejemplo.html

```
<div>
   <a href=’’>Yo seré seleccionado</a>
   <a href=’’>Yo NO seré seleccionado</a>
   <p>Yo seré seleccionado</p>
   <p>Yo NO seré seleccionado</p>
</div>
```

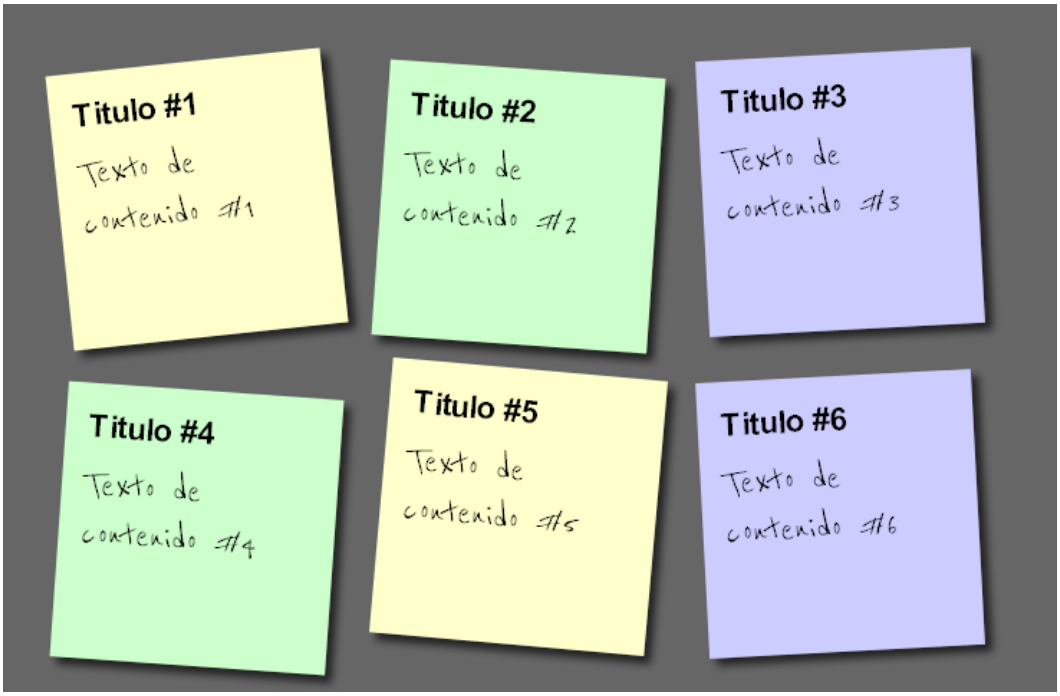
Sección de Prácticas

Práctica 1: Elaboración de “sticky notes” de colores con CSS3

En esta práctica construiremos una lista de HTML en un grupo de “sticky notes” o papeles adhesivos, de esos que se utilizan en oficinas.

El efecto es soportado en los navegadores modernos actualizados. Para los navegadores antiguos se pueden visualizar algunas de las propiedades.

RESULTADO FINAL



Paso 1: Construyendo el esqueleto y los cuadrados de los “stickies”

Arrancamos diseñando lo que sería la estructura básica de HTML que puede ser vista en todos los navegadores. Los sticky´s estarán representados en una lista no ordenada (o ´ul´) que será modificada con estilos de CSS3.

En este paso se deben realizar 2 archivos: **sticky.html**, que contiene la estructura en HTML y **estilo.css**, para guardar los estilos CSS3. Los dos archivos deben ser guardados en la misma carpeta.

A continuación se presenta el código de cada uno.

sticky.html

```
<!DOCTYPE html>
<html lang=’es’>
<head>
   <meta http-equiv="content-type" content="text/html; charset=utf-8" />
   <title>Paso 1 – Sticky Notes</title>
   <link rel="stylesheet" href="estilo.css"/>
<head>
<body>
   <ul>
      <li>
        <a href="#">
          <h2>Titulo #1</h2>
          <p>Texto de contenido #1</p>
        </a>
     </li>
     <li>
       <a href="#">
         <h2>Titulo #2</h2>
          <p>Texto de contenido #2</p>
       </a>
     </li>
     <li>
       <a href="#">
         <h2>Titulo #3</h2>
          <p>Texto de contenido #3</p>
       </a>
     </li>
     <li>
       <a href="#">
         <h2>Titulo #4</h2>
          <p>Texto de contenido #4</p>
       </a>
     </li>
     <li>
       <a href="#">
         <h2>Titulo #5</h2>
          <p>Texto de contenido #5</p>
       </a>
     </li>
     <li>
       <a href="#">
         <h2>Titulo #6</h2>
          <p>Texto de contenido #6</p>
       </a>
     </li>
   </ul>
</body>
</html>
```

estilo.css

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

body{
  font-family: arial,sans-serif;
  font-size: 100%;
  margin: 3em;
  background: #666;
  color: #fff;
}

h2,p{
  font-size: 100%;
  font-weight: normal;
}

ul,li{
  list-style:none;
}

ul{
  overflow:hidden;
  padding:3em;
}

ul li a{
  text-decoration:none;
  color:#000;
  background:#ffc;
  display:block;
  height:10em;
  width:10em;
  padding:1em;
}

ul li{
  margin:1em;
  float:left;
}
```

El resultado obtenido debe ser similar a la siguiente imagen:

Paso 2: Agregando sombras y fuentes personalizadas

Para darle más vida a los cuadrados que acabamos de relizar le agregaremos un par de sombras y haremos un cambio en la fuente utilizando el servicio gratuito Google Web Fonts.

Debajo del código donde agregamos el tag ´link´ a nuestro estilo (**estilo.css**), introducimos el siguiente código HTML:

```
<link href="http://fonts.googleapis.com/css?family=Reenie+Beanie:regular" rel="stylesheet" type="text/css">
```

Estas líneas permiten comunicarnos de forma gratuita al servicio de fuentes de tipografia para la web de Google y utilizar la fuente que escojamos. En este caso, escogemos la fuente “Reenie Beanie”.

El resultado de dicho cambio produce el siguiente archivo:

Nota: *las líneas resaltadas en* ***azul claro*** *indican los cambios que fueron realizados en este paso con respecto al paso anterior.*

sticky.html

```
<!DOCTYPE html>
<html lang=’es’>
<head>
   <meta http-equiv="content-type" content="text/html; charset=utf-8" />
   <title>Paso 1 – Sticky Notes</title>
   <link rel="stylesheet" href="estilo.css"/>
   <link href="http://fonts.googleapis.com/css?family=Reenie+Beanie:regular"
rel="stylesheet" type="text/css">
<head>
<body>
   <ul>
      <li>
        <a href="#">
          <h2>Titulo #1</h2>
          <p>Texto de contenido #1</p>
        </a>
     </li>
     <li>
       <a href="#">
         <h2>Titulo #2</h2>
          <p>Texto de contenido #2</p>
       </a>
     </li>
     <li>
       <a href="#">
         <h2>Titulo #3</h2>
          <p>Texto de contenido #3</p>
       </a>
     </li>
     <li>
       <a href="#">
         <h2>Titulo #4</h2>
          <p>Texto de contenido #4</p>
       </a>
     </li>
     <li>
       <a href="#">
         <h2>Titulo #5</h2>
          <p>Texto de contenido #5</p>
       </a>
     </li>
     <li>
       <a href="#">
         <h2>Titulo #6</h2>
          <p>Texto de contenido #6</p>
       </a>
     </li>
   </ul>
</body>
</html>
```

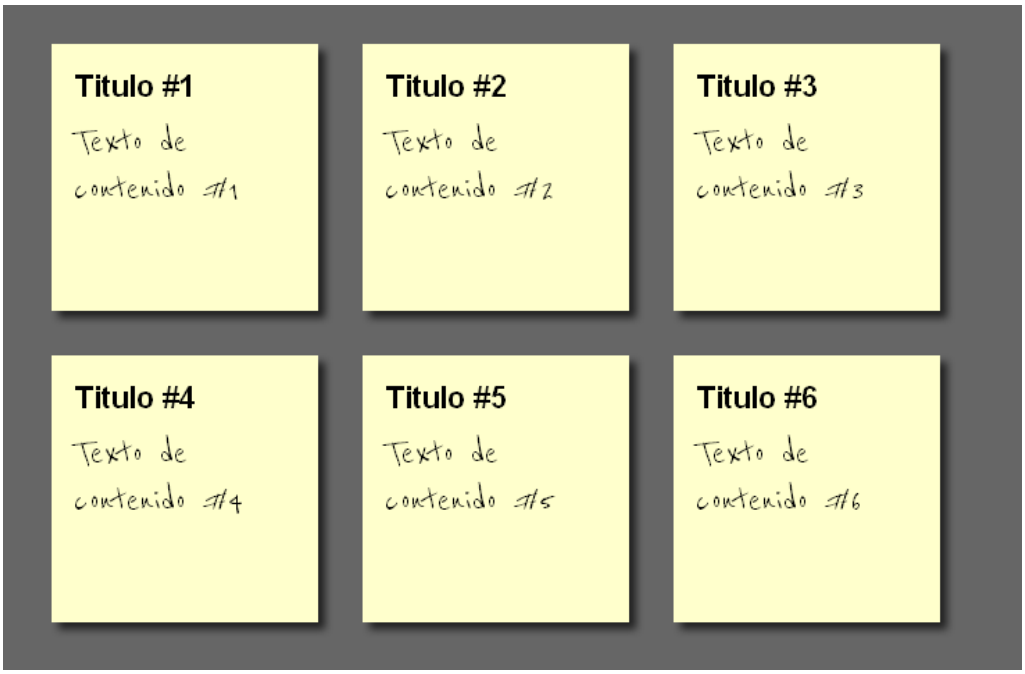
Ahora modificamos el archivo **estilo.css** para agregar el soporte a la fuente de Google y agregarle la sombra a los sticky notes. El archivo **estilo.css** quedará de la siguiente forma:

estilo.css

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

body{
  font-family:arial,sans-serif;
  font-size:100%;
  margin:3em;
  background:#666;
  color:#fff;
}

h2,p{
  font-size:100%;
  font-weight:normal;
}

ul,li{
  list-style:none;
}

ul{
  overflow:hidden;
  padding:3em;
}

ul li a{
  text-decoration:none;
  color:#000;
  background:#ffc;
  display:block;
  height:10em;
  width:10em;
  padding:1em;
  -moz-box-shadow:5px 5px 7px rgba(33,33,33,1);        /* Firefox */
  -webkit-box-shadow: 5px 5px 7px rgba(33,33,33,.7);   /* Safari+Chrome */
  box-shadow: 5px 5px 7px rgba(33,33,33,.7);           /* Opera */
}

ul li{
  margin:1em;
  float:left;
}

ul li h2{
  font-size:140%;
  font-weight:bold;
  padding-bottom:10px;
```

```
}

ul li p{
  font-family:"Reenie Beanie",arial,sans-serif;
  font-size:180%;
}
```

Para este paso, el resultado final sería el siguiente:



Paso 3: Rotando los Sticky notes

En este paso agregamos uno de los nuevos efectos que nos ofrece la tecnología CSS3: rotaciones.

Las rotaciones en CSS3 son consideradas como ¨Transformaciones¨, y poseen una sintaxis en la que podemos especificar el número de grados a los cuales queremos rotar un elemento.

Únicamente cambiaremos el archivo **estilo.css**, obteniendo el siguiente contenido:

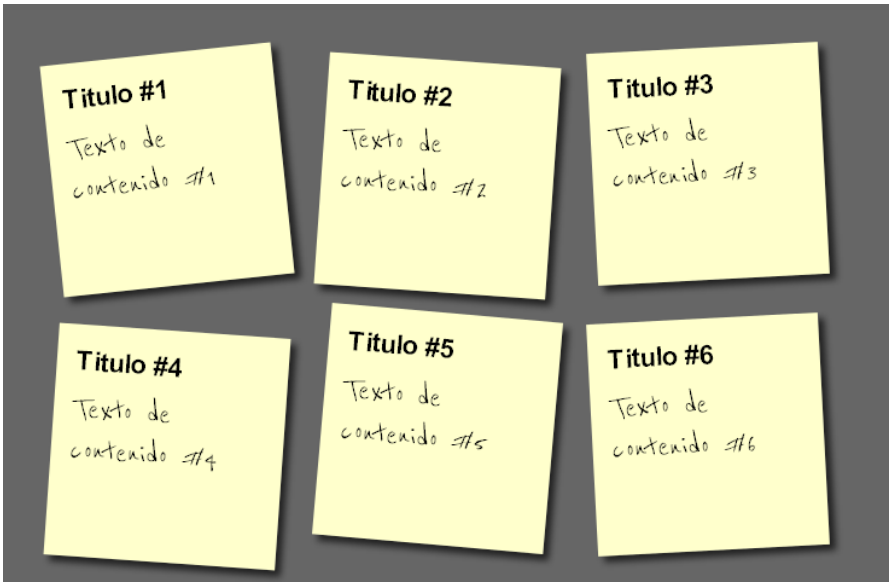
Nota: *las líneas subrayadas en* ***azul claro*** *indican los cambios que fueron realizados en este paso con respecto al paso anterior.*

estilo.css

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

body{
  font-family: arial,sans-serif;
  font-size:100%;
  margin:3em;
  background:#666;
  color:#fff;
}

h2,p{
  font-size:100%;
  font-weight:normal;
}

ul,li{
  list-style:none;
}

ul{
  overflow:hidden;
  padding:3em;
}

ul li a{
  text-decoration:none;
  color:#000;
  background:#ffc;
  display:block;
  height:10em;
  width:10em;
  padding:1em;
  -moz-box-shadow:5px 5px 7px rgba(33,33,33,1);        /* Firefox */
  -webkit-box-shadow: 5px 5px 7px rgba(33,33,33,.7);   /* Safari+Chrome */
  box-shadow: 5px 5px 7px rgba(33,33,33,.7);           /* Opera */
}

ul li{
  margin:1em;
  float:left;
}

ul li h2{
  font-size:140%;
  font-weight:bold;
  padding-bottom:10px;
}

ul li p{
```

```
  font-family:"Reenie Beanie",arial,sans-serif;
  font-size:180%;
}

ul li a{
       -webkit-transform:rotate(-6deg);
       -o-transform:rotate(-6deg);
       -moz-transform:rotate(-6deg);
}

ul li:nth-child(even) a{
  -o-transform:rotate(4deg);
  -webkit-transform:rotate(4deg);
  -moz-transform:rotate(4deg);
  position:relative;
  top:5px;
}

ul li:nth-child(3n) a{
  -o-transform:rotate(-3deg);
  -webkit-transform:rotate(-3deg);
  -moz-transform:rotate(-3deg);
  position:relative;
  top:-5px;
}

ul li:nth-child(5n) a{
  -o-transform:rotate(5deg);
  -webkit-transform:rotate(5deg);
  -moz-transform:rotate(5deg);
  position:relative;
  top:-10px;
}
```

El paso 3 debe generar la siguiente pantalla:



Paso 4: Acercamiento al pasar el mouse sobre los stickies (hover)

Nuevamente estamos usando una funcionalidad de CSS3: transformación por escalamiento. Este tipo de transformación nos servirá para implementar un efecto de acercamiento (zooming) sobre el "sticky note".

El código fuente del archivo **estilo.css** se muestra a continuación:

Nota: *las líneas subrayadas en* ***azul claro*** *indican los cambios que fueron realizados en este paso con respecto al paso anterior.*

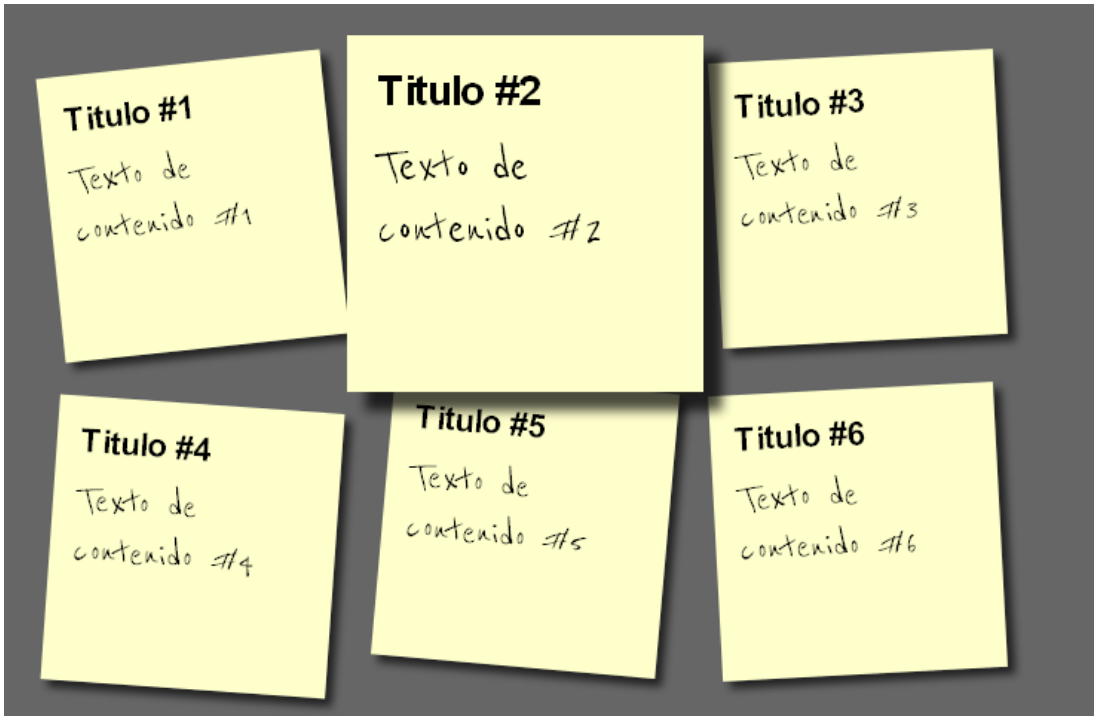
estilo.css

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

body{
  font-family:arial,sans-serif;
  font-size:100%;
  margin:3em;
  background:#666;
  color:#fff;
}

h2,p{
  font-size:100%;
  font-weight:normal;
}

ul,li{
  list-style:none;
}

ul{
  overflow:hidden;
  padding:3em;
}

ul li a{
  text-decoration:none;
  color:#000;
  background:#ffc;
  display:block;
  height:10em;
  width:10em;
  padding:1em;
  -moz-box-shadow:5px 5px 7px rgba(33,33,33,1);        /* Firefox */
  -webkit-box-shadow: 5px 5px 7px rgba(33,33,33,.7);   /* Safari+Chrome */
  box-shadow: 5px 5px 7px rgba(33,33,33,.7);           /* Opera */
```

```
}

ul li{
  margin:1em;
  float:left;
}

ul li h2{
  font-size:140%;
  font-weight:bold;
  padding-bottom:10px;
}

ul li p{
  font-family:"Reenie Beanie",arial,sans-serif;
  font-size:180%;
}

ul li a{
   -webkit-transform:rotate(-6deg);
   -o-transform:rotate(-6deg);
   -moz-transform:rotate(-6deg);
}

ul li:nth-child(even) a{
  -o-transform:rotate(4deg);
  -webkit-transform:rotate(4deg);
  -moz-transform:rotate(4deg);
  position:relative;
  top:5px;
}

ul li:nth-child(3n) a{
  -o-transform:rotate(-3deg);
  -webkit-transform:rotate(-3deg);
  -moz-transform:rotate(-3deg);
  position:relative;
  top:-5px;
}

ul li:nth-child(5n) a{
  -o-transform:rotate(5deg);
  -webkit-transform:rotate(5deg);
  -moz-transform:rotate(5deg);
  position:relative;
  top:-10px;
}

ul li a:hover,ul li a:focus{
  -moz-box-shadow:10px 10px 7px rgba(0,0,0,.7);
  -webkit-box-shadow: 10px 10px 7px rgba(0,0,0,.7);
  box-shadow:10px 10px 7px rgba(0,0,0,.7);
  -webkit-transform: scale(1.25);
  -moz-transform: scale(1.25);
```

```
  -o-transform: scale(1.25);
  position:relative;
  z-index:5;
}
```

Este paso debería producir un resultado similar al siguiente:



Paso 5: Agregando efectos de sombras y transiciones suaves

Para finalizar la práctica, damos los últimos toques técnicos para que los sticky tengan un aspecto profesional: introducimos sombras mediante la propiedad box-shadow y efectos de transición suaves para darle un toque sutil a la animación

El código fuente del archivo **estilo.css** se muestra a continuación:

Nota: *las líneas subrayadas en* ***azul claro*** *indican los cambios que fueron realizados en este paso con respecto al paso anterior.*

estilo.css

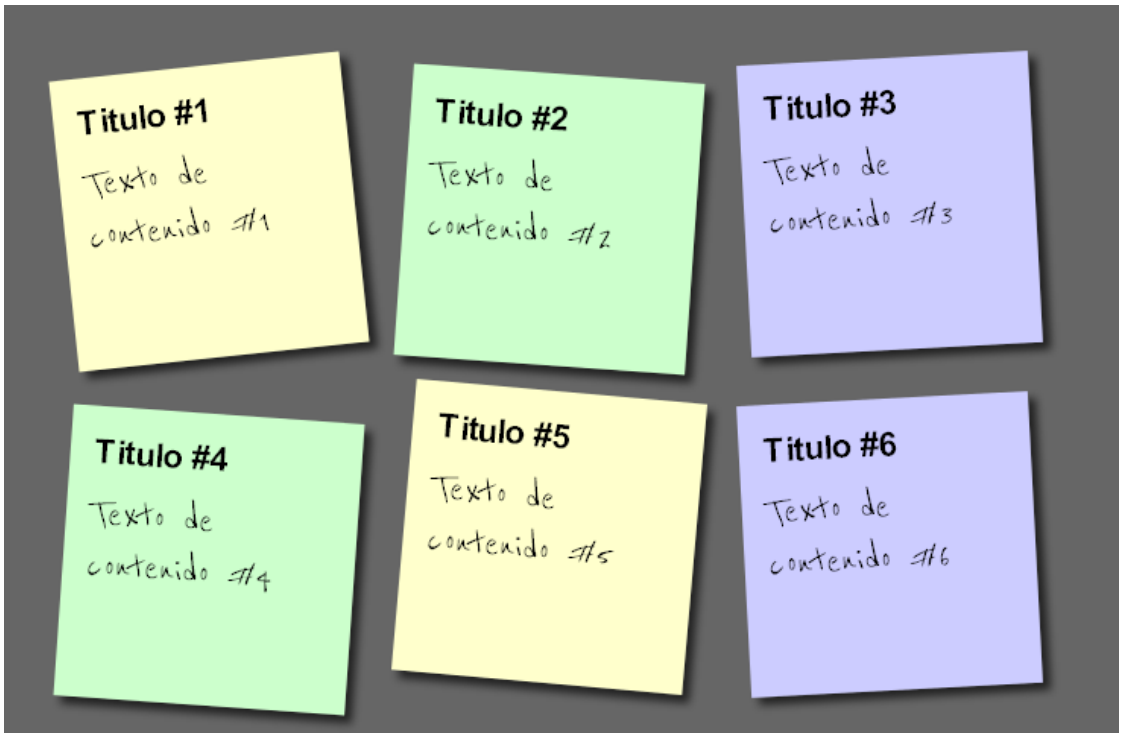
```
*{
  margin:0;
  padding:0;
}
```

```
body{
  font-family:arial,sans-serif;
  font-size:100%;
  margin:3em;
  background:#666;
  color:#fff;
}

h2,p{
  font-size:100%;
  font-weight:normal;
}

ul,li{
  list-style:none;
}

ul{
  overflow:hidden;
  padding:3em;
}

ul li a{
  text-decoration:none;
  color:#000;
  background:#ffc;
  display:block;
  height:10em;
  width:10em;
  padding:1em;
  -moz-box-shadow:5px 5px 7px rgba(33,33,33,1);        /* Firefox */
  -webkit-box-shadow: 5px 5px 7px rgba(33,33,33,.7);   /* Safari+Chrome */
  box-shadow: 5px 5px 7px rgba(33,33,33,.7);           /* Opera */
}

ul li{
  margin:1em;
  float:left;
}

ul li h2{
  font-size:140%;
  font-weight:bold;
  padding-bottom:10px;
}

ul li p{
  font-family:"Reenie Beanie",arial,sans-serif;
  font-size:180%;
}

ul li a{
   -webkit-transform:rotate(-6deg);
```

```
   -o-transform:rotate(-6deg);
   -moz-transform:rotate(-6deg);
}

ul li:nth-child(even) a{
  -o-transform:rotate(4deg);
  -webkit-transform:rotate(4deg);
  -moz-transform:rotate(4deg);
  position:relative;
  top:5px;
}

ul li:nth-child(3n) a{
  -o-transform:rotate(-3deg);
  -webkit-transform:rotate(-3deg);
  -moz-transform:rotate(-3deg);
  position:relative;
  top:-5px;
}

ul li:nth-child(5n) a{
  -o-transform:rotate(5deg);
  -webkit-transform:rotate(5deg);
  -moz-transform:rotate(5deg);
  position:relative;
  top:-10px;
}
ul li a:hover,ul li a:focus{
  -moz-box-shadow:10px 10px 7px rgba(0,0,0,.7);
  -webkit-box-shadow: 10px 10px 7px rgba(0,0,0,.7);
  box-shadow:10px 10px 7px rgba(0,0,0,.7);
  -webkit-transform: scale(1.25);
  -moz-transform: scale(1.25);
  -o-transform: scale(1.25);
  position:relative;
  z-index:5;
}

ul li a{
  text-decoration:none;
  color:#000;
  background:#ffc;
  display:block;
  height:10em;
  width:10em;
  padding:1em;
  -moz-box-shadow:5px 5px 7px rgba(33,33,33,1);
  -webkit-box-shadow: 5px 5px 7px rgba(33,33,33,.7);
  box-shadow: 5px 5px 7px rgba(33,33,33,.7);
  -moz-transition:-moz-transform .15s linear;
  -o-transition:-o-transform .15s linear;
  -webkit-transition:-webkit-transform .15s linear;
}
```

```
ul li:nth-child(even) a{
  -o-transform:rotate(4deg);
  -webkit-transform:rotate(4deg);
  -moz-transform:rotate(4deg);
  position:relative;
  top:5px;
  background:#cfc;
}

ul li:nth-child(3n) a{
  -o-transform:rotate(-3deg);
  -webkit-transform:rotate(-3deg);
  -moz-transform:rotate(-3deg);
  position:relative;
  top:-5px;
  background:#ccf;
}
```

La implementación completa de todos los pasos debería darnos la siguiente imagen:



Práctica 2: Tablas redondeadas con CSS3

En el pasado nos hemos enfrentado al reto de crear tablas con bordes redondeados empleando como solución el uso de imágenes y complicadas clases en el CSS para lograr que se puedan acoplar con el código HTML.

Con HTML5 y CSS3 tenemos la posibilidad de crear de manera sencilla tablas con bordes redondeados sin utilizar imágenes y escribiendo un código CSS sencillo.

En esta práctica lograremos hacer una tabla como la siguiente:

Tabla con estilo "Zebra" y Pie

#	Top 10 Peliculas	Year
1	The Shawshank Redemption	1994
2	The Godfather	1972
3	The Godfather: Part II	1974
4	The Good, the Bad and the Ugly	1966
5	Pulp Fiction	1994
6	12 Angry Men	1957
7	Schindler's List	1993
8	One Flew Over the Cuckoo's Nest	1975
9	The Dark Knight	2008
10	The Lord of the Rings: The Return of the King	2003

¿Qué resulta tan interesante en estas tablas?

Las tablas que crearemos en esta práctica son diseñadas completamente en CSS3 y poseen tres (3) elementos que las hacen destacar:

- Bordes redondeados sin imágenes
- Facilidad de actualización (no se agrega ningún id o clase extra).
- Amigable a la vista del usuario y fácil de leer.

Para empezar a construir nuestra tabla, diseñamos el esqueleto de la misma empleando el siguiente código HTML:

tablasRedondeadas.html

```
 <!DOCTYPE html>
<html>
<head>
    <title>Practica 2 Tablas Redondeadas con CSS3</title>
    <link rel='stylesheet' href="tablas.css" type='text/css'/>
</head>

<body>
<h2>Tabla con estilo "Zebra" y Pie</h2>

<table class="zebra">
    <thead>
    <tr>
        <th>#</th>
        <th>Top 10 Peliculas</th>
        <th>Year</th>

    </tr>
    </thead>

    <tfoot>
    <tr>
        <td> </td>
        <td></td>
        <td></td>
    </tr>
    </tfoot>
    <tr>

        <td>1</td>
        <td>The Shawshank Redemption</td>

        <td>1994</td>
    </tr>
    <tr>
        <td>2</td>
        <td>The Godfather</td>
        <td>1972</td>

    </tr>

    <tr>
        <td>3</td>
        <td>The Godfather: Part II</td>
        <td>1974</td>
    </tr>
    <tr>
        <td>4</td>
```

```
        <td>The Good, the Bad and the Ugly</td>

        <td>1966</td>
    </tr>
    <tr>
        <td>5</td>
        <td>Pulp Fiction</td>
        <td>1994</td>
    </tr>

    <tr>
        <td>6</td>
        <td>12 Angry Men</td>
        <td>1957</td>
    </tr>
    <tr>
        <td>7</td>
        <td>Schindler's List</td>

        <td>1993</td>
    </tr>
    <tr>
        <td>8</td>
        <td>One Flew Over the Cuckoo's Nest</td>
        <td>1975</td>
    </tr>

    <tr>

        <td>9</td>
        <td>The Dark Knight</td>
        <td>2008</td>
    </tr>
    <tr>
        <td>10</td>
        <td>The Lord of the Rings: The Return of the King</td>

        <td>2003</td>
    </tr>
</table>


</body>
</html>
```

Como podemos ver, nuestra tabla está trabajando con HTML y no HTML5, no obstante, nuestro navegador reconoce el documento como un documento de HTML5. Lo definimos como tal ya que trabajaremos con nuevas propiedades del CSS3 soportadas sólo por HTML5 y no HTML4.

Seguidamente creamos la hoja de estilos en el archivo **tablas.css** para darle vida a la tabla. Para darle una aspecto elegante y aumentar la facilidad de lectura, podemos intercalar el color de cada fila utilizando el selector `:nth-child(even)`, que se encarga de seleccionar

únicamente las filas pares y agregarle las reglas asociadas al mismo. El código CSS para realizarlo es el siguiente:

tablas.css

```
body {
    width: 600px;
    margin: 40px auto;
    font-family: 'trebuchet MS', 'Lucida sans', Arial;
    font-size: 14px;
    color: #444;
}

table {
    *border-collapse: collapse; /* IE7 and lower */
    border-spacing: 0;
    width: 100%;
}


.zebra td, .zebra th {
    padding: 10px;
    border-bottom: 1px solid #f2f2f2;
}

.zebra tbody tr:nth-child(even) {
    background: #f5f5f5;
    -webkit-box-shadow: 0 1px 0 rgba(255,255,255,.8) inset;
    -moz-box-shadow:0 1px 0 rgba(255,255,255,.8) inset;
    box-shadow: 0 1px 0 rgba(255,255,255,.8) inset;
}

.zebra th {
    text-align: left;
    text-shadow: 0 1px 0 rgba(255,255,255,.5);
    border-bottom: 1px solid #ccc;
    background-color: #eee;
    background-image: -webkit-gradient(linear, left top, left bottom,
from(#f5f5f5), to(#eee));
    background-image: -webkit-linear-gradient(top, #f5f5f5, #eee);
    background-image:    -moz-linear-gradient(top, #f5f5f5, #eee);
    background-image:     -ms-linear-gradient(top, #f5f5f5, #eee);
    background-image:      -o-linear-gradient(top, #f5f5f5, #eee);
    background-image:         linear-gradient(top, #f5f5f5, #eee);
}
```

En este código también observamos el uso de 3 propiedades nuevas de CSS3: `box-shadow`, `text-shadow` y `linear-gradient`.

- **box-shadow,** y sus versiones específicas para cada navegador, permiten agregar una sombra exterior al borde del elemento seleccionado. La sombra puede ser de cualquier color y si le agregamos el valor `inset` se convierte en una sombra interior.

- **text-shadow**: es similar a box-shadow, pero sólo puede ser utilizada para textos.
- **linear-gradient:** aún está en fase de definición, pero nos permite crear degrades de colores sin necesidad de imágenes.

Ahora procedamos a colocar los bordes redondeados en la tabla. Para colocarlos utilizamos la propiedad `border-radius` que acepta 4 valores posibles, cada uno de ellos representando el radio de curvatura en pixeles para cada una de las esquinas de la tabla. Agregamos, pues, el siguiente código CSS al mismo archivo con el que venimos trabajando:

tablas.css

```
.zebra th:first-child {
    -moz-border-radius: 6px 0 0 0;
    -webkit-border-radius: 6px 0 0 0;
    border-radius: 6px 0 0 0;
}

.zebra th:last-child {
    -moz-border-radius: 0 6px 0 0;
    -webkit-border-radius: 0 6px 0 0;
    border-radius: 0 6px 0 0;
}

.zebra th:only-child{
    -moz-border-radius: 6px 6px 0 0;
    -webkit-border-radius: 6px 6px 0 0;
    border-radius: 6px 6px 0 0;
}

.zebra tfoot td {
    border-bottom: 0;
    border-top: 1px solid #fff;
    background-color: #f1f1f1;
}

.zebra tfoot td:first-child {
    -moz-border-radius: 0 0 0 6px;
    -webkit-border-radius: 0 0 0 6px;
    border-radius: 0 0 0 6px;
}

.zebra tfoot td:last-child {
    -moz-border-radius: 0 0 6px 0;
    -webkit-border-radius: 0 0 6px 0;
    border-radius: 0 0 6px 0;
}

.zebra tfoot td:only-child{
    -moz-border-radius: 0 0 6px 6px;
    -webkit-border-radius: 0 0 6px 6px
    border-radius: 0 0 6px 6px
}
```

En el código anterior podemos observar el uso de los selectores de CSS3 `:first-child`, `:last-child` y `:only-child`, los cuales permiten aplicar las reglas CSS al primer y al último elementos hijos de la tabla (`:first-child` y `:last-child` respectivamente) y cuando alguna sección de la tabla tiene un solo elemento se utiliza el selector `:only-child`.

Soporte con los Browsers

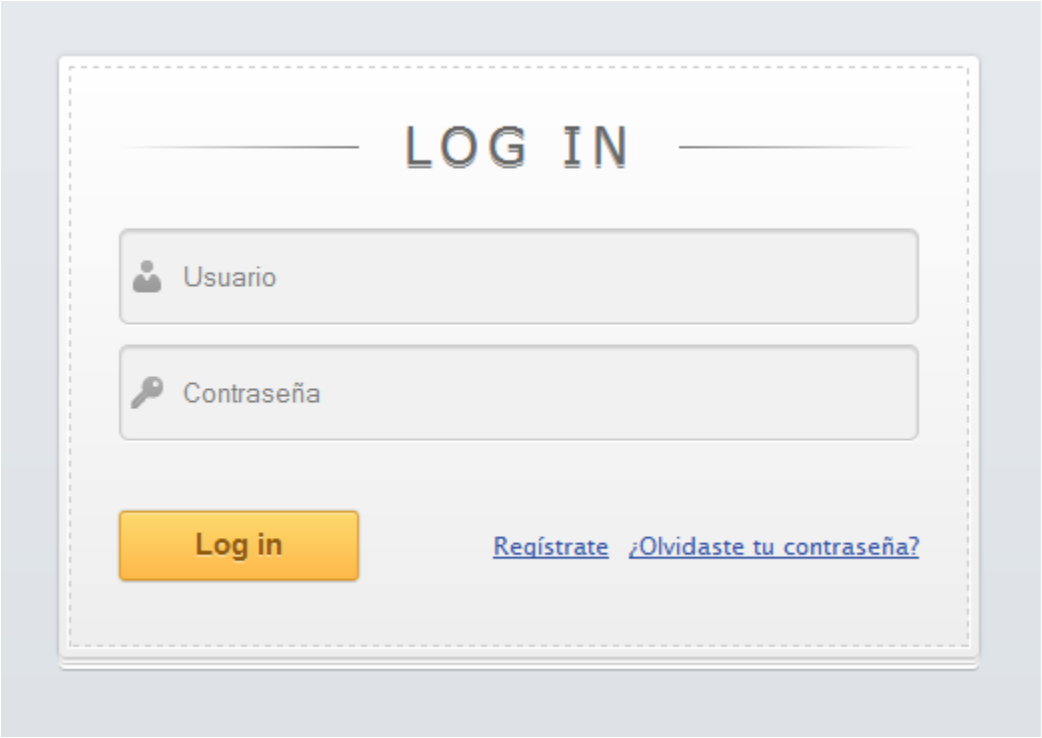
Para los Browsers o navegadores web modernos (últimas versiones de Opera, Mozilla, Chrome, Internet Explorer), se puede trabajar sin problemas con los nuevas propiedades del CSS3. Sin embargo, para exploradores tales como Internet Explorer 8 y versiones anteriores, simplemente no aplica el efecto de bordes redondeados.

Como solución sencilla se recomienda la actualización constante de los navegadores, con esto estaríamos garantizando el funcionamiento de nuestro código. Recuerda que si no actualiza sus aplicaciones, éstas con el tiempo dejarán de funcionar.

Práctica 3: Crear un formulario de login con CSS3

Como ya conocemos, CSS3 está habilitado para crear una cantidad de nuevas posibilidades para diseñar e implementar mejores formularios webs. También HTML5 tiene un importante rol cuando se trata de crear formularios más fáciles de usar sin necesitar ningún código javascript.

En esta práctica crearemos un formulario de login con un estilo elegante como el siguiente:



Comenzamos creando el esqueleto del formulario con el siguiente código HTML:

login.html

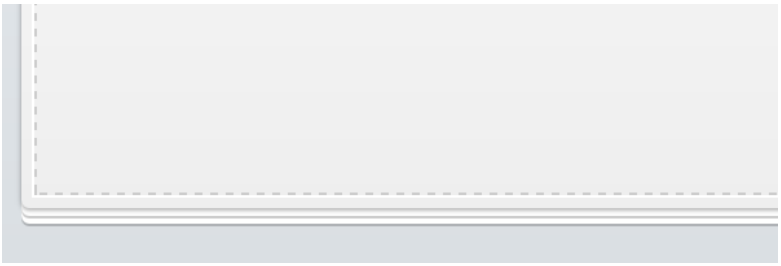
```
<!DOCTYPE html>
<html>
<head>
    <meta content="text/html; charset=utf-8" http-equiv="Content-Type">
    <title>Crear un formulario de login usando CSS3 y HTML5 </title>
    <link type="text/css" rel="stylesheet" href="estilos.css">
</head>

<body>

<form id="login">
    <h1>Log In</h1>
```

```
    <fieldset id="inputs">
        <input id="username" type="text" placeholder="Usuario" autofocus
required>
        <input id="password" type="password" placeholder="Password" required>
    </fieldset>
    <fieldset id="actions">
        <input type="submit" id="submit" value="Log in">
        <a href="">Olvidaste tu contraseña?</a><a href="">Regístrate</a>
    </fieldset>
</form>

</body>
</html>
```

En este código podemos observar nuevos atributos en las etiquetas, los cuales forman parte de las especificaciones de HTML5. Veamos cuáles son:

- **placeholder**: es una breve pista (una palabra o una frase corta) utilizada para dar una idea al usuario de lo que debe escribir en el elemento.
- **required**: especifica si el elemento es requerido para el envío del formulario.
- **autofocus:** especifica si el elemento representa un control que debe ser enfocado automáticamente luego que la página termine de ser cargada.

Luego definimos el CSS que utilizará el formulario. Empezamos definiendo un efecto de sombreado especial en el borde inferior del formulario:



Este código lo colocamos en el archivo **estilos.css**:

estilos.css:

```
html, body
{
    height: 100%;
}

body
{
```

```
    font: 12px 'Lucida Sans Unicode', 'Trebuchet MS', Arial, Helvetica;
    margin: 0;
    background-color: #d9dee2;
    background-image: -webkit-gradient(linear, left top, left bottom,
from(#ebeef2), to(#d9dee2));
    background-image: -webkit-linear-gradient(top, #ebeef2, #d9dee2);
    background-image: -moz-linear-gradient(top, #ebeef2, #d9dee2);
    background-image: -ms-linear-gradient(top, #ebeef2, #d9dee2);
    background-image: -o-linear-gradient(top, #ebeef2, #d9dee2);
    background-image: linear-gradient(top, #ebeef2, #d9dee2);
}

/*--------------------*/

#login
{
    background-color: #fff;
    background-image: -webkit-gradient(linear, left top, left bottom,
from(#fff), to(#eee));
    background-image: -webkit-linear-gradient(top, #fff, #eee);
    background-image: -moz-linear-gradient(top, #fff, #eee);
    background-image: -ms-linear-gradient(top, #fff, #eee);
    background-image: -o-linear-gradient(top, #fff, #eee);
    background-image: linear-gradient(top, #fff, #eee);
    height: 240px;
    width: 400px;
    margin: -150px 0 0 -230px;
    padding: 30px;
    position: absolute;
    top: 50%;
    left: 50%;
    z-index: 0;
    -moz-border-radius: 3px;
    -webkit-border-radius: 3px;
    border-radius: 3px;
    -webkit-box-shadow:
          0 0 2px rgba(0, 0, 0, 0.2),
          0 1px 1px rgba(0, 0, 0, .2),
          0 3px 0 #fff,
          0 4px 0 rgba(0, 0, 0, .2),
          0 6px 0 #fff,
          0 7px 0 rgba(0, 0, 0, .2);
    -moz-box-shadow:
          0 0 2px rgba(0, 0, 0, 0.2),
          1px 1px   0 rgba(0,   0,   0,   .1),
          3px 3px   0 rgba(255, 255, 255, 1),
          4px 4px   0 rgba(0,   0,   0,   .1),
          6px 6px   0 rgba(255, 255, 255, 1),
          7px 7px   0 rgba(0,   0,   0,   .1);
    box-shadow:
          0 0 2px rgba(0, 0, 0, 0.2),
          0 1px 1px rgba(0, 0, 0, .2),
          0 3px 0 #fff,
          0 4px 0 rgba(0, 0, 0, .2),
```

```
          0 6px 0 #fff,
          0 7px 0 rgba(0, 0, 0, .2);
}



#login:before
{
    content: '';
    position: absolute;
    z-index: -1;
    border: 1px dashed #ccc;
    top: 5px;
    bottom: 5px;
    left: 5px;
    right: 5px;
    -moz-box-shadow: 0 0 0 1px #fff;
    -webkit-box-shadow: 0 0 0 1px #fff;
    box-shadow: 0 0 0 1px #fff;
}
```

Aunque sea un poco largo, la mayoría de cosas las hemos visto ya en prácticas anteriores. Cabe mencionar el uso de la propiedad `box-shadow` (y sus versiones para los diferentes navegadores) que se utiliza para dibujar las múltiples sombras y/o capas que se encuentran en el borde inferior del formulario. Vemos también el uso de las propiedades `border-radius` y `linear-gradient` para colocar al formulario los bordes redondeados y los degrades de colores, respectivamente.

Por último, nos encontramos ante el uso del selector `:before` en última regla definida. Este selector sirve para generar **pseudo-elementos** o elementos parciales, que son elementos que no se escriben en el código HTML pero que son generados y mostrados en los navegadores modernos para darle estilo a otros elementos. Se utilizan en su mayoría para crear estilos complejos sin necesidad de modificar el código HTML que creamos.

Ahora definamos los efectos del título “Login”. Veamos el resultado:

En la imagen podemos ver en detalle que la letra posee una sombra, para lo cual se utiliza la propiedad `text-shadow;` también se muestran unas líneas difuminadas a los lados del

título, los cuales se generan con la propiedad `linear-gradient`. A continuación se muestra el código CSS que se debe agregar al archivo que venimos trabajando para crear estos efectos:

estilos.css

```
h1
{
    text-shadow: 0 1px 0 rgba(255, 255, 255, .7), 0px 2px 0 rgba(0, 0, 0, .5);
    text-transform: uppercase;
    text-align: center;
    color: #666;
    margin: 0 0 30px 0;
    letter-spacing: 4px;
    font: normal 26px/1 Verdana, Helvetica;
    position: relative;
}

h1:after, h1:before
{
    background-color: #777;
    content: "";
    height: 1px;
    position: absolute;
    top: 15px;
    width: 120px;
}

h1:after
{
    background-image: -webkit-gradient(linear, left top, right top, from(#777),
to(#fff));
    background-image: -webkit-linear-gradient(left, #777, #fff);
    background-image: -moz-linear-gradient(left, #777, #fff);
    background-image: -ms-linear-gradient(left, #777, #fff);
    background-image: -o-linear-gradient(left, #777, #fff);
    background-image: linear-gradient(left, #777, #fff);
    right: 0;
}

h1:before
{
    background-image: -webkit-gradient(linear, right top, left top, from(#777),
to(#fff));
    background-image: -webkit-linear-gradient(right, #777, #fff);
    background-image: -moz-linear-gradient(right, #777, #fff);
    background-image: -ms-linear-gradient(right, #777, #fff);
    background-image: -o-linear-gradient(right, #777, #fff);
    background-image: linear-gradient(right, #777, #fff);
    left: 0;
}
```

Finalmente agregamos estilos para el resto de los elementos del formulario, utilizando en su mayoría las mismas propiedades que hemos venido utilizando. Los colocamos al final del archivo **estilos.css:**

estilos.css

```
fieldset
{
    border: 0;
    padding: 0;
    margin: 0;
}

/*--------------------*/

#inputs input
{
    background: #f1f1f1 url(imagenes/login-sprite.png) no-repeat;
    padding: 15px 15px 15px 30px;
    margin: 0 0 10px 0;
    width: 353px; /* 353 + 2 + 45 = 400 */
    border: 1px solid #ccc;
    -moz-border-radius: 5px;
    -webkit-border-radius: 5px;
    border-radius: 5px;
    -moz-box-shadow: 0 1px 1px #ccc inset, 0 1px 0 #fff;
    -webkit-box-shadow: 0 1px 1px #ccc inset, 0 1px 0 #fff;
    box-shadow: 0 1px 1px #ccc inset, 0 1px 0 #fff;
}

#username
{
    background-position: 5px -2px !important;
}

#password
{
    background-position: 5px -52px !important;
}

#inputs input:focus
{
    background-color: #fff;
    border-color: #e8c291;
    outline: none;
    -moz-box-shadow: 0 0 0 1px #e8c291 inset;
    -webkit-box-shadow: 0 0 0 1px #e8c291 inset;
    box-shadow: 0 0 0 1px #e8c291 inset;
}

/*--------------------*/
#actions
{
```

```
    margin: 25px 0 0 0;
}

#submit
{
    background-color: #ffb94b;
    background-image: -webkit-gradient(linear, left top, left bottom,
from(#fddb6f), to(#ffb94b));
    background-image: -webkit-linear-gradient(top, #fddb6f, #ffb94b);
    background-image: -moz-linear-gradient(top, #fddb6f, #ffb94b);
    background-image: -ms-linear-gradient(top, #fddb6f, #ffb94b);
    background-image: -o-linear-gradient(top, #fddb6f, #ffb94b);
    background-image: linear-gradient(top, #fddb6f, #ffb94b);

    -moz-border-radius: 3px;
    -webkit-border-radius: 3px;
    border-radius: 3px;

    text-shadow: 0 1px 0 rgba(255,255,255,0.5);

     -moz-box-shadow: 0 0 1px rgba(0, 0, 0, 0.3), 0 1px 0 rgba(255, 255, 255,
0.3) inset;
     -webkit-box-shadow: 0 0 1px rgba(0, 0, 0, 0.3), 0 1px 0 rgba(255, 255, 255,
0.3) inset;
     box-shadow: 0 0 1px rgba(0, 0, 0, 0.3), 0 1px 0 rgba(255, 255, 255, 0.3)
inset;

    border-width: 1px;
    border-style: solid;
    border-color: #d69e31 #e3a037 #d5982d #e3a037;

    float: left;
    height: 35px;
    padding: 0;
    width: 120px;
    cursor: pointer;
    font: bold 15px Arial, Helvetica;
    color: #8f5a0a;
}

#submit:hover,#submit:focus
{
    background-color: #fddb6f;
    background-image: -webkit-gradient(linear, left top, left bottom,
from(#ffb94b), to(#fddb6f));
    background-image: -webkit-linear-gradient(top, #ffb94b, #fddb6f);
    background-image: -moz-linear-gradient(top, #ffb94b, #fddb6f);
    background-image: -ms-linear-gradient(top, #ffb94b, #fddb6f);
    background-image: -o-linear-gradient(top, #ffb94b, #fddb6f);
    background-image: linear-gradient(top, #ffb94b, #fddb6f);
}

#submit:active
{
```

```
    outline: none;
     -moz-box-shadow: 0 1px 4px rgba(0, 0, 0, 0.5) inset;
     -webkit-box-shadow: 0 1px 4px rgba(0, 0, 0, 0.5) inset;
     box-shadow: 0 1px 4px rgba(0, 0, 0, 0.5) inset;
}

#submit::-moz-focus-inner
{
  border: none;
}

#actions a
{
    color: #3151A2;
    float: right;
    line-height: 35px;
    margin-left: 10px;
}

/*--------------------*/

#back
{
    display: block;
    text-align: center;
    position: relative;
    top: 60px;
    color: #999;
}
```

En este último código es importante mencionar el uso de los selectores `:hover, :active y :focus`, los cuales se utilizan para modificar una regla CSS durante los eventos: pasar el mouse sobre un elemento (evento hover), hacer click en un elemento y mantener presionado el click (evento active), hacer click en un elemento y mantener el foco en el mismo (evento focus). De esta forma, podemos atender con CSS parte de los eventos que en años anteriores hubiésemos tenido que hacer obligatoriamente con Javascript.

Soporte de los navegadores

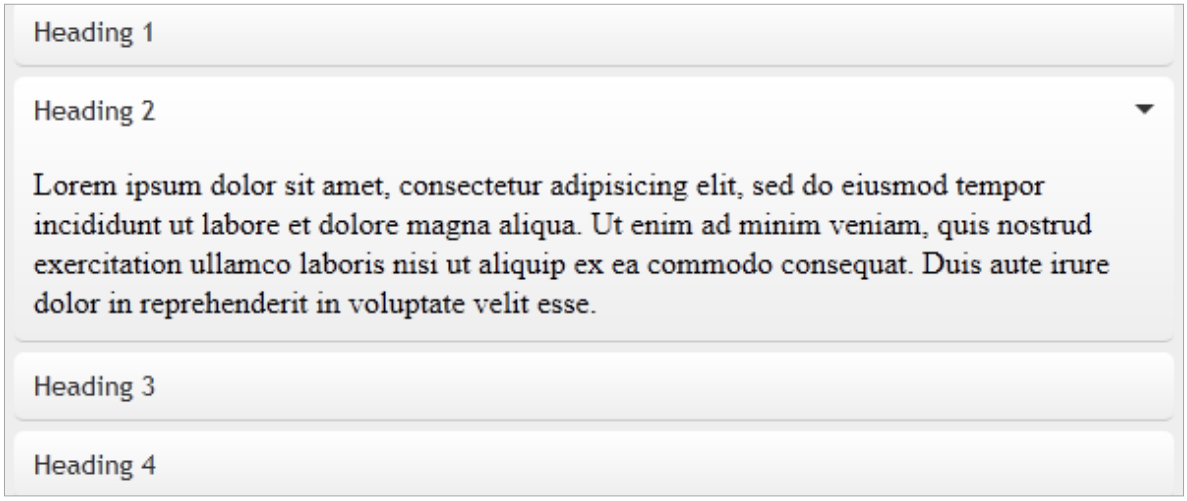
El código de esta práctica solo funcionará completamente en la mayoría de los navegadores modernos de CSS3 (últimas versiones de Opera, Mozilla Firefox, Google Chrome e Internet Explorer). Sin embargo, no funcionará de forma completa en el navegador Internet Explorer, en sus versiones 8 e inferiores. Es importante utilizar los navegadores modernos, que nos ofrecen los últimos avances en el mundo de la web, y mantenerlos actualizados. Sino actualizamos nuestros navegadores, eventualmente empezarán a fallar.

Practica 4: Crear un Acordeón con CSS3

En esta práctica aprenderás cómo crear un acordeón animado con instrucciones sencillas de CSS3.

Pero primero que todo, ¿qué es un acordeón en el mundo de la web? En términos simples, y al igual que en la vida real, es un elemento organizador que permite empaquetar muchos contenidos en un espacio reducido.

De esta forma, al finalizar está práctica tendremos el siguiente resultado:



Procedamos entonces a crear el esqueleto del acordeón con el siguiente código HTML:

acordeon.html

```
<!DOCTYPE html>
<html>
<head>
  <title>Crear un acordeon con CSS3 </title>
  <link type="text/css" rel="stylesheet" href="estilos.css">
  <!--[if lt IE 9]>
       <script
src="http://html5shim.googlecode.com/svn/trunk/html5.js"></script>
  <![endif]-->
</head>

<body>
<div class="accordion">
             <section id="one">
                    <h2><a href="#one">Heading 1</a></h2>

                    <div>
```

```
                        <p>Lorem ipsum dolor sit amet, consectetur adipisicing
elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim
ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea
commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit
esse.</p>
                  </div>
            </section>
            <section id="two">
                  <h2><a href="#two">Heading 2</a></h2>
                  <div>
                        <p>Lorem ipsum dolor sit amet, consectetur adipisicing
elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim
ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea
commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit
esse.</p>

                  </div>
            </section>
            <section id="three">
                  <h2><a href="#three">Heading 3</a></h2>
                  <div>
                        <p>Lorem ipsum dolor sit amet, consectetur adipisicing
elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim
ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea
commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit
esse.</p>
                  </div>
            </section>
            <section id="four">
                  <h2><a href="#four">Heading 4</a></h2>
                  <div>
                        <p>Lorem ipsum dolor sit amet, consectetur adipisicing
elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim
ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea
commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit
esse.</p>
                  </div>
            </section>
            <section id="five">
                  <h2><a href="#five">Heading 5</a></h2>
                  <div>
                        <p>Lorem ipsum dolor sit amet, consectetur adipisicing
elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim
ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea
commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit
esse.</p>
                  </div>
            </section>
      </div>
</body>
</html>
```

Observemos cómo está definido el acordeón: de acuerdo al código anterior, el acordeón es un contenedor global (el `<div>` con la clase "`accordion`") que en su interior está compuesto por una serie de paneles (representados por los distintas etiquetas `<section>`) cada una con su título y contenido.

Ahora diseñamos el estilo visual que nos proveerá la "magia" del CSS3 al acordeón. Lo colocamos en el siguiente archivo:

estilos.css

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

section
{
	display: block;
}

.accordion
{
	background-color: #eee;
	border: 1px solid #ccc;
	width: 600px;
	padding: 10px;
	margin: 50px auto;

	 /* Bordes redondeados */
	-moz-border-radius: 3px;
	-webkit-border-radius: 3px;
	border-radius: 3px;

	 /* Sombra del contenedor */
	-moz-box-shadow: 0 1px 0 #999;
	-webkit-box-shadow: 0 1px 0 #999;
	box-shadow: 0 1px 0 #999;
}

.accordion section
{
	border-bottom: 1px solid #ccc;
	margin: 5px;

	/* Fondo en degradado de colores */
	background-color: #fff;
	background-image: -webkit-gradient(linear, left top, left bottom,
from(#fff), to(#eee));
	background-image: -webkit-linear-gradient(top, #fff, #eee);
	background-image:    -moz-linear-gradient(top, #fff, #eee);
	background-image:     -ms-linear-gradient(top, #fff, #eee);
```

```
	background-image:	-o-linear-gradient(top, #fff, #eee);
	background-image:	   linear-gradient(top, #fff, #eee);

	/* Bordes redondeados */
	-moz-border-radius: 5px;
	-webkit-border-radius: 5px;
	border-radius: 5px;
}

.accordion h2,
 .accordion p
{
	margin: 0;

}


.accordion p
{
	padding: 10px;
}

.accordion h2 a
{
	display: block;
	position: relative;
	font: 14px/1 'Trebuchet MS', 'Lucida Sans';
	padding: 10px;
	color: #333;
	text-decoration: none;
	-moz-border-radius: 5px;
	-webkit-border-radius: 5px;
	border-radius: 5px;
}

.accordion h2 a:hover
{
	background: #fff;
}

.accordion h2 + div
{
	height: 0;
	overflow: hidden;

	 /* Transiciones */
	-moz-transition: height 0.3s ease-in-out;
	-webkit-transition: height 0.3s ease-in-out;
	-o-transition: height 0.3s ease-in-out;
	transition: height 0.3s ease-in-out;
}

.accordion :target h2 a:after
{
```

```
content: '';
position: absolute;
	right: 10px;
	top: 50%;
	margin-top: -3px;
	border-top: 5px solid #333;
	border-left: 5px solid transparent;
	border-right: 5px solid transparent;
}

.accordion :target h2 + div
{
	height: 100px;
}
```

Como ya vimos en las prácticas anteriores, aprovechamos las nuevas propiedades de CSS3 tales como `box-shadow`, `border-radius` y `linear-gradient` para dar al acordeón una estética agradable a la vista. Sin embargo, el efecto de expansión y contracción del acordeón se debe gracias a la propiedad `transition` (y sus versiones para cada navegador). En el CSS de la práctica se definió de la siguiente manera:

```
transition: height 0.3s ease-in-out;
```

En este caso se hace una transición sobre la altura (`height`) del panel al cual se le hace click para expandirlo o contraerlo, con una duración de 0.3 segundos (`0.3 s`) y con un efecto de entrada-salida (`ease-in-out`). Existen muchas otras transiciones de CSS3, te invitó a conocerlas, un rápido vistazo a Google te dará increíbles recursos sobre las mismas.

Soporte de los navegadores

El código de esta práctica solo funcionará en aquellos navegadores que soporten la pseudo-clase `:target` de CSS3, la cual es soportada por la mayoría de los navegadores modernos (últimas versiones de Opera, Mozilla Firefox, Google Chrome e Internet Explorer).

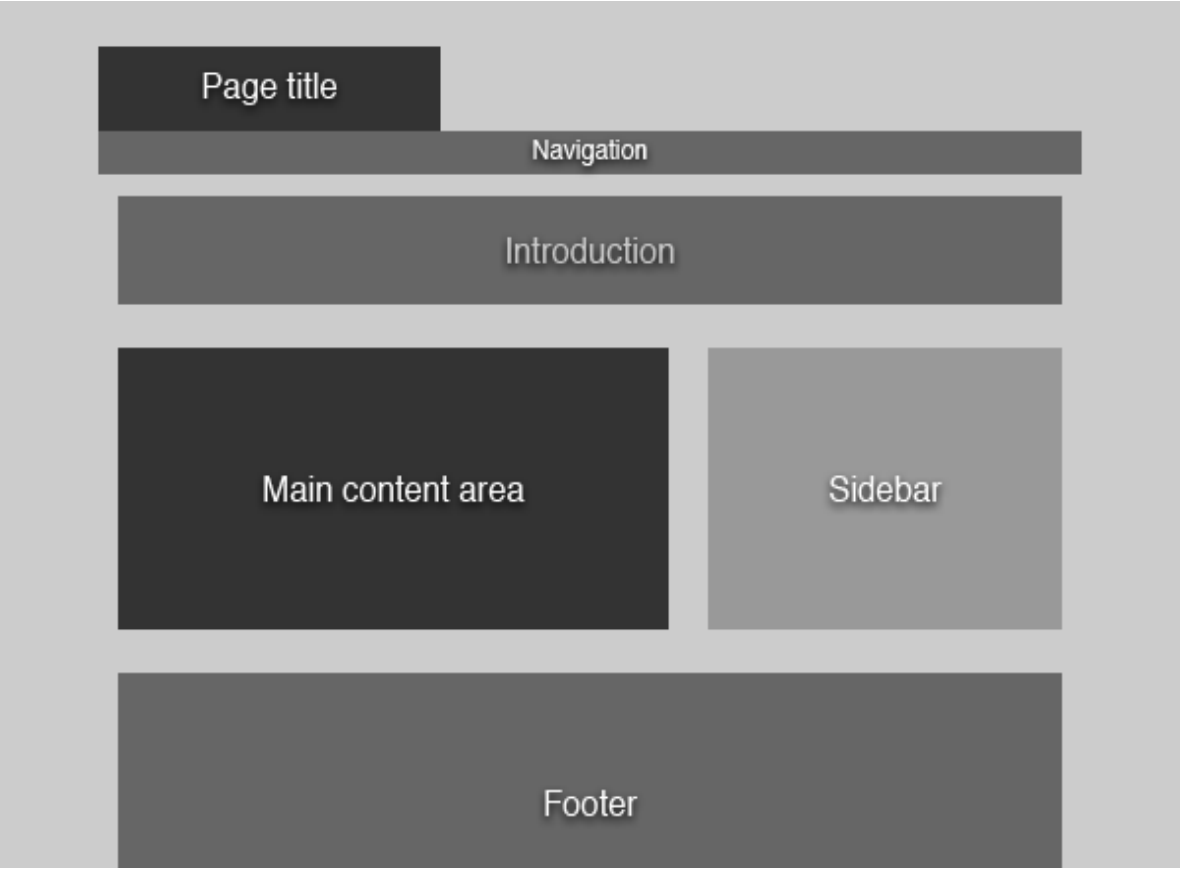
Sección de Proyecto

En esta sección vamos a crear una página tipo blog con HTML5 y CSS3. Paso a paso iremos construyendo el cuerpo de la página con el estilo correspondiente.

El diseño de este blog está basado en los diseños actuales: Cabecera con título, navegación horizontal, área de contenido con comentarios y formulario, sidebar y pie de página.

De esta forma comenzamos la definición de nuestro blog:

Estructura Básica:



Creamos el archivo **blog.html** el cual contendrá las etiquetas que nos permitirán delimitar los espacios correspondientes a cada tipo de contenido.

Es importante recordar que debemos crear secciones plenamente identificadas y delimitadas para que tanto como los usuarios y el programador puedan visualizarlas mejor.

Comenzamos entonces con el código de la estructura para nuestro blog.

Paso 1:

blog.html

```
<!doctype html>
<html>
<head>
  <title>Page title</title>
</head>
<body>
   <header>
      <h1>Page title</h1>
   </header>
   <nav>
      <!– Navigation –>
   </nav>
   <section id=”intro”>
      <!– Introduction –>
   </section>
   <section>
      <!– Main content area –>
   </section>
   <aside>
      <!– Sidebar –>
   </aside>
   <footer>
      <!– Footer –>
   </footer>
</body>
</html>
```

En lugar de utilizar divs para contener las diferentes secciones de la página, ahora estamos utilizando las etiquetas semánticas apropiadas.

Paso 2:

Delimitando la navegación

La navegación es delimitada de la misma forma en que lo haríamos en HTML 4 o XHTML, utilizando una lista desordenada. La clave es que esta lista se ubica dentro de las etiquetas nav.

```
<nav>
   <ul>
      <li>
         <a href=”#”>Blog</a>
      </li>
      <li>
```

```
         <a href=”#”>About</a>
      </li>
      <li>
         <a href=”#”>Archives</a>
      </li>
      <li>
         <a href=”#”>Contact</a>
      </li>
      <li class=”subscribe”>
         <a href=”#”>Subscribe via. RSS</a>
      </li>
   </ul>
</nav>
```

Paso 3:

Delimitando la introducción

Ya hemos definido una nueva sección en el documento mediante la etiqueta section. Ahora sólo necesitamos algo de contenido.

```
<section id=”intro”>
   <header>
      <h2>Do you love flowers as much as we do?</h2>
   </header>

   <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod
tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam,
quis nostrud exercitation ullamco laboris nisi ut.</p>

</section>
```

Añadimos un id a la etiqueta section para que la podamos identificar más tarde cuando coloquemos los estilos. Utilizamos la etiqueta header para envolver todo lo que se encuentra alrededor del elemento introductorio h2. Además de describir un documento entero, la etiqueta header también debería usarse para describir las secciones individuales.

Paso 4:

Delimitando el área de contenido principal

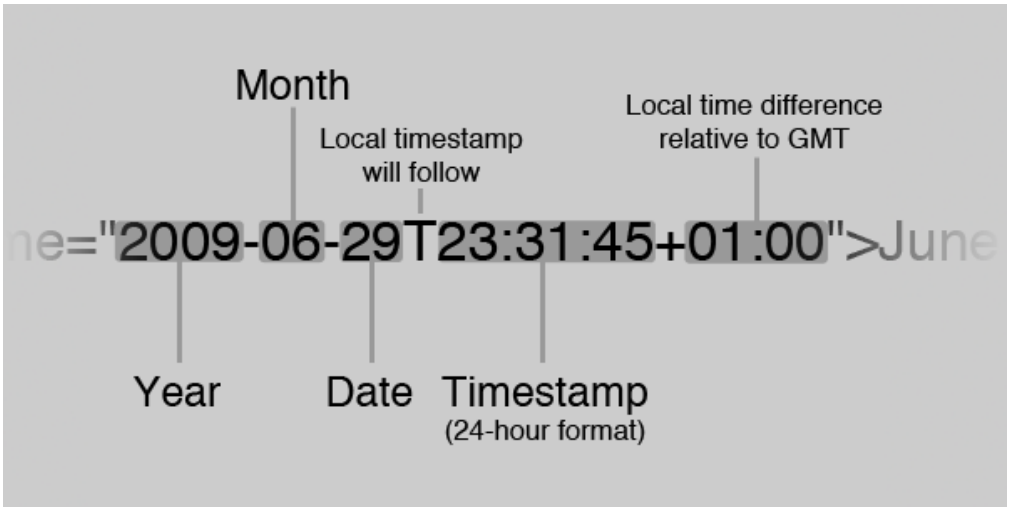
Nuestra área de contenido principal consiste de tres secciones: los posts del blog, los comentarios y el formulario de comentarios. Utilizando nuestros conocimientos sobre las nuevas etiquetas estructurales de HTML 5, debería ser fácil escribir el código.

4.1 Delimitando el post del Blog

Comenzamos una nueva sección y envolvemos todo el post del blog en una etiqueta article. La etiqueta article se usa para denotar una entrada independiente en un blog, discusión, enciclopedia, etc. y es ideal para usarla aquí. Dado que estamos viendo los detalles de un sólo post, sólo tenemos un artículo, pero en la página principal del blog envolveremos cada post en una etiqueta article.

```
<section>
   <article class=”blogPost”>
      <header>
         <h2>This is the title of a blog post</h2>

         <p>Posted on <time datetime=”2009-06-29T23:31:45+01:00″>June 29th
2009</time> by <a href=”#”>Mads Kjaer</a> - <a href=”#comments”>3
comments</a></p>
       </header>

       <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Proin euismod
tellus eu orci imperdiet nec rutrum lacus blandit. Cras enim nibh, sodales
ultricies elementum vel, fermentum id tellus. Proin metus odio, ultricies eu
pharetra dictum, laoreet id odio…</p>

    </article>
</section>
```

El elemento header es utilizado para presentar la cabecera y la información meta sobre el post. Le informamos a los usuarios cuándo fue escrito el post, quién lo escribió y cuantos comentarios tiene. La estampilla de tiempo está dentro de una etiqueta. Ésta también es nueva en HTML 5 y se usa para señalar un momento específico de tiempo. Los contenidos del atributo datetime deberían ser:



1. El año seguido por un guión del medio (-)

2. El mes seguido por un guión del medio (-)

3. La fecha

4. Una T mayúscula que denota que especificaremos el tiempo local

5. El tiempo local en el formato: hh:mm:ss

6. La zona horaria relativa al GMT.

4.2 Delimitando los comentarios

Delimitar los comentarios es bastante directo. No se necesita usar etiquetas ni atributos.

```
<section id=”comments”>
   <header>
      <h3>Comments</h3>
   </header>
   <article>
      <header>
         <a href=”#”>Usuario1 Nombre</a> on <time datetime=”2009-06-
29T23:35:20+01:00″>June 29th 2009 at 23:35</time>
      </header>

      <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do
eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim
veniam, quis nostrud exercitation ullamco laboris nisi ut.</p>

   </article>
   <article>
      <header>
         <a href=”#”>Usuario2 Nombre</a> on <time datetime=”2009-06-
29T23:40:09+01:00″>June 29th 2009 at 23:40</time>
      </header>

      <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do
eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim
veniam, quis nostrud exercitation ullamco laboris nisi ut.</p>

   </article>
</section>
```

4.3 Delimitando el formulario de “ingresa tu comentario”

En HTML 5 se han introducido varias mejoras en lo que respecta a formularios. Ya no tenemos que hacer la validación del lado del cliente de los campos, mails requeridos, etc, sino que el navegador se encarga de eso por nosotros.

```
<form action=”#” method=”post”>
   <h3>Post a comment</h3>

   <p>
      <label for=”name”>Name</label>
      <input name=”name” id=”name” type=”text” required />
   </p>

   <p>
      <label for=”email”>E-mail</label>
      <input name=”email” id=”email” type=”email” required />
   </p>

   <p>
      <label for=”website”>Website</label>
      <input name=”website” id=”website” type=”url” />
   </p>

   <p>
      <label for=”comment”>Comment</label>
      <textarea name=”comment” id=”comment” required></textarea>
   </p>

   <p>
      <input type=”submit” value=”Post comment” />
   </p>
</form>
```

Ahora hay dos tipos nuevos de entradas: e-mail y URL. E-mail especifíca que el usuario debería ingresar una dirección de correo electrónica válida y, URL que el usuario debería ingresar una dirección de un sitio web válida. Si escribes required como atributo, el usuario no puede enviar un campo incompleto. “Required” es un atributo booleano, nuevo para HTML 5. Esto quiere decir que el atributo no se puede declarar sin ningún valor.

Paso 5: Delimitando la Sidebar y el pie de página

El código de la sidebar y el pie de página es extremadamente simple. Algunas secciones con algo de contenido dentro, el aside apropiado y las etiquetas footer.

Por los momentos no hemos definido ningún estilo para nuestro blog lo cual se convierte en nuestro objetivo a partir de ahora.

Creamos el archivo **estilo.css**

Paso 1: Programación básica

Para comenzar vamos a definir algunas reglas básicas en lo que concierne a la tipografía, el color de fondo de la página, etc. Reconocerás todo esto de CSS 2.1

```
/* Reseteamos los estilos de margin y padding */
{
  margin: 0;
  padding: 0;
}

/* Le dice al navegador que dibuje los elementos de HTML 5 como block */
header, footer, aside, nav, article {
   display: block;
}

body {
   margin: 0 auto;
   width: 940px;
   font: 13px/22px Helvetica, Arial, sans-serif;
   background: #f0f0f0;
}

h2 {
   font-size: 28px;
   line-height: 44px;
   padding: 22px 0;
}

h3 {
   font-size: 18px;
   line-height: 22px;
   padding: 11px 0;
}

p {
  padding-bottom: 22px;
}
```

Primero, reseteamos los estilos de margen y padding con una regla simple. Con esto bastará.

Luego le decimos al navegador que dibuje todos los nuevos elementos HTML 5 como un bloque. Los navegadores no tienen problemas con los elementos que no reconocen, pero no saben cómo deberían dibujar esos elementos por defecto. Debemos decirles esto hasta que el estándar sea implementado en todos lados.

Hasta que los navegadores implementen por completo HTML 5 y CSS 3 necesitaremos definir los valores en unidades relativas, por lo que es conveniente seleccionar el tamaño de la fuente en píxeles y no en ems ó %.

Paso 2: Dándole estilo a la navegación

Es importante notar que el ancho del cuerpo ha sido definido como 940px y que fue centrado. Nuestra barra de navegación necesita hacer span en todo el ancho de la ventana, por lo que tendremos que aplicar algunos estilos adicionales:

```
nav {
   position: absolute;
   left: 0;
   width: 100%;
   background: url(”nav_background”);
}
```

Posicionamos el elemento nav de forma absoluta, lo alineamos a la izquierda de la ventana y hacemos que haga span en todo el ancho. Centraremos la lista anidada para mostrarla dentro de los límites del diseño:

```
nav ul {
   margin: 0 auto;
   width: 940px;
   list-style: none;
}
```

Ahora definiremos algunos estilos adicionales para hacer que los ítems de navegación luzcan más mejores y los alinearemos dependiendo de nuestro diseño.

```
nav ul li {
   float: left;
}

nav ul li a {
   display: block;
   margin-right: 20px;
   width: 140px;
   font-size: 14px;
   line-height: 44px;
   text-align: center;
   text-decoration: none;
   color: #777;
}

nav ul li a:hover {
   color: #fff;
}

nav ul li.selected a {
   color: #fff;
}
```

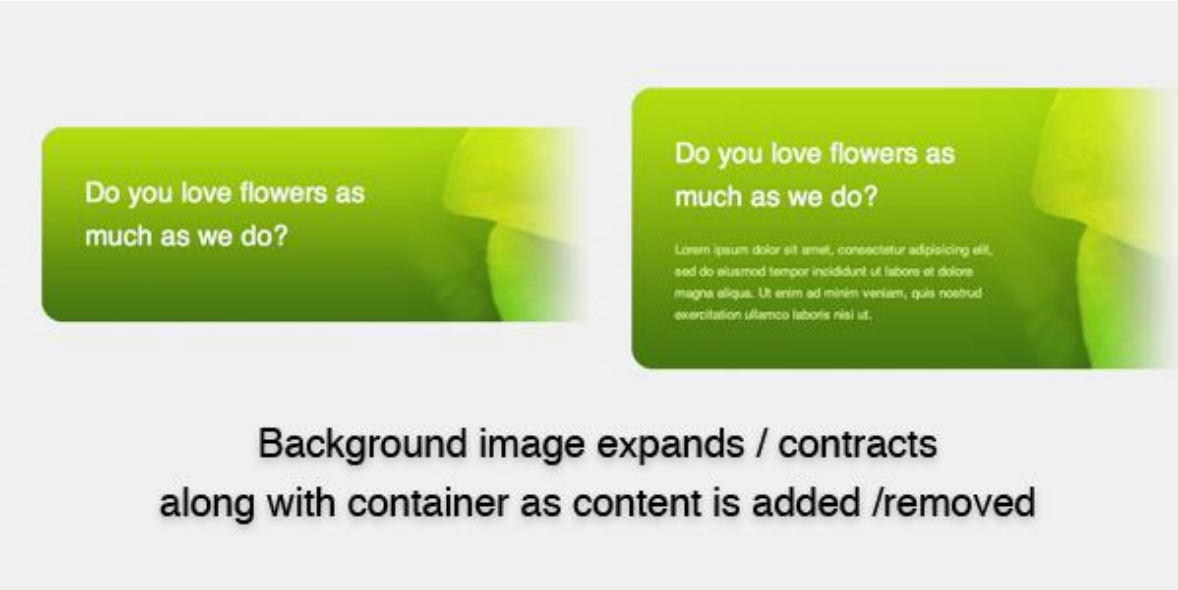
```
nav ul li.subscribe a {
   margin-left: 22px;
   padding-left: 33px;
   text-align: left;
   background: url(”rss.png”) left center no-repeat;
}
```

Paso 3: Dándole estilo a la introducción

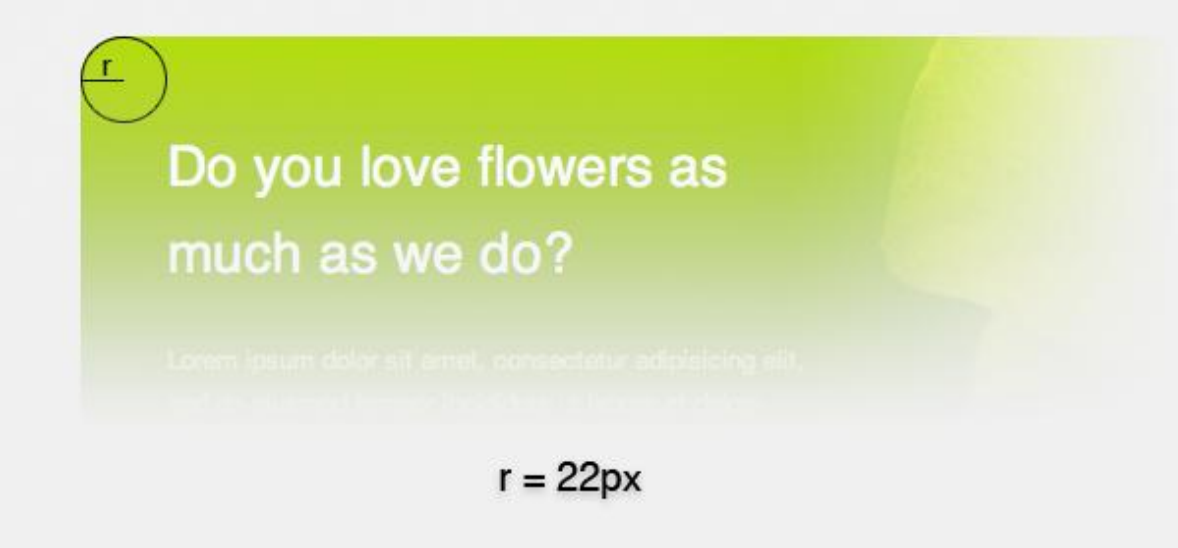
El código de la introducción es bastante simple: Una sección con un titular y un párrafo de texto. Sin embargo, usaremos algo de CSS 3 para hacerlo lucir más atractivo.

```
#intro {
   margin-top: 66px;
   padding: 44px;
   background: #467612 url(”intro_background.png”) repeat-x;
   background-size: 100%;
   border-radius: 22px;
}
```

Estamos utilizando dos propiedades nuevas. La primera es background-size, que nos permite escalar la imagen de fondo. En nuestro caso, la escalamos al 100% en ambos ejes. Esto es algo que no se podía hacer en CSS 2.1 sin código no-semántico y sin tener varios problemas con el navegador.



La segunda propiedad nueva es border-radius, que aplica esquinas redondeadas al elemento. Puedes especificar valores diferentes para cada esquina o seleccionar sólo algunas esquinas redondeadas.



Desafortunadamente, ninguna de las propiedades son implementadas por completo en los navegadores. Pero a pesar de esto, podemos obtener algo de soporte utilizando atributos vendor-specific. Background-size es soportado por las nuevas versiones de Safari, Opera y Konqueror. Y border-radius es soportado por las nuevas versiones de Safari y Firefox.

```
#intro {
   …
   /* Background-size not implemented yet */
   -webkit-background-size: 100%;
   -o-background-size: 100%;
   -khtml-background-size: 100%;
   /* Border-radius not implemented yet */
   -moz-border-radius: 22px;
   -webkit-border-radius: 22px;
}
```

Dado que tenemos un color de fondo definido, no habrá problemas mayores en los navegadores que no soporten la función background-size, como Firefox por ejemplo. Ahora sólo debemos darle estilo al titular y al texto.

```
#intro h2, #intro p {
   width: 336px;
}
#intro h2 {
   padding: 0 0 22px 0;
   font-weight: normal;
   color: #fff;
}
#intro p {
   padding: 0;
   color: #d9f499;
}
```

La imagen de la flor puede añadirse fácilmente dándole a #intro una segunda imagen de fondo, algo que CSS 3 nos permite realizar.

```
#intro {
 …
 background: #467612 url(”intro_background.png”) top left (287px 100%) repeat-x,
url(”intro_flower.png”) top right (653px 100%) no-repeat;
 …
}
```

Le damos a las dos imágenes de fondo dimensiones explícitas para asegurarnos que no se sobrepongan, y listo.

Paso 4: Dándole estilo al área de contenido y a la sidebar

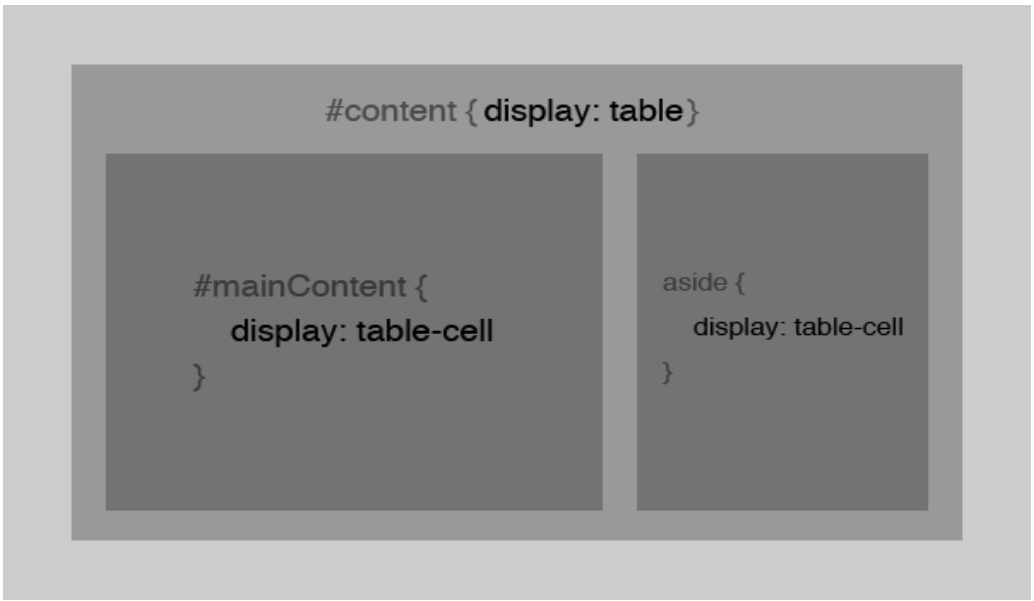
El área de contenido y la sidebar serán alineadas una al lado de la otra. Tradicionalmente, haríamos esto mediante floats, pero en CSS lo haremos con tablas.

En CSS 3 podemos hacer que los elementos se comporten como tablas sin que esto se note en el código. Para comenzar, necesitaremos algunos divs para agrupar las secciones de una forma un poco más lógica.

No devolvemos a nuestro archivo **blog.html**, para agregar los divs esto lo hacemos ubicándonos sobre el tag `<section>` de la siguiente manera:

```
<div id=”content”>
   <div id=”mainContent”>
      <section>
         <!– Blog post –>
      </section>

      <section id=”comments”>
         <!– Comments –>
      </section>

      <form>
         <!– Comment form –>
      </form>
   </div>

   <aside>
      <!– Sidebar –>
   </aside>
</div>
```

Todo sigue teniendo sentido semánticamente, pero ahora puedes darle estilo. Queremos que el div #content se comporte como una tabla, con #mainContent y aside como celdas de la tabla. Mediante CSS 3, esto es sencillo:



```
#content {
   display: table;
}

#mainContent {
   display: table-cell;
   width: 620px;
   padding-right: 22px;
}

aside {
   display: table-cell;
   width: 300px;
}
```

Paso 5: Dándole estilo al post del blog

Columnas múltiples

Las columnas múltiples de texto antes eran imposibles a menos que se separara el texto de forma manual, pero con CSS 3 es muy fácil. Aunque tendremos que añadir un div alrededor de los párrafos múltiples para que funcione en los navegadores actuales.

De nuevo nos devolvemos al archivo **blog.html**.

```
<div>
   <p>Lorem ipsum dolor sit amet…</p>
   <p>Pellentesque ut sapien arcu…</p>
   <p>Vivamus vitae nulla dolor…</p>
   …
</div>
```

Ahora podemos agregar dos propiedades simples en nuestra hoja de estilos:

```
.blogPost div {
   column-count: 2;
   column-gap: 22px;
}
```

Deseamos dos columnas y un espacio de 22px entre ellas. El div adicional se necesita porque actualmente no hay una forma soportada de hacer que un elemento tenga un span mayor que una columna. En el futuro, sin embargo, podremos especificar la propiedad span de la columna, y podremos escribir sólo:

```
.blogPost {
   column-count: 2;
```

```
   column-gap: 22px;
}

.blogPost header {
   column-span: all;
}
```

Por supuesto las propiedades **column-count** y **column-gap** son soportados sólo por algunos navegadores, **Safari and Firefox**. Así que por ahora debemos utilizar la propiedad **vendor-specific**.

```
.blogPost div {
   /* Column-count not implemented yet */
   -moz-column-count: 2;
   -webkit-column-count: 2;
   /* Column-gap not implemented yet */
   -moz-column-gap: 22px;
   -webkit-column-gap: 22px;
}
```

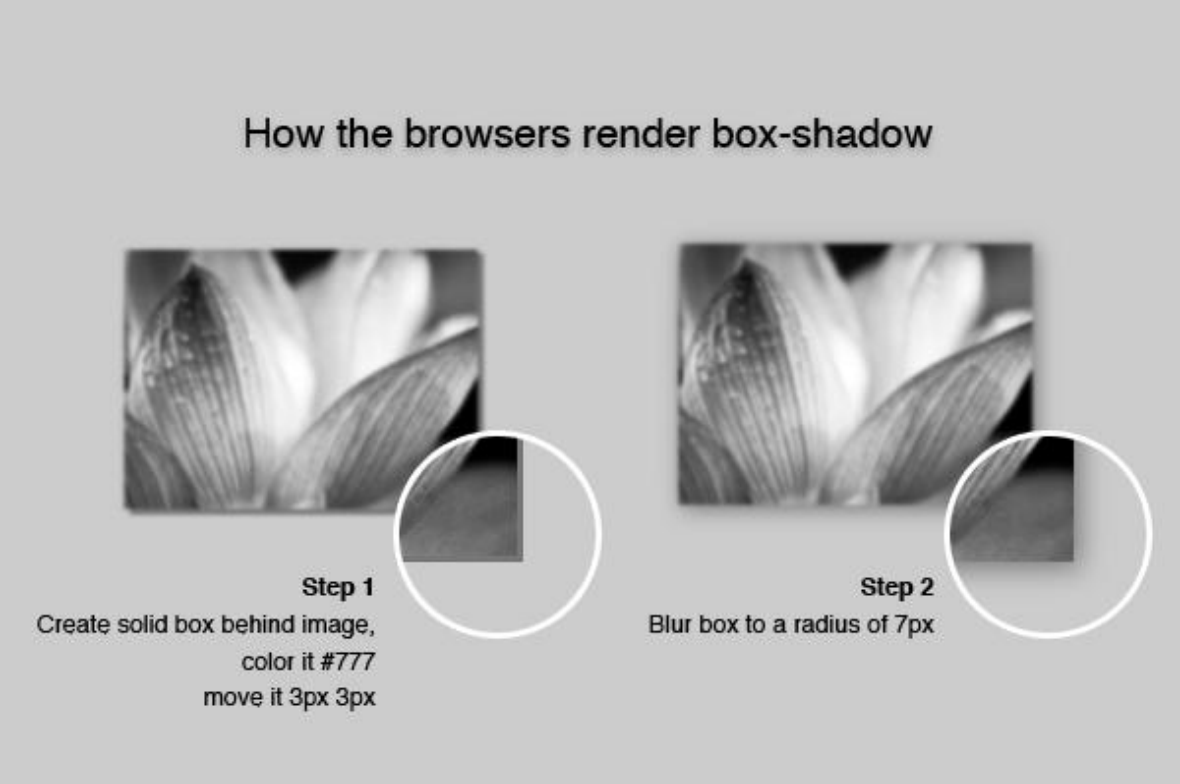
Paso 6: Sombra de caja (box-shadow)

Si miras con atención la imagen del post notarás una drop-shadow. Podemos generar esto mediante CSS 3 y la propiedad box-shadow.

```
.blogPost img {
   margin: 22px 0;
   box-shadow: 3px 3px 7px #777;
}
```

Los primeros "3px" le indican al navegador dónde queremos que la sombra pare horizontalmente. Los segundos "3px" le dicen dónde queremos que pare verticalmente. Los últimos "7px" indican cuan borroso debería ser el borde. Si lo programas en "0" será completamente sólido. Por último, definimos el color base de la sombra. El color se verá desvanecido por supuesto, dependiendo de cuanta borrosidad y sombra hayamos seleccionado.

No resulta sorprendente que esta propiedad todavía no haya sido implementada en todos los navegadores. De hecho, sólo funciona en Firefox 3 y Safari (aunque quizás también lo haga en Chrome, dado que ambos usan el motor Webkit). En ambos casos deberemos usar la propiedad vendor-specific.



Para Safari y Chrome:

```
.blogPost img {
   margin: 22px 0;
   -webkit-box-shadow: 3px 3px 7px #777;
}
```

Para Firefox 3:

```
.blogPost img {
   margin: 22px 0;
   -moz-box-shadow: 3px 3px 7px #777;
}
```

Paso 7: Realizar Zebra-striping en los comentarios

Zebra-striping, o resaltar cada segundo elemento de una serie (como las rayas intercaladas de una cebra), ha involucrado tradicionalmente la selección de todos los elementos por medio de javascript, luego el loopeo a través de ellos y finalmente el resaltado de todos los elementos impares. CSS 3 introduce la pseudo-clase “nth-child”, que nos permite llevar a

cabo esta función de forma increíblemente fácil sin necesidad de javascript. Lo usaremos para hacer zebra-stripe sobre los comentarios.

```
section#comments article:nth-child(2n+1) {
   padding: 21px;
   background: #E3E3E3;
   border: 1px solid #d7d7d7;
   /* Border-radius not implemented yet */
   -moz-border-radius: 11px;
   -webkit-border-radius: 11px;
}
```

EL valor raro "2n+1″ es en verdad bastante simple si entiendes lo que quiere decir:

- 2n selecciona cada segundo ítem. Si escribes 3n seleccionará cada tercer ítem, 4n cada cuarto ítem, y así sucesivamente.
- El +1 le dice al navegador que empiece en el elemento 1. En la programación todas las series comienzan en 0, por lo que el elemento 1 es en verdad el segundo de la serie.

Alternativamente, podrías sólo escribir:

```
section#comments article:nth-child(odd) {
    …
}
```

Dado que el estándar incluye los dos valores más usados como taquigrafía, par e impar. El resto del estilado de comentarios debería ser fácil de comprender con nuestros nuevos conocimientos.

Paso 8: Dándole estilo al formulario de comentario, el pie de página y la Sidebar

Deberemos reutilizar algunas de las técnicas de CSS3 aprendidas para dar estilo al formulario de "nuevo comentario", al footer y a la sidebar. En el formulario de comentario y el footer, usaremos el mismo tipo de estilo de tabla que en el diseño principal. En la sidebar, en cambio, utilizaremos border-radius para agregar esquinas redondeadas a las diferentes secciones.

Comenzamos entonces:

Para el formulario de "**nuevo comentario**" solicitamos el nombre de la persona, su email, su página web y el comentario que va a postear. El código HTML para el formulario es el siguiente:

```
<form action="#" method="post">
   <h3>Post a comment</h3>

   <p>
      <label for="name">Name</label>
      <input name="name" id="name" type="text" required />
   </p>

   <p>
      <label for="email">E-mail</label>
      <input name="email" id="email" type="email" required />
   </p>

   <p>
      <label for="website">Website</label>
      <input name="website" id="website" type="url" />
   </p>

   <p>
      <label for="comment">Comment</label>
      <textarea name="comment" id="comment" required></textarea>
   </p>

   <p>
       <input type="submit" value="Post comment" />
   </p>
</form>
```

Para dar estilo el formulario, le colocamos un estilo basado principalmente en colores y bordes grises que le dan al mismo un toque de elegancia. Para ello, agregamos al archivo **estilo.css** el siguiente código:

```
form {
   margin-top: 21px;
   padding-top: 22px;
   border-top: 1px solid #d7d7d7;
}

form p {
   display: table;
   margin-bottom: 22px;
   padding: 0 22px;
}

form label {
```

```
   display: table-cell;
   width: 140px;
   padding-right: 20px;
   text-align: right;
   font-weight: bold;
   vertical-align: top;
}

form input[type="text"], form input[type="email"], form input[type="url"] {
   display: table-cell;
   width: 300px;
   height: 20px;
   border: 1px solid #d7d7d7;
}

form textarea {
   width: 300px;
   height: 100px;
   border: 1px solid #d7d7d7;
}

form input[type="submit"] {
   margin-left: 162px;
}
```

En el código anterior utilizamos un nuevo selector de CSS 3 para acceder a los atributos de un elemento. Por ejemplo la regla: `form input[type="submit"]` está aplicándose a todos los elementos `input` que se encuentran dentro del elemento `form` y que contienen a su vez el atributo `type` con el valor `submit`.

Seguidamente construyamos el **footer** o **pie de página,** el cual es una sección de la página que podemos utilizar para diferentes cosas: dar información acerca de tu empresa; colocar el mapa de navegación de la página; colocar links a páginas web amigas; colocar links de los artículos más populares, etc. Para nuestro caso, utilizamos el footer para colocar información acerca del blog, el listado de las páginas amigas y el listado de links a los artículos de mayor popularidad. Cada información la colocamos en una sección distinta de la otra, utilizando la etiqueta `<section>` de HTML5 de la siguiente manera (esto código lo agregamos en el lugar apropiado para el footer en el archivo **blog.html**:

```
<footer>
   <div>
      <section id="about">
         <header>
            <h3>About</h3>
         </header>

         <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do
eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim
veniam, quis nostrud exercitation ullamco <a href="#">laboris nisi ut
aliquip</a> ex ea commodo consequat. Duis aute irure dolor in reprehenderit in
```

```
voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint
occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim
id est laborum.</p>

        </section>

        <section id="blogroll">
            <header>
               <h3>Blogroll</h3>
            </header>

            <ul>
               <li><a href="#">NETTUTS+</a></li>
               <li><a href="#">FreelanceSwitch</a></li>
               <li><a href="#">In The Woods</a></li>
               <li><a href="#">Netsetter</a></li>
               <li><a href="#">PSDTUTS+</a></li>
            </ul>
         </section>

         <section id="popular">
            <header>
               <h3>Popular</h3>
            </header>

            <ul>
               <li><a href="#">This is the title of a blog post</a></li>
               <li><a href="#">Lorem ipsum dolor sit amet</a></li>
               <li><a href="#">Consectetur adipisicing elit, sed do
eiusmod</a></li>
               <li><a href="#">Duis aute irure dolor</a></li>
               <li><a href="#">Excepteur sint occaecat cupidatat</a></li>
               <li><a href="#">Reprehenderit in voluptate velit</a></li>
               <li><a href="#">Officia deserunt mollit anim id est
laborum</a></li>
               <li><a href="#">Lorem ipsum dolor sit amet</a></li>
            </ul>
         </section>
    </div>

</footer>
```

Ahora colocamos el CSS para darle una mejor apariencia a estas secciones que desarrollamos. El siguiente código lo colocamos en el archivo **estilo.css**:

```
footer {
   position: absolute;
   left: 0;
   width: 100%;
   background: #222;
}
```

```
footer div {
   display: table;
   margin: 0 auto;
   padding: 44px 0;
   width: 940px;
   color: #777;
}

footer div section {
   display: table-cell;
   width: 300px;
}

footer div #about, footer div #blogroll {
   padding-right: 20px;
}

footer h3 {
   color: #FFF;
}

footer a {
   color: #999;
}

footer a:hover {
   color: #FFF;
   text-decoration: none;
}

footer ul {
   margin: 0 0 0 40px;
   list-style: square;
   color: #565656;
}

footer ul li a {
   display: block;
}
```

Cabe resaltar una explicación especial de la propiedad `margin` dentro de la regla `footer div`. Cuando a la propiedad `margin` se le coloca el valor `0 auto` se indica que se quiere centrar el elemento al cual se le está aplicando; en este caso particular, se está centrando a todos los `div` dentro del elemento `footer`.

Finalmente, definimos el **panel lateral** (o sidebar) de la página, utilizando la etiqueta de HTML5 `<aside>` que se utiliza para colocar un contenido tangencial o secundario. Para el presente ejemplo, el panel lateral se compone de la sección de listado de categorías y la sección listado de artículos archivados, los cuales colocamos cada uno en etiquetas `<section>` distintas. El código HTML para definir el panel lateral es el siguiente:

```
<aside>
   <section>
      <header>
         <h3>Categories</h3>
      </header>

      <ul>
         <li><a href="#">Lorem ipsum dolor</a></li>
         <li><a href="#">Sit amet consectetur</a></li>
         <li><a href="#">Adipisicing elit sed</a></li>
         <li><a href="#">Do eiusmod tempor</a></li>
         <li><a href="#">Incididunt ut labore</a></li>
      </ul>

   </section>
   <section>
      <header>
         <h3>Archives</h3>
      </header>

      <ul>
         <li><a href="#">December 2008</a></li>
         <li><a href="#">January 2009</a></li>
         <li><a href="#">February 2009</a></li>
         <li><a href="#">March 2009</a></li>
         <li><a href="#">April 2009</a></li>
         <li><a href="#">May 2009</a></li>
         <li><a href="#">June 2009</a></li>
      </ul>
    </section>
</aside>
```

Para estilizar el sidebar utilizamos el siguiente CSS que agregamos en el archivo **estilo.css,** en el cual establecemos bordes redondeados para las 2 secciones del panel lateral, usando la propiedad `border-radius` (y sus versiones específicas para cada navegador) la cual indica el radio necesario para formar el borde redondeado.

```
aside section {
   margin: 22px 0 0 22px;
   padding: 11px 22px;
   background: url("images/sidebar_section_background.png") repeat-x;

   /* Border-radius not implemented yet */
   -moz-border-radius: 11px;
   -webkit-border-radius: 11px;
   border-radius: 11px;
}

aside section ul {
   margin: 0 0 0 22px;
   list-style: none;
}
```

```
aside section ul li a {
   display: block;
   text-decoration: none;
   color: #000;
}

aside section ul li a:hover {
   text-decoration: underline;
}
```

Una vez que HTML 5 y CSS 3 sean implementados en todos los navegadores será mil veces más sencillo construir un sitio web de lo que ya es. Finalmente podremos dejar de utilizar floats para los diseños y pasaremos un tiempo considerablemente menor escribiendo javascript para poder escalar nuestras imágenes de fondo o hacer zebra-stripe en nuestras tablas.

Referencias

- **Andrés Nieto**. *Las principales diferencias entre HTML5 y HTML4.* Publicado en: http://www.anieto2k.com/2007/06/16/las-principales-diferencias-entre-html5-y-html4/

- **El Webmaster**. HTML5 – ¿Qué novedades trae la nueva versión de este lenguaje? Publicado en: http://www.elwebmaster.com/actualidad/html-5-%C2%BFque-novedades-trae-la-nueva-version-de-este-lenguaje

- **Red Team**. *Slick Login Form with HTML5 + CSS3.* Publicado en: http://www.red-team-design.com/slick-login-form-with-html5-css3

- **W3 Schools**. *CSS3 Selectors.* Publicado en: http://www.w3schools.com/cssref/css_selectors.asp

- **El Webmaster.** *HTML5 y CSS3 – Técnincas y tips para adaptarte a lo nuevo*. Publicado en: http://www.elwebmaster.com/articulos/html5-y-css3-tecnicas-y-tips-para-adaptarte-a-lo-nuevo

- **Andrés Nieto**. *Selectores CSS que deberías conocer.* Publicado en: http://www.anieto2k.com/2006/09/06/selectores-css-que-deberias-conocer/

- **Red Team**. *Slick Login Form with HTML5 + CSS3.* Publicado en: http://net.tutsplus.com/tutorials/html-css-techniques/create-a-sticky-note-effect-in-5-easy-steps-with-css3-and-html5/

- **NetTuts+**. Create a sticky note effect in 5 easy steps with CSS3 and HTML5. Publicado en: http://net.tutsplus.com/tutorials/html-css-techniques/create-a-sticky-note-effect-in-5-easy-steps-with-css3-and-html5/