

# UD1\_HTML 5 y CSS Índice

<b>1.</b>	<b>XHTML.....</b>	<b>3</b>
1.1	Principios fundamentales del XHTML.....	3
1.1.1	Estructura de un documento XHTML.....	3
1.2	Diferencias entre HTML 4.0 y XHTML 1.0.....	5
1.3	Validación XHTML.....	6
<b>2.</b>	<b>HTML 5.....</b>	<b>7</b>
2.1	El largo camino de HTML4 a HTML5.....	7
2.2	La primera página HTML5.....	8
2.2.1	Eliminación de etiquetas y atributos no deseables.....	9
2.3	Organizar el contenido con etiquetas de bloques de contenido.....	9
2.4	Etiquetas que dotan de mayor semántica a la página.....	11
2.5	Insertar audio y vídeo.....	12
2.6	Formularios en HTML5.....	13
2.7	Almacenar datos en local gracias a Web Storage.....	16
2.8	Otras características de HTML5.....	16
2.9	Referencia de etiquetas HTML5.....	17
<b>3.</b>	<b>CSS3.....</b>	<b>20</b>
3.1	Sombreado para texto.....	21
3.2	Color en CSS3.....	21
3.3	Background en CSS3.....	22
3.3.1	Background con varias imágenes.....	23
3.4	Degradados.....	24
3.5	Bordes redondeados.....	25
3.6	Selectores de atributos en CSS3.....	26
3.7	Pseudo-clases con CSS3.....	27
3.8	Pseudo-elementos.....	30
3.9	Trabajar con columnas en CSS3.....	31
3.10	Declarar variables.....	32

# 1. HTML 5

---

## El largo camino de HTML4 a HTML5

Tim Berners-Lee desarrolló el lenguaje HTML en 1989. Este se hizo muy popular ya que el concepto de etiquetas para crear páginas web era muy sencillo y fácil de aprender.

Para evitar que una compañía en particular obligue a los usuarios a utilizar una tecnología en concreto se crea el World Wide Web consortium (W3C), un organismo de normalización abierto constituido por representantes de diferentes compañías: Apple, Microsoft, Adobe, Sun, Google, IBM, Oracle, etc. Este consorcio ha creado muchas de las tecnologías que se utilizan hoy en día como HTML, XML, protocolos para servicios Web (SOAP), y gráficos en formato PNG entre otros. Cada uno de estos estándares son propuestos, definidos, ratificados y publicados con consentimiento de todo el grupo. Uno de los primeros estándares que siguió todo este proceso fue el lenguaje HTML.

La primera versión de HTML es muy diferente a la que usamos hoy en día, ya que no tenía en cuenta el diseño ni incluía herramientas para dar formato al texto. Entre los años 1989 y 1997, el lenguaje HTML pasó por cuatro ratificaciones importantes. El estándar de HTML4 se publicó en 1997 y en él se proponía utilizar hojas de estilo en cascada nivel 1 (CSS1) para controlar el diseño de páginas, gráficos en formato PNG como estándar de mapa bits no sujetos a patentes, y normas DOM (Document Object Model, Modelo de objetos del documento) para que las aplicaciones JavaScript funcionaran coherentemente en todos los navegadores Web. También se introdujo la primera versión de XML para controlar la estructura de datos.

Los navegadores fueron adoptando este estándar, siendo Microsoft, con Internet Explorer 8 la que más tardó en respaldarlo (12 años). Los nuevos navegadores (Safari de Apple, Opera, Chrome de Google y Firefox de Mozilla), y los dispositivos de acceso a Internet (iPhone de Apple, Android de Google, Pre de Plan y Blackberry de RIM) saben que HTML5 será el próximo estándar de referencia, y por esa razón lo están adoptando.

HTML 5 se prevé esté listo como especificación de implementación recomendada en el 2012. ¿Quiere esto decir que vamos a tener que esperar hasta 2012 para aprovechar las ventajas de HTML 5? Esto no es así ya que hay partes de HTML5 que son:

- Fiables desde ya
- Apenas utilizables
- Por desgracia extrañas
- Interesantes pero ignorables
- Fundamentalmente para experimentar con aplicaciones Web.

Podemos decidir utilizar HTML5 o seguir haciendo las cosas como las hacíamos hasta ahora. Lo que debemos valorar es que se pueden introducir elementos de HTML 5 en una página conviviendo con elementos de versiones anteriores. Lo que sí sería recomendable es incorporar algunas de las etiquetas que ayudan a definir los elementos de la página, ya que en el futuro cada vez más aplicaciones y buscadores se valdrán de estas etiquetas; utilizarlas favorecerá su posicionamiento y el acceso a sus contenidos.

# La primera página HTML5

La primera línea HTML de cualquier página web sirve para identificar el tipo de lenguaje utilizado, y se le conoce como DOCTYPE (Document Type Declaration, Declaración del tipo de documento). El nuevo estándar sólo tiene un DOCTYPE:

```
<!DOCTYPE html>
```

con el cual informa al navegador que el código utilizado es HTML5. Esta declaración puede ir en mayúsculas o en minúsculas.

HTML5 simplifica la sintaxis con respecto a las versiones anteriores. Por ejemplo:

	en HTML4.01	en HTML5
<b>meta</b>	<code>&lt;meta HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=UTF-8"&gt;</code>	<code>&lt;meta charset=UTF-8&gt;</code>
<b>script</b>	<code>&lt;script type="text/javascript" src="jquery.js"&gt;&lt;/script&gt;</code>	<code>&lt;script src="jquery.js"&gt;&lt;/script&gt;</code>
<b>declarar una CSS</b>	<code>&lt;link rel="stylesheet" type="text/css" href="styles.css" /&gt;</code>	<code>&lt;link rel="stylesheet" href="styles.css" /&gt;</code>

Así, podemos crear fácilmente una página en HTML5 con referencia a una hoja de estilos externa:

```
<!DOCTYPE html>
<html>
  <head>
    <META charset=utf-8>
    <title>Empezando con HTML5</title>
    <link rel="stylesheet" href="mihojadeestilos.css">
  </head>
  <body>
    <p>Esto es HTML</p>
  </body>
</html>
```

Para comprobar algunas de las posibilidades que ofrece HTML5 puedes probar el divertido logo que creó Google en conmemoración a *Les Paul*. Está creado íntegramente utilizando estas tecnologías, algo que, hasta hace poco, habría sido impensable crear sin el uso de flash o java. Introduce el siguiente código en el body de tú página HTML5:

```
<center><iframe src="http://www.google.com/logos/2011/lespaul.html" width="600"
height="350" frameborder="0" scrolling="no"> </iframe></center>
```

Observa que dependiendo de la cuerda que rasgues sonará una nota u otra. Puedes pulsar el botón de grabación y se guardará la melodía que interpretes. Al detener la grabación, Google te proporciona un enlace corto que te permitirá compartir dicha grabación con los demás.

En eso se resume HTML 5: *permite la interacción del usuario y proporciona una experiencia muy similar a la que se obtendría con una aplicación de escritorio, pero sin necesidad de realizar instalaciones.*

## 1.1.1 Eliminación de etiquetas y atributos no deseables

Los atributos de formato de muchas etiquetas han sido eliminados. Deben definirse en los estilos CSS. Así, se pretende separar el formato del contenido definitivamente. Prácticas que hasta ahora eran desaconsejadas, como el atributo background de <body> o los atributos align y border, pasan a estar prohibidas. Algunos ejemplos más son los atributos cellpadding, cellspacing de las tablas, o las etiquetas <u>, <font>, <center> o <strike>.

Se han eliminado etiquetas problemáticas como <frame>. Para mostrar una página dentro de otra se deberá utilizar <iframe>. También se eliminan otras etiquetas que habían caído en desuso <applet> (se utiliza <embed>).

# Organizar el contenido con etiquetas de bloques de contenido

Uno de los objetivos de HTML5 es facilitar la búsqueda, organización e intercambio de millones y millones de páginas. Con este lenguaje se quiere que el contenido de cada página tenga sentido. HTML5 nos permitirá:

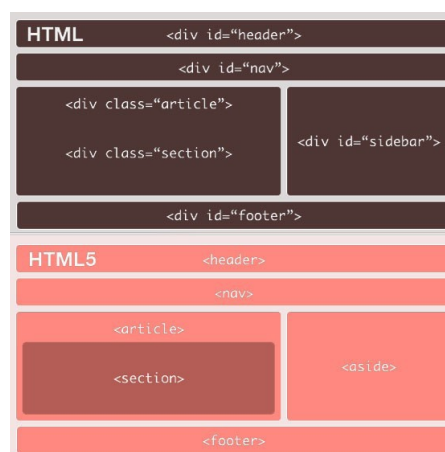
- Organizar el contenido general de una página en bloques.
- Crear una estructura de contenido a nivel de texto.

En HTML4 no existen muchas herramientas para definir el contenido de una página Web. Las más comunes son `P` para identificar un párrafo y `DIV` para una sección de contenido. No hay ninguna etiqueta que describa su contenido.

La mayoría de las webs tienen un formato común, formado por elementos como cabecera, pie, navegadores, etc. Tomando en cuenta la utilización que damos a algunos elementos `DIV` con identificadores muy comunes, los desarrolladores crearon *etiquetas estructurales que permiten definir las partes básicas de una página y estructurar su contenido*.

En la imagen siguiente podemos observar un layout típico que utiliza *divs* con atributos *id* y/o *class* para estructurarlos. Estos contienen un *header*, *footer* y una barra de navegación horizontal debajo del header. El contenido principal contiene un *article* y el *sidebar* se encuentra a la derecha.

HTML5 introduce nuevos elementos para representar cada una de estas secciones, con lo cual en el ejemplo de la imagen, los elementos *div* pueden ser reemplazados con los nuevos elementos: *header*, *nav*, *section*, *article*, *aside*, y *footer*.



## section

Identifica claramente un bloque de contenidos. Agrupa una parte de documento con contenido de temática similar, por ejemplo una introducción o un subapartado. En un blog, sería la zona donde están todos los posts. En un video de youtube, habría un *section* para el video, uno para los datos del video, otro para la zona de comentarios.

Una sección puede contener otras secciones.

## article

Define zonas únicas de contenido independiente. En el home de un blog, cada post sería un *article*. En un post del blog, el post y cada uno de sus comentarios sería un *article*.

### Cuándo utilizar *div*, *section* y *article*

A veces nos surgen dudas a la hora de diferenciar *div*, *section* y *article*. Básicamente, la diferencia está en la carga semántica que le queramos dar al contenido. ¿Cuándo usamos un *div*, un *section* o un *article*?

Usaremos *article* para representar cualquier información que tenga sentido propio por si misma y sea propensa a la reutilización, por ejemplo, mediante sindicación. Un ejemplo:

un post de un blog, el resumen de ese post en una portada, un hilo en un foro, un comentario...

Usaremos `section` para agrupar contenido relacionado. Puede servirnos para enmarcar un conjunto de artículos, o incluso nuevas y diferentes secciones, o artículos que se dividan en secciones... la variedad es infinita.

`div` no tiene ninguna intención semántica, es un simple “agrupador” de contenido que nos ayudará a pintarlo con CSS.

## aside

Con el elemento `aside` se representa una parte de la página que abarca un contenido tangencialmente relacionado con el contenido que lo rodea, pero que no pertenece a ella, por lo que se le puede considerar un contenido independiente. Tiene la misma función que las barras laterales de referencia o que las secciones laterales de los libros y artículos.

Este elemento puede utilizarse para efectos tipográficos, barras laterales, elementos publicitarios, para grupos de elementos de la navegación, u otro contenido que se considere separado del contenido principal de la página.

## header

Representa la cabecera de la página o de una sección. Las cabeceras, además del título, pueden incluir un logo, subtítulos o información adicional.

## footer

Contiene la información que suele estar en el pie de un documento, autor, copyright... independientemente si está situada en el pie del documento, a un lado o donde sea. Al igual que `header` también puede usarse con las etiquetas `article` o `section` si estas lo requieren.

## hgroup

La función del elemento `hgroup` es agrupar un conjunto de encabezados en un sólo bloque para que puedan ser manejados como una unidad. El elemento `hgroup` se utiliza para agrupar una serie de h1-h6 cuando el encabezado tiene múltiples niveles, como subtítulos, títulos alternativos o slogans.

Dentro de un elemento `hgroup` solo podemos manejar etiquetas de tipo encabezado, por lo que no se permite introducir párrafos o algún otro tipo de etiqueta.

## nav

La etiqueta `nav` define una sección de vínculos de navegación. No todos los enlaces de un documento deben ser un elemento `nav`. La función de esta nueva etiqueta consiste en identificar los grupos de enlaces que, una vez agrupados, componen la navegación de la página. Algunos contenidos que se pueden agrupar gracias a esta etiqueta son: los enlace principales que generalmente se encuentran en la esquina superior derecha de una página web, los enlaces que remiten a datos como Anterior y Siguiente o las migas de pan.

Navegadores, tales como lectores de pantalla para usuarios con discapacidad, puede utilizar este elemento para determinar si se omite la representación inicial de este contenido.

Además, un lector de pantalla podría tener un comando (atajo de teclado) para acceder directamente a los elementos de navegación de una página.

Ejemplo de un blog hecho con HTML5:

```
<header>
  <hgroup>
    <h1>El blog de Mary</h1>
    <h2>Este es el blog de cocina de Mary</h2>
  </hgroup>
</header>
<nav>
  Aquí va la botonera de navegación
</nav>
<section>
  <article>Aquí va un post, con su titulo en h2</article>
  <article>Aquí va un post, con su titulo en h2</article>
  <article>Aquí va un post, con su titulo en h2</article>
</section>
<aside>
  Barra lateral con cosas que nadie lee, como cuentas de twitter,
  facebook, posts viejos, etc.
</aside>
<footer>
  Pie de pagina, copyright, etc.
</footer>
```

## Etiquetas que dotan de mayor semántica a la página

Existen otras etiquetas que dotan de mayor semántica a la página web. Estas son:

### dialog

Con frecuencia encontramos conversaciones y comentarios en la Web. Ahora con la etiqueta `dialog` podemos identificarlos fácilmente. Consta de 3 partes:

- `dialog`: Abre y cierra la totalidad de la conversación.
- `dt`: identifica al interlocutor
- `dd`: identifica el diálogo o comentario

### figure

La etiqueta `figure` identifica una imagen y su descripción como parte de un conjunto. Este elemento puede ser usado para ilustraciones, diagramas, fotos, listas de código, etc, que están referidos desde el contenido principal del documento

`figure` contiene una etiqueta adicional `legend` con la que se identifica el texto asociado a la imagen. Por ejemplo:

```
<figure>
  <legend>Fig.10 Etiquetas obsoletas en HTML5</legend>
  
</figure>
```

### mark

Sirve para resaltar partes del texto.

```
<p>Podemos dar <mark>énfasis</mark> al texto con la etiqueta <mark>mark</mark></p>
```

## time

Permite identificar un contenido como medida de tiempo. Este elemento puede ser utilizado como una forma de codificar las fechas y horas de una forma legible por máquina, de modo que, por ejemplo, los agentes de usuario pueden ofrecer para añadir recordatorios de cumpleaños o eventos programados en el calendario del usuario, y los motores de búsqueda puede producir resultados más inteligentes.

A continuación se muestran dos ejemplos:

```
<p>Las clases empiezan a las <time>8:20</time> de la mañana.</p>
```

```
<p>Mi cumpleaños es el <time datetime="2010-01-06">día de reyes.</time>.</p>
```

## Insertar audio y vídeo

Una de las características más sonadas de HTML 5 es su soporte multimedia, que permite la reproducción de archivos de audio/vídeo, o bien la conexión y reproducción de una fuente de audio/vídeo. Esto se hace en HTML 5 con las etiquetas `audio` y `video`, respectivamente.

Html 5 permite la reproducción de video sin utilizar flash. Los formatos más utilizados para la reproducción de video en html5 son: `.mp4` `.ogv` y `.webm`

Como siempre, cada navegador tiene su estándar. Para convertir a estos formatos se pueden usar programas como <http://www.mirovideoconverter.com> o [www.firefog.com](http://www.firefog.com)

Para visualizar un vídeo en HTML5 el código es el siguiente:

```
<video src="movie.ogg" controls="controls">
  Su navegador no soporta la etiqueta video de HTML5
</video>
```

El atributo `control` muestra los botones de play, pause y volumen. Se recomienda incluir los atributos `width` y `height`.

Entre las etiquetas `<video>` `</video>` introduciremos un mensaje que se visualizará en los navegadores que no soporten el elemento `video` de HTML5.

El **audio** se inserta casi de la misma manera que el vídeo:

```
<audio src="archivo.mp3" controls="controls">
  Su navegador no soporta la etiqueta audio de HTML5
</audio>
```

Hay que tener cuidado aquí, ya que no todos los formatos de audio y video son soportados por todos los navegadores:

- MPEG-4 (mp4) y MPEG-1 (mp3) → de alta calidad, propietario, era un estándar de facto. Soportado por Chrome, Safari, IE y la practica totalidad de los reproductores comerciales.
- Ogg Theora (ogg,ogv) y Ogg Vorbis (ogg) → menor calidad, libre. Soportado por Chrome, Firefox y Opera.
- Vp8 (WebM) → alta calidad, comprado por Google y liberado, es el llamado a acabar con la guerra. Actualmente lo soportan Chrome, Firefox 4, Opera, Safari (instalando los codecs) e IE9 con un plugin.

Otros atributos, aplicables tanto al audio como al video son:

- `autoplay`: que reproduce automáticamente el audio/video
- `autobuffer`: que permite la carga automática del archivo (sus posibles valores son `true` o `false`)
- `loop`: que ejecuta la reproducción en bucle (cuando finaliza vuelve a empezar)

- `type`: que indica el tipo de archivo o fuente (audio/mp3; video/mp4; video/ogg...)

## Formularios en HTML5

Una de las características más importantes que nos brinda el nuevo standard Html5 son las mejoras y nuevos elementos disponibles para el manejo de formularios. Hay que tener en cuenta que la nueva especificación de HTML es soportada de diferente manera por los distintos navegadores y no todas sus características se comportan igual

A pesar de todo debemos comenzar a diseñar formularios con HTML5 ya que hasta ahora, la forma que se utiliza para validar los formularios “del lado del cliente” era usando javascript. De ese modo ayudabamos al usuario a completar correctamente los formularios para recopilar coherentemente su información, desde el registro y acceso de una cuenta hasta procesos de pago de compra y otros.

Las nuevas características para manejo de formularios html5 permiten prescindir de javascript para realizar validaciones “del lado del cliente”, esto me permite como diseñador crear un formulario completo sin recurrir a un desarrollador, aumentando la productividad, sobre todo en equipos pequeños de trabajo.

De momento el navegador que mejor soporta las nuevas características asociadas a formularios es Opera, pero no por ello debemos dejar de usarlas, ya que para otros no será visibles pero serán muy útiles para quienes utilicen navegadores más modernos.

### nuevos atributos

- ***autofocus***: Hacemos foco en el input que tenga asociado este valor al cargar la página.  
`<input autofocus type=text>`
- ***placeholder***: Para ofrecer una pista de lo que el usuario debe introducir.  
`<input type=text placeholder="(555) 555-5555">`
- ***accept***: Permite que sólo el tipo de archivo determinado pueda ser cargado en el formulario.  
`<input type=file accept="image/*">`
- ***multiple***: Permite seleccionar múltiples archivos para ser cargados de una vez por el formulario.  
`<input type=file multiple>`
- ***max / min / step***: Para delimitar el rango de valores numéricos máximos, mínimos y múltiplos permitidos dentro de un rango.  
`<input type=number min=0 max=100 step=5>`
- ***required***: Atributo booleano que determina si el elemento es obligatorio o no. (Utilizado junto con “*pattern*”, “*max / min*”, “*email*” y otros nuevos atributos, nos permite prescindir en gran medida de Javascript para validar nuestros formularios y hacero sólo con el navegador.)  
`<input type=text required name=texto>`
- ***autocomplete***: Permite especificar si un elemento puede o no ser autocompletado por el navegador basado en entradas previas del usuario.  
`<input type=text name=name autocomplete=off>`
- ***pattern***: Para validar un elemento en base a una expresión regular.  
`<input type=text name=pattern-test pattern="[0-9][A-Z]{3}">`
- ***novalidate***: Evita la validación del elemento al enviar el formulario.  
`<form novalidate>`



```

        <input type=number name=num step=5>
    </form>

```

- **formaction:** Para sobrescribir el comportamiento por defecto del formulario.

```

<form action="http://ejemplo.com">
    <input type=submit value=Submit formaction="http://nuevoejemplo.com">
</form>

```

- **formmethod:** Para sobrescribir el método por defecto del formulario.

```

<form method=post>
    <input type=submit value=Submit formmethod=get>
</form>

```

## nuevos elementos

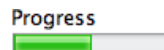
- **progress:** Con esta etiqueta se puede representar el progreso de cualquier evento. Por ejemplo, el progreso de descarga de un archivo, o la transferencia de datos. `progress` tiene dos atributos: `value` y `max`. El primero corresponde al valor de una descarga en un momento preciso; mientras que el segundo se refiere al valor total. En el siguiente ejemplo representamos la descarga de un archivo:

```

<progress value="24598" max="100000">25%</progress>

```

La etiqueta `progress` es estática. Por lo tanto debe vincularse con un programa JavaScript que permita seguir el proceso de los que se está rastreando. Así se vería en Opera:



- **meter:** Define una escala conocida dentro de un rango específico o una fracción.

```

<p>La altura es de <meter min=0 max=150 value=98>98cm</meter>.</p>

```

- **datalist:** Si asociamos un text input a un Datalist (lista de valores) al hacer foco en ese input aparece un dropdown mostrando el contenido del elemento Datalist.

```

<form>
    <label for="source">Cómo nos conociste?</label>
    <datalist id="sources">
        <select name="source">
            <option>elija por favor...</option>
            <option value="television">Televisión</option>
            <option value="radio">Radio</option>
            <option value="periodico">Periódico</option>
            <option>Otros</option>
        </select>
        Si ha seleccionado otros, por favor, especifique:
    </datalist>
    <input id="source" name="source" list="sources">
    <input type="submit">
</form>

```

- **keygen:** Genera un par de claves de control privada que se guarda en local y pública que se envía al servidor.

```

<keygen name=key>

```

- **output:** Se utiliza para realizar cálculos entre campos (por ejemplo, la suma de números de distintos inputs)

```

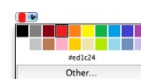
<form>
    <input name="number_1" type="number"> + <input name="number_2" type="number">
    = <output onforminput="value = number_1.valueAsNumber +
        number_2.valueAsNumber"></output>
</form>

```

## nuevos tipos de campos

Uno de los principales cambios en los formularios de HTML5 es la aparición de nuevos atributos para la etiqueta central de entrada `input`. Estos son algunos de ellos:

- **color:** Para seleccionar un color de una paleta



`<input type="color">`

- **number:** Específico para números

`<input type="number" min="0" max="100" step="2" value="6">`



- **range:** Para seleccionar un valor entre dos valores predeterminados

`<input type="range" min="0" max="10" step="2" value="6">`



- **tel:** Para números de teléfono. (En realidad, no prueba que sea un número, para validar un formato en particular habría que complementarlo con "pattern").

`<input type="tel">`

- **search:** Para utilizarlo en las cajas de búsqueda. En algunos navegadores no se notará la diferencia con el input de texto mientras que en otros se mostrará con las esquinas redondeadas.

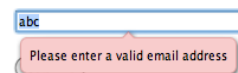
`<input type="search">`

- **url:** Para escribir direcciones web

`<input type="url">`

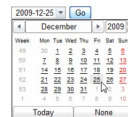
- **email:** Para valores únicos o múltiples de direcciones de correo

`<input type="email">`



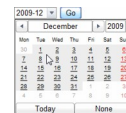
- **date:** Calendario para seleccionar un día.

`<input type="date" max="2023-04-20">`



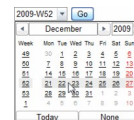
- **month:** Calendario para seleccionar los meses.

`<input type="month">`



- **week:** Fecha para ingresar la semana del año

`<input type="week">`



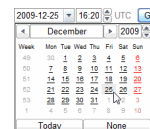
- **time:** Campo para las horas y minutos

`<input type="time">`



- **datetime:** Para indicar una fecha exacta

`<input type="datetime">`



- **datetime-local:** Campo para la fecha y hora local.

`<input type="datetime-local">`

Tendremos que esperar unos años hasta que estas características sean totalmente compatibles con los diferentes navegadores, Opera es uno de los que mayor compatibilidad tiene, seguido por Chrome y Safari.

## Almacenar datos en local gracias a Web Storage

Cuando queremos almacenar datos de una página Web, por lo general utilizamos cookies. Gracias al nuevo estándar de almacenamiento Web de HTML5 *Web Storage*, podemos guardar más datos que antes. Con este sistema, la información se almacena en una base de datos en el interior de los navegadores.

La principal ventaja de tener una base de datos dentro de un navegador es que podemos programar aplicaciones que se almacenan en local y que siguen funcionando aún cuando no estemos conectados a Internet. Esta función ya está disponible en los servicios Gmail, Calendar y Docs de Google. Estas aplicaciones funcionan aún cuando no estemos conectados.

La utilización de aplicaciones Web sin conexión no tiene mayor sentido para los PC tradicionales, pero sí para las aplicaciones de dispositivos móviles. Pensemos que estamos esperando el tren y decidimos consultar nuestro correo electrónico. Empezamos a contestar nuestros mensajes y mientras escribimos el tren se pone en marcha y nos desconecta de la red. ¿Qué sucede? Pues gracias al almacenamiento Web de HTML5, las aplicaciones del teléfono siguen funcionando y nosotros podemos contestar nuestros mensajes e incluso escribir unos nuevos sin siquiera enterarnos de que estábamos desconectados. Lo que hace Web Storage es que guarda los contactos, correos y respuestas en una base de datos local, en espera de una nueva contestación. Todo esto sucede sin que lo sepamos.

Google, Opera, Apple y Firefox utilizan la base de datos de código abierto SQLite para almacenar datos en HTML5 y navegar sin conexión.

Para almacenar datos de una página Web es indispensable utilizar JavaScript.

## Otras características de HTML5

- **Canvas:** El elemento canvas puede definirse como un entorno para crear imágenes dinámicas. El propio elemento es relativamente simple. Lo único que hay que especificar al usarlo son sus dimensiones:

```
<canvas id="entorno_canvas" width="360" height="240">
</canvas>
```

Cualquier cosa que escribamos entre la apertura y cierre de la etiqueta canvas solamente será interpretado por navegadores que no soportan aún la nueva etiqueta.

```
<canvas id="entorno_canvas" width="360" height="240">
  <p>Tu navegador no soporta canvas: Aquí deberías de ver una imagen :</p>
  
</canvas>
```

En el ejemplo anterior, la imagen solo sería visible por aquellos navegadores que aún no soporten canvas.

Canvas es un nuevo componente que permitirá dibujar, por medio de las funciones de un API, en la página todo tipo de formas, que podrán estar animadas y responder a interacción del usuario. El potencial de canvas reside en su habilidad para actualizar su contenido en tiempo real. Si usamos esa habilidad para responder a eventos de usuario, podemos crear herramientas y juegos que anteriormente a la nueva especificación hubiesen requerido de un plugin externo como Flash.

- **Web Workers:** son procesos que requieren bastante tiempo de procesamiento por parte del navegador, pero que se podrán realizar en un segundo plano, para que el usuario no tenga que esperar que se terminen para empezar a usar la página. Para ello se dispondrá también de un API para el trabajo con los Web Workers.
- **Geolocalización:** Una más de las novedades que nos trae HTML5 es la posibilidad de geolocalizarnos desde el propio navegador; una vez cargada la página web averiguar nuestras coordenadas en función de nuestra IP.

Esta geolocalización en HTML5 es gracias a una API cuyo responsable es el Geolocation Working Group, que no es el mismo que el encargado de desarrollar HTML5 (que es el HTML5 Working Group).

- **Aplicaciones web Offline:** Existirá otro API para el trabajo con aplicaciones web, que se podrán desarrollar de modo que funcionen también en local y sin estar conectados a Internet.

- **Nuevas APIs para interfaz de usuario:** temas tan utilizados como el "drag & drop" (arrastrar y soltar) en las interfaces de usuario de los programas convencionales, serán incorporadas al HTML 5 por medio de un API.

## Referencia de etiquetas HTML5

Etiqueta	Descripción
<a href="#"><u>&lt;!--...&gt;</u></a>	Define un comentario
<a href="#"><u>&lt;!DOCTYPE&gt;</u></a>	Define el tipo de documento
<a href="#"><u>&lt;a&gt;</u></a>	Define un hipervínculo
<a href="#"><u>&lt;abbr&gt;</u></a>	Define una abreviatura
<a href="#"><u>&lt;acronym&gt;</u></a>	No se admite en HTML5
<a href="#"><u>&lt;dirección&gt;</u></a>	Define la información de contacto del autor / propietario de un documento / artículo
<a href="#"><u>&lt;applet&gt;</u></a>	No se admite en HTML5
<a href="#"><u>&lt;area&gt;</u></a>	Define un área dentro de una imagen de mapa de
<a href="#"><u>&lt;article&gt;</u></a> <i>Nueva</i>	Define un artículo
<a href="#"><u>&lt;aside&gt;</u></a> <i>Nueva</i>	Define el contenido, aparte de contenido de la página
<a href="#"><u>&lt;audio&gt;</u></a> <i>Nueva</i>	Define el contenido de sonido
<a href="#"><u>&lt;b&gt;</u></a>	Define el texto en negrita
<a href="#"><u>&lt;base&gt;</u></a>	Especifica la URL base / blanco de todas las URLs relativas en un documento
<a href="#"><u>&lt;basefont&gt;</u></a>	No se admite en HTML5
<a href="#"><u>&lt;bdi&gt;</u></a> <i>Nueva</i>	Aislados de una parte del texto que pueda ser formateada en una dirección distinta de otro texto fuera de ella
<a href="#"><u>&lt;BDO&gt;</u></a>	Reemplaza la dirección del texto actual
<a href="#"><u>&lt;big&gt;</u></a>	No se admite en HTML5
<a href="#"><u>&lt;blockquote&gt;</u></a>	Define una sección en la que se cita de otra fuente
<a href="#"><u>&lt;body&gt;</u></a>	Define el cuerpo del documento
<a href="#"><u>&lt;br&gt;</u></a>	Define un salto de línea única
<a href="#"><u>&lt;button&gt;</u></a>	Define un botón que se pulse
<a href="#"><u>&lt;canvas&gt;</u></a> <i>Nueva</i>	Se utiliza para dibujar los gráficos, sobre la marcha, a través de scripting (normalmente JavaScript)
<a href="#"><u>&lt;caption&gt;</u></a>	Define una leyenda de la tabla
<a href="#"><u>&lt;center&gt;</u></a>	No se admite en HTML5
<a href="#"><u>&lt;cite&gt;</u></a>	Define el título de una obra
<a href="#"><u>&lt;code&gt;</u></a>	Define una pieza de código informático
<a href="#"><u>&lt;col&gt;</u></a>	Especifica las propiedades de cada columna dentro de un elemento <colgroup>
<a href="#"><u>&lt;colgroup&gt;</u></a>	Especifica un grupo de una o más columnas en una tabla de formato
<a href="#"><u>&lt;comando&gt;</u></a> <i>Nueva</i>	Define un botón de comando que el usuario puede invocar
<a href="#"><u>&lt;datalist&gt;</u></a> <i>Nueva</i>	Especifica una lista de opciones predefinidas para los controles de entrada

Etiqueta	Descripción
<a href="#"><u>&lt;dd&gt;</u></a>	Define una descripción de un elemento en una lista de definiciones
<a href="#"><u>&lt;sup&gt;</u></a>	Define un texto que ha sido borrado de un documento
<a href="#"><u>&lt;details&gt;</u></a> <i>Nueva</i>	Define los detalles adicionales que el usuario puede mostrar u ocultar
<a href="#"><u>&lt;em&gt;</u></a>	Define un término de definición
<a href="#"><u>&lt;dir&gt;</u></a>	No se admite en HTML5
<a href="#"><u>&lt;div&gt;</u></a>	Define una sección en un documento
<a href="#"><u>&lt;dl&gt;</u></a>	Define una lista de definiciones
<a href="#"><u>&lt;dt&gt;</u></a>	Define un plazo (un elemento) en una lista de definiciones
<a href="#"><u>&lt;em&gt;</u></a>	Define el texto subrayado
<a href="#"><u>&lt;embed&gt;</u></a> <i>Nueva</i>	Define un contenedor para una aplicación externa o contenido interactivo (un plug-in)
<a href="#"><u>&lt;fieldset&gt;</u></a>	Grupos de elementos relacionados en un formulario
<a href="#"><u>&lt;figcaption&gt;</u></a> <i>Nueva</i>	Define un título para un elemento <figure>
<a href="#"><u>&lt;figure&gt;</u></a> <i>Nueva</i>	Especifica autónomo contenido
<a href="#"><u>&lt;font&gt;</u></a>	No se admite en HTML5
<a href="#"><u>&lt;footer&gt;</u></a> <i>Nueva</i>	Define un pie de página de un documento o sección
<a href="#"><u>&lt;form&gt;</u></a>	Define un formulario HTML para la entrada del usuario
<a href="#"><u>&lt;frame&gt;</u></a>	No se admite en HTML5
<a href="#"><u>&lt;frameset&gt;</u></a>	No se admite en HTML5
<a href="#"><u>&lt;h1&gt; a &lt;h6&gt;</u></a>	Define encabezados HTML
<a href="#"><u>&lt;head&gt;</u></a>	Provee información sobre el documento
<a href="#"><u>&lt;header&gt;</u></a> <i>Nueva</i>	Especifica una introducción o un grupo de elementos de navegación para un documento
<a href="#"><u>&lt;hgroup&gt;</u></a> <i>Nueva</i>	Agrupar un conjunto de elementos que <h1> <h6> cuando una partida tiene varios niveles
<a href="#"><u>&lt;hr&gt;</u></a>	Define un cambio en el contenido temático
<a href="#"><u>&lt;html&gt;</u></a>	Define la raíz de un documento HTML
<a href="#"><u>&lt;i&gt;</u></a>	Define una parte de texto en una voz alternativa o estado de ánimo
<a href="#"><u>&lt;iframe&gt;</u></a>	Define un marco en línea
<a href="#"><u>&lt;img&gt;</u></a>	Define una imagen
<a href="#"><u>&lt;input&gt;</u></a>	Define un control de entrada
<a href="#"><u>&lt;ins&gt;</u></a>	Define un texto que se ha insertado en un documento
<a href="#"><u>&lt;keygen&gt;</u></a> <i>Nueva</i>	Define un campo generador de par de claves (para formularios)
<a href="#"><u>&lt;kbd&gt;</u></a>	Define la entrada del teclado
<a href="#"><u>&lt;label&gt;</u></a>	Define una etiqueta para un elemento de entrada
<a href="#"><u>&lt;legend&gt;</u></a>	Define un título para un fieldset, <figure> o elemento <details>
<a href="#"><u>&lt;li&gt;</u></a>	Define un elemento de la lista
<a href="#"><u>&lt;link&gt;</u></a>	Define la relación entre un documento y un recurso externo (la mayoría utiliza para vincular a hojas de estilo)

Etiqueta	Descripción
<a href="#"><u>&lt;map&gt;</u></a>	Define un cliente-lado de la imagen de mapa de
<a href="#"><u>&lt;mark&gt;</u></a> <i>Nueva</i>	Define el texto marcado / destacado
<a href="#"><u>&lt;menu&gt;</u></a>	Define una lista / menú de comandos
<a href="#"><u>&lt;meta&gt;</u></a>	Define los metadatos de un documento HTML
<a href="#"><u>&lt;meter&gt;</u></a> <i>Nueva</i>	Define una medida escalar dentro de un rango conocido (un indicador)
<a href="#"><u>&lt;nav&gt;</u></a> <i>Nueva</i>	Define los vínculos de navegación
<code>&lt;noframes&gt;</code>	No se admite en HTML5
<a href="#"><u>&lt;noscript&gt;</u></a>	Define un contenido alternativo para los usuarios que no soporten scripts del lado del cliente
<a href="#"><u>&lt;object&gt;</u></a>	Define un objeto incrustado
<a href="#"><u>&lt;ol&gt;</u></a>	Define una lista ordenada
<a href="#"><u>&lt;optgroup&gt;</u></a>	Define un grupo de opciones relacionadas en una lista desplegable
<a href="#"><u>&lt;option&gt;</u></a>	Define una opción en una lista desplegable
<a href="#"><u>&lt;output&gt;</u></a> <i>Nueva</i>	Define el resultado de un cálculo
<a href="#"><u>&lt;p&gt;</u></a>	Define un párrafo
<a href="#"><u>&lt;param&gt;</u></a>	Define un parámetro para un objeto
<a href="#"><u>Si su canción</u></a>	Define el texto preformateado
<a href="#"><u>&lt;progress&gt;</u></a> <i>Nueva</i>	Representa el progreso de una tarea
<a href="#"><u>&lt;q&gt;</u></a>	Define una cita corta
<a href="#"><u>&lt;rp&gt;</u></a> <i>Nueva</i>	Define lo que para mostrar en los navegadores que no soportan anotaciones ruby
<a href="#"><u>&lt;rt&gt;</u></a> <i>Nueva</i>	Define una explicación / pronunciación de los caracteres (para Asia del Este tipografía)
<a href="#"><u>&lt;ruby&gt;</u></a> <i>Nueva</i>	Define una anotación de ruby (para Asia del Este tipografía)
<a href="#"><u>&lt;S&gt;</u></a>	Define el texto que ya no es correcto
<a href="#"><u>&lt;samp&gt;</u></a>	Define la salida de muestra de un programa de computadora
<a href="#"><u>&lt;script&gt;</u></a>	Define un script del lado del cliente
<a href="#"><u>&lt;section&gt;</u></a> <i>Nueva</i>	Define una sección en un documento
<a href="#"><u>&lt;select&gt;</u></a>	Define una lista desplegable
<a href="#"><u>&lt;small&gt;</u></a>	Define el texto más pequeño
<a href="#"><u>&lt;source&gt;</u></a> <i>Nueva</i>	Define múltiples recursos de los medios de comunicación para los elementos de los medios de comunicación (<video> y <audio>)
<a href="#"><u>&lt;span&gt;</u></a>	Define una sección en un documento
<code>&lt;strike&gt;</code>	No se admite en HTML5
<a href="#"><u>&lt;strong&gt;</u></a>	Define el texto importante
<a href="#"><u>&lt;style&gt;</u></a>	Define la información de estilo para un documento
<a href="#"><u>&lt;sub&gt;</u></a>	Define el texto subíndice
<a href="#"><u>&lt;summary&gt;</u></a> <i>Nueva</i>	Define un título visible de un elemento de <details>
<a href="#"><u>&lt;sup&gt;</u></a>	Define el texto en superíndice

Etiqueta	Descripción
<a href="#"><code>&lt;table&gt;</code></a>	Define una tabla
<a href="#"><code>&lt;tbody&gt;</code></a>	Grupos del cuerpo contenido en una tabla
<a href="#"><code>&lt;td&gt;</code></a>	Define una celda de una tabla
<a href="#"><code>&lt;textarea&gt;</code></a>	Define un control de entrada de varias líneas (área de texto)
<a href="#"><code>&lt;tfoot&gt;</code></a>	Grupos del contenido del pie en una tabla
<a href="#"><code>&lt;th&gt;</code></a>	Define una celda de encabezado de una tabla
<a href="#"><code>&lt;thead&gt;</code></a>	Grupos de la cabecera del contenido de una tabla
<a href="#"><code>&lt;hora&gt;</code></a> <i>Nueva</i>	Define una fecha / hora
<a href="#"><code>&lt;title&gt;</code></a>	Define un título para el documento
<a href="#"><code>&lt;tr&gt;</code></a>	Define una fila de una tabla
<a href="#"><code>&lt;track&gt;</code></a> <i>Nueva</i>	Define pistas de texto para los elementos de los medios de comunicación ( <code>&lt;video&gt;</code> y <code>&lt;audio&gt;</code> )
<code>&lt;tt&gt;</code>	No se admite en HTML5
<code>&lt;u&gt;</code>	No se admite en HTML5
<a href="#"><code>&lt;ul&gt;</code></a>	Define una lista desordenada
<a href="#"><code>&lt;var&gt;</code></a>	Define una variable
<a href="#"><code>&lt;video&gt;</code></a> <i>Nueva</i>	Define un vídeo o una película
<a href="#"><code>&lt;wbr&gt;</code></a> <i>Nueva</i>	Define un posible salto de línea
<code>&lt;xmp&gt;</code>	No se admite en HTML5

## 2. CSS3

Aún cuando las etiquetas HTML5 distribuyen y organizan el contenido de un sitio Web de forma muy descriptiva, el aspecto visual de la página es controlado por las hojas de estilo en cascada de nivel 3.

El objetivo inicial de CSS, separar el contenido de la forma, se cumplió ya con las primeras especificaciones del lenguaje. Sin embargo, el objetivo de ofrecer un control total a los diseñadores sobre los elementos de la página ha sido más difícil de cubrir. Las especificaciones anteriores del lenguaje tenían muchas utilidades para aplicar estilos a las webs, pero los desarrolladores aun continúan usando trucos diversos para conseguir efectos tan comunes o tan deseados como los bordes redondeados o el sombreado de elementos en la página. CSS3 aporta más control sobre los elementos de la página.

Vamos a ver algunas de las características que aporta CSS3:

### Sombreado para texto

Es muy fácil aplicar sombras a textos con CSS3. La propiedad `text-shadow` de CSS le permite crear una copia un poco borrosa, ligeramente desplazada del texto, que termina pareciéndose a una sombra del mundo real. Según la especificación CSS3, la propiedad `text-shadow` puede tener los siguientes valores:

```
none | [<shadow> , ] * <shadow> ,
```

`<shadow>` se define como:

```
[<color>? <length> <length> <length>? | <length> <length> <length>? <color>? ],
```

donde los dos primeros representan la longitud horizontal y vertical y la tercera un radio opcional de difuminado. Vamos a ver algunos ejemplos.

Podemos hacer una simple sombra de esta forma, por ejemplo:

```
text-shadow: 2px 2px 3px #000;
```

## Una sombra simple

Si no queremos que se vea difuminado bastaría con poner a 0 la opción de difuminado, y variando el color se vería:

```
text-shadow: 2px 2px 0 #888;
```

## No me gusta el difuminado

Podemos hacer que el texto se vea borroso:

```
text-shadow: 0px 0px 5px #000;
```

## Texto borroso

Por último, puede añadir "más de una sombra", lo que permite crear efectos especiales como:

```
text-shadow: 0 0 4px blanco, 0-5px 4px #FFFF33, 2px-10px 6px #FFDD33,  
-15px 11px-2px #FF8800, 25px 18px-2px #FF2200
```



Al igual que todas las propiedades CSS se puede modificar `text-shadow` sobre la marcha utilizando JavaScript.

Existen muchas ventajas en la utilización de texto en lugar de imágenes: No utilizar imágenes disminuye el consumo de ancho de banda y de conexiones HTTP. Se mejora la accesibilidad, tanto para las personas que utilizan lectores de pantalla como para los motores de búsqueda. Y el zoom de la página funcionará mejor porque el texto se puede ampliar en lugar de utilizar la interpolación de píxeles para aumentar la escala de una imagen.

## Color en CSS3

La paleta de colores ha sido considerablemente ampliada en CSS3. Así podemos definirlo:

- Color name: Con nombres para los colores como red, pink o black.
- Full hexadecimal: Valor hexadecimal compuesto por seis valores alfanuméricos.
- Short hexadecimal: Valor hexadecimal compuesto por tres valores alfanuméricos.
- RGB: Combinación de rojo, verde y azul.
- RGBA: Combinación de rojo, verde azul y una transparencia (alfa).
- HSL: Combinación de tonalidad, saturación luminancia.
- HSLA: Combinación de tonalidad, saturación, luminancia y una transparencia (alfa).

Vamos a ver como se definiría el rojo utilizando los distintos métodos:

```
.color_name { color: red; }  
.fullHex    { color: #FF0000; }  
.shortHex   { color: #F00; }  
.rgb        { color: rgb(255,0,0); }  
.rgba       { color: rgba(255,0,0,100); }  
.hsl        { color: hsl(0%, 100%, 50%); }  
.hsla       { color: hsla(0%, 100%, 50%, 100%); }
```



# Background en CSS3

Las diferencias más significativas que ofrece la propiedad background en CSS3 son:

- La posibilidad de colocar varias imágenes de fondo
- El tamaño de nuestro fondo
- La posición de estas imágenes.

Vamos a ver un ejemplo aplicando unos estilos a la etiqueta DIV

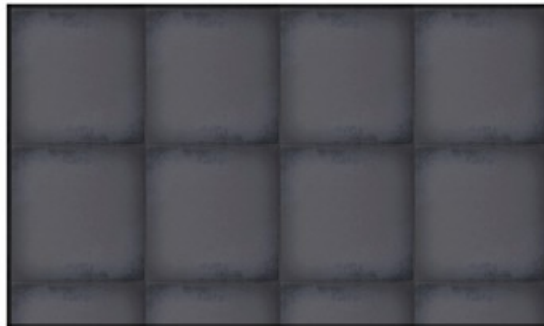
```
div{
  width:600px;
  height:350px;
  border: 5px solid hsla(30, 8%, 5%, .5); }
```

Ahora añadiremos a nuestro background una imagen cuyas medidas no son múltiplos de las de nuestro DIV con lo cual quedaría:

```
div {
  width:600px;
  height:350px;
  border: 5px solid hsla(30, 8%, 5%, .5);

  background-image:url(azulejo.png);
  /*Tamaño de la imagen 125 x 125*/
}
```

con lo cual se vería como se muestra a continuación, ya que la imagen de los azulejos no es múltiplo de nuestro DIV y los azulejos son cortados en la parte inferior. Pudiendo ser cortados también en la parte de la derecha.



Para evitar esto que nos ha sucedido, deberíamos emplear Background-size asignándole dos valores, uno para el ancho y otro para el alto. Si queremos mantener las proporciones en la imagen o ratio debemos poner uno de los dos valores auto y que se redimensione en función a su otro valor en px. Si dejamos el segundo valor en blanco el navegador lo interpretará como auto también. Así, nuestra imagen se adapta al tamaño del background.

```
div {
  width:600px;
  height:350px;
  border: 5px solid hsla(30, 8%, 5%, .5);

  background-image:url(azulejo.png);
  /*Tamaño de la imagen 125 x 125*/

  background-size: 50px auto;
}
```

De este modo, la imagen de los azulejos se adaptó perfectamente a nuestro DIV sin perder las proporciones y sin que nuestra imagen haya sido cortada por ningún lado.



Como unidad de medida hemos utilizado **px** haciendo referencia al tamaño que queremos para nuestra imagen de fondo. También podemos emplear tantos por ciento. Éstos hacen referencia al tamaño del contenedor. Si cambiamos los valores de `background-size`

```
background-size: 50% 100%;
```

se vería:



### 2.1.1 Background con varias imágenes

Con CSS3 podemos insertar varias imágenes de fondo fácilmente, empleando el mismo elemento `background` y separando las imágenes por comas. Cada una de estas imágenes podrá ser posicionada, repetida y dimensionada, pudiendo aplicar a éstas las mismas variaciones que cuando trabajamos con una sola imagen.

Vamos a colocar dos imágenes encima del azulejo. Las imágenes se muestran en orden inverso a su colocación. Es decir, la imagen de azulejos que tenemos intención que vaya detrás del todo hay que colocarla en nuestro código la última, de la misma forma con las demás.

Como hay navegadores que no soportan este tipo de `background`, es muy recomendable colocar un estilo `background` antes del de múltiples imágenes. De esta forma si el navegador no reconoce `background` para varias imágenes habrá tomado el estilo definido una línea anterior. Así, en nuestro ejemplo, si queremos incluir una imagen de un príncipe y otra de una rana sobre el azulejo:

```
div {
  width:600px;
  height:350px;
  border: 5px solid hsla(30, 8%, 5%, .5);

  background-size: 50% 100%;

  /*Para los navegadores que no soportan varias imágenes*/
  background:url(azulejo.png);

  /*En caso de soportar varias imágenes sustituye el estilo por este*/
  background: url(principe.png),
              url(rana.png),
              url(azulejo.png);
}
```

Donde la propiedad `background-size` se la aplica a todas las imágenes:



Ahora añadiremos valores de posicionamiento para colocar nuestra rana y nuestro saltamontes cada uno en un azulejo y al mismo tiempo le vamos a decir que no se repita la imagen. Estos valores de posicionamiento pueden ser px y tantos por ciento %.

```
background:url(principe.png) 0% 5px no-repeat,
            url(rana.png) 100% 5px no-repeat,
            url(azulejo-pequeno.png);
```

Al principe le indicamos que se sitúe en el eje X a 0% y a 5 px en el eje Y. Del mismo modo colocamos la rana.



Por último, podríamos indicar un tamaño para cada una de nuestras imágenes en la propiedad `background-size`. El orden para dar estos valores es de arriba hasta abajo en la lista de imágenes y se aplicará de la siguiente forma.

```
background-size: auto, auto, 50% 100%;
```

Con el valor **auto** le estamos indicando que tome el valor original de la imagen para `principe.png` y `rana.png` quedando nuestra imagen:



## Degradados

Con CSS3, además de los colores sólidos, también podemos hacer degradados. Estos son de dos tipos: lineales y radiales.

Al definir un degradado tenemos que definir los siguientes elementos:

- Type: radial o lineal

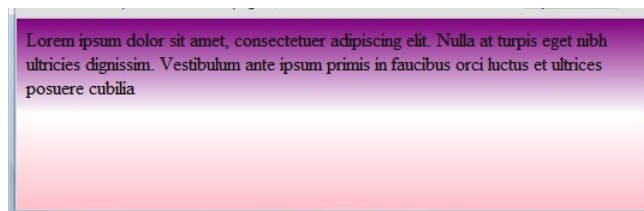
- Point: Dos valores separados por un espacio que definen donde empieza y donde termina el degradado (se usa un número, un porcentaje, o las palabras clave top, bottom, left y right)
- Radius: Es un número que sólo se utiliza para el degradado radial.
- Stop: Determina la intensidad de la mezcla de colores como un porcentaje o número entre 0 y 1 (como 0,75 o 75%) y un color.

Al definir todas estas características juntas creamos un degradado. Los degradados pueden utilizarse con las siguiente definiciones:

- Background-image
- Border-image
- List-style-image
- Content property

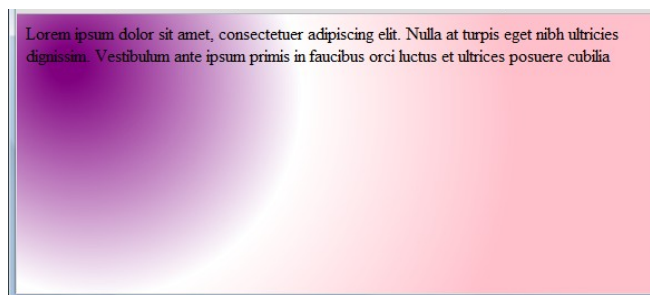
A continuación se muestra un ejemplo de un degradado que reemplaza la imagen de fondo y va desde el color púrpura al rosa. La función stop combina los colores hasta la mitad del color blanco.

```
body {
    background-image: -webkit-gradient(linear, left top, left bottom,
        from(purple), to(pink), color-stop(0.5 , white), color-stop(0.5 , white));
}
```



Para crear un degradado radial sería:

```
body {
    background-image: -webkit-gradient(radial, 45 45, 15, 100 100, 350,
        from(purple), to(pink), color-stop(0.5 , white));
}
```



## Bordes redondeados

CSS 3 incorpora nuevas propiedades para el control de bordes de los elementos. La propiedad `border-radius` permite definir bordes redondeados en las esquinas, especificando las medidas del radio que deben darse a la curva de las esquinas. Por ejemplo:

```
border-radius: 5px;
```

Definiría un radio de 5 píxeles en el redondeo de las esquinas del elemento. Por el momento para Mozilla el nombre del atributo es `-moz-border-radius` y para los navegadores basados en WebKit, como Google Chrome o Safari es `-webkit-border-radius`.

Vamos a ver un ejemplo donde queremos que todos los div tengan un borde redondeado en las esquinas de radio de 7 píxeles.:

```
DIV {
  border: 1px solid #000000;
  -moz-border-radius: 7px;
  -webkit-border-radius: 7px;
  padding: 10px;
}
```

Esto es un div con los bordes redondeados. Pero el estilo definido por -moz-border-radius: 7px; sólo se verá en navegadores basados en Mozilla.

Se pueden definir los valores para el radio de las cuatro esquinas por separado. De esta manera:

```
-moz-border-radius: 7px 27px 100px 0px;
```

con lo cual se vería:

Esto es un div con los bordes redondeados. Pero el estilo definido por -moz-border-radius: 7px; sólo se verá en navegadores basados en Mozilla.

## Selectores de atributos en CSS3

Vamos a ver algunos de los selectores de atributos más significativos de CSS2 y CSS3. Mediante los selectores de CSS podemos seleccionar diferentes elementos de nuestra web para dar a éstos una forma determinada aplicando estilos.

Con los selectores de atributos, podemos aplicar estilos definidos, a los elementos de nuestra web que tengan determinados atributos y éstos un valor determinado.

### [atr]

Aplica un estilo determinado a los elementos que tengan un determinado atributo, independientemente del valor que tenga éste. Por ejemplo, si queremos que todos nuestros objetos que tengan el atributo title aparecerán con el nuevo estilo:

```
[title]{
  border:5px dotted #333;
}
```

### div[atr]

Aplica un estilo determinado a los “tipos de elementos” que tengan un atributo especificado, independientemente del valor que tenga éste. Por ejemplo, si queremos que todos nuestros objetos div que tengan el atributo title aparecerán con el nuevo estilo sería:

```
div[title]{
  border:5px dotted #333;
}
```

### [atr=valor]

Aplica un estilo a los elementos que tengan un determinado atributo y su valor se igual que el que nosotros hemos especificado. Por ejemplo, si queremos que todos nuestros objetos div que tengan el atributo title con valor Azul aparecerán con el nuevo estilo aplicado.

```
[title="Azul"]{
  background-color:#039;
}
```

### [atr ~= valor]

Aplica un estilo determinado a los elementos que tengan en el valor de un atributo una lista de palabras y una de ellas coincida. Por ejemplo, para aplicar el nuevo estilo a nuestros elementos que tengan el atributo title y en su valor aparezca la palabra verde.

```
[title~="verde"]{
    background-color:#039;
}
```

### [atr |= "valor"]

Aplica estilos a los elementos en los que el valor de su atributo *comience* por una determinada palabra *y a su vez lleve un guión*. Por ejemplo, si queremos aplicar un estilo a los elementos cuyo atributo title comienza por “Logo-“ sería:

```
[title |= "Logo"]{
    background-color:#0F0;
}
```

### [atr^="valor"]

Aplica estilos a todos los elementos que tengan el atributo seleccionado y *su valor comience* por unos caracteres en concreto. Por ejemplo, si queremos aplicar un estilo a los elementos cuyo atributo title comience por Logo sería:

```
[title ^= "Logo"]{
    border:2px double #C00;
}
```

### [atr \$= "valor"]

Aplica estilos a todos los elementos que tengan el atributo seleccionado y su valor *termine* por unos caracteres en concreto. Por ejemplo, si queremos aplicar un estilo a los elementos cuyo atributo title termine por css3 sería:

```
[title $= "css3"]{
    border:2px double #C00;
}
```

Este selector podríamos usarlo por ejemplo para **asignar una imagen determinada dependiendo de la extensión**, pdf, rar, txt, doc, jpeg.

### [attr \*= "val"]

Aplica estilos a todos los elementos que tengan el atributo seleccionado y en su valor unos caracteres concretos, sin importar la posición de éstos. Por ejemplo, si queremos aplicar un estilo a los elementos cuyo atributo title contenga en cualquier posición “de”:

```
[title *= "de"]{
    border:2px double #C00;
}
```

## Pseudo-clases con CSS3

Otro método de dar formato a una página web es a través de las pseudo-clases CSS. Su uso más común es con las etiquetas que identifican enlaces como se visualiza a continuación:

```
a:link {color:#FF0000;} /* enlace no visitado*/
a:visited {color:#00FF00;} /* enlace visitado */
a:hover {color:#FF00FF;} /* ratón sobre el enlace */
a:active {color:#0000FF;} /* enlace seleccionado*/
```

## Las pseudo-clases y clases CSS

Las pseudo-clases se pueden combinar con clases CSS:

```
a.red:visited {color:#FF0000;}
```

```
<a class="red" href="mienlace.html">CSS3</a>
```

Si el enlace del ejemplo anterior se ha visitado, se mostrará en rojo.

## root

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

```
:root{
  font-family:Arial, Helvetica, sans-serif;
  color:#cff;
}
```

## first-child

El primer hijo de un elemento. En el siguiente ejemplo, el selector equivale a cualquier elemento `p` que es el primer hijo de cualquier elemento.

```
p:first-child
{
  color:blue;
}
```

Y el siguiente selector equivale a cualquier elemento `p` que es el primer hijo de un elemento `div`

```
div > p:first-child
{
  color:blue;
}
```

Para todos los elementos `i` que se encuentran en un `p` que es primer hijo, sería:

```
p:first-child i
{
  color:blue;
}
```

## last-child

El último hijo de un elemento. Funciona igual que el anterior pero para el último hijo.

## only-child

El único hijo de un padre. Para seleccionar los enlaces que son hijos únicos de su padre:

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

## Nth-child(n)

El enésimo hijo de un padre. De este modo seleccionamos el hijo situado en segunda posición del `div` de clase ejemplo.

```
div.ejemplo:nth-child(2){
  color:blue;
}
```

### `nth-child (even)`

Con este ejemplo seleccionaremos los elementos `p` que tengan una posición par, para esto empleamos la palabra `even`.

```
p:nth-child(even) {  
    color:blue;  
}
```

### `nth-child (odd)`

Con este ejemplo seleccionaremos los elementos `p` que tengan una posición impar

```
p:nth-child(odd) {  
    color:#9FF;  
}
```

### `Nth-last-child(n)`

El enésimo hijo de su padre, contando del último al primero.

### `nth-of-type()`

Con esta pseudo-clase podremos seleccionar mediante la posición, elementos hijos de una forma alterna pero de un determinado tipo. Por ejemplo: Si tenemos un `div` con un `h2` y tres párrafos, para seleccionar el segundo párrafo sería:

```
p:nth-of-type(2) {  
    color:blue;  
}
```

### `nth-last-of-type()`

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

### `first-of-type`

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

### `only-of-type`

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

### `empty`

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

### `disabled`

Elemento desactivado



## enabled

Elemento activado

## Checked

Elemento seleccionado

## lang

Selecciona elementos en función de su idioma. Los navegadores utilizan los atributos `lang`, las etiquetas `<meta>` y la información de la respuesta del servidor para determinar el idioma de cada elemento.

```
p:lang(es) { color: red; }
```

No podemos confundir la pseudo-clase `:lang(xx)` con el selector de atributos `[lang|=xx]`, tal y como se muestra con el siguiente ejemplo:

```
*[lang|=es] { ... } /* selector de atributo */
*:lang(es) { ... } /* pseudo-clase */

<body lang="es">
  <p>Por otra parte, los navegadores...</p>
</body>
```

El selector `*[lang|=es]` selecciona todos los elementos de la página que tengan un atributo llamado `lang` cuyo valor empiece por `es`. En el ejemplo anterior, solamente el elemento `body` cumple con la condición del selector.

Por otra parte, el selector `*:lang(es)` selecciona todos los elementos de la página cuyo idioma sea el español, sin tener en cuenta el método empleado por el navegador para averiguar el idioma de cada elemento. En este caso, tanto el elemento `body` como el elemento `p` cumplen esta condición.

# Pseudo-elementos

Una de las novedades de CSS3 es la introducción de pseudo-elementos que permiten controlar aspectos particulares de un elemento; como por ejemplo, el estilo de la primera letra de una palabra. CSS3 tiene cuatro pseudoelementos:

## first-letter

Se utiliza para darle un estilo especial a la primera letra de un texto.

```
p:first-letter
{
  color:red;
  font-size:xx-large;
}
```

## first-line

Se utiliza para darle un estilo especial a la primera línea del texto.

## before

El pseudo-elemento `before` puede ser utilizado para insertar algún contenido antes de que el contenido de un elemento. El siguiente ejemplo se inserta una imagen antes de cada elemento `h1`

```
h1:before
{
    content:url(logo.gif);
}
```

## after

El pseudo-elemento `after` puede ser utilizado para insertar algún contenido después de que el contenido de un elemento.

Los pseudo-elementos `:before` y `:after` se utilizan en combinación con la propiedad `content` de CSS para añadir contenidos antes o después del contenido original de un elemento. En el siguiente ejemplo se añade el texto Capítulo - delante de cada encabezado `h1` y un punto detrás de cada párrafo de la página:

```
h1:before {
    content: "Capítulo - ";
}
p:after {
    content: ".";
}
```

# Trabajar con columnas en CSS3

En cualquier página web resulta difícil estructurar un texto en columnas. Por lo general es necesario crear un complejo sistema de tablas para lograrlo.

En CSS3 tenemos nuevas propiedades que nos permiten trabajar con columnas. Es posible especificar el número de columnas que necesitamos y el navegador asignará automáticamente el ancho de cada columna para que quepa la cantidad que especificamos. Otra manera de trabajar con columnas con CSS3 es especificando el ancho de las columnas y se crearán las columnas que quepan de ese ancho; además se pueden especificar el margen entre cada columna, así como definir que haya una especie de borde que separe las columnas.

Como con la mayoría de las propiedades de CSS3, necesitamos agregar prefijos específicos de cada servidor. Para los navegadores basados en Webkit como Safari o Chrome se necesita el prefijo `-webkit-`; para los navegadores de Mozilla, como Firefox, se necesita `-moz-`; para Opera se utiliza `-o-`. Esperemos que se adopte CSS3 como un estándar para ya no necesitar estos prefijos.

- `Column-count` permite especificar el número de columnas que se desean. El navegador repartirá el contenido en el número de columnas especificadas. El navegador calcula el ancho de cada columna para que quepa dentro del espacio especificado, y también calculará el alto de las columnas.
- `Column-width` permite especificar el ancho de las columnas y el navegador calculará la cantidad de columnas que caben en el espacio asignado.

Aparte de las dos propiedades para definir las columnas, tenemos otras dos para dar estilo a las columnas:

- `column-gap` se utiliza para especificar un margen entre cada columna.

- `column-rule` es como un borde entre cada columna y tiene las mismas propiedades que la propiedad `border`: `width`, `color` mediante el uso de los estilos `column-rule-width`, `column-rule-color`, and `column-rule-style`.

También se puede especificar la altura de cada columna con la propiedad `column-height`. Esta se debe usar con cuidado ya que las columnas no van a exceder esta altura y si el contenido no cabe en el espacio disponible el navegador va a agregar nuevas columnas haciendo que se salga del espacio especificado.

```
.columnas{
  -moz-column-count: 3;
  -webkit-column-count: 3;
  -o-column-count: 3;
  column-count: 3;
  -moz-column-gap: 20px;
  -webkit-column-gap: 20px;
  -o-column-gap: 20px;
  column-gap: 20px;
}
```

## Declarar variables

CSS3 permite el uso de variables. La sintaxis para declarar variables es:

```
@variables { keyColor: #FF0; }
```

una vez declarada, puedes invocar esta variable siempre que la necesites dentro de tu código CSS de la siguiente manera:

```
h1 { color: var(keyColor); }
```

Por ejemplo, podrías declarar toda tu paleta de colores con variables al principio del CSS y después usarlas:

```
@variables { firstcolor: #f00; secondcolor: #300; thirdcolor: #300; }
```