GUÍA COMPLETA DE CSS

Antonio Navajas Ojeda
@ajnavajas
www.antonionavajas.com

Si tienes cualquier duda o consulta acerca de su contenido, no dudes en contactar conmigo a través de twitter a @ajnavajas

Curso de CSS3

Introducción

Este libro trata sobre CSS3, la última versión del Cascade Style Sheet. El autor da por hecho que conoces tanto HTML como las versiones anteriores de CSS. Si no es así te recomiendo visitar el maravilloso curso de HTML y CSS de Cristalab y el tutorial de introducción a HTML 5.

Para sacarle partido a este curso necesitarás las siguientes herramientas:

- Un editor de texto plano o editor de código para HTML y CSS. Yo uso Komodo, aunque reconozco Sublime Text o Notepad++ son excelentes opciones.

- Las últimas versiones de un par de navegadores que sigan los estándares como Firefox o Chrome. A lo largo de todo este libro encontraréis avisos de lo poco compatible que es todo con Internet Explorer. Bueno, las versiones 9 y 10 parece que pretenden solucionar en gran medida esto.

***Importante**. Dada la reducida compatibilidad actual de los navegadores con CSS3 es aconsejable añadir los prefijos **-moz-** y **-webkit-** en todos los ejemplos que realicemos. De manera opcional podéis probar con **-o-** y **-ms-**aunque no os puedo asegurar su funcionamiento en estos motores.*

Sobre CSS3

Creo que si estás leyendo este libro, sobra decirte que CSS son las siglas de “Hojas de estilo en cascada” y que sirve para dar estilo a nuestra estructura creada en HTML. CSS3 es la última versión hasta la fecha y presenta como principales características **mayor control sobre el estilo** de los elementos de nuestra página web y mayor número de efectos visuales. Las ventajas de usar CSS3 son:

- Código más simple para muchas tareas
- Mayores opciones de gráficas
- Usar CSS3 te asegura ligar un 20% más.Estadística calculada a ojo. El valor real puede oscilar entre el 100 y -100%.

Como desventaja, en el momento de escribir este curso, CSS3 no tiene compatibilidad 100% con ningún navegador. Los ejemplos de este curso irán indicando si requieren específicamente del uso de Firefox y Chrome para su correcta visualización. El autor no se responsabiliza por los traumas o frustraciones creados a usuarios de Internet Explorer.

La cosa no acaba ahí. Como es habitual a la hora de crear un estándar surgen 20 versiones del mismo, cada una con sus propuestas. En este libro no voy a tratar de aspectos únicos de Webkit o Gecko (los motores de Chrome y Firefox respectivamente) sino que procuraré que aquello que explique sea aplicable a ambos navegadores.

> ***Consejo:*** *Prueba siempre los ejemplos en varios navegadores. Cada uno puede interpretar de forma ligeramente diferente nuestro CSS. Tampoco olvides utilizar los prefijos.*

Tema 1. Selectores

He querido comenzar este libro con la parte más teórica. Algunos podrían pensar que todo el bloque de Selectores, Pseudoclases y Pseudoelementos es un poco coñazo. Tal vez lo sea en comparación a las sombras, los bordes redondeados y las transiciones. Sin embargo para ser un auténtico ninja del CSS conocer el manejo de estas herramientas es fundamental y es la línea que separa al buen front-end del mediocre.

En estos tres primeros temas veremos los nuevos selectores, pseudo-elementos y pseudo-clases. No voy a entrar en detalle sobre los conceptos en si porque entiendo que ya conocíais el CSS 2.1. Si no, aquí mismo tenéis el curso de CSS básico.

> ***Importante:*** *Los selectores son el "abc" del CSS. Hay que conocerlos a fondo si queremos crear un código profesional.*

Un selector en CSS es la herramienta que **nos sirve para *seleccionar* uno o varios elementos** de los que tenemos en nuestro HTML. Cuando decimos seleccionar nos referimos a referenciar ese elemento para aplicarle el estilo. CSS3 incluye todos los selectores de CSS2.1 y añade algunos más de forma que es mucho más sencillo referenciar elementos.

Selectores de atributos en CSS3

En primer lugar encontramos 3 nuevos selectores de atributos:

1. **Código :**

```
elemento[atributo^="valor"]
```

Selecciona los elementos con ese atributo y que su valor comienza por la cadena de texto indicada en "valor".

2. **Código :**

```
elemento[atributo$="valor"]
```

Selecciona los elementos con ese atributo y que su valor termina por la cadena de texto indicada en "valor".

3. **Código :**

```
elemento[atributo*="valor"]
```

Selecciona los elementos con ese atributo y que su valor contiene la cadena de texto indicada en "valor".

Por ejemplo

Código :

```
/* Selecciona todos los enlaces que
lleven a una página que contenga la
palabra ejemplo*/
a[href*="ejemplo"]{...}

/*Selecciona todos los enlaces que
apunten a una dirección de correo*/
a[href^="mailto:"]{...}

/*Selecciona todos los enlaces que
apuntan a páginas .php*/
a[href$=".php"]{...}
```

- Esto es un enlace a un ejemplo y aparecerá en rojo
- Esto es un enlace a una dirección de correo y aparecerá en azul
- Esto es un enlace a una página php y aparecerá en verde

Ver ejemplo

Selector general de hermanos

El **selector general de hermanos** tiene un funcionamiento similar al *selector adyacente* de CSS2.1.

Mientras que en el adyacente la condición es que los elementos fuesen consecutivos en el código HTML, el selector general de hermanos tiene en consideración todos los elementos que sean consecutivos **aunque no lo sean en el código HTML**.

Código :

```
/*Selector adyacente en CSS2.1*/
h1 + h2{...}
/*Selector general de hermanos de CSS3*/
h1 ~ h2{...}

<h1>Título</h1>
<h2>Subtítulo adyacente</h2>

<h1>Título</h1>
<p> párrafo de separación</p>
<h2>Subtítulo con selector general de
hermanos</h2>
```

Título

Subtítulo adyacente

Título

párrafo de separación

Subtítulo con selector general de hermanos

Ver ejemplo

Tema 2. Pseudo elementos de CSS3

En este capítulo abordaremos los pseudo-elementos del CSS3 y sus cambios respecto a CSS2.1.

En la versión 2.1 contábamos con 4 pseudo-elementos:

- *:first-line*. Selecciona la primera línea.
- *:first-letter*.Selecciona la primera letra.
- *:before*. Nos posiciona al inicio del contenido de un elemento.
- *:after*. Nos posiciona al final del contenido de un elemento.

La siguiente imagen muestra dónde apuntaría el selector en cada uno de los casos:

CSS3 conserva estos pseudo-elementos aunque cambia su sintaxis. Para usarlos **escribiremos "::" en lugar de ":"**.

Código :

```
p::first-letter{
    color:red;
}
p::first-line{
    color:blue;
}
h1::after{
    content:".";
}
h1::before{
    content:"Ejemplo de ";
}

<div id="ejemplo">
        <h1>pseudo-elementos</h1>
        <p>Esto es un ejemplo usando
pseudo elementos en CSS3 donde pasamos
coloreamos de rojo la primera letra
con first-letter, coloreamos de azul
la primera l&iacute;nea con first-
line, e incluimos en el título el
texto "Ejemplo de" y "." con before y
after respectivamente.
</div>
```

Ver ejemplo

CSS3 añade un nuevo pseudo-elemento:

- *::selection*. Selection referencia al texto seleccionado por el usuario.

Si usamos

Código :

```
p::selection{
        background-color:blue;

}
```

Cuando seleccionemos el texto del párrafo, la selección tendrá color de fondo azul en vez del típico gris.

> ***Importante:*** *Este selector me ha dado problemas con algunas versiones de Firefox. Es aconsejable usar siempre los prefijos -webkit- y -moz- en todas vuestras pruebas.*

Ver ejemplo

Tema 3. Pseudo-clases.

Uno de los puntos donde CSS3 hace grandes incorporaciones es en las pseudo-clases. Gracias a las nuevas pseudo-clases tendremos un control mucho más ajustado de los elementos HTML.

> ***Consejo:*** *Ya que Internet Explorer 6-8 no soporta la mayoría de pseudo-clases se han desarrollado algunas librerías de javascript que realizan las mismas funciones para estos navegadores. La que mejores resultados me ha dado ha sido* ***select[ivizr]*** *que podréis descargar de su página oficial.*

A continuación os explico en detalle cada una de ellas con algunos ejemplos.

- *:nth-child(N)*. Selecciona los elementos en base a sus posiciones en una lista de elementos hijos dentro de un elemento padre. N es el número de la posición.

Código :

```
/*Selecciona los elementos impares de la
clase .nthchild*/
.nthchild:nth-child(odd){
   color:red;
}
```

- *:nth-last-child(N)*. Selecciona los elementos tomando de referencia el último elemento de una lista. N es

el número de posición desde el último elemento, donde el valor 1 equivaldría al último.

Código :

```
/*Selecciona el penúltimo elemento de la clase .nthlastchild*/
.nthlastchild:nth-last-child(2){
   color:blue;
}
```

- *:nth-of-type(N)*. Selecciona los elementos de un tipo concreto dentro de los hijos de un elemento padre.

Código :

```
/*Selecciona los párrafos pares dentro de una capa*/
 div>p:nth-of-type(even){
   color:red;
}
```

Con las pseudo-clases, pseudo-elementos y selectores nos convertiremos en verdaderos *francotiradores* del código.

- *:nth-last-of-type(N)*. Igual que el anterior tomando como referencia el último elemento de la lista.

- *:last-child*. Selecciona el último elemento de la lista de elementos hijos. Sería el análogo al :first-child de CSS2 y es equivalente a :nth-last-child(1).

- *:first-of-type*. Selecciona el primer elemento de un tipo concreto dentro de la lista de hijos.

Código :

```
/*Selecciona el primer párrafo dentro de una capa*/
div>p:first-of-type{
   font-style:italic;
}
```

- *:last-of-type*. Selecciona el último elemento de un tipo.

Código :

```
/*Selecciona el último párrafo dentro de una capa*/
div>p:last-of-type{
   font-style:italic;
}
```

- *:only-child*. Selecciona el elemento si es el único elemento hijo.

- *:only-of-type*. Selecciona el elemento si es el único elemento hijo de ese tipo.

- *:root*. Selecciona el elemento raíz del documento.

- *:empty*. Selecciona los elementos que no tienen hijos.

- *:enabled*. Selecciona los elementos de la interfaz que tengan un estado de enable.

- *:disabled*. Selecciona los elementos de la interfaz que tengan un estado disabled.

- *:checked*. Selecciona los *checkboxes* o *radio buttons* que estén activados.

- *:not(S)*. Selecciona los elementos que **no coincidan** con el selector especificado.

Código :

```
/*Selecciona los h3 que no pertenezcan a
la clase .rojo*/
h3:not(.rojo){
   font-style:italic;
}
```

Podéis ver ejemplos funcionales aquí.

Tema 4. Efectos de texto y tipografía.

En anteriores temas hemos visto selectores, pseudo-elementos y pseudo-clases que son herramientas fundamentales para lograr un código más depurado y limpio. Estoy seguro que más de un diseñador habrá encontrado este comienzo muy pesado y aburrido. A partir de ahora veremos características de CSS más visuales con la que podrás explotar tu creatividad.

Los diseñadores web se han encontrado durante bastante tiempo con enormes limitaciones en cuanto a texto se refiere. CSS3 nos da un ápice de libertad en este sentido ya que por fin se asienta como estándar la implementación de *font-face*, nos permite editar textos en varias columnas y soluciona algunos problemas de visualización con *word-warp*.

@font-face

Font-face nace en realidad con la versión 2 de CSS aunque hasta ahora no ha empezado a funcionar y prosperar adecuadamente. La implementación de font-face es la siguiente:
Código :

```
@font-face{
   font-family:<nombre_fuente>;/*El
nombre con el que nos referiremos a la
fuente*/
```

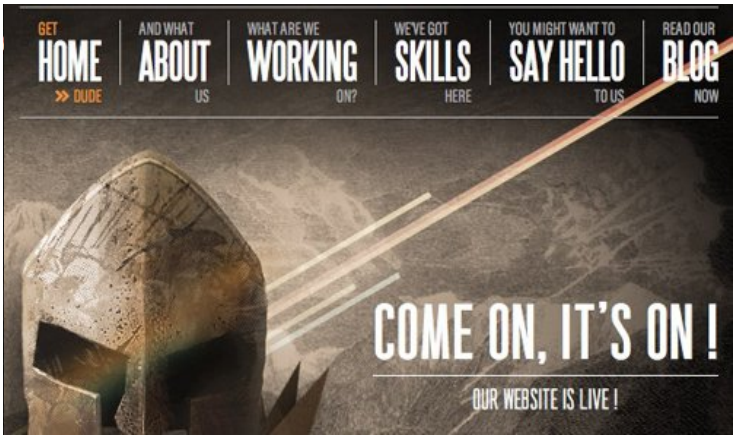
```
   src: <source>[,<source>]*;/*La ruta de
donde cargamos el archivo del tipo*/
   [font-weight:<weigth>];/*Peso de la
fuente*/
   [font-style:<style>];/*Estilo*/
}
```

Veamos un ejemplo:

Código :

```
@font-face{
            font-family:"Deliciosa";
            src:url('fonts/Delicious-
Roman.otf');
        }
        #ejemplo{
            font-family:"Deliciosa";
        }
```

Notad que el nombre que indicamos en font-family es el nombre que actuará como *alias* de la tipografía, por lo tanto es irrelevante su valor, siempre que nos refiramos a ella de la misma manera.



En epicagency.net usan una tipografía especial para sus títulos y menús.

Existe el riesgo de que un navegador determinado no interprete correctamente algunos formatos de fuentes. Para asegurarse habría que importar todos los formatos:

Código :

```
src: url('Delicious-Roman.eot');
   src: local('?'), url('Delicious-
Roman.woff') format('woff'),
url('Delicious-Roman.ttf')
 format('truetype'), url('Delicious-
Roman.svg#webfont57ztNrX6')
 format('svg');
```

Consejo: *Existen aplicaciones que permiten exportar las tipografías de un formato a otro.*

Podéis ver el resultado en el siguiente ejemplo
(Testeado en Chrome 9)

Existen diferentes aplicaciones online que permiten además crear tipografías a partir de símbolos. Esto nos ayuda a disponer de iconos vectoriales y por tanto reescalables. Una de las propuestas en este sentido que me ha parecido más interesante es Icomoon.

Columnas en CSS3

Si queremos crear una estructura de varias columnas contamos con cuatro nuevos atributos. Al día que se escribe este curso estos atributos están únicamente incorporados de manera experimental en navegadores derivados de Mozilla (Firefox) y Webkit (Chrome y Safari), por lo que hay que añadir el prefijo -moz- o -webkit- según el caso. Os describo los atributos:

- *column-count*. Indica el número de columnas que queremos tener.
- *column-width*. Define el ancho de cada columna.
- *column-gap*. Define la separación entre columnas.
- *column-rule*. Crea una línea de separación entre las columnas.

Por ejemplo:

Código:

```
.ejemplo{
            -moz-column-count: 3;
            -webkit-column-count: 3;
            -moz-column-width: 100px;
            -moz-column-gap: 20px;
            -webkit-column-width: 100px;
            -webkit-column-gap: 20px;
/*column-rule se centra en el espacio
dedicado al gap*/
            -webkit-column-rule: 1px
solid #ccc;/*Observad que indicamos,
ancho, estilo y color.
            -moz-column-rule: 1px solid
#ccc;
        }
```

Podéis ver un ejemplo de multi-columna aquí.

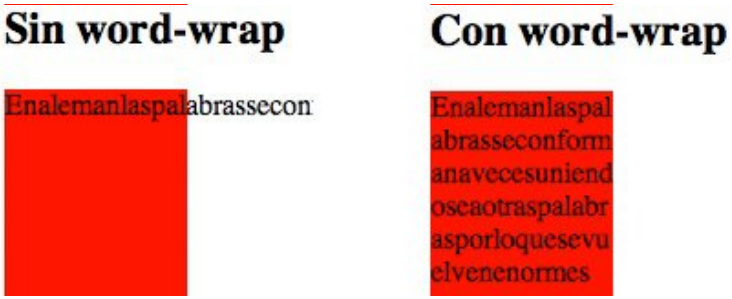
word-wrap

En ocasiones algunas palabras cuya longitud excede el ancho de de la capa que lo contiene, por lo que ignorando toda media, la muy hija de un diccionario se sale arruinando el diseño.

El atributo word-warp soluciona esto "rompiendo" la palabra y situando el resto en las filas inferiores. Personalmente me parece una solución a medias, ya que lo correcto sería la inclusión de un guión en la ruptura de la palabra, al igual que cuando escribimos en un editor de textos.

Código:

```
word-wrap:break-word; /*Rompe las palabras*/
word-wrap:normal; /*Se porta de la forma habitual*/
```



Os dejo un ejemplo.

Conclusiones

Aunque a la fecha de la edición de este tema ninguno de los atributos aquí descritos tiene una implementación fiable en el 100% de los navegadores, es cierto que se

ha notado un cambio de tendencias en el diseño web, con páginas que empiezan a basar su diseño en el atractivo de las tipografías.

Por último me gustaría recomendaros visitar e investigar Google Font Drectory.

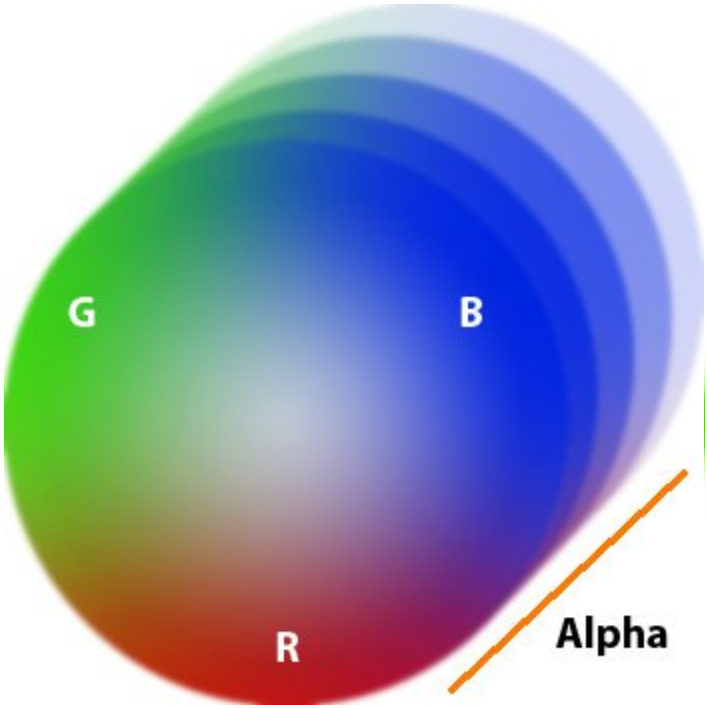
Tema 5. Color.

Tradicionalmente en CSS los valores de los colores se han expresado en valores RGB. CSS3 da una vuelta de tuerca a este sistema y añade por un lado el **canal alfa**, y por el otro los sistemas HSL y HSLA.

Colores RGBA

CSS3 nos permite añadir un **canal alfa** (de transparencia) al sistema de color RGB.

Esta nueva notación a la que llamaremos *rgba* permite gracias a su cuarto canal la posibilidad de indicar la opacidad en tantos por 1, es decir, tomando 0 como la máxima transparencia y 1 como la máxima opacidad.



Actualmente cuando queremos crear alguna capa con transparencia creamos una imagen png con la opacidad que deseemos y guardamos conservando el canal alfa. Después en el css definimos esa imagen de fondo y eventualmente le damos el valor *transparent*.

Código :

```
background:transparent url("fondo.png")
repeat;
```

Ahora solamente necesitamos definir el color con el modelo RGBA. Su uso es el mismo que*rgb(r,g,b);* pero añadiendo el cuarto valor:

Código:

```
body{
background-color:rgba(100,20,40,0.5);
}
```

Además este tipo de color actúa sobre cualquier elemento susceptible de atribución de color, como los textos.

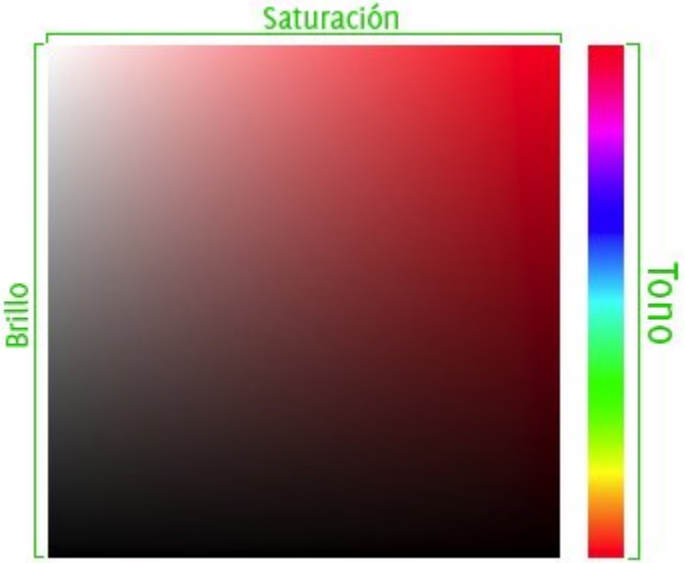
Este es un ejemplo de capas usando RGBA en css3maker.com

Para no perder las costumbres, Internet Explorer 8 no da soporte a este modelo por lo que para verlo tendremos que usar una versión 9 o superior del programa de Microsoft.

Aquí tenéis un ejemplo.

Colores HSL y HSLA

CSS3 añade además un nuevo modelo de color conocido como HSL. Estas siglas provienen del inglés *Hue, Saturation, Ligthness* o lo que es lo mismo, **tono, saturación y brillo**.



El modelo *hsl* es tal vez el más intuitivo de los modelos de color digitales

La sintaxis es la siguiente:

Código :

```
/*atributo:
hsl(tono,saturación,brillo);*/
background-color: hsl(360,100%,20%);
```

Nótese que el valor tono puede tomar valores del 0 al 360 donde:

- 0, sería el rojo.
- 120, sería el verde.
- 240, sería el azul.
- 360, volvería a ser rojo.

Además, como en el caso del modelo RGB, al HSL se le puede añadir un **canal alpha** para definir la transparencia del color. El resultado en código sería el siguiente, dando resultados análogos a RGBA:

Código :

```
hsla(300,130%,65%,10%);
```

Tema 6. Efectos de borde.

CSS3 trae una serie de novedades en el manejo de bordes de elementos. Tareas tan complicadas hasta ahora como crear una esquina redondeada o usar imágenes en los bordes serán ahora asombrosamente sencillos.

Bordes redondeados con border-radius

Hasta ahora era un auténtico quebradero de cabeza y una pérdida de tiempo crear un borde redondeado teniendo que utilizar imágenes de fondo o recurriendo a hacks. El atributo *border-radius* simplifica su implementación como vemos a continuación:

Código:
```
border-radius:15px;
```

Como resultado tendríamos una capa con **todas** las esquinas redondeadas.

> ***Consejo:*** *Como siempre, para optimizar la compatibilidad con los navegadores es conveniente usar los prefijos -webkit- y -moz-.*

Si quisiéramos usar diferentes radios en cada esquina la sintaxis shorthand sería:

Código:
```
border-radius:15px 0px 25px 0px;
```

La imagen superior usa un sólo radio para todas las esquinas, mientras que la inferior lo varía.

Importante: *Los valores dados al atributo border-radius se empiezan a aplicar* ***desde la esquina superior izquierda en sentido horario***

Para indicar el radio de cada esquina sin usar shorthands seguimos la siguiente sintaxis:

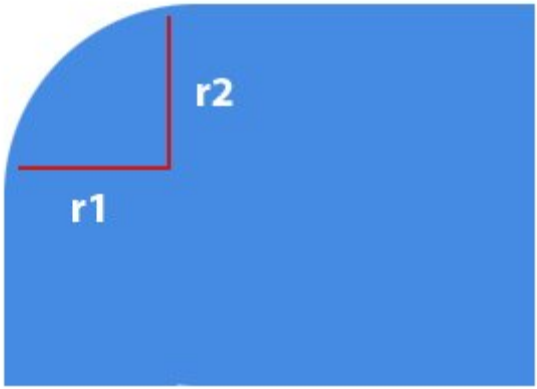
Código:
```
/*border-y-x-radius: valor;*/

border-top-left-radius: 15px;

/*Equivale a:*/

border-radius: 15px 0px 0px 0px;
```

Para darle una vuelta de tuerca, podemos indicar **dos radios** en lugar de uno de la manera que se muestra en la siguiente imagen:

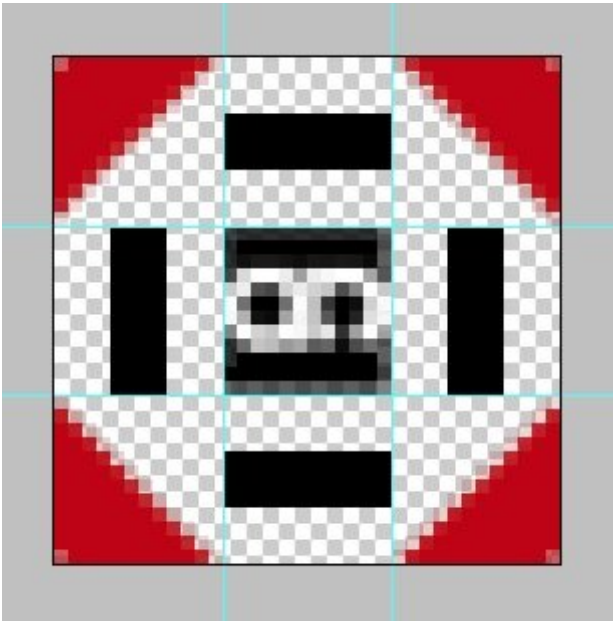


border-radius: r1 / r2;

Podéis ver un ejemplo del resultado en ambos casos.

Bordes con imágenes

El atributo *border-image* nos habilita para usar imágenes de fondo para los bordes de nuestros elementos.
Para entender el funcionamiento de este atributo, empezaremos por crear una imagen de prueba, en mi caso es esta:



Observad como las guías la dividen en una malla de 9 secciones, cada una de 12 x 12 px. No es obligatorio pero nos servirá para entender mejor el concepto.
La sintaxis a usar es la siguiente:

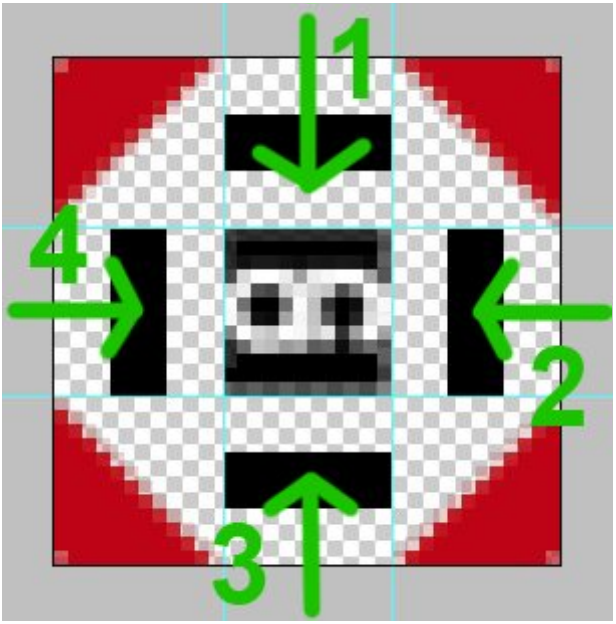
border-image: url("ruta de la imagen") arriba derecha abajo izquierda modificador modificador;

Código:

```
border-image: url("borde.png") 12 12 12 12 repeat repeat;
```

La url hace clara referencia a la imagen que vamos a usar de borde. Para entender perfectamente las cuatro medidas que damos, pensemos en las guías del dibujo.

Con la primera medida indicaríamos a que distancia estaría la guía horizontal superior, con la segunda la guía vertical derecha, con la tercera la guía horizontal inferior y con la cuarta la guía vertical izquierda. Como dicen que una imagen vale más que mil palabras:



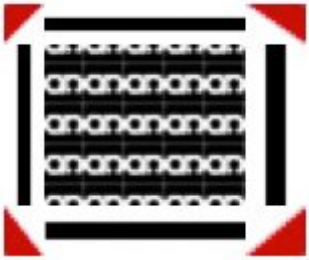
Lógicamente si sólo diésemos una medida, esta se aplicaría a todas las distancias y para las mismas podemos indicar cantidades porcentuales.
Los modificadores indican como actuar en caso de que el borde sea mayor que las medidas de la imagen.

Puede tomar tres valores:

- *stretch*. "Estira" cada uno de los sectores de la malla para adaptarlos a la distancia de borde a cubrir.

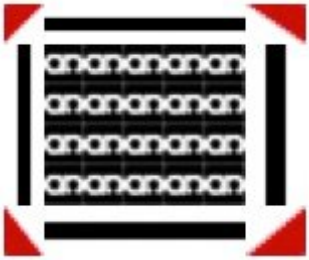
Fijaos en la zona central y como se ha estirado ocupando todo el espacio.

- *repeat*. Repite el patrón de cada sector de la malla a lo largo de la distancia de borde.



La imagen se repite, y en ocasiones se queda cortada.

- *round*. Repite el patrón de cada sector de la malla **modificando su escala** para adaptar las repeticiones a la distancia del borde.



La imagen se repite adaptándose para que no se corte.

Importante. *En chrome round reproduce el mismo efecto que repeat*

En este ejemplo veréis el resultado de cada uno.

Tema 7. Nuevas propiedades de fondo.

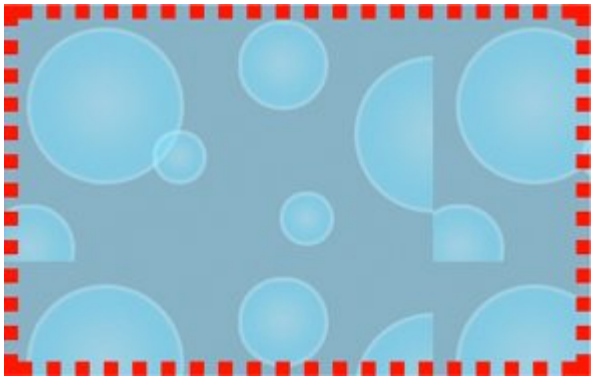
Veamos las nuevas propiedades *background-origin* y *background-clip*, así como la modificación que ha sufrido la propiedad *background* para permitir múltiples imágenes de fondo en un mismo elemento.

background-origin

> ***Importante:*** *A estas alturas supongo que sabéis lo que tengo que decir de la compatibilidad de Internet Explorer 8 con este atributo. Por si las dudas lo confirmo: no es compatible.*

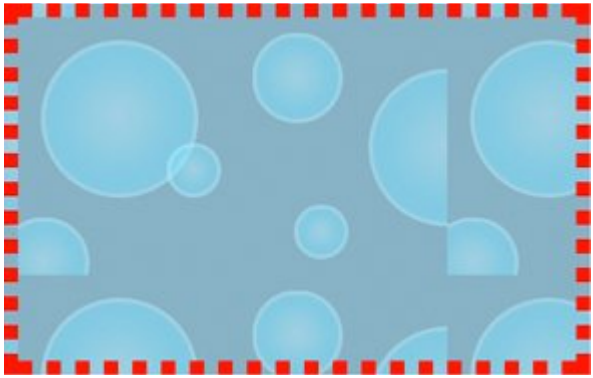
Podríamos decir que el atributo *background-origin* es una versión mejorada de *background-position* que nos permite definir el origen de coordenadas desde el cual posicionamos la imagen. Podemos darle tres valores:

- *border-box.* Toma como origen de coordenadas el borde del elemento.



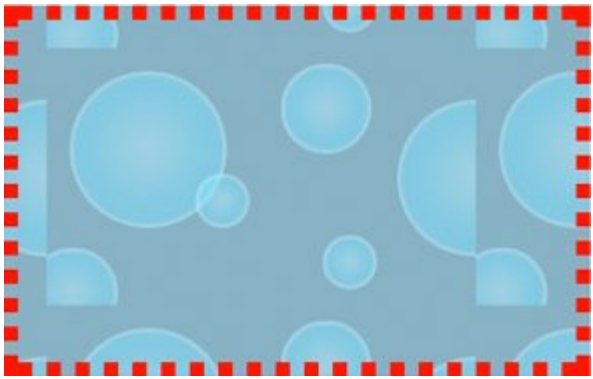
La imagen se incrusta al comienzo del border

- *padding-box.* Toma como origen de coordenadas el padding del elemento.



Esta vez se incrusta justo al final del borde, donde empieza el padding

- *content-box.* Toma como origen de coordenadas el punto donde comienza el contenido del elemento.



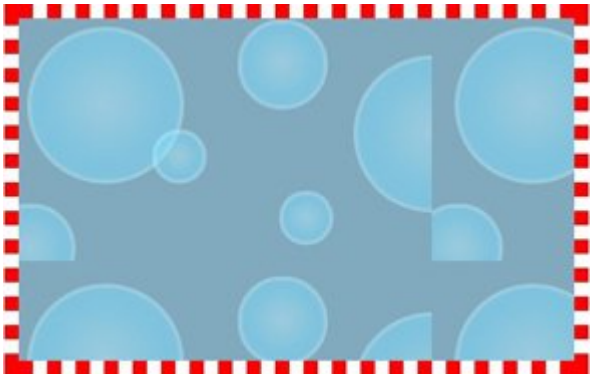
La imagen comienza al acabar el padding, justo donde empieza el contenido

Aquí tenéis una demostración.

background-clip

En ocasiones tendremos bordes con transparencias bajo los cuales no queremos que se muestre el fondo, aunque la propiedad *background-origin* tenga el valor *border-box*. Para estos casos contamos con el atributo *background-clip* que "enmascara" nuestro fondo tomando como referencia el borde, el padding o el contenido. Lo valores pueden ser:

- *border*. Es el valor por defecto. La imagen no sufre cambios.
- *padding*. Empieza a enmascarar a partir del inicio padding, por lo que no se mostrará fondo bajo nuestro borde.
- *content*. Enmascara a partir del contenido, por lo que no se muestra fondo en el borde ni en el padding.



Este es un ejemplo de como actúa el atributo con el valor *padding*. Aunque el fondo se empieza a aplicar al comienzo del borde, sólo se mostrará dentro del área de padding y content.

Aquí tenéis la demo de la imagen.

Múltiples fondos en CSS3

Hasta ahora en CSS2 el uso de varios fondos era algo complejo que pasaba por superponer varias capas aplicando a cada una de ellas un fondo distinto. CSS3 permite asignar a un mismo elemento varios fondos mediante el atributo *background-image* por lo que este trabajo se facilita enormemente.

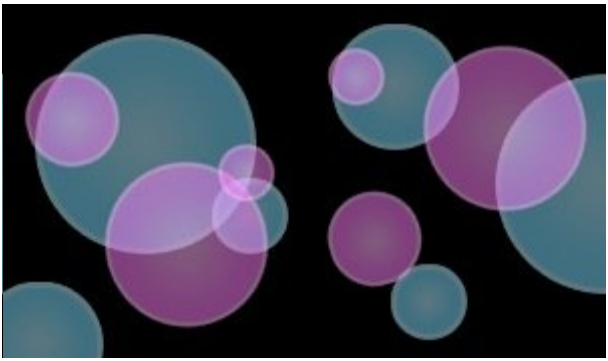
La sintaxis para asignar varios fondos es la siguiente:

Código:

```
background-image:url("imagen"),
url("imagen");
```

Como podéis ver, simplemente añadimos una coma al valor dado para incluir uno nuevo.

Si tomásemos dos imágenes como las de arriba y aplicásemos *background-image* de la manera citada obtendríamos este resultado:



Código:

```
background-image:url("bg2.png"),
url("bg1.png");
```

Como es habitual, os dejo un ejemplo.

Tema 8. Efectos de sombra y brillo.

En este tema veremos el uso de *box-shadow* y *text-shadow* con los que podremos crear sombras y brillos. Además como regalo os dejo una aplicación práctica: Textos 3D.

box-shadow

Hasta ahora aplicar un efecto de sombra a cualquier elemento de nuestro html era un proceso entretenido donde teníamos que cargar imágenes creadas previamente en nuestro programa de edición de imágenes favorito, recortarlas, ajustarlas, etc.

Con el atributo *box-shadow* podemos aplicar sombras a nuestras capas con pasmosa facilidad. La sintaxis de *box-shadow* es la siguiente:

Código:

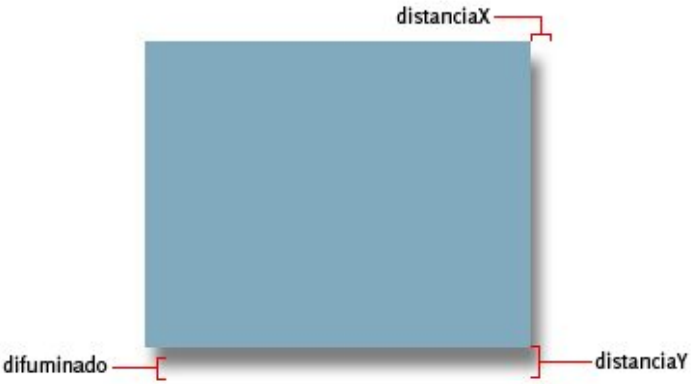
```
box-shadow: distanciaX distanciaY
difuminado color;
```

Aplicado de la siguiente forma:

Código:

```
box-shadow: 5px 10px 7px rgba(0,0,0,0.5);
```

El resultado es el que podéis ver en la siguiente imagen:



Notad que el difuminado se mide **de fuera hacia dentro**

Si quisiéramos añadir varias sombras, sólo tendríamos que añadir un nuevo valor al atributo usando una coma como separador:

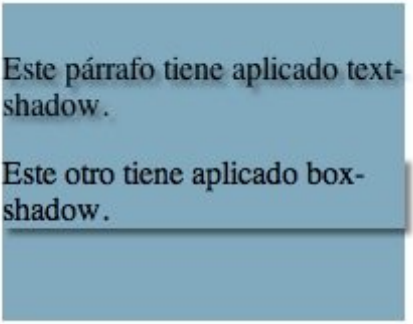
Código:

```
box-shadow: 5px 10px 7px rgba(0,0,0,0.5),
-5px -10px 7px rgba(0,0,0,0.5);
```

Fácil ¿verdad? Podéis ver el ejemplo aquí.

text-shadow

El atributo *text-shadow* se incluyó inicialmente en el CSS2 pero fué eliminado en CSS2.1. Por suerte para todos CSS3 vuelve a incluirlo. Seguro que te preguntas en qué se diferencia de*box-shadow*. Mientras que *box-shadow* crea sombras a en forma de caja, *text-shadow* realiza una sombra ajustada a los propios caracteres de un texto. En la siguiente imagen podéis ver la diferencia de los atributos aplicados a textos:



box-shadow se aplica al párrafo como contenedor, no al contenido.

La sintaxis de *text-shadow* es igual a la de *box-shadow*:

Código:

```
text-shadow: distanciaX distanciaY
difuminado color;
```

Para asignar varias sombras usamos la coma como separador. Por ejemplo:

Código:

```
text-shadow: 5px 10px 7px
rgba(0,0,0,0.5), -5px -10px 7px
rgba(0,0,0,0.5);
```

Aquí tenéis la demo.

Brillo

Si queremos crear un resplandor en vez de una sombra, tenemos que darle valor 0 a ambas distancias y aplicar un color claro.

Código:

```
text-shadow: 0px 0px 20px
rgba(0,255,0,1);
```

Si queremos aumentar la intensidad del brillo, le añadimos más valores de sombra.

Código:

```
text-shadow:
0px 0px 20px rgba(0,255,0,1),
0px 0px 20px rgba(0,255,0,1),
```

```
0px 0px 20px rgba(0,255,0,1);
```

Textos en 3D con text-shadow

Veamos una aplicación práctica de *text-shadow* que dará como resultado un efecto distinto a una simple dombra paralela.

Sabemos que podemos aplicar diferentes sombras a un texto, y que podemos decidir que el difuminado de la sombra sea 0. Partiendo de esta base, podemos crear un efecto de pseudo 3D a cualquier texto.

Primero tenemos que aplicar el efecto sombra a nuestro texto. Creamos un texto blanco y una sombra a una distancia *x* e *y* de un color gris claro y con difuminado 0:

Código:

```
text-shadow: 1px 1px 0px
rgba(230,230,230,1);
```

Ahora mismo tiene poco de 3D. Probemos a ir añadiéndole unas cuantas sombras más, aumentando en cada una 1 pixel las distancias *x* e *y* y haciendo el gris más oscuro:

Código :

```
text-shadow:
1px 1px 0px rgba(230,230,230,1),
2px 2px 0px rgba(200,200,200,1),
3px 3px 0px rgba(180,180,180,1),
4px 4px 0px rgba(160,160,160,1);
```

Eso ya tiene algo más de aspecto 3D, pero aun le podemos dar un aspecto más realista. Aplicaremos una nueva sombra siguiendo el método anterior, con la diferencia que esta vez le daremos color negro. Por último añadimos una sombra mayor que todas las demás, de color gris claro y con difuminado:

Código:

```
text-shadow:
1px 1px 0px rgba(230,230,230,1),
2px 2px 0px rgba(200,200,200,1),
3px 3px 0px rgba(180,180,180,1),
4px 4px 0px rgba(160,160,160,1),
/*Añadimos*/
5px 5px 0px rgba(0,0,0,1),
8px 8px 20px rgba(0,0,0,0.5);
```

Esta vez el resultado merece la pena:

Me río yo del VRay, 3DStudio y Papervision

Os dejo este ejemplo con cariño.

Tema 9. Transformaciones.

Una de las aportaciones de CSS3 que en mi opinión añaden más dinamismo al diseño son las **transformaciones**. A partir de ahora podremos modificar la **rotación, inclinación o escala** de nuestros elementos.

transform en CSS3

El atributo *transform* nos permite, como su propio nombre indica, transformar un elemento. Su sintaxis es la siguiente:

Código :

```
transform: tipo(cantidad);
```

El valor tipo puede tomar cuatro valores, y cada uno de ellos realiza una función diferente:

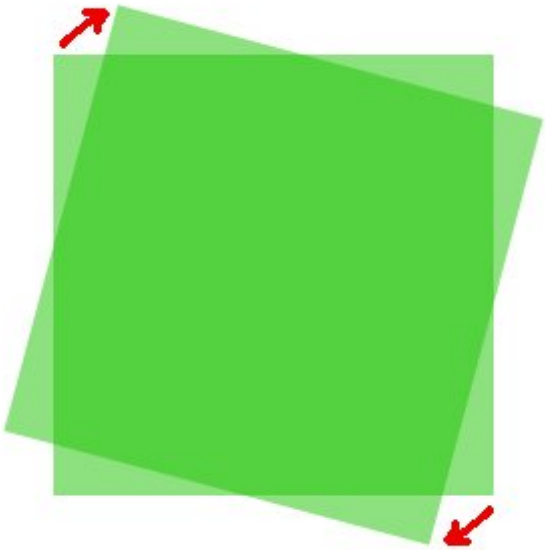
- *rotate*. Nos permite girar los elementos un número de grados. La sintaxis es:

Código :

```
transform: rotate(25deg);
```

Importante: *He notado que las transformaciones aplicadas a textos no funcionan bien con* ***-webkit-***

. Como solución podemos aplicarlas a las capas contenedoras. En ***-moz-*** *sí se puede aplicar directamente sobre el texto sin problemas.*

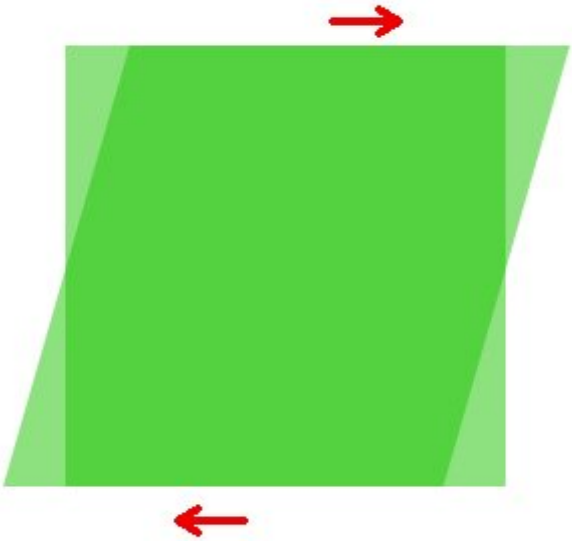


El valor que demos al giro se aplicará en sentido horario

- *skew*. Podemos inclinar un elemento tanto en coordenadas *X* como *Y*. El valor se expresa en grados y la sintaxis es la siguiente:

Código :

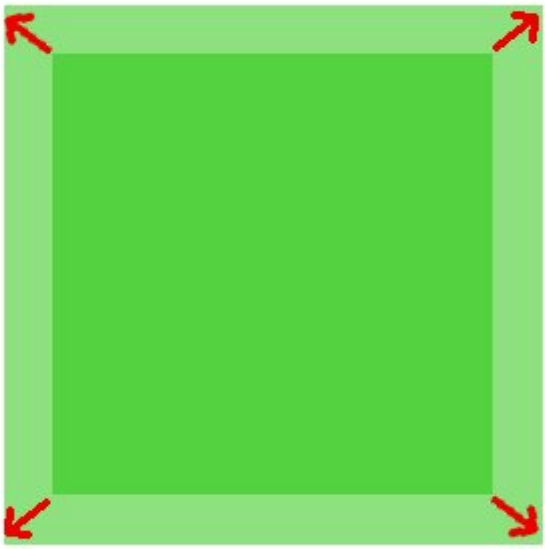
```
/*transform: skew(gradosX, gradosY);*/
transform: skew(15deg, 3deg);
```



- *scale*. Con este tipo podremos escalar nuestro elemento tanto en *X* como en *Y* en una cantidad expresada en tantos por uno:

Código:

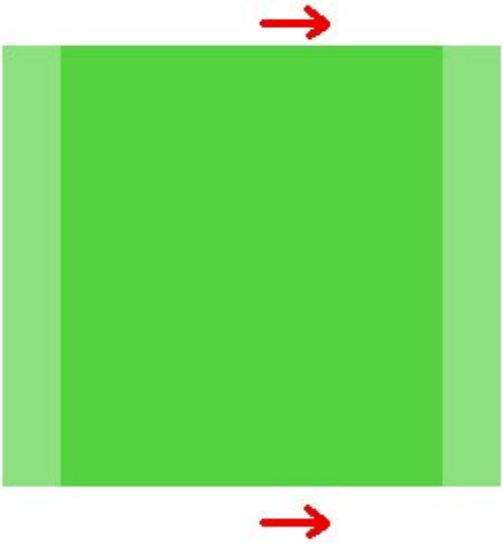
```
/*transform: scale(escalaX,escalaY);*/
transform: scale(1.5,0.6);*/
```



- *translate*. Podemos desplazar el elemento tanto en *X* como en *Y*.

Código:

```
/*transform: translate(desplazamientoX, desplazamientoY);*/
transform:translate(12px, 19px);
```



Se pueden aplicar diferentes transformaciones a un mismo elemento simplemente escribiéndolas de manera consecutiva:

Código:

```
transform: scale(1.6) skew(10deg)
translate(5px) rotate(12deg);
```

Os dejo un ejemplo sencillo de transformaciones en este link.

Tema 10. Transiciones.

Actualmente CSS3 permite la posibilidad de cambiar el estilo de un elemento mediante una transición animada. Este es tal vez el aspecto que me ha resultado más atractivo de todo CSS3 y es curioso que cuando redacté este mismo curso en Cristalab no escribí nada acerca de ellas. Sin embargo es una de las características con mayor potencial y más efectista de cuantas hayamos revisado hasta ahora.

Existen 4 propiedades base para crear transiciones:

- transition-property: Define la propiedad que vamos a animar mediante la transición.

- transition-duration: La duración en segundos de la animación.

- transition-timing-function: La función matemática aplicada a la animación. Puede tomar 5 valores predefinidos (*linear*, *ease*, *ease-in*, *ease-out* o *ease-in-out*) o *cubic-bezier(v, v, v, v)* para definir tus propias funciones. Su valor por defecto es *ease*.

- transition-delay: Indica el retardo en segundos. El valor por defecto es 0.

También se puede aplicar el método *shorthand*:

Independientemente del orden:

transition: [propiedad] [duración] [función] [retardo]

Por ejemplo

Código:

```
div{
	width: 100px;
	transition: width 1s ease-in 0.2s;
	-o-transition: width 1s ease-in
0.2s;
	-moz-transition: width 1s ease-in
0.2s;
	-webkit-transition: width 1s ease-
in 0.2s;
}

div:hover{
	width: 500px;
	transition: width 1s ease-in 0.2s;
	-o-transition: width 1s ease-in
0.2s;
	-moz-transition: width 1s ease-in
0.2s;
	-webkit-transition: width 1s ease-
in 0.2s;
}
```

En este ejemplo modificamos el ancho de una capa durante un segundo empezando 0.2 segundos después de hacer *hover* sobre el div.

El caso es que podemos aplicar transiciones a varias propiedades a la vez, simplemente separando las sentencias por comas:

Código:

```
transition: width 1s, height 1s;
```

En el caso de que fuesen todas las propiedades las que queramos cambiar mediante un efecto de transición, podemos indicar como variable propiedad el valor *all* que las engloba todas.

```
transition: all 1s ease-out;
```

Tema 11. Animaciones CSS.

Tradicionalmente cuando queríamos incluir cualquier tipo de animación en nuestras páginas webs teníamos que recurrir a tecnologías como Flash o JavaScript. Si bien la potencia de los dos citados es descomunal en comparación, las nuevas funciones de animación de CSS3 abren una nueva puerta a nuestra imaginación, sin tener que depender de tecnologías extras.

> ***Importante:*** *En la fecha en la que se realiza este curso, la compatibilidad de las animaciones se limita a los navegadores basados en webkit como Chrome o Safari.*

Fotogramas claves de las animaciones en CSS3

Para aquellos que son *flasheros* el concepto de **fotograma clave** no necesita explicación. Si no te encuentras entre los adeptos del programa de Adobe, es importante que entiendas bien este concepto antes de continuar.

Un fotograma clave no es más que un **punto destacado en el tiempo de nuestra animación**. Cualquier animación consta al menos de dos fotogramas claves: el punto inicial y el punto final. Imaginad que nuestra animación es como una

carretera:

No se en vuestra ciudad, pero en la mía los semáforos son verdes

El primer semáforo actuaría como **fotograma clave** inicial y el segundo como el final. Entre uno y otro se produciría nuestra animación, que no es más que el desplazamiento del coche hacia la derecha.

@keyframes

En CSS3 creamos animaciones completas mediante *@keyframes*, que son un conjunto de fotogramas clave. Su sintaxis es la siguiente:

Código:

```
/*@keyframe nombreAnimacion{
     puntoDelKeyframe{
          atributosIniciales;
     }
     puntoDelKeyframe{
          nuevosAtributos;
     }

     .......

     puntoDelKeyframe{
         últimosAtributos;
     }*/
}
```

Un poco extraño ¿verdad? Tranquilo, es más fácil de lo que parece. Veamos que tendríamos que hacer para desplazar nuestro coche a la derecha:

Código:

```
@keyframe animacionCoche{
     /*Indicamos que salimos de la
posición 0*/
      from{
           left:0px;
     }
     /*Indicamos que al final la posición
debe ser 350*/
     to{
           left:350px;
     }
}
```

Podemos crear animaciones más complejas estableciendo fotogramas claves intermedios mediante porcentajes:

Código:

```
@keyframe animacionCoche{
      from{
           left:0px;
     }
    /*Hasta el 65% de la reproducción
solo queremos que se desplace 10 pixels*/
     65%{
           left:10px;
     }
     to{
```

```
            left:350px;
      }
}
```

Aplicando la animación a un elemento

Hemos creado nuestra animación. Para aplicarla sobre un elemento de nuestra página, deberemos acudir al mismo y agregarle el atributo de animación:

Código:

```
#coche{
      animation-name: animacionCoche;
      animation-duration: 3s;
      animation-iteration-count: 1;
      position:relative;
}
```

> ***Importante:*** *Fijaos que el atributo position tiene valor* ***relative***

En este enlace podéis ver una animación que convierte varios atributos como la posición y el ancho de un objeto.

Hay tres nuevos atributos que no habíamos visto antes. Estos y otros tantos conforman los atributos destinados a la animación que describo a continuación:

- *animation-name*. Indica a que animación responde nuestro elemento. Es posible definir más de una animación usando comas como separador.

- *animation-duration*. Define la duración en segundos de nuestra animación.

- *animation-iteration-count*. Define cuantas veces se reproduce la animación. Podemos darle el valor *infinite* para que se reproduzca hasta el fin de los tiempos.

- *animation-direction*. Por defecto nuestra animación se reproducirá hacia delante, como es lógico. Sin embargo si le damos el valor *alternate*, al reproducirse completamente la animación esta volverá a reproducirse en sentido opuesto, es decir, como si se rebobinase.

- *animation-delay*. Indica si se produce un retardo en el inicio de la animación.

- *animation*. El shorthand de todos los anteriores.

- *animation-timing-function*. Define como progresa la animación entre los keyframes mediante ecuaciones matemáticas.

animation-timing-function

Es posible que la definición que acabas de leer sobre el atributo *animation-timing-function* te ha dejado pensando *WTF*. este atributo sirve para aplicar un efecto de suavizado a la animación.

Por defecto responde al valor *ease*, pero puede responder a diferentes valores:

- *ease*

- *linear*
- *ease-in*
- *ease-out*
- *ease-in-out*

Es terriblemente complejo describir estos efectos con palabras, así que lo mejor es que veáis estos ejemplos.