GUÍA COMPLETA DE CSS

Por: Geovanny Quilarque

Con CSS 3, más control sobre la forma

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 aún continúan usando trucos diversos para conseguir efectos tan comunes o tan deseados como los bordes redondeados o el sombreado de elementos en la página.

CSS 1 ya significó un avance considerable a la hora de diseñar páginas web, aportando mucho mayor control de los elementos de la página. Pero como todavía quedaron muchas otras cosas que los diseñadores deseaban hacer, pero que CSS no permitía especificar, éstos debían hacer uso de trucos para el diseño. Lo peor de esos trucos es que muchas veces implica alterar el contenido de la página para incorporar nuevas etiquetas HTML que permitan aplicar estilos de una manera más elaborada. Dada la necesidad de cambiar el contenido, para alterar al diseño y hacer cosas que CSS no permitía, se estaba dando al traste con alguno de los objetivos para los que CSS fue creado, que era el separar por completo el contenido de la forma.

CSS 2 incorporó algunas novedades interesantes, que hoy ya utilizamos habitualmente, pero CSS 3 todavía avanza un poco más en la dirección, de aportar más control sobre los elementos de la página.

Así pues, la novedad más importante que aporta CSS 3, de cara a los desarrolladores de webs, consiste en la incorporación de nuevos mecanismos para mantener un mayor control sobre el estilo con el que se muestran los elementos de las páginas, sin tener que recurrir a trucos o hacks, que a menudo complicaban el código de las web.

Propiedades nuevas en CSS 3

He aquí una lista de las principales propiedades que son novedad en CSS3.

Bordes

- border-color
- border-image
- border-radius
- box-shadow

Fondos

- background-origin
- background-clip
- background-size
- hacer capas con múltiples imágenes de fondo

Color

- colores HSL
- colores HSLA
- colores RGBA
- Opacidad

Texto

- text-shadow
- text-overflow
- Rotura de palabras largas
- Web Fonts

Interfaz

- box-sizing
- resize
- outline
- nav-top, nav-right, nav-bottom, nav-left

Selectores

- Selectores por atributos

Modelo de caja básico

- overflow-x, overflow-y

Degradados CSS3

- Degradados lineales
- Degradados radiales
- Degradados lineales de repetición
- Degradados radiales de repetición

Otros

- media queries
- creación de múltiples columnas de texto
- propiedades orientadas a discurso o lectura automática de páginas web
- animaciones CSS3

Explicaciones y test de nuevas características CSS 3

Artículos prácticos con explicaciones de las nuevas características de la especificación CSS 3, útiles para hacer todo tipo de efectos gráficos de las webs modernas.

Bordes redondeados en CSS 3

Las características de CSS 3 incluyen bordes redondeados, a través del atributo border-radius, que define la curvatura que debe tener el borde del elemento.

CSS 3 incorpora nuevas propiedades para el control de bordes de los elementos. Ahora se permiten bordes con las esquinas redondeadas, bordes con imágenes (incluso varias imágenes se pueden utilizar para definir el aspecto del borde), sombras, etc.

Tenemos la propiedad border-radius, que permite definir bordes redondeados en las esquinas, especificando las medidas del radio que deben darse a la curva de las esquinas. Su uso sería aproximado al que vemos a continuación:

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

Definiría un radio de 5 píxeles en el redondeo de las esquinas del elemento. Por el momento Mozilla ha adoptado este atributo con un nombre especial, que es válido para productos como Firefox, mientras que las especificaciones de CSS3 no hayan alcanzado el estado "Candidate Recommendation", que es cuando se supone que los distintos navegadores deben implementarlas. El nombre del atributo por el momento es:

```
-moz-border-radius
```

Los navegadores basados en WebKit, como Google Chrome o Safari, también soportan las esquinas redondeadas de CSS 3, pero el atributo border-radius tampoco funciona directamente, como en el caso de Firefox, sino que hay que utilizar un "alias":

```
-webkit-border-radius
```

Por el momento, para navegadores Mozilla y WebKit que son los primeros en adaptarse a CSS3, podemos utilizar el atributo border-radius de la siguiente manera:

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

Con esto conseguimos que todos los div tengan un borde redondeado en las esquinas de radio de 7 píxeles. Fijarse en el uso de los atributos -moz-border-radius y -webkit-border-radius, que sirven para lo mismo, pero en navegadores basados en Mozilla y basados en WebKit.

Pero el atributo border-radius tiene otras posibles configuraciones, en la que se pueden definir los valores para el radio de las cuatro esquinas por separado. De esta manera:

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

Así estaríamos definiendo un borde redondeado con radio de 7 pixel para la esquina superior izquierda, luego 27px para la esquina superior derecha, de 100px para la inferior derecha y 0px para la inferior izquierda. (Hay que explicar que un border-radius de 0px es un borde con esquina en ángulo recto)

Los bordes redondeados con CSS 3, como se podrá imaginar, sólo se ven si se tiene definido algún borde visible al elemento que se los asignamos, ya sea solid, dotted, etc. Eso es lo que definen las especificaciones de CSS3, aunque en el momento de escribir el artículo debo señalar que incluso Mozilla o Firefox (el único que por ahora soporta este atributo de CSS3) no funciona del todo correctamente con esto y sólo muestra los bordes redondeados con solid.

Múltiples imágenes de fondo con CSS

Cómo conseguir que un elemento de la página tenga varias imágenes de fondo a la vez, con CSS básico y con características de CSS 3.

Con el atributo background-image podemos conseguir que un elemento de la página tenga un fondo de imagen. Esto debemos saberlo, puesto que es algo básico de las hojas de estilo en cascada (CSS).

Una de las nuevas características de CSS 3 consiste en la posibilidad de declarar varios fondos de imagen a un elemento de la página. Lo que antes hemos visto que es posible, creando varios elementos anidados y colocando un fondo en cada uno, se puede hacer en CSS 3 con un solo elemento, al que aplicaremos varios fondos distintos.

El HTML del ejemplo de varias imágenes de fondo sería el siguiente:

```
<div id="fondos">
   texto de un único elemento
   ...
</div>
```

Ahora veamos el CSS 3 válido para este ejemplo:

```
<style type="text/css">
#fondos{
      background: url(fondo3.png) bottom right no-repeat,
      url(fondo2.png) center no-repeat,
      url(fondo1.gif) center repeat;
      width: 300px;
}
</style>
```

Sólo cabe comentar que las distintas imágenes de fondo se tienen que escribir en la declaración CSS separadas por comas. Además, las imágenes que declaramos se van colocando de modo que la primera aparece sobre las siguientes. Así pues, en esta declaración, fondo1.gif, que está colocada como último fondo, es la que aparece detrás del todo.

Colores RGBA en CSS 3

Veremos qué son los colores RGBA y su notación, que se incluyen en la especificación de Hojas de Estilo en Cascada CSS 3.

Como sabemos, los colores en HTML se expresan en valores RGB, igual que en CSS, que admite diversas notaciones para definir el color, a través de números en hexadecimal e incluso en decimal. Todo esto suponemos no será un misterio para los lectores, que sin duda habrán experimentado con CSS y probablemente estén familiarizados con las distintas notaciones para especificar color en las hojas de estilo.

Ahora queremos hablar de una nueva notación, que no es simplemente una manera nueva de expresar lo mismo, sino una que nos permite definir colores por medio de valores adicionales. Se trata de la notación RGBA, que a partir de CSS 3 está disponible para los desarrolladores.

La notación RGBA es una manera de especificar colores en la que se definen cuatro valores. Los tres primeros son los bien conocidos canales RGB (rojo, verde y azul) y el cuarto parámetro es el canal Alpha, que no es más que el grado de transparencia u opacidad del color. El canal Alpha es un valor entre cero y uno, siendo 0 totalmente transparente y 1 totalmente opaco.

En el mundo del diseño quizás ya habremos visto otros formatos o sistemas que soportan colores con canal Alpha y por ello puede que estemos familiarizados con este parámetro. El formato de imagen PNG, que tanto nos gusta por soportar transparencia que se ve correctamente en todos los fondos posibles, implementa justamente este canal alpha en la definición del color para conseguir una transparencia ideal.

Ahora, por medio de los colores en RGBA en CSS 3, podremos aplicar nuevas transparencias a los colores que especificamos con CSS, abriendo nuevas posibilidades a los diseñadores sin necesidad de complicarse con pequeños trucos como el uso de imágenes de fondo semitransparentes en PNG, etc. Además, como los colores RGBA se pueden aplicar a cualquier elemento que soporte asignación de color, las aplicaciones aumentan todavía más. El único pero, al menos a la hora de escribir este artículo, es que CSS 3 no está ampliamente soportado por todos los navegadores. Por ejemplo Internet Explorer 8 no lo soporta por el momento.

Notación de color RGBA

Para definir un color RGBA, se deben especificar cuatro valores, de la siguiente manera:

```
rgba(255, 125, 0, 0.5);
```

Los tres primeros valores son números en sistema decimal, que corresponden con los valores de rojo, verde y azul. Siempre tienen que ser números entre 0 y 255.

El cuarto valor es un número entre 0 y 1. Por ejemplo 0 sería totalmente transparente, 1 sería totalmente opaco y 0.5 sería una transparencia al 50%, es decir, mitad opaco mitad transparente.

Ejemplos de estilos CSS con colores definidos por RGBA

Ahora veamos varios ejemplos de colores definidos con CSS y la notación RGBA.

```
<div style="background-color: rgba(0, 0, 255, 0.1);">Esta capa tiene fondo azul, casi transparente</div>

<span style="color: rgba(0,255,0,0.8);">Este texto es verde y tiene un poco de transparencia</span>
```

Código de ejemplo:

```
<html>
<head>
   <title>Colores RGBA con CSS 3</title>
<style type="text/css">
div.cuadrado{
   width: 150px;
   height: 40px;
   border: 1px solid #dddddd;
   margin: 5px;
}
div.textogrande{
   font-family: verdana, arial, helvetica;
   font-weight: bold;
   font-size: 40pt;
}
</style>
</head>

<body>
<h1>Colores RGBA con CSS 3</h1>

<h2>Ejemplo de capas con fondo azul y varias transparencias</h2>
<div class="cuadrado" style="background-color: rgba(0, 0, 255, 0.1);"></div>
<div class="cuadrado" style="background-color: rgba(0, 0, 255, 0.4);"></div>
<div class="cuadrado" style="background-color: rgba(0, 0, 255, 0.7);"></div>
<div class="cuadrado" style="background-color: rgba(0, 0, 255, 1);"></div>

<h2>Ejemplo de capas con fondo verde y varias transparencias, sobre una capa con fondo amarillo</h2>
<div style="background-color: #ff3; padding: 10px;">
<div class="cuadrado" style="background-color: rgba(0, 255, 0, 0.1);"></div>
<div class="cuadrado" style="background-color: rgba(0, 255, 0, 0.4);"></div>
<div class="cuadrado" style="background-color: rgba(0, 255, 0, 0.7);"></div>
<div class="cuadrado" style="background-color: rgba(0, 255, 0, 1);"></div>
</div>

<h2>Ejemplo de capas con fondo naranja y varias transparencias, sobre una capa con una imagen de fondo</h2>
```

```
<div style="background-image:
url(http://www.desarrolloweb.com/articulos/ejemplos/photoshop/fondo-nieve/nieve.gif);
padding: 10px;">
<div class="cuadrado" style="background-color: rgba(255, 125, 0, 0.1);"></div>
<div class="cuadrado" style="background-color: rgba(255, 125, 0, 0.4);"></div>
<div class="cuadrado" style="background-color: rgba(255, 125, 0, 0.7);"></div>
<div class="cuadrado" style="background-color: rgba(255, 125, 0, 1);"></div>
</div>

<h2>Ejemplo de texto de color rojo y varias transparencias, sobre una capa con una
imagen de fondo</h2>
<div style="background-image:
url(http://www.desarrolloweb.com/articulos/ejemplos/photoshop/fondo-nieve/nieve.gif);
padding: 10px;">
<div class="textogrande" style="color: rgba(200, 0, 0, 0.3);">Este texto está para que
se vea que puede ser también medio transparente</div>
<div class="textogrande" style="color: rgba(200, 0, 0, 0.7);">Este texto está para que
se vea que puede ser también medio transparente</div>
</div>

</body>
</html>
```

Word-wrap en CSS 3

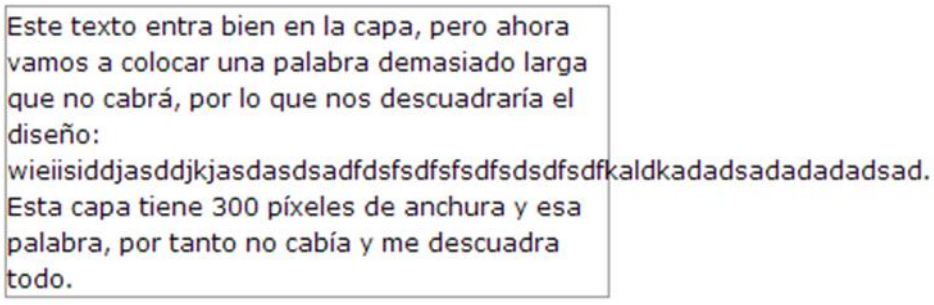
Una propiedad de CSS 3 que sirve para romper las palabras que son demasiado largas y no caben enteras por la anchura de una caja.

El atributo word-wrap tiene dos posibles valores: normal o break-word.

```
word-wrap: normal;
```

Hace que las palabras no se corten, lo que sería el comportamiento normal que conocíamos hasta ahora, ocurriendo que las palabras largas nos puedan descuadrar nuestro diseño.

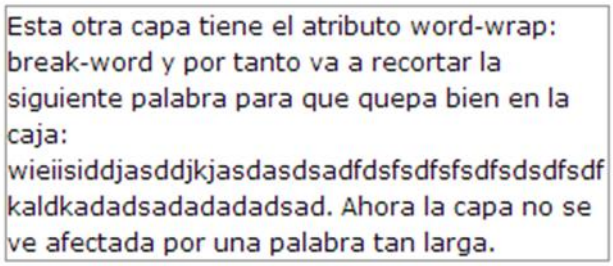
Ahora podemos ver una caja que tenía una anchura de 300 px y que por culpa de una palabra muy larga se deforma la caja o el texto aparece por fuera.




```
word-wrap: break-word;
```

Con este otro valor de word-wrap: break-word, lo que conseguimos es que la palabra se recorte, para caber en el ancho que habíamos definido.

Ahora veamos una caja donde hemos colocado el atributo para que recorte las palabras:



Textos multi-columna con CSS 3

CSS 3 incorpora nuevos atributos para que el navegador se encargue de producir texto multicolumna, es decir, que maquete directamente el texto en varias columnas sin tener que hacer nosotros nada.

Para crear una estructura multi-columna utilizaremos varios atributos, que servirán para modelizar las columnas:

colum-width: servirá para definir la anchura de las distintas columnas a crear.

column-gap: nos permitirá definir el espacio en blanco entre columnas.

column-rule: servirá para crear un filete o línea divisoria entre las columnas.

Estas etiquetas por el momento no las soporta tal cual ningún navegador. Sin embargo Safari, Google Chrome y Firefox ya implementan el multicolumna, de manera experimental y hasta que las especificaciones de CSS 3 estén dispuestas para incorporar en los navegadores. Para ello, podemos utilizarlas si les ponemos un prefijo, que sería "-moz-" en el caso de Firefox y "-webkit-" para el navegador Safari o Chrome.

Un ejemplo de multicolumna, para que funcione en estos navegadores sería:

```
.multicolumna{
   -moz-column-width: 15em;
   -moz-column-gap: 15px;
   -webkit-column-width: 15em;
   -webkit-column-gap: 15px;
   -webkit-column-rule: 1px solid #ccc;
   -moz-column-rule: 1px solid #ccc;
}
```

Otra posibilidad para crear un multicolumna será definir simplemente el número de columnas que querríamos incorporar en el texto, por medio del atributo column-count, de esta manera:

```
.multicolumna5columnas{
   -moz-column-count: 5;
   -moz-column-gap: 2em;
```

```
   -moz-column-rule: 1px solid #ccf;
   -webkit-column-count: 5;
   -webkit-column-gap: 2em;
   -webkit-column-rule: 1px solid #ccf;
}
```

Especificar el número de columnas es quizás más cómodo, pero en páginas fluidas puede funcionar peor, porque no se sepa con certeza qué anchura va a tener el lugar donde se muestran los textos y por tanto unas veces queden columnas muy anchas y otras muy estrechas.

El sistema de múltiples columnas se está encuentra en estado de borrador y forma parte de un módulo aparte dentro de las especificaciones de CSS 3. Si se desea encontrar más información sobre el tema en la W3C se puede consultar la URL http://www.w3.org/TR/css3-multicol/

Bordes con imágenes en CSS 3

El atributo border-image y varios otros de CSS 3 harán posible la utilización de imágenes como bordes de los elementos de la página, sin código HTML especial, simplemente con hojas de estilo.

Distintas especificaciones sobre border-image

Las especificaciones de CSS 3 están en etapa de desarrollo. El organismo W3C, que se encarga de definir los estándares de Hojas de Estilo en Cascada, ha alterado algunas veces las especificaciones del atributo border-image y relacionados. Es por ello que todavía hay algunas diferencias entre lo que los navegadores entienden con este atributo y lo que recomienda el W3C. Esto quiere decir que en el futuro todavía puede cambiar el funcionamiento de este atributo en los distintos navegadores, tal como anuncia Mozilla en su centro de desarrollo.

Ejemplo de border-image

Por ejemplo, tendríamos este elemento HTML:

```
<div id="capaborde">
Esta capa le voy a poner un borde arriba
</div>
```

Y ahora podríamos aplicar estilos para crear un borde en la imagen:

```
#capaborde{
-moz-border-image: url(sello.png) 2 2 2 2 stretch stretch;
-webkit-border-image: url(sello.png) 2 2 2 2 stretch stretch;
padding:20px;
width: 100px;
}
```

Como se puede ver, los atributos para bordes de imágenes no tienen el nombre definitivo, que será border-image, sino unos propios para cada uno de los dos navegadores que implementan esta nueva característica de CSS 3. El atributo -moz-border-image es para el navegador Firefox y

otros productos de la compañía Mozilla y el atributo -webkit-border-image es para cualquier navegador basado en WebKit, como Safari o Chrome.

La imágen que estamos utilizando como borde es la siguiente:

Otros atributos para hacer borde con imágenes

A parte del atributo border-image, existen otros numerosos atributos para definir los bordes de manera independiente para cada uno de los lados o vértices de un elemento HTML. Quizás conviene esperar un poco antes de dar unas explicaciones concisas y ejemplos sobre este y otros atributos, puesto que han cambiado bastante últimamente.

Sin embargo, según la última especificación de CSS 3, tenemos los siguientes atributos:

border-image-source: Para indicar la URL de la imagen que vamos a utilizar como borde.

border-image-slice: Indica el espacio de la imagen que será visible como borde, en los cuatro lados del elemento, es decir, top, right, bottom y left.

border-image-width: Para indicar la anchura del borde.

border-image-outset: Nos sirve para indicar el área en la que la imagen de borde se extiende más allá del área del elemento, en los 4 lados del mismo.

border-image-repeat: Permite marcar si se desea o no que se repita la imágen del borde haciendo un mosaico o si se desea que se estire, etc.

border-image: Se utilizaría como una manera resumida de especificar varios atributos de borde con imágenes al mismo tiempo.

Sombras en CSS 3 con box-shadow

Crear sombras en CSS3 con el atributo box-shadow. Por fin podremos aplicar sombras a los elementos de la página, sin usar imágenes, Javascript ni nada extra, simplemente con un atributo de CSS 3.

CSS 3 supone una nueva revolución en el diseño web, aportando soluciones a muchas de las prácticas que utilizan los diseñadores para decorar las páginas web. Esto quiere decir que, muchas de las cosas que antes hacíamos con técnicas de diseño que requerían uso de imágenes, como las

sombras o las esquinas redondeadas, a partir de ahora las vamos a poder especificar simplemente desde CSS.

Atributo box-shadow

El atributo box-shadow requiere varios valores para especificar las características de la sombra, como difuminado, separación de la sombra y la propia caja o color. La sintaxis es como esta:

```
box-shadow: 5px -9px 3px #000;
```

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

Desplazamiento horizontal de la sombra: La sombra de un elemento suele estar un poco desplazada con respecto al elemento que la produce y su posición será en función del ángulo con el que llegue la luz. En el caso de este ejemplo el primero de los valores, 5px, quiere decir que la sombra aparecerá 5 píxeles a la derecha. Si la sombra quisiéramos que apareciera un poco hacia la izquierda del elemento original que la produce, pondríamos un valor negativo a este atributo. Cuanto más desplazamiento tenga una sombra, el elemento que la produce parecerá que está más separado del lienzo de la página.

Desplazamiento vertical de la sombra: El segundo valor que colocamos en el atributo box-shadow es el desplazamiento vertical de la sombra con respecto a la posición del elemento que la produce. Este valor es similar al desplazamiento horizontal. Valores positivos indican que la sombra aparecerá hacia abajo del elemento y valores negativos harán que la sombra aparezca desplazada un poco hacia arriba. En el caso del anterior ejemplo, con -9px estamos indicando que la sombra aparecerá desplazada 9 píxeles hacia arriba del elemento.

Difuminado: El tercer valor indica cuánto queremos que esté difuminado el borde de la sombra. Si el difuminado fuera cero, querría decir que la sombra no tiene ningún difuminado y aparece totalmente definida. Si el valor es mayor que cero, como en nuestro ejemplo 3px, quiere decir que la sombra tendrá un difuminado de esa anchura, 3 píxeles en el ejemplo.

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

Compatibilidad de las sombras CSS con navegadores

Lo cierto es que las CSS 3 todavía están en fase de especificación, aunque ya se encuentran muy avanzadas y los navegadores más modernos ya han comenzado a implementarlas. No obstante, el W3C todavía no ha liberado las especificaciones de esta versión de las Hojas de Estilo en Cascada y hasta que empiece a recomendar su implementación los clientes web no tienen por qué entenderlas.

Por el momento podemos utilizar box-shadow en las versiones más moderas del navegador Opera. Por su parte, navegadores basados en Mozilla y WebKit tienen soporte a esta funcionalidad de CSS3, pero a través de unos atributos CSS con una ligera variación en su nombre.

Atributo box-shadow para navegadores basados en Mozilla, como Firefox: De manera temporal, Firefox es capaz de interpretar el atributo -moz-box-shadow, por ejemplo:

```
-moz-box-shadow: 1px 1px 0px #090;
```

Atributo box-shadow para navegadores basados en WebKit, como Safari o Google Chrome: En estos momentos y de manera temporal, navegadores como Chrome o Safari entienden el atributo: -webkit-box-shadow, por ejemplo:

```
-webkit-box-shadow: 3px 3px 1px #fc8;
```

Como podremos imaginar, si deseamos ampliar al máximo la compatibilidad con box-shadow, necesitaríamos indicar tanto el propio atributo box-shadow (que funciona en Opera y en el futuro funcionará en todos los navegadores), así como -moz-box-shadow y -webkit-box-shadow para que funcione en las versiones actuales de Firefox, Safari, Chrome, etc.

Ejemplos de Sombras CSS3

Ahora veamos varios ejemplos de sombras creadas directamente con CSS 3 y el atributo box-shadow, con sus variantes para compatibilidad temporal en los navegadores Mozilla o WebKit.

```
#cajasombra{
   background-color: #ddd;
   width: 300px;
   padding: 10px;

   box-shadow: 5px 5px 0 #333;
   -webkit-box-shadow: 5px 5px 0 #333;
   -moz-box-shadow: 5px 5px 0 #333;
}
```

Esto crearía una capa con un gris claro como color de fondo y una sombra desplazada abajo y a la derecha en 5 píxeles y sin difuminado. Además, hemos definido un color de sombra gris oscuro para el elemento.

```
#sombraclara{
   width: 200px;
   padding: 10px;
   background-color: #999;
   color: #fff;

   box-shadow: 2px 2px 2px #ffc;
   -webkit-box-shadow: 2px 2px 2px #ffc;
   -moz-box-shadow: 2px 2px 2px #ffc;
}
```

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

```
#sombraredondeada{
   background-color: #090;
   color: #fff;
   width: 400px;
   padding: 10px;
   -moz-border-radius: 7px;
   -webkit-border-radius: 7px;

   box-shadow: 15px -10px 3px #000;
   -webkit-box-shadow: 15px -10px 3px #000;
   -moz-box-shadow: 15px -10px 3px #000;
}
```

En este tercer ejemplo tenemos un caso curioso de sombra, pues está aplicada sobre un elemento que tiene las esquinas redondeadas con CSS 3. Así pues, la sombra también debe tener las sombras redondeadas, para ajustarse al elemento que la produce. Ambos navegadores con compatibilidad a sombras y CSS 3 funcionan correctamente con sombras y bordes redondeados.

Resplandor exterior con CSS3

Cómo realizar un elemento que tenga un resplandor exterior con CSS3 y la propiedad box-shadow.

Una sombra es un resplandor exterior, pero esta es de color claro en vez de ser oscura. Pero para poder verlos del todo lo que se debe hacer es que se ponga un fondo oscuro en la página de esta forma se podrán apreciar.

Para poder realizar el resplandor que he comentado, lo que necesitamos es una sombra blanca en nuestra hoja de estilos.

-moz-box-shadow: 0px 0px 30px #ffffff;

-webkit-box-shadow: 0px 0px 30px #ffffff;

box-shadow: 0px 0px 30px #ffffff;

En nuestro diseño web cambiando el RGB, podremos cambial el color del resplandor, en este otro caso de color verde:

-moz-box-shadow: 0px 0px 30px #A3FF0F;

-webkit-box-shadow: 0px 0px 30px #A3FF0F;

box-shadow: 0px 0px 30px #A3FF0F;

El ejemplo del que nos referimos en nuestro diseño CSS es el siguiente:

```
<!DOCTYPE HTML PUBLIC “-//W3C//DTD HTML 4.01//EN”
“http://www.w3.org/TR/html4/strict.dtd”
>
<html lang=”en”>
<head>
<title>Hacer un resplandor exterior con CSS3</title>
<style type=”text/css”>
body{
font-family: Trebuchet MS, verdana, sans-serif;
background-color: #000;
color: #fff;
}
#resplandorblanco{
-moz-box-shadow: 0px 0px 30px #ffffff;
-webkit-box-shadow: 0px 0px 30px #ffffff;
box-shadow: 0px 0px 30px #ffffff;
padding: 10px;
border: 1px solid #fff;
width: 160px;
margin: 40px;
}
#resplandorverde{
-moz-box-shadow: 0px 0px 30px #A3FF0F;
-webkit-box-shadow: 0px 0px 30px #A3FF0F;
box-shadow: 0px 0px 30px #A3FF0F;
padding: 10px;
border: 1px solid #66ff00;
width: 160px;
margin: 40px;
}
</style>
</head>
<body>
<div id=”resplandorblanco”>
Resplando exterior
</div>
<br>
<br>
<div id=”resplandorverde”>
El resplandor es de color verde
</div>
</body>
</html>
```

Propiedad background-origin de CSS 3

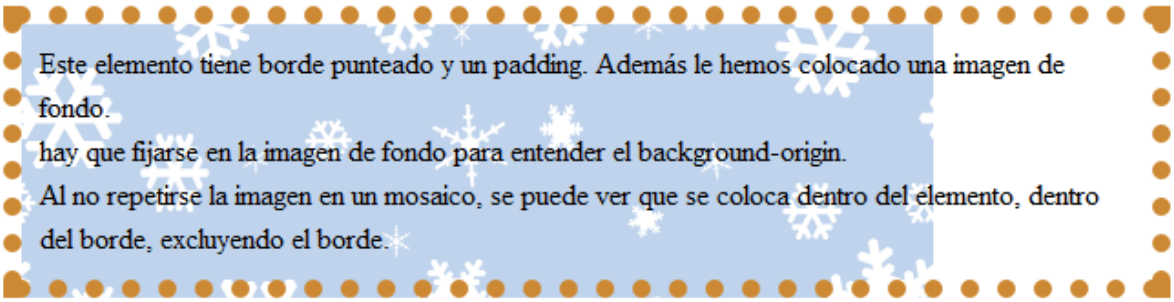
La propiedad de CSS 3 background-origin permite decidir la posición de la imagen de fondo con respecto al borde, padding o el contenido del elemento.

CSS permite especificar que los elementos tengan un fondo con una imagen y además, con algunos atributos como background-position o background-repeat podemos definir cosas como la posición de la imagen de fondo se traslade a otro lugar o que se forme un mosaico. Esas propiedades son bastante utilizadas en el diseño de páginas web y quizás ya las dominemos. Ahora bien, con CSS 3 tenemos la posibilidad de especificar nuevos estilos o modos de comportamiento de las imágenes.

Qué es background-origin

Cuando colocamos una imagen de fondo en un elemento de HTML se posiciona dentro del espacio de ese elemento. Por defecto una imagen de fondo aparece como un mosaico, repitiendo la imagen a lo largo de todo el espacio de ese elemento. Pero para colocar esa imagen el navegador utiliza un origen de coordenadas, que veríamos más fácilmente si hacemos que la imagen no se repita, es decir, que no se haga un mosaico en el fondo.

Observar el ejemplo de la siguiente imagen:



Se ha colocado una imagen de fondo a ese elemento, pero la imagen no se repite. Se puede ver el amarillo del fondo de color asignado al elemento y la imagen de fondo en azul. Ahora apreciemos dónde comienza la imagen. Veremos que está colocada dentro del elemento, dentro del cuerpo y sin tener en cuenta el borde. Sabemos que con background-position podríamos cambiar esa posición, pero en CSS 3 existe el atributo background-origin que también nos servirá para trasladar esa imagen pero de otra manera.

Con background-origin podemos definir el origen de coordenadas sobre el que se va a colocar la imagen de fondo, para que sea el borde del elemento, su padding o su contenido. Veamos sus posibles valores con una explicación:

border-box:
Este valor significa que el origen de coordenadas de la imagen será el borde del elemento. En este caso estamos indicando que la imagen empiece donde empieza el borde del elemento, si es que tiene borde.

```
background-origin: border-box;
```

padding-box:

Este valor es el utilizado por defecto en CSS 3 y es tal como se posiciona la imagen en navegadores que sólo entienden CSS 2 o inferior. Cuando indicamos background-origin: padding; queremos que el eje de coordenadas donde se va a colocar la imagen sea el espacio destinado al elemento, incluyendo su posible padding y sin contar el posible borde.

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

content-box:
El tercero de los posibles valores de background-origin sirve para que el origen de coordenadas para la posición de la imagen de fondo sea el lugar donde empieza el contenido del elemente, osea, sin tener en cuenta sus posibles bordes y padding.

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

Atributos CSS3 overflow-x y overflow-y

Descripción de los atributos de CSS3 overflow-x y overflow-y, que sirven para definir cómo renderizar un contenido cuando sobrepasa los límites de un contenedor en la horizontal o vertical.

Valores posibles para overflow-x y overflow-y

En estos dos nuevos atributos podemos colocar varios valores distintos, que nos servirán para definir diferentes tipos de comportamientos ante el desborde del contenido de una capa. Tanto overflow-x como overflow-y comparten el mismo abanico de valores posibles, pero los podemos especificar por separado, para la coordenada X y la Y. De ese modo, no tienen por qué definirse los mismos valores cuando surgen desbordamientos en la horizontal y en la vertical.

Visible: Esto hace que el contenido que no cabía en la capa se muestre igualmente en el navegador. Es el comportamiento predeterminado.

Hidden: Sirve para decirle al navegador que no muestre cualquier contenido que se salga de las dimensiones especificadas en el contenedor.

Scroll: Permite mostrar unas barras de desplazamiento para que el usuario pueda hacer scroll sobre el contenido y ver áreas que quedarían fuera del contenedor. En el caso que se aplique este atributo, las barras de desplazamiento aparecerían siempre en el contenedor, independientemente de si el contenido sobrepasa o no las dimensiones del contenedor.

Auto: Este valor indica que las barras de desplazamiento deben aparecer solo en el caso que el contenido supere los límites del contenedor. Es decir, es lo mismo que scroll, pero no aparecerían siempre las barras de desplazamiento, sino solamente cuando sean necesarias.

No-display: Este comportamiento a día de hoy no está implementado en los navegadores, pero serviría para que, en caso que un contenido sobrepase el límite asignado al contenedor, la capa completa contenedora sea eliminada de la página. El efecto sería el mismo que si hubiésemos colocado display:none en el contenedor (si es que había contenido que saliese de sus dimensiones, claro).

No-content: Esto provocaría que cualquier contenido, que no cupiese en las dimensiones del contenedor, fuera eliminado como si le hubiésemos colocado el atributo visibility:hidden. Osea, en diferencia del atributo anterior, lo que se elimina es el contenido y no el contenedor entero.

Como se puede comprobar, las opciones son diversas y permitirían bastantes combinaciones distintas para comportamientos en una capa, definiendo por separado lo que debe ocurrir en la horizontal y en la vertical.

Introducción a @font-face de CSS

Fuentes en CSS 3. Sintaxis y principales características de la regla CSS @font-face, que nos permite utilizar cualquier tipografía en una página web.

Importar fuentes tipográficas mediante CSS con @font-face

La mencionada regla @font-face nace con CSS 2 pero hasta CSS 3 no empieza a funcionar y prosperar. En un principio sólo funcionaba en IE 5 y únicamente admitía formatos de fuente .eot, pero con el paso del tiempo otros navegadores ampliaron su soporte, comenzando con Safari 3,1. En la actualidad admite otros tipos de formatos tipográficos como son .ttf y .otf y funciona también con los navegadores Opera 10, Firefox 3,1 y por supuesto, todas las versiones superiores a los navegadores ya citados.

Así pues, nada nos impide ya hacer uso de esta @font-face, para poder utilizar cualquier fuente en nuestra web, con la garantía que se verá perfectamente en todos los navegadores más actuales.

Su sintaxis es la siguiente:

```
@font-face{
   font-family:<nombre_fuente>;
   src: <source>[,<source>]*;
   [font-weight:<weigth>];
   [font-style:<style>];
}
```

Con esto definimos el tipo de letra y su ubicación en nuestro servidor. Si queremos utilizar dicha fuente tan solo tenemos que llamarla con font-family en las reglas de estilo que deseemos.

Debemos tener en cuenta que si no encuentra la fuente en nuestro servidor, cogerá la siguiente por defecto que tengamos definida en nuestra hoja de estilos.

Ejemplo de uso de @font-face

A continuación colocamos el código de un ejemplo con dos tipos de fuentes, una con formato .eot y otra con formato .otf. El primero se ven en todos los navegadores y en el segundo IE coge otra letra por defecto ya que no admite el formato .otf.

Código de la hoja de estilos:

```
@font-face {
   font-family: Vivaldi;
```

```
   font-style: normal;
   font-weight: normal;
   src: url(VIVALDI0.eot);
}

@font-face{
   font-family: "gothic";
   font-style: normal;
   font-weight: normal;
   src: url(gothic.otf);
}

H1{
   font-family: "gothic";
}

H2{
   font-family: "Vivaldi";
}
```

Y a continuación el código HTML para ver el resultado:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">

<html>
<head>
   <title>Prueba con estilos de letra distintos</title>
   <link rel="STYLESHEET" type="text/css" href="estilo-font-face.css">
</head>

<body>

   <h1>Estamos probando la letra Gothic (no se verá en IE)</h1>
   <h2>Aqui la letra Vivaldi</h2>
</body>
</html>
```

Sombras en el texto con text-shadow de CSS

Cómo aplicar sombras y otros efectos en los textos con CSS y el atributo text-shadow.

El atributo text-shadow de CSS sirve para crear sombras en el texto, pero realmente con un poco de práctica e imaginación nos puede dar soporte a muchos otros efectos interesantes. En este artículo explicaremos dicha regla de estilos y ofreceremos ejemplos variados para demostrar su versatilidad.

Sombra sólida

```
h1{
   text-shadow: 1px 2px #999;
}
```

Así estamos modificando los encabezamientos de nivel 1 para que tengan una sombra sólida de color gris. Los valores que estamos indicando en la sombra son:

- Desplazamiento de la sombra a la derecha (1px).
- Desplazamiento de la sombra hacia abajo (2px).
- Color de la sombra (#999).

Sombra desenfocada

La sombra sólida está bien, pero en muchos casos vamos a desear hacer un efecto de desenfocado de la sombra, que es mucho más realista y a menudo más atractivo visualmente. Para ello podemos definir un valor adicional, que es el tamaño del difuminado.

```
h2{
   text-shadow: 3px 3px 2px #696;
   color: #666;
}
```

Aquí hemos definido una sombra con 3px de desplazamiento abajo y a la derecha y 2px de difuminado o desenfoque. Además la sombra es de color verdoso. También se ha definido el color del texto, con el atributo "color", pero eso no tiene nada que ver con la sombra.

Colocar varias sombras en un mismo elemento

Podemos definir varias sombras diferentes sobre un mismo elemento de la página, con lo que se pueden obtener efectos variados y algunos de ellos bastante llamativos. Para ello se pueden colocar las sombras que se deseen separadas por comas.

```
h3{
   text-shadow: 10px 8px #ccf, -10px 12px #fcf, -8px -12px #cfc, 12px -5px #fc9;
   color: #999;
}
```

Esto no tiene ningún misterio, simplemente se irán colocando todas las sombras que definamos, pero habrá que tener un poco de criterio para hacer efectos que merezcan la pena.

Efectos diversos con sombras CSS

El atributo text-shadow es un excelente recurso para hacer distintos tipos de efectos gráficos que resultan visualmente atractivos, más aun teniendo en cuenta que se hacen con texto simple y asignado únicamente algunas reglas de estilo. A continuación veremos varios ejemplos que podemos anotarnos como inspiración, pero la gama de posibilidades va mucho más allá.

Sombra "Giga":
Podemos utilizar varias sombras sólidas para generar una supersombra para nuestro texto.

```
h2.sombragiga{
   text-shadow: #f83 -1px 1px, #f83 -2px 2px, #f83 -3px 3px, #f83 -4px 4px, #f83 -5px
5px;
   color: #060;
   letter-spacing: 1px;
}
```

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

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

Contornear el texto con un trazo:
Con cuatro sombras sólidas a un píxel de distancia del texto, situadas a los cuatro lados, podemos conseguir un efecto de trazo alrededor del texto.

```
h2.contornear{
   text-shadow: -1px 0 #090, 1px 0 #090, 0 1px #090, 0 -1px #090;
   color: #fff;
}
```

Texto en relieve:
Con una sombra oscura y otra clara podemos conseguir un efecto de relieve sobre el texto. Puede ser un relieve o un bajo relieve, dependiendo de dónde coloquemos ambas sombras.

```
h2.relieve {
   text-shadow: 1px 1px white, -1px -1px #333;
   background-color: #ddd;
   color: #ddd;
   padding: 10px;
}
```

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

```
h1.pixelart{
   text-shadow: 1px 1px #666, 2px 2px #86D6D3, 3px 3px #666, 4px 4px #86D6D3;
   color: #ccc;
}
```

Degradados con CSS 3

Presentación de las características de los degradados con CSS3, que permiten hacer todo tipo de gradientes sin necesidad de usar imágenes.

Los degradados son uno de los recursos que utilizan los diseñadores para decorar las webs y la verdad es que dan mucho juego para mejorar el aspecto de la página. No obstante, hasta la llegada de CSS 3, también tenían una desventaja importante, ya que para implementarlos necesitábamos usar imágenes como fondo de los elementos. Ello tiene algunas desventajas, como una mayor carga de peso en la página y la necesidad de fabricar los archivos gráficos con un programa de diseño, con la molestia adicional que necesitaríamos usar de nuevo el programa de diseño gráfico, para producir una nueva imagen, en el momento que queramos retocar el degradado.

Degradados lineares:

En los que se crea un gradiente que va de un color a otro de manera lineal. Puede ser de arriba a

abajo, de izquierda a derecha y viceversa. Incluso se puede conseguir un degradado en un gradiente de una línea con cualquier ángulo.
Los degradados lineares se consiguen con el atributo background asignándole el gradiente con la propiedad "linear-gradient" de CSS 3. Un ejemplo puede verse a continuación:

```
div{
   height: 130px;

   width: 630px;

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

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

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

}
```

Imagen del renderizado del degradado lineal con CSS3, tal como aparecería en Firefox 4

linear-gradient(orange, pink);

Degradados circulares:

En ellos se implementa un gradiente que se distribuye radialmente, desde un punto del elemento hacia fuera, de manera circular, que puede tener el mismo valor de radio (para hacer degradados en círculos perfectos) o con valores de radio variables (lo que generaría elipses).

El valor que asignamos a background en este caso será por medio del atributo "radial-gradient", además de toda la serie de parámetros necesarios para definir el degradado según nuestras intenciones.

```
div.circular{
   background: -webkit-radial-gradient(#0f0, #06f);
   background: -moz-radial-gradient(#0f0, #06f);
   border: 1px solid #333;
   height: 200px;
   width: 250px;
}
```

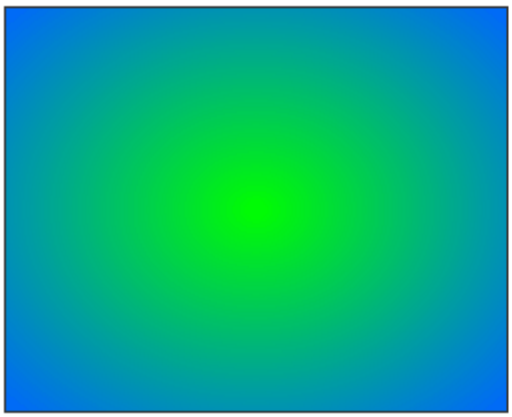


Imagen del renderizado del degradado radial con CSS3, tal como aparecería en Firefox 4

radial-gradient(#0f0, #06f);

Degradado lineal, linear-gradient de CSS3

Explicación detallada de los degradados lineales de CSS3 que conseguimos con la propiedad linear-gradient. Crear degradados de colores, que se distribuyen en un gradiente lineal.

El degradado de colores más sencillo que podemos crear es el degradado lineal. Con CSS 3 se pueden especificar con tan solo definir una serie de parámetros en la propiedad linear-gradient, que nos permiten configurar todo tipo de gradientes de dos o más colores, sin la necesidad de usar imágenes.

Sintaxis de creación de degradados lineales

```
background: linear-gradient(parámetros);
```

O bien

```
background-image: linear-gradient(parámetros);
```

Como vemos, para asignar un degradado a un elemento, tenemos que utilizar la propiedad linear-gradient sobre un atributo background, o background-image. Todos los elementos que soportan imágenes de fondo permiten también colocar degradados de fondo. Además, tendremos que indicar una serie de parámetros variables para la creación del degradado, que son los que realmente tienen alguna dificultad de entender. Estos parámetros son los siguientes:

A) Origen-y/o-angulo del degradado:
El primer parámetro sería el origen desde donde comenzará el degradado y/o el ángulo de disposición del gradiente de color. Podemos decir que el degradado comience desde arriba, abajo o desde una esquina cualquiera. Por defecto los degradados serán distribuidos en un gradiente en línea recta, pero además podemos indicar un ángulo distinto con el que se vaya produciendo el gradiente de color.

B) lista-de-colores y opcionalmente, el lugar hasta donde se debe mostrar cada uno:
Luego colocaremos los colores, todos los que queramos, que deben utilizarse en el degradado,

separados por comas. Además, si lo deseamos, podemos definir las paradas de color "color stops", que consiste en declarar el lugar desde donde debe empezar el gradiente del color.

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

Esto hace un degradado desde el color naranja hacia el rosa. Todos los demás parámetros quedarían con sus valores predeterminados y el resultado sería que el degradado se realiza en toda la altura del elemento, de arriba a abajo, en un gradiente vertical, comenzando el naranja en la parte de arriba y acabando en rosa en la parte de abajo.

```
background: linear-gradient(top left, #fff, #f66);
```

Este degradado comienza en la esquita superior izquierda y se crea un gradiente que va hacia la esquina opuesta. Por tanto, el degradado formará un gradiente oblicuo, en diagonal desde la esquina superior izquierda, donde estaría el blanco (#fff), hasta la esquina inferior derecha, donde estaría el rosa (#f66).

```
background: -webkit-linear-gradient(180deg, #f0f, #f66);
```

Este degradado define su dirección por medio de un ángulo expresado en grados. 0 grados haría que el degradado comenzara en la parte de la izquierda y 180deg indica que el degradado empezaría justo por el lado contrario, es decir, por la derecha. De modo que en la parte de la derecha tendríamos el color morado y en la izquierda tendríamos el rosado.

```
background: linear-gradient(#00f 50%, #000);
```

Este degradado tiene lo que se llama un "color stop" es decir, una parada de color, que está asignada con el 50% en el primer color. Quiere decir que el primer color estaría homogeneo (sin degradado) hasta el 50% del tamaño del elemento y que luego comenzaría a degradarse hacia el segundo color.

```
background: linear-gradient(45deg, #66f, #f80, #ffc);
```

Este degradado tiene una disposición en diagonal, por los 45 grados que se han definido. Además, podemos ver que hemos definido más de dos colores en el degradado. Podemos poner tantos como queramos, separados por comas. Como no hay "color stops" los tres colores se distribuyen de manera equitativa, desde la esquina inferior izquierda hasta la superior derecha.

```
background: linear-gradient(45deg, #66f 10%, #f80 30%, #ffc 60%);
```

Este degradado se hace también en diagonal, desde la esquina inferior izquierda, igual que el anterior, pero hemos definido una serie de paradas de color (color stops), con lo cual la distribución del gradiente no es homogénea. El primer color empezaría a degradarse hacia el segundo cuando se llega al 10% del tamaño del elemento. El degradado hacia el segundo color se completaría al llegar al 30% y a partir de ese punto empezaría a degradarse hacia el tercer color. El degradado entre el segundo y tercer color se realizaría desde el 30% al 60% del tamaño del elemento y se completaría cuando estamos en el 60%. A partir de ese último color stop (60%) tendríamos el último color de manera homogénea hasta el 100% del tamaño. Por tanto, el color

predominante veremos que es el tercero, que tiene un 40% (100% del elemento - 60% del espacio donde veremos degradados) del espacio para mostrarse con su RGB tal cual fue definido.

```
background: linear-gradient(45deg, #66f 160px, #f80 180px, #ffc);
```

Este es el mismo degradado que el anterior, pero con paradas de color distintas. Además, estamos definiendo esos "color stops" con medidas en píxeles en lugar de porcentajes.

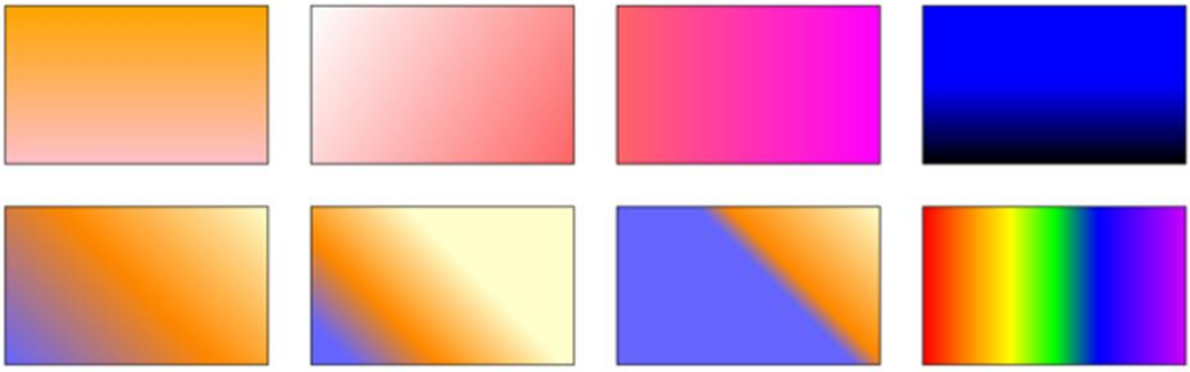
```
Nota: se aconseja no mezclar unidades CSS distintas en las paradas de color, como podría ser:

background: linear-gradient(left, #66f 60%, #f80 50px)

Porque en ese caso, dependiendo del anchura del elemento, la segunda parada de color, con 50px, podría estar en un punto anterior al 60% de su tamaño, lo que podría provocar que la distribución del gradiente de color fuera imposible de realizar.

background: linear-gradient(left, #f00, #f80, #ff0, #0f0, #00f, #60f, #c0f);
```

Este degradado, que empieza en la izquierda y con un gradiente recto hacia la derecha, tiene varios colores, los del arcoiris.



Degradados radiales con CSS3

Veremos ahora cómo conseguir degradados CSS3 en un gradiente de color que se distribuirá de forma radial, creando tanto círculos como elipses.

Los degradados radiales, que incluyen tanto los que tiene forma circular en general, como los que tienen forma de elipse, se consiguen a través del atributo radial-gradient, de CSS3. De modo que, mediante la aplicación de distintos parámetros, conseguiremos todas las posibilidades. Como veremos a continuación, tienen una forma de definirse muy parecida a la que vimos con los degradados lineares, aunque en este caso tendremos algunos otros parámetros a tener en cuenta, lo que puede dificultar un poquito más su entendimiento.

La sintaxis resumida será la siguiente:

```
background: radial-gradient(parámetros);
```

O bien

```
background-image: radial-gradient(parámetros);
```

Los parámetros que podemos indicar en la declaración radial-gradient() es donde realmente radica la dificultad y a la vez la potencia de este tipo de degradados. No obstante, la mayoría de los parámetros son opcionales, por lo que podemos hacer degradados radiales bastante simples, que tomarán parámetros por defecto. En realidad, como veremos, lo único que necesitaremos siempre es definir dos o más colores para realizar el gradiente de color.

El listado de parámetros que podremos indicar es el siguiente:

A) Posición inicial del gradiente circular:
Los degradados radiales comienzan en un punto cualquiera del fondo de un elemento y se extienden hacia fuera de ese punto con formas circulares o de elipse. Luego, para definirlos, necesitaremos una forma de especificar dicho punto de inicio del degradado. El punto se especifica con una o dos coordenadas, que pueden tener distintas unidades CSS. Si se omite, se entiende que el degradado tiene que comenzar en el punto central del fondo del elemento.

B) Forma y/o dimensión:
La forma puede ser circular o elipse, para lo cual especificamos las palabras "circle" o "ellipse". El tamaño lo expresamos con otra serie de palabras clave, que indican hasta donde debe crecer el círculo o elipse: closest-side | closest-corner | farthest-side | farthest-corner | contain | cover. Por ejemplo, closest-side indica que el círculo o elipse debe crecer hasta el lado más cercano. La palabra farthest-corner indicaría que debe crecer hasta la esquina más lejana. Contain sería lo mismo que decir closest-side y cover sinónimo de farthest-corner.

Alternativa a B) Tamaño:
De manera alternativa a especificar la forma y dimensión del degradado -punto B) anterior-, podemos indicar un par de medidas en cualquier unidad CSS o porcentajes. Esas medidas se utilizarían para generar un círculo o una elipse del tamaño deseado para nuestro gradiente. La primera medida sería para la anchura de la elipse y la segunda sería para la altura (si ambas son iguales se mostraría una forma circular en el degradado. Si son distintas, sería una elipse. El tamaño debe ser siempre positivo.

C) Colores del degradado:
Para acabar, se deben indicar cuantos colores se deseen, separados por comas, con la posibilidad de indicar las paradas de color que se deseen.

background: radial-gradient(#0f0, #06f);

Esto hace un degradado desde el verde al azul turquesa, con todos los otros parámetros perdeterminados. Haría un gradiente de forma circular, con su punto de inicio en el centro del elemento, en verde, haciendo que se llegase al azul turquesa en los bordes del elemento.

background: radial-gradient(top left, #fff, #f66);

En este caso hemos definido el punto de inicio del gradiente con "top left". Se trata de la esquina superior izquierda, donde aparecerá el blanco y el degradado tendría forma circular tendiendo hacia rosa, ocupando el 100% del elemento.

background: radial-gradient(200px 30px, #f0f, #000);

Este degradado también declara la posición inicial del gradiente, pero lo hace mediante las coordenadas definidas con medidas en píxeles. Es circular y ocupa el 100% del espacio disponible en el elemento.

background: radial-gradient(center, #00f, #000 50%);

En este declaramos la posición inicial con center, el comportamiento predeterminado, que coloca el inicio del degradado en centro, tanto vertical como horizontal. El detalle es que el degradado se realiza desde el centro hasta el 50% del tamaño del elemento, ya que le hemos puesto una parada de color ("color stop") de 50% en el último color.

Introducción a las animaciones CSS

Las animaciones CSS nos permiten realizar efectos que hasta ahora estaban sólo disponibles con otros tipos de tecnologías. Veremos los principales aspectos a conocer sobre las animaciones CSS 3.

Como todos posiblemente sepamos, hasta el momento, las animaciones en las páginas web siempre se tenían que realizar utilizando diversas tecnologías accesorias, más allá del simple HTML o CSS. El primer sistema que alcanzó gran popularidad para realizar una animación de elementos bastante fluida y espectacular fue la tecnología Flash y luego le acompañaron algunos otros sistemas como Silverlight, de características similares. Sin embargo, todo esto son tecnologías propietarias, que requieren la instalación de un plugin para funcionar en el navegador, lo que impide que sean universales, por mucha aceptación que hayan llegado a tener.

Paralelamente existen varios otros soportes para animación que sí forman parte de las tecnologías de creación de páginas web universales, pero que no llegan ni de lejos a las posibilidades de animación que podríamos desear. Nos referimos a los GIF animados, que tanto se utilizaron al principio y que ahora están prácticamente olvidados, así como a Javascript que también permite hacer animaciones a base de cambiar atributos CSS de manera progresiva a lo largo de un tiempo.

Bien, pues con CSS 3 viene una nueva forma de realizar animaciones totalmente novedosa y que resultará mucho más sencilla que el uso que podemos conocer con Javascript. Pero lo que es más importante, que soporta muchos más tipos de animación que hasta ahora estaban reservados a tecnologías como Flash, como pueden ser rotaciones, ampliaciones y reducciones del tamaño vectoriales, etc.

Esto no se queda ahí, ya que además se han implementado una ciertas interacciones con el usuario y que se consiguen únicamente con CSS 3. Además, todo ello sin tener que programar, lo que puede resultar mucho más agradable y al alcance de los desarrolladores menos técnicos.

Ventajas de las animaciones CSS 3

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

Por tanto, una de las ventajas es que nos podemos olvidar de Flash si queremos hacer dinamismos espectaculares en nuestra web. Dejar a Flash de lado además implica que no tenemos que preocuparnos por el posicionamiento de la página que tantos quebraderos de cabeza provoca cuando nuestra web esta creada en enteramente en Flash. Todo esto sin entrar en el tema de la accesibilidad, en el que Flash es un verdadero quebradero de cabeza.

Pero, como dejábamos entrever, las ventajas más importantes serían la compatibilidad y la facilidad de implementación, al usar un lenguaje que ya resulta familiar para el desarrollador. La compatibilidad viene dada por por el uso de un sistema abierto y regulado por el W3C, al que todos los navegadores tarde o temprano se adaptarán. Y la facilidad de desarrollo porque sólo trabajaremos en nuestros sitios con el lenguaje CSS y no existirá la necesidad de dominar otros lenguajes de programación como ocurría con Flash.

Conceptos básicos para la animación CSS

Seguimos con el tema de las animaciones CSS y repasamos los conceptos y propiedades más básicas para poder crear una animación simple con CSS 3.

Fotograma clave

Los fotogramas claves son valores iniciales y finales que debe tener la animación CSS. Estas localizaciones, en teoría, las sabemos a ciencia cierta, es decir, siempre conocemos en que punto vamos a empezar y en cual vamos a terminar la animación, así como su duración. Pero podemos crear otros fotogramas clave, no solamente los de inicio y fin, que correspondan con puntos intermedios del movimiento. Las reglas que determinan estos valores es lo que llamamos fotogramas clave dentro de CSS.

Su sintaxis sería algo parecido a esto:

```
@keyframes 'nombre_fotograma_clave' {

   0% {
      left: 100px;
   }

   40% {
```

```
      left: 150px;
   }

   60% {
      left: 75px;
   }

   100% {
      left: 100px;
   }

}
```

Esta animación estaría compuesta de 4 fotogramas clave, el porcentaje es en el momento de la animación en el que va a producirse ese fotograma y los px son la longitud y la alineación donde se colocaría el fotograma dentro del DIV en que se encaje.

El código de dicho DIV sería el siguiente:

```
DIV {
   animation-name: 'nombre-fotograma-clave';
   animation-duration: 45s;
   animation-iteration-count: 10;
}
```

Los atributos de estilo para esta capa que se ven en el código anterior son los siguientes:

- **animation-name:** el nombre del fotograma clave.
- **animation-duration:** la duración de la animación.
- **animation-iteration-count:** la veces que se repite.

Propiedades sobre la animación aplicables en el DIV

Además de las propiedades que hemos citado en el párrafo anterior, tenemos otra serie de atributos que se pueden aplicar a la animación y que se colocan en el DIV.

Esta sería una lista de las propiedades adicionales, aplicables para definir las animaciones que especificamos en el DIV:

- **animation-timing-function:** se aplica entre los fotogramas clave, no sobre toda la animación y describe como progresa la animación a lo largo de un ciclo.
- **animation-direction:** esta propiedad define el sentido de la animación. Si especificamos “alternate” y los ciclos de iteraccion son impares, la animación irá en la dirección normal, si no, se realizará en la dirección inversa
- **animation-delay:** propiedad que nos indica el momento en el que comenzará la animación. Si el valor es 0 se ejecuta en cuanta se carga la página.

- **animation:** esta propiedad combina las anteriores de una forma resumida.

Código completo para una animación CSS

A continuación veremos un código CSS donde estamos definiendo una animación, aunque todavía hay algunas cosas que tenemos que contaros antes de hacer nuestra primera página de prueba. Pero de momento aquí tenéis un ejemplo utilizando el fotograma clave y las propiedades de una animación:

```
DIV {
   animation-name: 'movimiento-diagonal';
   animation-duration: 5s;
   animation-iteration-count: 10;
}

@keyframes 'movimiento-diagonal' {
   from {
      left: 0;
      top: 0;
   }

   to {
      left: 100px;
      top: 100px;
   }
}
```

Este ejemplo lo que nos mostraría seria una animación en la que se mueve un elemento de la esquina inferior izquierda a la esquina superior derecha, ese movimiento va a tardar 5 segundos y se va a repetir 10 veces.

Animación de un texto con CSS 3

Realizamos un ejemplo básico de animación CSS 3 sobre una capa con un simple texto. Veremos cómo hacer funcionar este primer ejemplo de animación CSS en navegadores basados en Webkit.

En este artículo vamos a realizar nuestro primer ejemplo de animación CSS. Veremos que todo es bastante sencillo y que con unas pocas líneas de código CSS se pueden hacer cosas bastante interesantes. Por lo menos, para los que hayan estudiado un poco sobre la animación realizada con otras tecnologías como Javascript, quedará claro que las animaciones CSS son mucho más fáciles y rápidas de producir.

Lo primero que tenemos que hacer es crearnos nuestros fotogramas clave, para ello utilizamos el siguiente código en nuestra hoja de estilos.

```
@-webkit-keyframes movimiento-diagonal {
   from {
      left: 0px;
   }

   to {
```

```
      left: 100px;
   }
}
```

Como ya se comentó anteriormente, lo que se ha definido en el código anterior es un par de fotogramas clave, que corresponden con el inicio y el fin de la animación. En la práctica ésto hace que se nos mueva nuestro texto, de izquierda a derecha de 0px a 100 px.

Una vez que tenemos este primer paso, vamos a darle las propiedades necesarias a la capa DIV, para terminar de definir nuestra animación:

```
#anim {
   -webkit-animation-name: movimiento-diagonal;
   -webkit-animation-duration: 3s;
   -webkit-animation-iteration-count: infinite;
   -webkit-animation-direction: alternate;/*para que vuelva a su posicion inicial */
   width: 100px;
   background-color: Teal;
   color: #fff;
   position: relative;
   padding: 2px;
}
```

En la primera linea le damos el nombre a la animación, que tiene que ser el mismo que el del fotograma clave.

En la segunda le damos una duración de 3 segundos, es decir, la animación tardará en hacer el recorrido sólo 3 segundos.

En la tercera le decimos que lo repita infinitas veces.

La propiedad -webkit-animation-direction: alternate hace que el texto, una vez que haga el recorrido, vuelva a su posición inicial realizando el camino inverso.

Y por último le damos un ancho, y color de fondo y de texto, así como una posición relativa, ya que de lo contrario no funcionaria nuestra animación.

Con esto nuestro código CSS estaría completo, ahora sólo nos quedaría el código HTML que sería tan simple como esto:

```
<html>
<head>
   <title>Animacion CSS 3</title>
   <link rel="stylesheet" href="animacion-css.css" type="text/css">
</head>
<body>

   <div id="anim">Esto es una animación</div>

</body>
</html>
```

Código completo de este ejemplo a continuación:

animacion-css.css

```
@-webkit-keyframes movimiento-diagonal {
    from {
        left: 0px;
    }

    to {
        left: 100px;
    }
}

#anim {
    -webkit-animation-name: movimiento-diagonal;
    -webkit-animation-duration: 3s;
    -webkit-animation-iteration-count: infinite;
    -webkit-animation-direction: alternate;/*para que vuelva a su posicion inicial */
    width: 100px;
    background-color: Teal;
    color: #fff;
    position: relative;
    padding: 2px;
}
```

animacion-css.html

```
<html>
<head>
    <title>Animacion CSS 3</title>
    <link rel="stylesheet" href="animacion-css.css" type="text/css">
</head>
<body>

    <div id="anim">Esto es una animación</div>

</body>
</html>
```

Álbum con efectos en CSS 3

Creamos un álbum de fotos con estilos impactantes de CSS 3 y animación, sin utilizar Javascript o jQuery.

Este ejemplo, que verán que es relativamente sencillo, hace años sería muy complicado de hacer y tendríamos que emplear (aparte de HTML y CSS) muchas líneas de código y conocimientos avanzados de Javascript o de algún framework como jQuery. Es una muestra excelente de cómo CSS 3 sustituirá a Javascript en algunas parcelas de sus usos habituales.

En este caso lo primero que vamos a ver va a ser nuestro código HTML, para ir viendo los cambios que vamos realizando según vamos metiendo propiedades a nuestro CSS.

Lo primero que tenemos son las imágenes fijas en nuestro HTML.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">

<html>
<head>
   <title>Álbum de fotos con CSS</title>
   <link rel="stylesheet" href="album.css" type="text/css">
</head>

<body>

   <ul>
      <li><img src="images/mi-gatito.jpg" alt="" border="0"></li>
      <li><img src="images/queso-gema.jpg" alt="" border="0"></li>
      <li><img src="images/atardecer-pescando.jpg" alt="" border="0"></li>
      <li><img src="images/moto.jpg" alt="" border="0"></li>
      <li><img src="images/tanto-monta.jpg" alt="" border="0"></li>
      <li><img src="images/doris.jpg" alt="" border="0"></li>
   </ul>
</body>
</html>
```

Con esto, tan sólo nos aparecerían todas las imágenes en una lista, sin estilos y sin nada. Entonces lo primero que vamos a hacer es darle un estilo a la etiqueta <UL> dejándola así: <UL class=fotos>

Y ese estilo lo definimos con el siguiente código CSS:

```
ul.fotos {
   width: 970px;
   margin: 0 0 18px -30px;
}
ul.fotos li {
   display: inline;
}
ul.fotos a {
   display: inline;
   float: left;
   margin: 0 0 27px 30px;
   width: auto;
   padding: 10px 10px 15px;
   text-align: center;
   color: #333;
   font-size: 18px;
}
ul.fotos img {
   display: block;
   width: 190px;
   margin-bottom: 12px;
}
```

Esto lo único que hace es darle un ancho a nuestro álbum, y redimensiona nuestras imágenes para que todas tengan el mismo tamaño inicial. Todavía no hemos animado nada, vamos paso a paso.

Ahora vamos a ver qué pasa si le damos más estilos a los <li>, pero esta vez tenemos que crear un enlace, aunque no apunte a ningún sitio para poder darle animación a nuestras imágenes. El enlace luego puede ir a la ampliación o a otro sitio, etc.

```
<ul class="fotos">
      <li><a href="#" title="Mi gato"><img src="images/mi-gatito.jpg" alt=""
border="0" ></a></li>
      <li><a href="#" title="El queso"><img src="images/queso-gema.jpg" alt=""
border="0" ></a></li>
      <li><a href="#" title="Valmayor"><img src="images/atardecer-pescando.jpg" alt=""
border="0" ></a></li>
      <li><a href="#" title="Mi moto"><img src="images/moto.jpg" alt="" border="0"
></a></li>
      <li><a href="#" title="Toledo"><img src="images/tanto-monta.jpg" alt=""
border="0" ></a></li>
      <li><a href="#" title="Mi coneja"><img src="images/doris.jpg" alt="" border="0"
></a></li>

</ul>
```

Para que nos salgan los títulos de las fotos tenemos que poner el atributo title en el enlace y no en la imagen.

Los nuevos estilos CSS serían los siguientes:

```
ul.fotos a:after {
   content: attr(title);
}
```

Este estilo junto con las siguientes lineas que añadiríamos al estilo “ul.fotos a”, nos mostrarían una caja y el título de cada foto. Las líneas que tenemos que añadir son las siguientes:

```
-webkit-box-shadow: 0 3px 6px rgba(0,0,0,.25);
-webkit-transform: rotate(-2deg);
-webkit-transition: -webkit-transform .15s linear;
```

Estos estilos son para la colocación de las cajas:

```
ul.fotos li:nth-child(3n) a {
   -webkit-transform: none;
   position: relative;
   top: -5px;

}
ul.fotos li:nth-child(5n) a {
   -webkit-transform: rotate(5deg);
   position: relative;
   right: 5px;

}

ul.fotos li a:hover {
   -webkit-transform: scale(1.25);
   -webkit-box-shadow: 0 3px 6px rgba(0,0,0,.5);
   position: relative;
   z-index: 5;
}
```

Con estos estilos lo que le decimos es que nos rote las fotos x grados y que cada 3 una la ponga recta, después en la 5 la rote algo más que las otras. Estos estilos podríamos hacerlos para

infinidad de imágenes, buscando cada cuantas fotos queremos rotar o poner rectas. Además el último estilo nos hace que al pasar el ratón por la imagen, esta se agrande con scale y se ponga recta si estaba rotada.

Menú animado con CSS 3

Nuevo ejemplo de animaciones CSS 3, en el que construimos un menú animado con HTML y CSS, sin necesidad de Javascript ni jQuery.

Lo primero que vamos a hacer es crearnos nuestro HTML, es decir, el código necesario que utilizaríamos para crearnos nuestro menú normalmente, eso si, realizado con DIVs y no con tablas.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 5.0 Transitional//EN">

<html>
<head>
   <title>Menú animado con CSS</title>
<link rel="stylesheet" href="menu.css" type="text/css">
</head>

<body>

<div class="menu-general">
<ul class="nav">
<li><a href="#">HTML</a></li>
<li><a href="#"</a></li>
<li><a href="#">Javascript</a></li>
<li><a href="#">ASP</a></li>
<li><a href="#">PHP</a></li>

</ul>
</div>
</div>
</body>
</html>
```

Bien pues ahora vamos a dar estilos a los DIV y al listado para crear nuestro menú dinámico. Lo primero que vamos a realizar es un fondo, simplemente para que el menú quede bien en nuestro ejemplo.

```
body{
   background: -webkit-gradient(linear, left top, left bottom, from(#34bdfc),
to(#f5f8fa));
}
```

Este fondo es de un color azul que va degradándose a un blanco de arriba hacia abajo.

A continuación vamos a darle estilos a nuestro menu-general:

```
.menu-general {
   position: relative;
   float: left;
}
```

Simplemente vamos a hacer que flote a la izquierda y que tenga una posición relativa (Esto es importante para que nuestra animación funcione correctamente).

Ahora ya pasamos a los estilos del listado de nuestro menú:

```
ul.nav {
    list-style: none;
    display: block;
    width: 200px;
    position: relative;
    top: 50px;
    left: 100px;
    padding: 60px 0 60px 0;
    -webkit-background-size: 50% 100%;
    -moz-background-size: 50% 100%;
    -o-background-size: 50% 100%;
}
```

Aquí lo que hacemos es quitar los guiones del listado, situamos el listado y le damos un tamaño. Al final le damos la animación de transformar el tamaño del 50% al 100%. esto nos dará la animación que queremos, es decir, que se haga grande el botón.

```
ul.nav li a {
    -webkit-transition: all 0.3s ease-out;
    -moz-transition: all 0.3s ease-out;
    -o-transition: all 0.3s ease-out;
    background: #f77e08;
    color: #174867;
    padding: 7px 15px 7px 15px;
    -webkit-border-top-right-radius: 10px;
    -moz-border-top-right-radius: 10px;
    -o-border-top-right-radius: 10px;
    -webkit-border-bottom-right-radius: 10px;
    -moz-border-bottom-right-radius: 10px;
    -o-border-bottom-right-radius: 10px;
    -webkit-border-top-left-radius: 10px;
    -moz-border-top-left-radius: 10px;
    -o-border-top-left-radius: 10px;
    -webkit-border-bottom-left-radius: 10px;
    -moz-border-bottom-left-radius: 10px;
    -o-border-bottom-left-radius: 10px;
    width: 100px;
    display: block;
    text-decoration: none;
    -webkit-box-shadow: 2px 2px 4px #0e169b;
    -moz-box-shadow: 2px 2px 4px #0e169b;
    -o-box-shadow: 2px 2px 4px #0e169b;
}
```

En este estilo lo que hacemos es construir los botones de nuestro menú. Le damos un color de fondo, unas esquinas redondeadas, y una sobra alrededor.

```
ul.nav li a:hover {
    background: #faef77;
    color: #67a5cd;
```

```
    padding: 7px 15px 7px 30px;
}
```

Por ultimo le damos estilos a nuestro menú para cuando pasemos por encima con el ratón.

Con esto ya tendría que funcionarnos nuestro menú, eso si, recordando siempre que en el momento de escribir este artículo sólo funcionaba completamente para los navegadores Safari y Chrome y parcialmente para Mozilla Firefox y Opera.

Botones con efectos CSS 3

Creamos una serie de botones con colores y efectos al pasar el ratón sobre ellos, únicamente con CSS 3.

En este artículo vamos a ver como crear botones con CSS3. Estos botones van a tener diferentes colores y tamaños, y todos ellos tendrán un pequeño efecto al pasar el ratón por encima.

Lo primero que vamos a ver seré el código html necesario para mostrar nuestro botón.
Partimos de que cada botón tendrá una clase que vendrá dada por el tamaño que queremos mostrar y el color del botón.

Si vemos el código nos quedará más claro:

```
<a class="button pequeno azul" href="#"><span>Botón</span></a>
```

Una vez que tenemos esto en nuestro archivo html, tenemos que irnos al archivo css donde vamos a crear los estilos necesarios para mostrar nuestros botones.

Empezamos creando el estilo boton:

```
.button, .button span {
    display: inline-block;
    -webkit-border-radius: 4px;
    -moz-border-radius: 4px;
    border-radius: 4px;
}
.button {
    white-space: nowrap;
    line-height:1em;
    position:relative;
    outline: none;
    overflow: visible;
    cursor: pointer;
    border: 1px solid #999;
border: rgba(0, 0, 0, .2) 1px solid;
    border-bottom:rgba(0, 0, 0, .4) 1px solid;
    -webkit-box-shadow: 0 1px 2px rgba(0,0,0,.2);
    -moz-box-shadow: 0 1px 2px rgba(0,0,0,.2);
    box-shadow: 0 1px 2px rgba(0,0,0,.2);
    background: -moz-linear-gradient(
        center top,
        rgba(255, 255, 255, .1) 0%,
```

```
        rgba(0, 0, 0, .1) 100%
    );
    background: -webkit-gradient(
        linear,
        center bottom,
        center top,
        from(rgba(0, 0, 0, .1)),
        to(rgba(255, 255, 255, .1))
    );
    -moz-user-select: none;
    -webkit-user-select:none;
    -khtml-user-select: none;
    user-select: none;
    margin-bottom:10px;
}
.button.full, .button.full span {
    display: block;
}
.button:hover, .button.hover {
    background: -moz-linear-gradient(
        center top,
        rgba(255, 255, 255, .2) 0%,
        rgba(255, 255, 255, .1) 100%
    );
    background: -webkit-gradient(
        linear,
        center bottom,
        center top,
        from(rgba(255, 255, 255, .1)),
        to(rgba(255, 255, 255, .2))
    );
}
.button:active, .button.active {
    top:1px;
}
.button span {
    position: relative;
    color:#fff;
    text-shadow:0 1px 1px rgba(0, 0, 0, 0.25);
    border-top: rgba(255, 255, 255, .2) 1px solid;
    padding:0.6em 1.3em;
    line-height:1em;
    text-align:center;
    white-space: nowrap;
}
```

Una vez que tenemos el estilo del botón vamos a crear diferentes estilos para los tamaños:

```
.button.pequeno span {
    font-size:12px;
}
.button.mediano span {
    font-size:16px;
}
.button.grande span {
    font-size:22px;
}
```

Con esto ya sólo nos queda los estilos para los colores, aquí podéis crear tantos estilos como colores queráis poder implementar en vuestros botones.
Un ejemplo de estilos para 3 colores por ejemplo sería el siguiente:

```
.button.rojo {
   background-color: #e62727;
}
.button.naranja {
   background-color: #ff5c00;
}

.button.azul {
   background-color: #00ADEE;
}
```

Y terminamos nuestro CSS quitando el subrayado a nuestros enlaces para que no aparezcan los botones con ello.

```
A{
   color: #0000cc;
   text-decoration: none;
}
```

Para finalizar os dejo el código completo del archivo .html y del .css

Boton.html

```
<!DOCTYPE html>
<html>

<head>

   <title>Botones CSS 3</title>
<link rel="stylesheet" href="botones-css3.css" type="text/css">




</head>

<body>
   <a class="button pequeno rojo" href="#"><span>Botón</span></a>
<br>
<a class="button mediano naranja" href="#"><span>Botón</span></a>
<br>
<a class="button grande azul" href="#"><span>Botón</span></a>

</body>
</html>
   <a class="boton pequeno gris" href="#"><span>Botón</span></a>

</body>
</html>
```

boton-css3.css

```
.button, .button span {
   display: inline-block;
   -webkit-border-radius: 4px;
   -moz-border-radius: 4px;
   border-radius: 4px;
}
.button {
   white-space: nowrap;
   line-height:1em;
   position:relative;
   outline: none;
   overflow: visible;
   cursor: pointer;
   border: 1px solid #999;
border: rgba(0, 0, 0, .2) 1px solid;
   border-bottom:rgba(0, 0, 0, .4) 1px solid;
   -webkit-box-shadow: 0 1px 2px rgba(0,0,0,.2);
   -moz-box-shadow: 0 1px 2px rgba(0,0,0,.2);
   box-shadow: 0 1px 2px rgba(0,0,0,.2);
   background: -moz-linear-gradient(
      center top,
      rgba(255, 255, 255, .1) 0%,
      rgba(0, 0, 0, .1) 100%
   );
   background: -webkit-gradient(
      linear,
      center bottom,
      center top,
      from(rgba(0, 0, 0, .1)),
      to(rgba(255, 255, 255, .1))
   );
   -moz-user-select: none;
   -webkit-user-select:none;
   -khtml-user-select: none;
   user-select: none;
   margin-bottom:10px;
}
.button.full, .button.full span {
   display: block;
}
.button:hover, .button.hover {
   background: -moz-linear-gradient(
      center top,
      rgba(255, 255, 255, .2) 0%,
      rgba(255, 255, 255, .1) 100%
   );
   background: -webkit-gradient(
      linear,
      center bottom,
      center top,
      from(rgba(255, 255, 255, .1)),
      to(rgba(255, 255, 255, .2))
   );
}
.button:active, .button.active {
   top:1px;
}
.button span {
```

```
    position: relative;
    color:#fff;
    text-shadow:0 1px 1px rgba(0, 0, 0, 0.25);
    border-top: rgba(255, 255, 255, .2) 1px solid;
    padding:0.6em 1.3em;
    line-height:1em;
    text-align:center;
    white-space: nowrap;
}

.button.pequeno span {
    font-size:12px;
}
.button.mediano span {
    font-size:16px;
}
.button.grande span {
    font-size:22px;
}

.button.rojo {
    background-color: #e62727;
}
.button.naranja {
    background-color: #ff5c00;
}

.button.azul {
    background-color: #00ADEE;
}

A{
    color: #0000cc;
    text-decoration: none;
}
```

Transiciones CSS3

Aplicaciones atractivas con transiciones CSS3.

Una aplicación atractiva tiene que provocar un impacto visual agradable en el usuario. Los usuarios siempre han de saber que cualquier orden suya (mediante clic del ratón, una pulsación en pantalla o cualquier otra forma) se recibe e interpreta de forma correcta en la aplicación, y las animaciones ofrecen una manera muy interesante de conseguir esto.

La nueva especificación HTML5 (aunque, en honor a la verdad, tendría que decir "la nueva especificación CSS3") incorpora una herramienta muy potente para gestionar animaciones sencillas: las transiciones.

Introducción

Al principio, el grupo de trabajo de CSS dentro del W3C se resistía a integrar las transiciones dentro de CSS argumentando que en realidad no son propiedades de los estilos, pero finalmente

los diseñadores y desarrolladores consiguieron convencerles de que las transiciones son en realidad estilos dinámicos y que pueden tener cabida dentro de un archivo CSS.

Según se puede leer en el sitio web del W3C, CSS3 Transitions permite crear variaciones progresivas sobre los siguientes tipos de propiedades:

1. Color: interpolación entre los componentes rojo, verde, azul y alpha (tratando a cada uno como un número, como se verá más adelante).
2. Longitud: interpolando entre valores expresados como números reales.
3. Porcentaje: interpolando entre valores expresados como números reales.
4. Entero: interpolado en pasos discretos (números completos). La interpolación tiene lugar en el espacio de los números reales y se convierte a entero utilizando la función floor().
5. Número: se interpola entre valores expresados como números reales (coma flotante).
6. Lista de transformación: puedes consultar la especificación CSS Transforms en: http://www.w3.org/TR/css3-2d-transforms/.
7. Rectángulo: se interpolan los valores x e y, y los componentes de anchura y altura (todos estos valores se tratan como números).
8. Visibilidad: se interpola en forma de pasos discretos. La interpolación se produce en el espacio de números entre el cero, el 1, donde 1 significa "visible" y el resto de valores son "hidden" (ocultos).
9. Sombra: se interpolan los valores de color, coordenadas x e y, y el componente de difuminado (blur), tratando todos los valores como de color o número según corresponda. En aquellos casos donde existen listas de sombras, la lista más corta se rellena al final con sombras de color transparente y todas las longitudes (x,y y difuminado), con valor cero.
10. Gradiente: se interpolan los valores de posición y color en cada parada. Tienen que ser del mismo tipo (radial o lineal) y con el mismo número de paradas para poder realizar la animación.
11. Servidor de dibujo (SVG): la interpolación solo está soportada para las transiciones de gradiente a gradiente y de color a color. Funcionan como se ha descrito antes para cada uno de esos valores.
12. Lista separada mediante espacios de los elementos anteriores: si la lista tiene el mismo número de elementos, cada elemento dentro de ella se interpola siguiendo las reglas anteriores. Si no, no se produce la interpolación.

13. Una propiedad abreviada: si todas las partes de una abreviatura se pueden animar, la interpolación se efectúa como si se hubiera especificado cada una de las propiedades de manera individual.

Y esta es la lista de propiedades que se pueden modificar mediante transiciones:

1. background-color (color)
2. background-image (solo gradientes)
3. background-position (porcentaje y longitud)
4. border-bottom-color (color)
5. border-bottom-width (longitud)
6. border-color (color)
7. border-left-color (color)
8. border-left-width (longitud)
9. border-right-color (color)
10. border-right-width (longitud)
11. border-spacing (longitud)
12. border-top-color (color)
13. border-top-width (longitud)
14. border-width (longitud)
15. bottom (longitud y porcentaje)
16. color (color)
17. crop (rectángulo)
18. font-size (longitud y porcentaje)
19. font-weight (número)
20. grid-* (diversos valores)
21. height (longitud y porcentaje)
22. left (longitud y porcentaje)

23. letter-spacing (longitud)
24. line-height (número, longitud y porcentaje)
25. margin-bottom (longitud)
26. margin-left (longitud)
27. margin-right (longitud)
28. margin-top (longitud)
29. max-height (longitud y porcentaje)
30. max-width (longitud y porcentaje)
31. min-height (longitud y porcentaje)
32. min-width (longitud y porcentaje)
33. opacity (número)
34. outline-color (color)
35. outline-offset (entero)
36. outline-width (longitud)
37. padding-bottom (longitud)
38. padding-left (longitud)
39. padding-right (longitud)
40. padding-top (longitud)
41. right (longitud y porcentaje)
42. text-indent (longitud y porcentaje)
43. text-shadow (sombra)
44. top (longitud y porcentaje)
45. vertical-align (palabras clave, longitud y porcentaje)
46. visibility (visibilidad)
47. width (longitud y porcentaje)
48. word-spacing (longitud y porcentaje)

49. z-index (entero)
50. zoom (número)

Declaraciones

Para declarar una transición en un archivo CSS nos basta con escribir el siguiente código:

```
transition-property: all;
transition-duration: 0.5s;
transition-timing-function: ease;
transition-delay: 0s;
```

Esta declaración indica que cualquier modificación del valor de cualquier propiedad debe hacerse en un intervalo de 0,5 segundos (y no de forma inmediata).

Y podemos también definir las transiciones a nivel individual para cada propiedad:

```
transition-property: opacity left top;
transition-duration: 0.5s 0.8s 0.1s;
transition-timing-function: ease linear ease;
transition-delay: 0s 0s 1s;
```

Finalmente, podemos utilizar la propiedad abreviada "transition" para definir todo lo necesario en una sola línea:

```
transition: all 0.5s ease 0s;
```

En esta versión abreviada se pueden incorporar todas las propiedades que queramos, separándolas con comas:

```
transition: opacity 0.5s ease 0s, left 0.8s linear 0s;
```

Las transiciones se disparan cuando se modifica una propiedad del objeto indicado. La modificación se puede hacer con JavaScript o utilizando CSS3 mediante la asignación de una nueva clase a una etiqueta.

Por ejemplo, si usamos IE10, tenemos la siguiente declaración CSS3:

```
-ms-transition-property: opacity left top;
-ms-transition-duration: 0.5s 0.8s 0.5s;
-ms-transition-timing-function: ease linear ease;
```

Retardo

La línea "transition-delay" determina el retardo que se produce entre el momento en que se modifica el valor de la propiedad y el comienzo de la transición.

Eventos

Al final de cada transición se dispara un evento llamado "TransitionEnd". Dependiendo de qué navegador utilicemos, el nombre cambia:

- `Chrome y Safari: webkitTransitionEnd`
- `Firefox: mozTransitionEnd`
- `Opera: oTransitionEnd`
- `Internet Explorer: MSTransitionEnd`

El evento nos pasa la siguiente información:

- PropertyName: Nombre de la propiedad animada
- ElapsedTime: La cantidad de tiempo, en segundos que ha estado desarrollándose la transición

Más sobre las transiciones CSS3

Puedo sugerir dos excelentes razones por las cuales las transiciones CSS3 van a ser muy útiles:

1. Aceleración por hardware: las transiciones basadas en CSS3 las maneja directamente la GPU (si existe en el equipo) dando lugar a resultados mucho más suaves. Y esto es verdaderamente importante en los dispositivos móviles, cuya capacidad de computación, en general, es limitada.
2. Mayor independencia entre el código y el diseño: desde mi punto de vista, el desarrollador no debería tener que ocuparse de las animaciones ni de nada relacionado con el diseño. Por esa misma razón, el diseñador/artista no tendría que ocuparse del JavaScript. Por eso me parece que CSS3 Transitions es una novedad realmente interesante para los diseñadores, que pueden describir todas las transiciones utilizando CSS sin involucrar a los programadores.

Soporte y fallback

Desde la versión PP3, IE10 (que ya podemos descargar con la versión Preliminar de Desarrollo de Windows 8 desde aquí) soporta las transiciones CSS3:

Sin duda, puesto que la especificación aún no está terminada (está en la fase *working draft*), nos va a tocar utilizar los prefijos de fabricante: -ms-, -moz-, -webkit-, -o-.

Podemos además ver que, obviamente, será necesaria una solución transparente para resolver la situación que se va a presentar en el resto de navegadores. Si el navegador no soporta la funcionalidad, tendremos que preparar un *fallback* programando con JavaScript.

Conviene tener preparado un método de fallback si las funcionalidades de tus sitios web dependen de las transiciones. Si no quieres hacerlo, deberías pensar en utilizar las transiciones solamente como mejoras de diseño. En este caso el sitio web seguirá funcionando, pero la experiencia completa solo se podrá ver en los navegadores que la soporten. Aquí solemos hablar de "mejoras progresivas" ya que cuanto más potente es el navegador, más funcionalidades nos permite ofrecer.