Manual de CSS 3

Miguel Angel Alvarez
Sara Alvarez
David Rousset
Jorge Vargas Vega
y otros...

desarrolloweb.com

desarrolloweb.com/manuales/css3.html

Introducción: Manual de CSS 3

Nuevas características de las especificaciones de Hojas de Estilo en Cascada nivel 3, CSS3, para definir el aspecto de las páginas web.

Las CSS han recibido una actualización importante que incluye características avanzadas tanto para aplicar aspecto avanzado en elementos de una página como para ayudarnos a realizar una maquetación más precisa.

En este manual vamos conociendo diversas características de las CSS aparecidas recientemente, con artículos cortos en los que podrás encontrar explicaciones y ejemplos de cada una por separado.

Encuentras este manual online en:
http://desarrolloweb.com/manuales/css3.html

Autores del manual

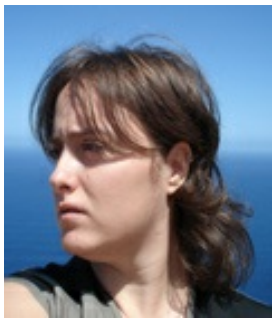
Las siguientes personas han participado como autores escribiendo artículos de este manual.

Miguel Angel Alvarez

Miguel es fundador de DesarrolloWeb.com y la plataforma de formación online EscuelaIT. Comenzó en el mundo del desarrollo web en el año 1997, transformando su hobby en su trabajo.

Sara Alvarez

Equipo DesarrolloWeb.com



Jhon Nadie

David Rousset

Desarrollador Evangelista de Microsoft, especialista en HTML5 y desarrollo Web.

Jorge Vargas Vega

Diseñador y Desarrollador Web

OldMith

Desarrollador Web. jQuery. Responsive Design. Wordpress. Friki por naturaleza.

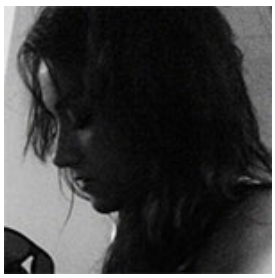
Jaime Peña Tresancos

Escritor. Colaborador habitual de revistas de tecnología y experto en nuevas tecnologías y programas Microsoft

Juan R. Castro Lurita

Thinker, Engineer especialista en web.

Diana Aceves



Licenciada en química, Diana comenzó con la programación con Java y finalmente su vocación la llevó a convertirse en desarrolladora front-end

CSS 3 qué ¿hay de nuevo?

La especificaciónde CSS3 viene con interesantes novedades que permitirán hacer webs más elaboradas y más dinámicas, con mayor separación entre estilos y contenidos. Dará soporte a muchas necesidades de las webs actuales, sin tener que recurrir a trucos de diseñadores o lenguajes de programación.

Introducción a CSS 3

CSS 3 trae grandes novedades para el diseño de webs y algunos navegadores comienzan a implementar CSS 3.

Desde que CSS comenzó han pasado muchos años y ya vamos por la especificación de CSS3, que incorpora una serie de novedades que vamos a tratar de resumir en este artículo.

Qué es CSS

Si no sabes lo que es CSS probablemente te interesaría comenzar leyendo nuestro manual de CSS o la sección de CSS a fondo. No obstante, cabría decir que CSS es un lenguaje para definir el estilo o la apariencia de las páginas web, escritas con HTML o de los documentos XML. CSS se creó para separar el contenido de la forma, a la vez que permite a los diseñadores mantener un control mucho más preciso sobre la apariencia de las páginas.

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

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.

> **Nota:** El estándar de CSS3 a día de hoy está bastante estable, los navegadores las soportan, pero realmente lo que es el estándar sigue en evolución, incorporando propiedades de estilos, redefiniendo sus posibilidades, etc. Es por ello que ninguna lista que se escriba en un momento dado sobre características de CSS3 podrá estar completa, a no se que se actualice semanalmente. En este listado tienes unas cuantas y en el manual de CSS3 tienes muchas otras. En la medida de nuestras posibilidades iremos actualizando.

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<7li>
- 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
- Flexbox

Parte de este listado de nuevas propiedades de CSS 3 lo he sacado de:

http://www.css3.info/preview/. Es un sitio en inglés, pero puede estar bien visitar para ir conociendo más detalles sobre CSS 3. Sin embargo, en ese sitio faltaban algunas cosas como los degradados o las animaciones, por lo menos cuando lo visitamos, por lo que hemos completado para la realización de este índice.

En futuros artículos ofreceremos algunas claves y explicaciones sobre varias de estas propiedades, al menos las más interesantes, así como ejemplos que sirvan para ir conociendo esta nueva especificación de CSS. Todo ello lo iremos colocando en el Manual de CSS 3.

Este artículo es obra de *Miguel Angel Alvarez*
Fue publicado por primera vez en *09/06/2008*
Disponible online en *http://desarrolloweb.com/articulos/introduccion-css3.html*

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.

Atributo gradiente de colores en borde con CSS y Firefox

Posibilidad de definir el un gradiente de color en el borde de los elementos con CSS, en un atributo no estándar de Firefox.

Investigando un poco con algunas de las propiedades nuevas que va a traer CSS 3, para completar la información del artículo Introducción a CSS 3, he dado con un atributo para cambiar el color del borde de los elementos de la página, que nos permite definir el color con una sucesión de distintos valores de colores. En la actualidad, con CSS podemos definir por separado los colores del borde de un elemento, de arriba, derecha, abajo e izquierda. Pero no nos referimos a poder dar un color por separado para cada parte del borde, sino aplicar varios colores distintos a una parte, por ejemplo a la parte de arriba del borde.

Esta propiedad la he visto comentada en algún lugar como de CSS 3, pero según veo en el borrador de la especificación de CSS 3 para bordes y fondos, publicado por el W3C, no aparece esta nueva característica como parte del nuevo estándar.

Por ello, habría que matizar que esta propiedad no es de CSS 3, sino que se trata de un atributo no estándar de CSS, creado por Mozilla y que, por tanto, se puede ver en su navegador más conocido: Firefox. Dicho de otra forma, es una extensión de CSS realizada por Mozilla. En la página de Mozilla podemos encontrar más información sobre esta extensión de CSS: https://developer.mozilla.org/en/CSS/-moz-border-bottom-colors

Los atributos a los que nos referimos, que permiten definir varios colores para cada una de las partes del borde, son los siguientes:

```
-moz-border-top-colors
-moz-border-right-colors
-moz-border-bottom-colors
-moz-border-left-colors
```

En cada uno de los atributos se define el color, pudiendo especificar una lista de colores, separados por espacios, que se aplicarán a cada una de las partes del borde, de dentro hacia fuera. Claro que el borde tiene que tener una anchura mayor que 1 para que se vean los distintos colores. Con una anchura de 2 pixel se podrán ver 2 colores distintos, con una

anchura de 3 podremos definir 3 colores y así sucesivamente.

Para ver una de las posibilidades que daría el uso de este atributo podemos ver un ejemplo en una página aparte. (Pero tener en cuenta que sólo lo veréis correctamente en Firefox).

El código para crear ese gradiente de colores sería el siguiente:

```
<style type="text/css">
.coloresborde{
   border-style: solid;
   border-width: 10px;
   -moz-border-top-colors: #ffcc99 #ffbb88 #ffaa77 #ff9966 #ff8855 #ff7744 #ff6633 #ff5522 #ff4411 #ff3300;
   -moz-border-right-colors: #ffcc99 #ffbb88 #ffaa77 #ff9966 #ff8855 #ff7744 #ff6633 #ff5522 #ff4411 #ff3300;
   -moz-border-bottom-colors: #ffcc99 #ffbb88 #ffaa77 #ff9966 #ff8855 #ff7744 #ff6633 #ff5522 #ff4411 #ff3300;
   -moz-border-left-colors: #ffcc99 #ffbb88 #ffaa77 #ff9966 #ff8855 #ff7744 #ff6633 #ff5522 #ff4411 #ff3300;
}
</style>
```

Es una pena que sea un atributo únicamente desarrollado por Mozilla, aunque si el soporte a estos atributos se lleva a cabo por más navegadores y la W3C lo tiene a bien, quizás en algún momento se convierta en un estándar de CSS.

Por el momento no vale para mucho más que una mera curiosidad y posibilidad de personalización especial para los usuarios de Firefox y otros navegadores basados en Mozilla.

Este artículo es obra de *Miguel Angel Alvarez*
Fue publicado por primera vez en *10/11/2008*
Disponible online en *http://desarrolloweb.com/articulos/atributo-gradiente-colores-borde-css-firefox.html*

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.

En este artículo vamos a explicar cómo realizar **bordes redondeados con CSS3**.

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 lo que respecta a Internet Explorer hay que decir que todavía no soporta este atributo de CSS 3, pero esperemos que los chicos de Microsoft lo puedan implementar pronto en el navegador, o que las especificaciones de CSS3 pasen rápido al estado "Candidate Recommendation", que sería el llamamiento para que se implementen en la generalidad de los navegadores.

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.

Podemos ver el ejemplo en una página aparte. Pero recordar que sólo se verá el efecto utilizando Firefox o navegadores basados en Mozilla.

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)

Podemos ver este ejemplo en una página aparte.

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.

Otra cosa que definen las especificaciones de CSS y que por el momento no está funcionando del todo bien, es que las imágenes de fondo deben ajustarse a los bordes redondeados. Ocurre, por el momento, que las imágenes de fondo salen por fuera de los bordes redondeados. Confiamos en que en el futuro esto pueda ser revisado y optimizado, para que todo funcione correctamente. Debemos recordar que en el momento de escribir el artículo todavía se tienen que terminar de definir las especificaciones de CSS 3 y después, los navegadores web deberán actualizarse en un cierto espacio de tiempo para soportarlas completamente.

Nota: Ofrecimos una lista de las características principales de CSS 3 en el artículo Introducción a CSS3.

Este artículo es obra de *Miguel Angel Alvarez*
Fue publicado por primera vez en *10/07/2008*
Disponible online en *http://desarrolloweb.com/articulos/bordes-redondeados-en-css-3.html*

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). Pero en este artículo de DesarrolloWeb.com vamos a mostrar cómo podríamos conseguir que un elemento de la web tuvisese varias imágenes de fondo al mismo tiempo. Colocar varias imágenes de fondo a un elemento en principio no se puede con CSS, por lo que mostraremos cómo hacerlo con una sencilla técnica alternativa. Pero también veremos que en CSS 3 existe la posibilidad de configurar varios fondos al mismo tiempo en un elemento, con una declaración de background-image y sin necesidad de realizar ningún tipo de técnica alternativa.

Referencia: Para conocer un poco de la reciente especificación de CSS3 os vendrá bien leer el artículo Introducción a CSS 3.

En CSS normal (CSS 1), como hemos dicho, podemos colocar un único fondo a un elemento de la página. Esto es algo soportado por todos los navegadores. Como no podemos poner más de 1

fondo por elemento, para colocar varios fondos al mismo tiempo, tenemos que utilizar varios elementos. A cada elemento le colocaremos un único fondo, pero al mostrarse los elementos en el mismo espacio, conseguiremos el efecto deseado de tener varios fondos de imagen a la vez.

En nuestro caso, vamos a utilizar varias capas DIV anidadas y cada una tendrá su fondo de imagen.

El código HTML que utilizaríamos para anidar varias capas DIV sería como sigue:

```
<div id="fondo1">
   <div id="fondo2">
      <div id="fondo3">
         contenido del elemento que va a tener 3 fondos de imagen distintos
         ...
      </div>
   </div>
</div>
```

Como se ha visto, podemos anidar varias capas DIV y cada una tendrá un identificador, o si lo preferimos una clase, para asignar estilos por CSS. Al estar anidados, todos los elementos DIV se mostrarán uno encima del otro.

Ahora veamos el código CSS para darle fondos a cada uno de estos elementos DIV:

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

Los fondos se superpondrán unos a otros, siendo el fondo1 el que se vea más abajo y el fondo3 el que se verá más arriba.

El resultado de este ejemplo se puede ver en una página aparte.

En principio todos los navegadores visualizarán perfectamente los fondos, pero como he utilizado imágenes transparentes en formato PNG e Internet Explorer 6 tiene problemas con la transparencia de los archivos PNG, podemos encontrar algún defecto, pero los fondos de los

elementos DIV se verán.

Colocar varios fondos de imagen a un elemento con CSS 3

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.

De momento, la posibilidad de incluir varios fondos de imagen a un elemento sólo está disponible en el navegador Safari, pero esperemos que pronto otros navegadores vayan incorporando esta funcionalidad de CSS 3.

Este ejemplo se puede ver en una página aparte.

Este artículo es obra de *Miguel Angel Alvarez*
Fue publicado por primera vez en *22/12/2008*
Disponible online en *http://desarrolloweb.com/articulos/multiples-imagenes-de-fondo-con-css.html*

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.

> **Nota:** En DesarrolloWeb.com hemos publicado varios artículos sobre CSS 3, que se pueden acceder desde el siguiente enlace: introducción a CSS3.

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>
```

Pero aparte de estos ejemplos, lo ideal es ver el efecto de transparencia en marcha, para poder hacernos una idea más aproximada de las posibilidades. Para ello hemos construido una página donde se muestran varias capas con colores y transparencias, tanto en el fondo como en el texto.

Ver ejemplo de colores CSS RGBA.

Pero nuevamente llamamos la atención de los lectores sobre el hecho que, dependiendo del navegador que se utilice, se verán o no los distintos colores, puesto que las CSS 3 no son compatibles con todos los sistemas. Este ejemplo funcionará bien en navegadores que cumplen los estándares, como Firefox, Chrome o Safari, pero no en navegadores como Internet Explorer, que hacen la guerra por su cuenta.

El código del anterior ejemplo es el siguiente:

```
<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>
```

Si se permite mi opinión, es una pena que esta gestión de color con canal alpha no esté disponible en Internet Explorer 8, en el momento de escribir estas líneas, porque así se hace complicado usar este tipo de efectos. No obstante, es de suponer que las personas de Microsoft pondrán al día su navegador un año de estos, para hacerlo compatible con las CSS 3 y los colores RGBA.

Este artículo es obra de *Miguel Angel Alvarez*
Fue publicado por primera vez en *25/08/2009*
Disponible online en *http://desarrolloweb.com/articulos/colores-rgba-css-3.html*

Word-wrap y overflow-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. Atributos word-wrap y overflow-wrap.

Estamos dando un repaso a las novedades que traerá la especificación CSS 3 y en este caso vamos a ver una propiedad interesante que servirá para que ciertas palabras que son demasiado largas y no caben en la anchura de una caja.

Como debemos saber, en el modelo de caja de los navegadores, las palabras se van colocando en líneas para ocupar toda la anchura disponible en la caja. Puede surgir un problema cuando tenemos una palabra muy larga, que no cabe en el ancho disponible del contenedor donde se

ha colocado. Entonces lo que ocurre es que la palabra aparece por fuera de la caja, o incluso puede ocurrir que la caja se deforma de tamaño, haciendo que el elemento contenedor amplíe la anchura lo suficiente para que quepa la palabra. En cualquier caso, el efecto resultante suele ser poco agradable, porque muchas veces nos descuadra nuestro diseño y hace que las páginas queden mal maquetadas.

Para evitar este efecto, en CSS 3 se ha incluido un atributo llamado word-wrap que sirve para especificar que las palabras que sean demasiado largas se deben cortar, de manera que quepan en el ancho disponible de la caja. Una propiedad muy útil que seguro que pronto se comenzará a utilizarse habitualmente.

> **Nota:** Hemos de agradecer a Microsoft la incorporación de esta nueva propiedad de CSS 3, ya que el atributo word-wrap comenzó siendo una etiqueta no estándar de CSS, que estaba disponible en Internet Explorer y debido a su utilidad, el W3C se decidió a incorporarla a la nueva especificación de este lenguaje de estilo.

Alternativas overflow-wrap y word-wrap

ACTUALIZADO EN 2016: En el primer borrador de la especificación de CSS3 al atributo para la ruptura de palabras se le conocía como word-wrap, sin embargo, en más recientes borradores se alteró su nombre para overflow-wrap. El uso es exactamente el mismo, sin embargo ha surgido un problema de soporte que a día de hoy todavía se arrastra, puesto que navegadores entienden un atributo y otros entienden otro.

Por tanto, la recomendación hoy para nosotros sería usar overflow-wrap siempre, pero mantener el antiguo atributo word-wrap en el código CSS, a modo de fallback, para que todos los navegadores interpreten uno u otro.

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

```
overflow-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.

Este texto entra bien en la capa, pero ahora vamos a colocar una palabra demasiado larga que no cabrá, por lo que nos descuadraría el diseño: wieiisiddjasddjkjasdasdsadfdsfsdfsfsdfsdsdfsdfkaldkadadsadadadadsad. Esta capa tiene 300 píxeles de anchura y esa palabra, por tanto no cabía y me descuadra todo.

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

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

Este atributo es actualmente soportado por todos los navegadores, tanto de ordenadores de escritorio como de móviles. Los navegadores de Microsoft Internet Explorer / Edge todavía soportan únicamente word-wrap, en octubre de 2016.

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

Esta otra capa tiene el atributo word-wrap: break-word y por tanto va a recortar la siguiente palabra para que quepa bien en la caja: wieiisiddjasddjkjasdasdsadfdsfsdfsfsdfsdsdfsdfkald kadadsadadadadsad. Ahora la capa no se ve afectada por una palabra tan larga.

Este artículo es obra de *Miguel Angel Alvarez*
Fue publicado por primera vez en *05/02/2009*
Disponible online en *http://desarrolloweb.com/articulos/word-wrap-css3.html*

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.

En este artículo vamos a tratar sobre las nuevas características de CSS para maquetación automática de textos en columnas, que se encontrarán disponibles cuando CSS 3 se implemente finalmente en los navegadores. Aunque todo esto no pase de un estado experimental, lo cierto es que ya podemos empezar a probarlo y utilizar en navegadores modernos, como muchas de las nuevas posibilidades que estamos comentando sobre CSS 3

Numerosas publicaciones maquetan el texto en diversas columnas, como criterio de diseño y para facilitar la lectura. Podríamos imaginarnos el texto de los periódicos si no se encontrase dividido en diversas columnas ¿no sería casi imposible seguir las líneas para hacer una lectura cómoda de la información si todo estuviera en una única columna?

Este problema no lo encontramos generalmente en las páginas web, porque el texto que cabe en el cuerpo de la página no es tan grande como el que cabría en una hoja de un diario. Además, generalmente la propia página ya se encuentra dividida en diversas columnas. No obstante, aunque quisiéramos, con las características de CSS 2 no sería muy fácil hacer un texto fluído que se adaptase automáticamente en columnas, de modo que éstas crecieran homogéneamente a medida que el texto crece o se reduce. CSS 3 soluciona este problema, o

más bien ofrecerá una solución al mismo, sencilla y automática.

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.

Podemos ver una página aparte con los ejemplos mostrados de maquetación en múltiples columnas. Pero recordad que sólo funcionará en Mozilla Firefox, Apple Safari y Google Chrome.

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/

Esperamos que estas características de CSS 3 se encuentren pronto disponibles, pues no cabe duda que la distribución por columnas nos abrirá nuevas e interesantes posibilidades para maquetación de textos en páginas web.

Este artículo es obra de *Miguel Angel Alvarez*
Fue publicado por primera vez en *09/10/2009*
Disponible online en *http://desarrolloweb.com/articulos/texto-multi-columna-css3.html*

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.

Vamos a hablar rápidamente sobre uno de los nuevos atributos de las CSS 3, que sirve para personalizar los bordes de los elementos HTML con imágenes. Publicamos este artículo dentro del Manual de CSS 3, que es una especie de compendio de ejemplos y técnicas que nos permitirá aplicar esta nueva especificación de las Hojas de Estilo en Cascada.

De manera resumida, border-image es un atributo que nos ayudará a aplicar nuevos estilos a las cajas con CSS, a través de la utilización de imágenes en los bordes de los elementos, incluso pudiendo escoger varias imágenes para varias de las partes de los bordes.

Colocar imágenes en los bordes es una tarea que ya se suele realizar en el diseño web, y para ello se suelen utilizar complementariamente técnicas con los lenguajes HTML y CSS. Es decir, en el momento de escribir este artículo, para que un elemento de la página, como puede ser una división o una tabla, párrafo, etc., tenga un borde a partir de una imagen, se necesita colocar algún código HTML adicional, con algún contenedor que nos permita luego definir estilos CSS para "imitar" ese borde de imagen. En cualquier caso, estemos o no familiarizados con las técnicas de uso de imágenes en los bordes, lo importante es que con CSS 3 vamos a poder hacer eso mismo simplemente escribiendo algunos nuevos atributos a los elementos que deseemos.

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.

Para las personas que deseen encontrar de primera mano las especificaciones de los atributos

para poner borde en las imágenes, recomendamos la lectura del sitio de la W3C.

Borrador de trabajo sobre las propiedades de borde de imagen. Noviembre de 2002 Borrador sobre borde de imagen. Diciembre 2009

Como se podrá ver, existen bastantes diferencias y es que también ha pasado bastante tiempo entre una y otra especificación. Además, la más actual de las dos tiene unas explicaciones basadas en esquemas mucho más entendibles.

Del mismo modo, tenemos acceso a las especificaciones que han puesto en marcha los más avanzados navegadores, que implementan ya de manera experimental algunos de los atributos para crear bordes en las imágenes.

Sitio para desarrolladores de Mozilla, sobre Border Image. Sitio para desarrolladores de Mac y su navegador Safari.

Ejemplo de border-image

Así pues, para que sirva únicamente a modo de demostración que es todo lo que se puede hacer por el momento en este artículo de DesarrolloWeb.com, vamos a poner un simple ejemplo sobre lo que permiten a fecha de hoy los navegadores Safari y Firefox sobre 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:

Y el ejemplo se puede ver en marcha en una página aparte, pero recordar que depende de vuestro navegador podréis ver o no el borde de imagen.

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.

Explicar con ejemplos cada uno de estos atributos sería sin duda interesante, pero nos llevaría varios artículos y además, merece la pena hacerlo cuando ya estén disponibles como especificación definitiva de CSS 3 y no en un simple borrador, así como cuando estén implementados en los navegadores.

Este artículo es obra de *Miguel Angel Alvarez*
Fue publicado por primera vez en *04/01/2010*
Disponible online en *http://desarrolloweb.com/articulos/bordes-imagenes-css3.html*

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.

Como vimos en el artículo introducción a CSS 3, No cabe duda que significará un grande avance para las personas que nos dedicamos a desarrollar webs, que nos permitirá ahorrar tiempo y sobre todo, economizar código fuente, lo que hará las páginas más ligeras y separará aun más el contenido de la forma.

A lo largo del Manual de CSS 3 ya vimos varios atributos nuevos de bastante importancia y variedad y ahora le toca el turno a box-shadow, el atributo de CSS3 que nos permitirá definir sombras a los elementos de la página.

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.

Para acabar, ponemos un enlace a una página donde puedes ver los ejemplos de sombras en CSS 3.

Ten en cuenta que debes probar estos ejemplos con Opera, Firefox, Safari o Chrome, que son los que actualmente soportan este atributo.

Si deseas ampliar las informaciones y practicar más con este nuevo atributo de CSS3 para la realización de sombras, te recomendamos leer el artículo sobre cómo hacer un resplandor exterior con CSS 3.

Este artículo es obra de *Miguel Angel Alvarez*
Fue publicado por primera vez en *22/03/2010*
Disponible online en *http://desarrolloweb.com/articulos/sombras-css3-box-shadow.html*

Resplandor exterior con CSS3

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

Hace ya tiempo que se publicó en DesarrolloWeb.com un análisis de las capacidades de la propiedad box-shadow, que sirve para producir una sombra en un contenedor, por medio de CSS. Es una de esas propiedades que vienen como novedad en la tercera especificación de CSS y que nos vendrán como anillo al dedo para conseguir ciertos efectos que antes sólo podíamos emular mediante el uso de imágenes semi-transparentes por canal alpha.

Así pues, lo que antes era un verdadero engorro y peso adicional innecesario para nuestras páginas, ahora se puede hacer con unas sencillas propiedades de CSS 3. Como decíamos, ya explicamos la propiedad box-shadow de CSS 3 y vimos ejemplos para utilizarla de modo que podamos producir sombras. En este artículo veremos una pequeña práctica para utilizar esa misma propiedad para obtener otro estilo de capa diferente, que es el resplandor exterior.

Nota: Algunas de estas propiedades de CSS 3 me recuerdan irremediablemente a los estilos de capa de Photoshop. A este efecto le he llamado "resplandor exterior" justamente por ser el nombre con el que le llaman en Photoshop, en la configuración de los estilos de capa.

Pues resulta que un resplandor exterior es justamente una sombra, pero en lugar de oscura, de color claro. Así que, haciendo uso de la misma propiedad CSS para la creación de sombras, podemos crear los resplandores que deseemos. Eso si, tendremos que colocar un fondo oscuro a nuestra página para verlos, por obvias razones de contraste.

En realidad estas son las propiedades CSS que necesitaremos para crear una sombra, pero de color blanco, con lo que se obtendrá el mencionado resplandor exterior.

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

La primera propiedad -moz-box-shadow sirve para Firefox, la segunda -webkit-box-shadow es para Safari y Chrome y la tercera propiedad box-shadow sirve para todos los navegadores que ya han incorporado CSS 3 como si fuera un estándar ya aprobado, como Opera e Internet Explorer 9.

> **Nota:** Más adelante, cuando CSS 3 esté en estado de implementación recomendada por el W3C, podremos usar solamente la propiedad box-shadow, que será compatible con todos los navegadores.

Si queremos hacer el resplandor con otro color, por ejemplo verde, tendríamos que cambiar únicamente el RGB de las propiedades CSS, tal como se puede ver a continuación.

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

Hemos preparado un ejemplo de página web donde he creado un par de resplandores. Como se puede ver, he colocado un fondo de página negro, para que los resplandores se puedan ver correctamente. Asimismo, he colocado algunos estilos adicionales, como un borde para resaltar el resplandor o un padding para hacer que la caja se vea un poquito más holgada.

Para que queden claros todos los estilos CSS que hemos definido para la realización de este ejemplo, vamos a mostrar a continuación el código fuente completo generado.

```
<!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">
        Esta capa tiene un resplandor exterior!!
    </div>

    <br>
    <br>

    <div id="resplandorverde">
        Esta capa tiene un resplandor exterior, pero ahora he utilizado un resplandor de color verde!!
    </div>
</body>
</html>
```

Para finalizar, podemos ver el ejemplo en marcha en una página aparte.

Este artículo es obra de *Jhon Nadie*
Fue publicado por primera vez en *13/01/2011*
Disponible online en *http://desarrolloweb.com/articulos/resplandor-exterior-css3.html*

Cómo hacer con CSS3 un bocadillo de cómic o globo de historieta

Por medio de formas CSS podemos implementar simplemente con una capa y estilos CSS3, los típicos globos de historietas o bocadillos de cómic.

En este artículo vamos a hacer un taller práctico de capacidades de CSS3 para realizar estilos con distintas formas, que nos pueden ayudar a crear esas bolas que sirven para incluir los textos con los que hablan los personajes en los cómics. En España yo he llamado siempre a estos recuadros "bocadillos" y creo que así los llaman la mayoría de las personas, pero también se pueden conocer como "globos". En inglés son "speech balloon", supongo que todos sabrán a lo que me refiero.

Antes de CSS3 no quedaba otro remedio que implementarlas con trucos que incluían diversas capas en el HTML y el uso de imágenes PNG, o GIF con fondos transparentes. Ya sabemos que esa no era la mejor solución, pues en el HTML tenemos que mantener simplemente el contenido y dejar la forma o el aspecto para CSS.

Ahora podemos usar algunas técnicas, medianamente sencillas, que nos ayudarán a realizar en el HTML una única unidad de contenido con el texto del bocadillo y por medio de CSS3 especificar el estilo para que se parezca a lo que sería el globo del cómic o historieta.

Para ver a lo que me refiero, simplemente podemos ver el ejemplo en funcionamiento con una presentación de varias capas con ejemplos de globos de cómic.

La técnica, formas en CSS3, triángulo + caja redondeada

Debo de reconocer que la técnica no es mía, sino que la he visto hacer y la he podido modificar un poco para adaptarla a mis necesidades. La idea original sobre cómo hacerla la tienes en CSS3 Shapes donde tienes un recurso con varios tipos de formas en CSS3 y la manera de realizarlas.

Comenzamos haciendo un triángulo con CSS3: El ejemplo del bocadillo está compuesto en verdad por dos formas distintas. Un triángulo y un cuadrado con esquinas redondeadas. Comenzamos haciendo el triángulo que es más sencillo.

Lo componemos a partir de una capa vacía:

```
<div id="triangulo">
</div>
```

Y aplicando estilos CSS en los que básicamente jugamos con los bordes, para producir las esquinas del triángulo. Si te fijas, tanto la altura como la anchura están a cero y con los bordes conseguimos definir los tres lados.

```
#triangulo{
width: 0;
height: 0;
borderbottom: 30px solid #55f;
borderright: 20px solid transparent;
```

```
borderleft: 20px solid transparent;
}
```

Este triángulo define su altura con el tamaño del "border bottom" y su anchura por los bordes de la derecha e izquierda en suma. Para que tenga lados iguales, la anchura por la derecha tiene que ser igual a la anchura por la izquerda. En realidad, es un triángulo isósceles con el ángulo más agudo el de arriba, lo que da la impresión de apuntar hacia arriba, aunque en la pantalla del ordenador se ve casi como un equilátero.

Jugando con las propiedades de "border" puedes cambiar fácilmente las características del triángulo. Puedes probar a cambiar las dimensiones de los bordes para producir otros ángulos. O bien puedes cambiar por ejemplo "border bottom" por "border top" para que el triángulo apunte hacia abajo en vez de apuntar hacia arriba.

######### La caja redondeada:

Es tan facil hacerla como usar una capa con borde redondeado, gracias al "borderradius" de CSS3.

```
<div id="bocadillo">
Probando el bocadillo o "globo de hablar", o como le llames.
</div>
```

Su CSS sería el siguiente:

```
#bocadillo {
padding: 10px;
width: 220px;
borderradius: 10px;
backgroundcolor: #f80;
fontfamily: "trebuchet ms", tahoma, verdana, sansserif;
fontsize: 1.4em;
}
```

Hay poco que comentar aquí. Simplemente observar el "border radius" y ajustar sus valores al gusto de cada uno o a las necesidades del diseño.

######### Técnica ":before":

La parte más interesante de todo esto es usar la técnica ":before" de CSS2 y que ya podemos usar con libertad casi en todos los navegadores (quitando los más viejos Internet Explorer que cada día menos deberían preocuparnos, aunque eso es otra discusión).

> **Nota:** Los ":before" o ":after" son unos pseudo-elementos de CSS2 que ya fueron explicados en otro artículo en DesarrolloWeb.com. Pseudo element en CSS (pseudo-elementos)

En nuestro caso la situación es simple: partimos de una capa del bocadillo, donde introducimos el texto que queremos que haya dentro del globo. Esa capa tiene simplemente los cantos redondeados. Luego colocamos con el pseudo-elemento :before lo que sería el triángulo del globo.

```
#bocadillo:before {
content:"";
position: absolute;
width: 0;
height: 0;
bordertop: 20px solid transparent;
borderright: 30px solid #f80;
borderbottom: 20px solid transparent;
margin: 10px 0 0 32px;
}
```

Como ves, con :before indicamos que nos coloque un texto antes del elemento "#bocadillo". El contenido será la cadena vacía (pues para hacer el triángulo partimos de una capa vacía). Luego colocamos los estilos CSS que necesitamos para hacer el triángulo, descritos anteriormente, más un margen para colocar el triángulo en el lugar desdeado con respecto al globo de texto.

Cambiar el lugar donde está la "punta" del bocadillo

A partir de aquí es bastante sencillo alterar nuestro bocadillo para que la punta esté en diferentes lugares, de modo que podamos maquetarlo apuntando hacia donde necesitemos. Ahora vamos a hacer un globo que tiene la punta hacia arriba.

Partimos siempre de una caja que contiene el texto que queremos que haya en el globo de texto.

```
<div id="bocadilloarriba">
Probando otro bocadillo con el triangulo arriba.
</div>
```

Luego aplicamos nuestros estilos CSS para convertirlo en un típico recuadro el texto hablado en un cómic o historieta.

```
#bocadilloarriba {
padding: 10px;
width: 220px;
borderradius: 10px;
backgroundcolor: #8f0;
fontfamily: "trebuchet ms", tahoma, verdana, sansserif;
fontsize: 1.4em;
}
```

```
#bocadilloarriba:before {
content:"";
position: absolute;
width: 0;
height: 0;
borderbottom: 30px solid #8f0;
borderright: 20px solid transparent;
borderleft: 20px solid transparent;
margin: 30px 0 0 90px;
}
```

¡Eso es todo! esperamos que sea útil para vosotros. Las aplicaciones son las que se deseen, pero creo que una de las más interesantes es para hacer divertidos "tip" o "tooptip" que podemos mostrar y ocultar para señalar cualquier tipo de información contextual, o remarcar de una manera simpática algunos elementos de nuestra página.

Lo bueno de esta técnica es que no mezclas el contenido con la presentación. Simplemente le colocas una capa normal y corriente con un texto y todos los estilos para la presentación los haces por medio de CSS3, de manera totalmente autónoma al contenido.

Este artículo es obra de *Miguel Angel Alvarez*
Fue publicado por primera vez en *21/06/2013*
Disponible online en *http://desarrolloweb.com/articulos/css3-bocadillo-comic-globo-hitorieta.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.

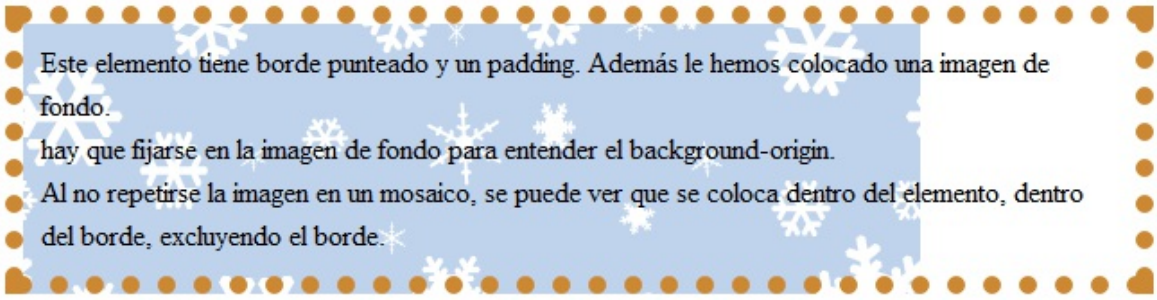
En este artículo exploraremos el nuevo atributo background-origin, que forma parte de la nueva especificación de Hojas de Estilo en Cascada. Veremos también ejemplos sobre cómo utilizar este atributo, dentro de las explicaciones que estamos presentando en el Manual de CSS 3.

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;
```

Para hacernos una idea más clara sobre los distintos valores de background-origin hemos

hecho un ejemplo, pero se ha de ver con navegadores compatibles con CSS (en seguida explicamos esto).

Ver un ejemplo en marcha en una página aparte.

> **Nota:** Si background-attachment tiene el valor "fixed" este atributo se ignorará.

Compatibilidad de background-origin

En el momento de escribir este artículo los navegadores aun no han implementado todas las nuevas características de las CSS. En la mayoría de casos apenas están a modo de prueba, hasta que las especificaciones de CSS 3 estén recomendadas para su implementación.

Por ello, los navegadores que han comenzado a soportar las CSS utilizan algunos prefijos para los nombres de las propiedades. En Mozilla Firefox debemos utilizar el atributo -moz-background-origin, en navegadores basados en webkit (como Chrome o Safari) se debe utilizar el atributo -webkit-background-origin.

Además, en navegadores basados en webkit y Mozilla no tienen en cuenta la terminación "-box" al final de los valores de los atributos, por lo que hay que eliminarla.

Por su parte, en Opera ya tienen implementada la funcionalidad con el nombre de atributo definitivo, y sus valores con la correspondiente terminación "-box", que es como aparece en la especificación actual del modelo de caja de CSS 3.

Este sería el código CSS para aplicar distintos valores de background-origin y que los entiendan todos los navegadores más modernos.

> **Nota:** El Internet Explorer 8, el más actual de los navegadores de Microsoft, en estos momentos, las CSS 3 no están implementadas de ningún modo, así que no podremos ver estos ejemplos en funcionamiento.

```
#pruebasfondo1{
   -moz-background-origin: border;
   -webkit-background-origin: border;
   background-origin: border-box;
}
#pruebasfondo2{
   -moz-background-origin: padding;
   -webkit-background-origin: padding;
   background-origin: padding-box;
}
#pruebasfondo3{
   -moz-background-origin: content;
```

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

Así pues, ya hemos aprendido a cambiar el eje de posicionamiento de la imagen de fondo con background-origin, pero volvemos a comentar que la posición de la imagen definitiva también la podemos marcar con background-position. De hecho, con el atributo background-origin estamos definiendo el eje de coordenadas sobre el que se va a aplicar el background-position para colocar definitivamente la imagen.

Para terminar, ponemos de nuevo el enlace al ejemplo marcha.

Este artículo es obra de *Miguel Angel Alvarez*
Fue publicado por primera vez en *27/05/2010*
Disponible online en *http://desarrolloweb.com/articulos/background-origin-css.html*

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.

En este artículo nos disponemos a analizar los atributos de hojas de estilo en cascada, en su tercera versión (CSS3), overflow-x y overflow-y. Estas son dos de las novedades de este lenguaje para definición de estilos en páginas web, que estamos comentando en el Manual de CSS3.

Realmente, aunque se trata de dos atributos nuevos en CSS3, el concepto de overflow no resultará tan novedoso para los estudiosos de las hojas de estilo, puesto que ya viene de versiones anteriores. En CSS2 se incorporó el atributo overflow que, como quizás sepas, sirve para indicar al navegador qué hacer cuando un contenido sobrepasa los límites de un contenedor. Sin embargo, ambos atributos ya estaban en el tintero del W3C desde hace tiempo, tanto es así que navegadores tan antiguos como Internet Explorer 6, tienen una implementación parcial de overflow-x y overflow-y , a pesar de haber sido incluidos formalmente en CSS a partir de la tercera versión del lenguaje.

Como habrás podido experimentar, de manera predeterminada, cuando un contenido sobrepasa los límites de las dimensiones de la capa donde está situado, el navegador lo muestra igualmente. En algunos casos, como podrás ver en Internet Explorer (al menos en versiones antiguas), ajusta las dimensiones de esa capa contenedora para que ese contenido se pueda ver en el navegador. En otros casos muestra igualmente el contenido, aunque fuera del área del contenedor (sin modificar las dimensiones del contenedor, como hace Firefox o Chrome). Pero ese comportamiento no es siempre el que deseamos y para ello se utiliza el atributo overflow.

Pues bien, overflow-x y overflow-y sirven para exactamente lo mismo que overflow, pero con la

diferencia que permiten especificar los comportamientos del navegador por separado, cuando surge un desbordamiento de un contenido en la horizontal y en la vertical respectivamente.

> **Referencia:** Nos hemos apoyado en la descripción de overflow para explicar lo que significa overflow-x y overflow-y. No obstante, si esa regla de estilo resulta nueva para ti y deseas entenderla mejor, te recomendamos leer el artículo El atributo Overflow de CSS.

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.

Merece la pena comentar un detalle sobre el comportamiento predeterminado de estos atributos. Como se dijo, en caso que no se especifique nada en overflow-x u overflow-y, sería como si hubiésemos aplicado el valor visible. Sin embargo, si definimos solamente uno de

ellos, el otro ya no tendría visible como valor predeterminado, sino auto.

Podemos ver varios ejemplos de la combinación de varios valores de overflow-x y de overflow-y.

Este artículo es obra de *Miguel Angel Alvarez*
Fue publicado por primera vez en *01/11/2011*
Disponible online en _http://desarrolloweb.com/articulos/overflow_x-overflow_y-css3.html_

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.

A la hora de escoger una fuente para usar en una página web, tradicionalmente se encontraba la limitación de que el usuario tuviera ese tipo de letra instalada en su ordenador, porque de no ser así, los textos se mostrarían con otras tipografías, distintas a las que habíamos elegido. Es por ello que el abanico de fuentes que podíamos utilizar, con la garantía que funcionasen bien en la mayoría de los visitantes, estaba bien reducido a las típicas Arial, Verdana, Times New Roman y poco más.

Bien, pues en la actualidad este problema ya esta solucionado y todo gracias a la regla CSS @font-face. Esta regla nos permite definir en nuestra hoja de estilos **cualquier tipo de tipografía, independientemente de si el usuario dispone de ella o no** y para ello lo único que debe preocuparnos es que este instalada en nuestro servidor. En el presente artículo explicaremos cómo se puede configurar nuestra hoja de estilos CSS para poder utilizar cualquier fuente que nosotros deseemos.

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.

Nota: No todos los navegadores admiten todos los formatos anteriormente citados.

- Firefox 3,6 soporta Opentype, Embedded Opentype y WOFF.
- I.E. soporta Embedded Opentype
- Opera soporta Truetype, Embedded Opentype y Opentype
- Safari 3,1 soporta Truetype, Embedded Opentype y Opentype
- Google Chrome soporta Truetype, Embedded Opentype

Si queremos trabajar con fuentes distintas a través de @font-face y nos preocupa la compatibilidad con todos los navegadores, comprobaremos que lo más aconsejable es subir las fuentes con formato .eot, que funciona en todos los navegadores. Esto es debido a que IE tan sólo soporta el formato .eot. Por ello, debemos convertir nuestras fuentes a dicho formato, para lo que podemos utilizar el programa Microsoft Weft, cuyo funcionamiento veremos detalladamente en otro artículo.

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>
```

Puedes ver el resultado en una página aparte.

Este artículo es obra de *Sara Alvarez*
Fue publicado por primera vez en *14/09/2010*
Disponible online en *http://desarrolloweb.com/articulos/intro-font-face.html*

CSS 3. Nuevas funcionalidades de fuentes. Ligaduras

Analizamos las nuevas funcionalidades de fuentes disponibles en CSS 3 y mediante el formato de fuente OpenType.

Abarcaremos aspectos diversos entre los que destacan la disponibilidad de mayor número de tipos y la presencia de ligaduras.

En concreto, en el presente artículo, trataremos:

- Fuentes *OpenType*, especificaciones y tipos de archivos.
- Soporte en los diversos navegadores.
- Cómo incluir las nuevas funcionalidades –CSS 3-.
- Las ligaduras y su presencia en las fuentes.
- Las fuentes y su documentación.
- Un ejemplo completo para probar y ver el efecto de los diversos tipos de ligaduras.

Nuevas funcionalidades de fuentes

Hay unas cuantas funcionalidades de fuentes que estarán presentes en los nuevos tipos y que conviene entender su significado, al menos de forma elemental.

Los *glifos* son la representación gráfica de un carácter, de varios o de la parte de uno, según los casos; así, una letra puede estar representada por varios glifos o la agrupación de dos letras por un glifo en un set de caracteres. Una fuente de caracteres se valorará en parte por el número de glifos que contenga –mejor cuantos más-. Véase: es.wikipedia.org/wiki/Glifo

Respecto a las *ligaduras*, es un signo formado por la unión de dos o más, por ejemplo @, que resulta de at. Véanse los enlaces que siguen, el primero para una descripción elemental, el segundo para lo referente a la especificación oficial del W3C –*World Wide Web Consortium*-: es.wikipedia.org/wiki/Ligadura_(tipograf%C3%ADa)) - www.w3.org/TR/css3-fonts/#font-variant-ligatures-prop

Por último, como referencia base para todo lo relativo al trabajo con fuentes, en la especificación CSS 3, deberemos acudir a la fuente oficial del W3C: www.w3.org/TR/css3-fonts

Fuentes OpenType -otf-, eot y woff

Todas ellas son fuentes *OpenType*. Estas fuentes son derivadas de las conocidas fuentes *TrueType*, a las que mejoran en diversos aspectos como la incorporación de un juego más amplio de caracteres, el uso de tipos pequeñas mayúsculas –*small caps*-, formas más detalladas y diversas con la incorporación de ligaduras y glifos y opciones de presentación de números.

Para una descripción sencilla de las fuentes OpenType se puede acudir a: es.wikipedia.org/wiki/OpenType

Los formatos de fuentes OpenType disponibles son:

- OTF: Es el formato base de un archivo de fuentes *OpenType (OpenType Font)*.
- EOT: Son una forma compacta de fuentes *OpenType*, diseñadas por Microsoft para ser utilizadas de forma embebida en documentos Web (*Embedded OpenType*). Posteriormente fue remitido al W3C para su aceptación: www.w3.org/Submission/2008/01.
- WOFF: Es un nuevo formato de fuente, desarrollado en 2009 por el W3C; consta esencialmente de una fuente *TrueType o una OpenType* y metadatos adicionales y

comprimido, para favorecer la distribución desde el servidor (*Web Open Font Format*): www.w3.org/TR/WOFF.

Todo lo referente a la especificación puede consultarse en la siguiente dirección URL: www.microsoft.com/typography/otspec

Una lista completa de todas las capacidades soportadas se encuentra en: www.microsoft.com/typography/otspec/featurelist.htm

Soporte en los diferentes navegadores

En el momento presente todos los navegadores importantes soportan, de una u otra forma –en sus versiones más recientes- los nuevos tipos de fuentes. En la tabla que sigue se muestran por tipos de formatos los que pueden emplearse en cada caso:

	OTF	EOF	WOFF
IE	x	x	x
Firefox	x	-	x
Opera	x	-	x
Opera Mobile	x	-	x
Chrome	x	-	x
Safari	x	-	x
Safari (iOS)	-	-	x
Android Browser	x	-	-

(Fuente: caniuse.com/#feat=fontface)

Cómo incluir las nuevas capacidades en los estilos

Se trata de emplear la regla de CSS 3 "font-feature-settings":

Regla	Descripción
font-feature-settings	Toma como parámetro una etiqueta o etiquetas separadas por comas y entre comillas. Véase seguidamente <feature-tag-value>.

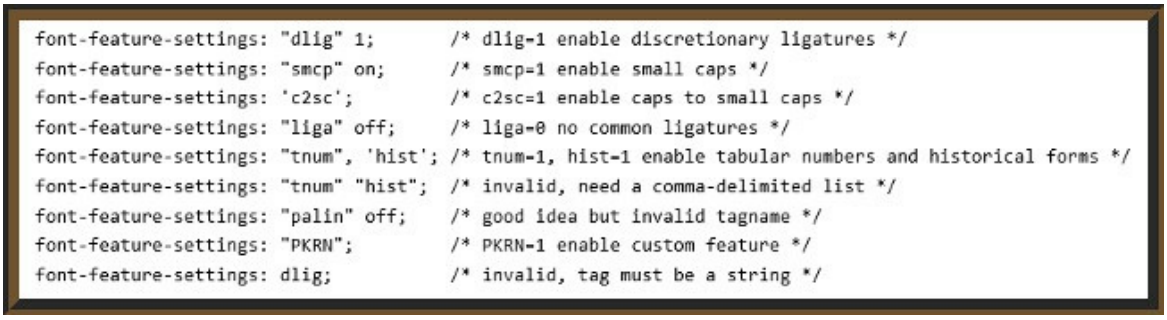
Para una referencia completa acúdase a: www.w3.org/TR/css3-fonts/#propdef-font-feature-settings

Con respecto a la sintaxis de la etiqueta de la regla, es:

```
<feature-tag-value> = <string> [<integer> | on | off ]
```

Es decir:

- En una cadena de texto, si hay varias directivas, deben estar separadas por comas. Véase por ejemplo el Listado 2.
- En su caso, un valor numérico si tiene significado para la etiqueta –dependerá de las especificaciones de la propia etiqueta y de la fuente en concreto-.
- La presencia de "on" u "off" si deseamos activar o desactivar lo que implica la o las etiquetas. Por defecto es el valor de "on", que podremos obviar.



```
font-feature-settings: "dlig" 1;        /* dlig=1 enable discretionary ligatures */
font-feature-settings: "smcp" on;       /* smcp=1 enable small caps */
font-feature-settings: 'c2sc';          /* c2sc=1 enable caps to small caps */
font-feature-settings: "liga" off;      /* liga=0 no common ligatures */
font-feature-settings: "tnum", 'hist'; /* tnum=1, hist=1 enable tabular numbers and historical forms */
font-feature-settings: "tnum" "hist";  /* invalid, need a comma-delimited list */
font-feature-settings: "palin" off;     /* good idea but invalid tagname */
font-feature-settings: "PKRN";          /* PKRN=1 enable custom feature */
font-feature-settings: dlig;            /* invalid, tag must be a string */
```

(Fuente: W3C)

Para una fuente instalada en el sistema, el código sería similar al del listado que sigue, para una fuente tipo *Gabriola*. Aquí se ha explicitado una ligadura *ss07*, algo que más abajo comentaremos.

Listado 1: Implementación de una fuente con ligaduras, caso de estar instalada en el sistema.

```
<style>
h2 {
font-family: Gabriola;
}

p.poesia {
font-family: Gabriola;
font-style: italic;
-moz-font-feature-settings: "ss07";
font-feature-settings: "ss07";
}

p.center {
text-align: center;
}
</style>
```

Para una fuente alojada en un archivo de fuentes, el código sería similar al del listado que sigue, obsérvese que se da un nombre a la fuente y la referencia del archivo o archivos con sus diversos tipos –este es opcional para los navegadores más actualizados-, tal como se comenta y se especifican los tipos en el apartado del enlace que se da y siguientes en el mismo texto: www.w3.org/TR/css3-fonts/#at-font-face-rule

Aquí se ha explicitado una doble ligadura liga y dlig, algo que seguidamente comentaremos.

Listado 2: Implementación de una fuente con ligaduras, caso de estar alojada en archivos externos.

```
<style>
@font-face {
font-family: megalopolis;
```

```
src: url(MEgalopolisExtra.woff) format("woff"),
url(MEgalopolisExtra.otf) format('opentype');
}

p.poesia {
font-family: megalopolis, sans-serif;
font-style: italic;
font-variant: small-caps;
-moz-font-feature-settings: "liga", "dlig";
font-feature-settings: "liga", "dlig";
}

p.center {
text-align: center;
}
</style>
```

Las ligaduras y su presencia en las diferentes fuentes

Lo primero que hay que tener presente es que las ligaduras son algo opcional y particular de cada fuente. Es decir, una fuente dada puede tener o no ligaduras, puede tener unas particulares y aun esas pueden ser diferentes de las de otras fuentes. Habrá que conocer las particularidades de cada una.

Sí podremos saber el significado global de cada tipo de ligadura, pero no así sus particularidades y si está implementada en todos los glifos o sólo en parte, sin atender a su documentación, véase más abajo.

Algunos ejemplos de etiquetas comunes se recogen en la tabla que sigue:

Etiqueta	Descripción
liga	Activa y desactiva las ligaduras normales o comunes. (CSS 3)
clig	Ídem que la anterior. (CSS 3)
diga	Activa y desactiva las ligaduras discrecionales. (CSS 3)
dlig	Ídem que la anterior. (CSS 3)
hist	Activa y desactiva las ligaduras históricas. (CSS 3)
hlig	Ídem que la anterior. (CSS 3)
salt	Activa y desactiva estilos alternativos. (CSS 3)
calt	Activa y desactiva las ligaduras contextuales alternativas. (CSS 3)
smcp	Activa y desactiva pequeñas letras mayúsculas. (CSS 3)
ss01 ... ss20	Activa y desactiva capacidades de estilos diversos. (CSS 3)

Pueden observarse ejemplos de estos tipos genéricos de ligaduras y muchos más en: www.w3.org/TR/css3-fonts/#propdef-font-variant-ligatures

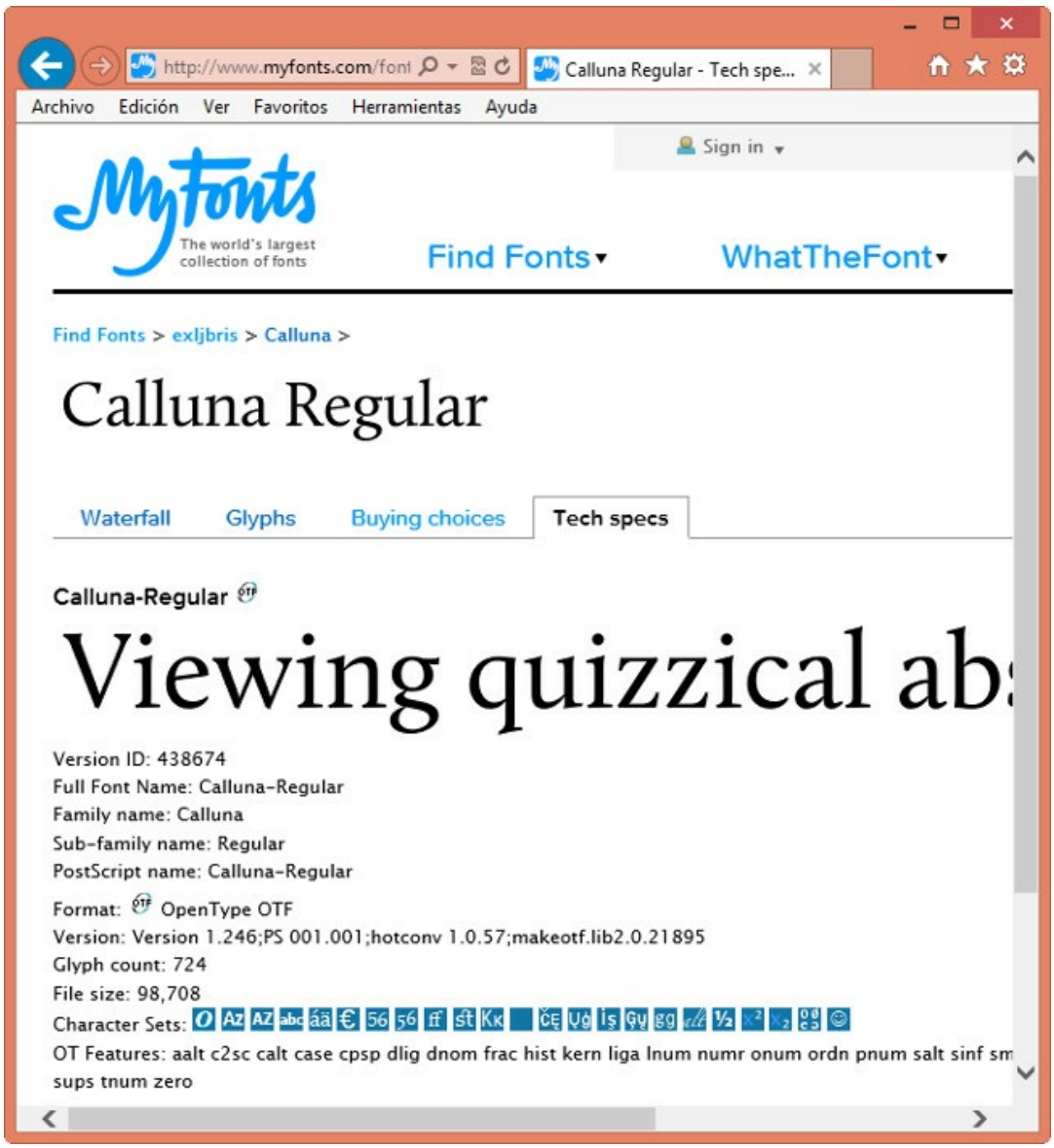
Las fuentes y su documentación

Como comentábamos, a la hora de obtener, instalar o distribuir una fuente, es imprescindible conocer sus capacidades de funcionalidades adicionales, como la presencia de ligaduras y el número de glifos, que nos orientarán sobre la calidad de la fuente.

Veámoslo con un ejemplo:

- Localice un sitio de compra y/o descarga de fuentes gratuitas de relevancia como algunos de los dos apuntados más abajo.
- Seleccione el tipo de fuente que se adapte a sus necesidades.

- Antes de incluirla en su carrito de la compra o de descargarla, debería poder disponer de una información o documentación en línea como la mostrada en la figura que sigue:



- En algunos lugares permiten incluir opciones de etiquetas adicionales a insertar en la compra de la fuente; un valor añadido de sumo interés.

- Normalmente la descarga incluirá un conjunto de fuentes de diversos tipos –OTF, WOFF, ...-, la documentación y archivos CSS y HTML de ejemplo de uso.

Algunos tipos de fuentes que nos serán útiles para nuestras pruebas:

- MEgalopolisExtra, la podrá obtener en: github.com/mozilla/bedrock/blob/master/media/fonts/MEgalopolisExtra.woff
- Calluna-Regular, la podrá obtener en los dos sitios siguientes, aunque de mucha mejor calidad en el segundo: github.com/TH-code/Wordpress-themes/blob/master/thijstoo/fonts/calluna-regular-webfont.woff - www.fontspring.com/fonts/exljbris/calluna
- Gabriola (instalada por defecto en los sistemas Windows 7/8 y Mac)
- Palatino Linotype (instalada por defecto en los sistemas Windows 7/8 y Mac)

Por otra parte, entre otros, dos lugares de interés para compra y descarga de fuentes gratuitas

en muchos casos son: www.fontspring.com/ - www.myfonts.com

Un programa para probar las ligaduras

A modo de ejemplo general y como utilidad de aplicación para sus experimentaciones con fuentes, vamos a implementar una aplicación que nos permitirá seleccionar y aplicar diferentes tipos de etiquetas de ligaduras sobre un texto y ver su efecto.

Constará de dos opciones generales, selección del tipo de ligadura y aplicarla sobre el texto tal cual o sobre el texto más la opción de mayúsculas pequeñas –*small caps*-.

Nuevamente hemos de insistir en que aquí incluiremos una serie de ligaduras bastante comunes, pero no todas las fuentes tienen todas y muchas tendrán otras diferentes, por lo que tendrá, en cada caso, que alterar el listado aquí propuesto.

El programa tiene prefijada la fuente *Gabriola*, algo que debería cambiar para cada caso de prueba y en el Listado 3 se supone está instalada en el sistema. En el Listado 4, se muestra como trabajar con las fuentes si se encuentran en archivos no instalados en el sistema.

El programa consta de los siguientes elementos funcionalmente relevantes:

- Una lista desplegable con las ligaduras predeterminadas –seleccionables-.
- Al pulsar los botones de comando de aplicar las ligaduras, se llama a la función *ligadura()*.
- Los botones de comando son del tipo *on/off*, es decir, la primera pulsación activa la ligadura, la segunda la desactivará.
- Si la ligadura no está aplicada aun, se llama a *font-feature-settings*, bien sea con mayúsculas pequeñas o sin ellas, según desde que botón se haya pulsado y se aplicará la ligaduras
- Si la ligadura ya estuviera aplicada, se llamara a *font-feature-settings* con una cadena vacía, con lo que se anularán todas las ligaduras.
- A modo de referencia visual, cuando está aplicada una ligadura, el texto se mostrará en azul.

Listado 3: Programa para probar ligaduras en fuentes OpenType.

```
<!doctype html>
<html>
<head>
<meta charset="utf-8" />
<title>CSS3 - Ligaduras</title>
<style>
h2 {
font-family: Gabriola;
}

p.poesia {
font-family: Gabriola;
}
```

```
p.center {
text-align: center;
}
</style>
<script>
var flag = true;
var caps = true;
var tipo = "";
function ligadura() {
if (flag) {
if (!caps) caps = true;
document.getElementById('A').style.cssText = "font-feature-settings: '" + tipo + "'";
document.getElementById('A').style.color = "blue";
flag = false;
}
else {
document.getElementById('A').style.cssText = "font-feature-settings: ''";
document.getElementById('A').style.color = "black";
flag = true;
}
}
function small() {
if (!flag) flag = true;
if (caps) {
document.getElementById('A').style.cssText = "font-feature-settings: '" + tipo + "' , 'smcp'";
document.getElementById('A').style.color = "blue";
caps = false;
}
else {
document.getElementById('A').style.cssText = "font-feature-settings: ''";
document.getElementById('A').style.color = "black";
caps = true;
}
}
</script>
</head>
<body>

<div style="position: relative; top: 10px; left: 150px; margin: 30px">
<h2>Saludo a Beatriz <span style="font-size: smaller">de Dante Alighieri<br />
<span style="font-size: smaller">(Traducción de Clemente Althaus)</span></span></h2>

<p id="A" class="poesia">
Tan honesta parece y tan hermosa<br />
mi casta Beatriz cuando saluda,<br />
que la lengua temblando queda muda<br />
y la vista mirarla apenas osa.<br />
<br />
Ella se va benigna y humillosa<br />
y oyéndose loar, rostro no muda<br />
y quien la mira enajenado duda<br />
si es visión o mujer maravillosa.<br />
<br />
Muéstrase tan amable a quien la mira<br />
que al alma infunde una dulzura nueva<br />
```

```
que solo aquél que la sintió la sabe.<br />
</p>
</div>
<hr />
<select onchange="tipo=value">
<option value="">sin ligadura</option>
<option value="liga">liga</option>
<option value="clig">clig</option>
<option value="diga">diga</option>
<option value="dlig">dlig</option>
<option value="hist">hist</option>
<option value="hlig">hlig</option>
<option value="salt">salt</option>
<option value="calt">calt</option>
<option value="aalt">aalt</option>
<option value="ss01">ss01</option>
<option value="ss02">ss02</option>
<option value="ss03">ss03</option>
<option value="ss04">ss04</option>
<option value="ss05">ss05</option>
<option value="ss06">ss06</option>
<option value="ss07">ss07</option>
</select>
<input type="button" value="Ligadura" onclick="ligadura()" />
<input type="button" value="Ligadura con Small Caps" onclick="small()" />
<p class="center">
<a href="http://jigsaw.w3.org/css-validator/check/referer">
<img style="border: 0; width: 88px; height: 31px"
src="http://jigsaw.w3.org/css-validator/images/vcss"
alt="¡CSS Válido!" />
</a>

<a href="http://www.w3.org/html/logo/">
<img src="http://www.w3.org/html/logo/badge/html5-badge-h-css3-semantics.png"
width="165" height="64" alt="HTML5 Powered with CSS3 / Styling, and Semantics"
title="HTML5 Powered with CSS3 / Styling, and Semantics">
</a>

<span style="font-size: xx-small">©Jaime Peña Tresancos, 2013</span>
</p>

</body>
</html>
```

Conclusiones

En el presente artículo hemos visto las características más importantes de los nuevos tipos de fuentes, centrándonos fundamentalmente en las ligaduras y su implementación. Finalmente hemos desarrollado un programa de pruebas completo de sus características.

Esperamos que todo lo expuesto les haya servido de ayuda. Hasta nuestro próximo artículo, felices horas de programación.

Este artículo es obra de *Jaime Peña Tresancos*
Fue publicado por primera vez en *17/10/2013*
Disponible online en *http://desarrolloweb.com/articulos/funcionalidades-fuentes-css3.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.

Antes de comenzar quiero aclarar que, a pesar de haber puesto el artículo dentro del Manual de CSS3, la regla de estilos text-shadow no pertenece a CSS3 sino que ya fue introducida en el nivel 2 de las Hojas de Estilo en Cascada. Sin embargo, hasta ahora no se había implementado dentro de los navegadores más comunes, por eso la estamos agrupando como novedad dentro de este Manual de CSS3.

Una vez comentado ese detalle sobre las sombras en texto, con CSS y sin utilizar un programa de diseño gráfico, en breves instantes comprobaremos lo fácil que crearlas. Comencemos viendo un ejemplo de declaración con text-shadow.

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 tienen 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 donde 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;
}
```

Todos los estilos de sombras ofrecidos en este artículo se pueden ver en una página aparte.

Conclusión a text-shadow

En definitiva, las sombras CSS que conseguimos con text-shadow nos ofrecen una nueva vía muy rápida y creativa para dar algunos toques de diseño en nuestros sitios web, sin tener que recurrir a Photoshop, u otro programa de diseño, como ocurría anteriormente.

Si buscamos artículos antiguos dentro de DesarrolloWeb.com veremos que se ofrecen talleres diversos para crear sombras en textos y en algún caso se ofrecen técnicas CSS para emular las sombras cuando no existía el atributo text-shadow. Todas las posibilidades anteriores necesitaban de un trabajo minucioso y la colaboración de otras técnicas o programas como editores gráficos.

Otra de las ventajas de usar CSS es que, si mañana deseamos cambiar cualquier cosa, como el

tamaño del texto, el efecto de la sombra o simplemente el color de la web y con ello el color de las sombras para que casen con la nueva cromática, únicamente tenemos que editar nuestra declaración de estilos.

Este artículo es obra de *Miguel Angel Alvarez*
Fue publicado por primera vez en *22/08/2011*
Disponible online en *http://desarrolloweb.com/articulos/sombras-texto-text-shadow-css.html*

CSS Media Queries

Las Media Queries, incorporadas en CSS3, son las primeras construcciones del lenguaje CSS que nos permiten definir estilos condicionales, aplicables únicamente en determinadas situaciones.

Las Media Queries son, en una traducción rápida, consultas de las características del medio donde se está visualizando una web y nos sirven para definir estilos condicionales, que solo se aplicarán en caso que esa consulta del medio sea satisfactoria.

Son una de las herramientas fundamentales para implementar el "Responsive Web Design" y han llegado de la mano de las CSS3, convirtiéndose en un aliado fundamental de cualquier diseñador web.

En la mayoría de los casos las Media Queries sirven para definir estilos diferentes para distintos tamaños de la pantalla. Son sencillas de entender y aplicar, aunque el estándar es bastante sofisticado, con diversas posibilidades. Existen muchos usos, algunos no tan habituales en el mundo del diseño actual, pero que podrán tener su protagonismo en algún momento.

Nota: A modo de curiosidad, pues realmente no se usa mucho en el diseño actual, las media queries pueden decirnos si el dispositivo tiene una relación de pantalla 4:3 o 16/9, o una profundidad de color dada, por ejemplo. En este artículo nos encargaremos de conocer los casos más comunes de uso y más adelante nos adentraremos en estas otras características adicionales.

Paso al frente de las CSS

Las Media Queries representan una evolución importante de CSS, puesto que suponen la primera estructura condicional en el lenguaje. Sabemos que CSS no es un lenguaje de programación, ni tiene motivos para parecerse, pero existen muchas cosas que se implementan en los lenguajes de programación y que nos vendrían bien en el desarrollo con CSS, como es el caso de los condicionales.

"Si ocurre ésto, entonces haz tal cosa"

Una construcción condicional como esta es tan útil y básica que, aunque CSS no sea un lenguaje de programación, necesita incorporarlas. Ejemplos de casos en los que nos vendría bien un condicional:

- Si la pantalla del usuario tiene estas características, entonces aplica estos estilos
- Si se imprime el documento en la impresora, aplica estos estilos.
- Si la pantalla del dispositivo tiene estas dimensiones y además está situado en posición horizontal (landscape), entonces aplica este CSS.

Situaciones como esas son básicas y los diseñadores pueden resolverlas usando las Media Queries. Son un paso al frente en cuanto a la separación entre el contenido y su representación, puesto que nos facilitan que el contenido de una página pueda adaptarse al medio donde se está reproduciendo, sólo a través de la definición de estilos, sin tener que tocar el HTML para nada.

Alternativas para implementar Media Queries

Aunque la utilidad es novedosa, la sintaxis es parecida a lo que ya conocemos de las CSS, por lo que nos resultará bien sencilla.

Para producir Media Queries debemos tener siempre en mente la expresión condicional, aquella que debe cumplirse para que se apliquen ciertos estilos. Además la expresión condicional puede tener incluso varias condiciones, usando operadores lógicos como "and" para combinarlas. De modo que las circunstancias que se deban cumplir para aplicar unas reglas CSS sean de lo más variadas.

Alternativa 1: La primera alternativa de las Media Queries es a través del atributo media de la etiqueta LINK. Como sabemos, esa etiqueta es la que se usa para enlazar una hoja de estilo con un documento HTML. En ese enlace podemos especificar condicionales que deben cumplirse para que los estilos enlazados se apliquen. Por ejemplo, que se esté imprimiendo un documento o que la pantalla tenga cierta anchura mínima.

Recordamos, la etiqueta LINK tiene esta forma:

```
<link rel="stylesheet" href="estilos-generales.css">
```

Pues ahora simplemente le podemos agregar el atributo "media" indicando la condición que se debe cumplir para que estos estilos se apliquen:

```
<link rel="stylesheet" href="estilo-imprimir.css" media="print">
```

Este atributo media="print" quiere decir que los estilos deben aplicarse sólo cuando la página se están mostrando para la impresión.

> **Nota:** Este uso ya lo vimos anteriormente en el artículo Estilos específicos para impresión.

```
<link rel="stylesheet" href="estilo-pantallas-grandes.css" media="(min-width:1200px)">
```

Este uso será seguramente más novedoso para ti. Quiere decir que esos estilos deben aplicarse sólo cuando la pantalla del usuario (En caso de ordenadores de escritorio, la ventana del navegador) tenga una anchura mínima de 1200 píxeles.

El problema de escribir tus Media Queries así es que tienes archivos de CSS separados. Es decir, aquellos estilos para impresión o para pantallas de 1200px están en archivos independientes, lo que es sencillo de administrar para nosotros, pero una mala práctica en términos de optimización de la web, puesto que se tienen que realizarse varias solicitudes al servidor distintas para traerse los CSS. En la práctica ralentiza la carga de la página en relación a hacer una única solicitud de un archivo CSS que contenga todo el código de los estilos.

Alternativa 2: Este método que vamos a ver ahora es más interesante y es el que se usa habitualmente a nivel profesional. Consiste en incorporar los estilos en una construcción @media donde se apliquen entre llaves todos los estilos que queremos para una consulta de medio dada.

```
@media (min-width: 500px) {
    h1{
        margin: 1%;
    }
    .estiloresponsive{
        float: right;
        padding-left: 15px;
    }
}
```

Como puedes ver, tenemos la sentencia @media en la que podemos indicar entre paréntesis las condiciones que deben cumplirse para que se aplique esta media query. En este caso será para pantallas que tengan una anchura mínima de 500 píxeles.

Luego entre llaves colocamos todas las reglas y atributos de estilos CSS que necesitemos aplicar en esta situación. Las reglas de estilos son las mismas que pondrías fuera de la estructura condicional. Cuando la sentencia entre paréntesis se evalúe como verdadera, se aplicarán todas ellas.

Operadores lógicos para las Media Queries

Para combinar diversas condiciones podemos usar los operadores lógicos, de una manera similar a como se usan en lenguajes de programación. Los que tenemos disponibles son:

- **Operador and:** las dos condiciones deben cumplirse para que se evalúe como verdadera.
- **Operador not:** es una negación de una condición. Cuando esa condición no se cumpla se aplicarán las media queries.
- **Operador only:** se aplican las reglas solo en el caso que se cunpla cierta circunstancia.
- **Operador or:** no existe como tal, pero puedes poner varias condiciones separadas por comas y cuando se cumpla cualquiera de ellas, se aplicarán los estilos de las media queries.

```
@media (max-width: 600px) and (orientation: landscape) {
    h1{
        color: red;
    }
}
```

Esta regla se aplicaría para pantallas con una anchura máxima de 600 píxeles y cuando la orientación está en horizontal.

> **Nota:** Ten en cuenta que la mayoría de smartphones simulan tamaños de pantalla mayores, haciendo una especie de dimensiones virtuales que faciliten la lectura de webs que no están diseñadas para Responsive Web Design. Por ello, lo más seguro es que tengas que poner el "viewport" en el documento HTML a algo como:
>
> ```
> <meta name="viewport" content="width=device-width, initial-scale=1.0" />
> ```
>
> Sin ese viewport tu móvil simularía que tiene unas dimensiones de unos 980 píxeles, cuando quizás la pantalla solo tenga una anchura real de 320 o 500 píxeles. Este asunto ya se explicó con detalle en el artículo sobre el viewport.

Para ver en funcionamiento la parte del orientation (landscape o portrait), tienes que usar un móvil o tablet y cambiar la posición de la pantalla, para que esté en horizontal o vertical.

> **Nota:** Si tienes un ordenador que no reconozca el cambio de orientación la posición siempre será considerada será landscape.

```
@media tv and (min-width: 1200px){
    h1{
        margin: 10%;
```

```
    }
}
```

Esta regla aplicaría en dispositivos de tipo televisión y cuya resolución minima de anchura sea de 1200 píxeles.

```
@media (min-width: 600px),  handheld and (orientation: portrait) {
    h1{
        color: green;
    }
}
```

Este mediaquery servirá para pantallas de minimas dimensiones 600 píxel y también para todos aquellos dispositivos handheld que estén en posición vertical.

> **Nota:** Handheld es un término inglés que sirve para especificar aquellos dispositivos que son de mano. Pequeños ordenadores que se llevan en la mano, como los palm o agendas electrónicas que aparecieron antes de popularizarse los smartphones o tablets. Por mis pruebas handheld no aplica a los móviles o tablets.

Las Media Queries tienen muchas más posibilidades que no cubrimos con este artículo. Las veremos más adelante, pero sin duda con lo que ahora sabes podrás resolver el 98% de las necesidades que te puedas encontrar en el mundo del diseño web.

Si te interesa saber más sobre este tema, en este otro artículo de DesarrolloWeb.com te contamos la estrategia para aplicar las Media Queries en Responsive Web Design.

Este artículo es obra de *Miguel Angel Alvarez*
Fue publicado por primera vez en *07/01/2015*
Disponible online en *http://desarrolloweb.com/articulos/css-media-queries.html*

Selectores de CSS que quizás no conozcas y debes revisar

Presentamos un listado de selectores de las CSS que puede que no conozcas y que resultan muy prácticos.

Las especificaciones de la W3C en cuanto a las Hojas de Estilo en Cascada no paran de crecer, eso nos da una idea de lo dinámicos que son los estándares y de la salud de los lenguajes con los que se construye la web. En este caso hacemos referencia a una especificación de **Selectors Level 4 (http://dev.w3.org/csswg/selectors4)** (Selectores de nivel 4) que no queremos que se confunda con CSS4, del que todavía no hay noticias en el momento de la publicación de este artículo. No existe confirmación alguna que se vaya a producir esa versión del lenguaje, aunque lo lógico será suponer que en algún momento se concrete.

Este artículo trata sobre selectores no tan usados en el día a día y que resultan prácticos para los desarrolladores. Pero antes sepamos qué son. **Los selectores son patrones que hacen corresponder conjuntos de elementos en la jerarquía de etiquetas del HTML y nos sirven para seleccionar partes de un documento llamadas nodos**. Son parte fundamental de CSS y aparte de en el lenguaje HTML, también se usan para seleccionar nodos en lenguajes como XML. Por medio de éstos, podemos definir a qué elementos se les aplican determinados estilos CSS.

La especificación de Selectores de nivel 4 de la que os hablamos en este texto nos da una referencia de los selectores que existen actualmente en CSS y algunos que quizás en el futuro deberían incorporarse en el lenguaje. Hay tanto selectores que se encuentran soportados por los navegadores actuales y pertenecientes a las CSS 1, 2 o 3, como otros que todavía no están disponibles para uso y quedan actualmente como mera curiosidad de lo que probablemente tendremos en un futuro. En esa especificación, de la cual hemos recuperado algunos "flashes", hay selectores para todos los gustos y utilidades diversas, nosotros te traemos unos pocos que quizás no son tan usados, pero que pueden ser prácticos.

1.- Negación con la pseudo-clase :not()

Esta pseudo-clase not() cuya referencia está en http://dev.w3.org/csswg/selectors4/#negation está presente en CSS3, pero en el borrador de los selectores nivel 4 hay algunas modificaciones que se han propuesto.

Es una especie de función para negar un selector dado, técnicamente le llaman "functional pseudo-classs" o pseudo-clase funcional, que recibe un partámetro que es el selector que se quiere negar. El resultado es que se accede a todos los selectores que no corresponden con aquel selector que se ha negado. Por ejemplo, si queremos seleccionar todos los botones que no están desactivados (no están "disabled"):

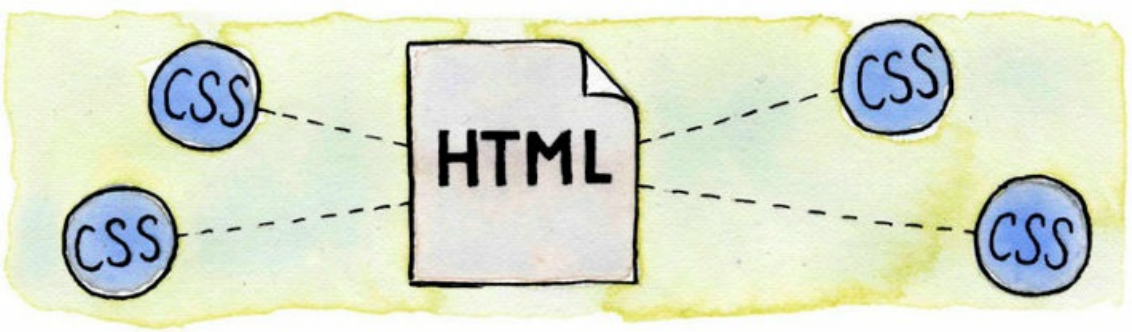
```
button:not([DISABLED])
```

Si tenemos un selector determinado llamado YYY (entendiendo ese YYY como cualquier selector, de etiqueta, class, identificador, etc) podemos seleccionar todos los elementos, menos los que seleccionaríamos con ese selector, con el código:

```
*:not(YYY)
```

La siguiente nserviría para seleccionar todos los elementos de la página HTML, excepto a los enlaces:

```
html|*:not(:link):not(:visited)
```



2.- La pseudo-clase :local-link

Esta pseudo-clase forma parte de la especificación de selectores de nivel 4 que hemos mencionado, encontrando la referencia en el borrador bajo el epígrafe The local link pseudo-class :local-link http://dev.w3.org/csswg/selectors4/#local-link-pseudo Se utiliza para aplicar cualquier propiedad CSS, pero solo a los enlaces que sean locales al documento donde estamos trabajando.

> **Nota:** Ojo que esta clase es para enlaces que vayan a este mismo documento, lo que se conocería como enlaces internos en la misma página. Pero también lo podemos hacer para enlaces que vayan a otras partes de este dominio, lo que te puede resultar útil para diferenciar el estilo CSS de los enlaces internos (a tu dominio) y los enlaces a otros dominios externos.

La estructura sería así:

```
nav :local-link { text-decoration: none; }
```

Pero además, podemos especificar diversas profundidades en los enlaces, entendiendo dichas profundidades como niveles en la ruta definidos por los directorios. La raíz del dominio tiene nivel cero, el primer directorio nivel 1 y así a continuación. De ese modo podemos definir las pseudo-clases como local-link(0) o local-link(1), para que solo apliquen a una profundidad determinada de enlaces dentro del mismo dominio. Para verlo más claro, tomemos en cuenta un ejemplo.

Vamos a suponer que estamos trabajando en un **documento alojado en la ruta http://www.mipaginaweb.com/2020/11/** (fíjate que esta ruta, el camino completo y además en el número de directorios. Comprobarás que tendría dos niveles, por tener dos directorios. Fíjate que acaba en una barra). Ahora dentro del código HTML de ese documento

tenemos los siguientes links o vínculos:

```
A) <a href="http://www.mipaginaweb.com">Contáctenos</a>
B) <a href="http://www.mipaginaweb.com/2014">2014</a>
C) <a href="https://www.mipaginaweb.com/2020/11">Noviembre</a>
D) <a href="http://www.mipaginaweb.com/2020/11/">Noviembre</a>
E) <a href="https://www.mipaginaweb.com/2020/12/">Diciembre</a>
F) <a href="http://mipaginaweb.com/2018/10/30">30 de octubre</a>
```

Y sus estilos:

```
1) a:local-link {...}
2) a:local-link(0) {...}
3) a:local-link(1) {...}
4) a:local-link(2) {...}
5) a:local-link(3) {...}
6) a:local-link(4) {...}
```

- El primer link A) recibiría el Estilo 2), pues se le aplica el estilo de la profundidad cero
- El link B) recibiría el Estilo 2) y el 3)
- El link C) recibiría el estilo 2), 3), y 4)
- El link D) recibiría el estilo 1), 2), 3), 4) y 5) (Fíjate que es el mismo link del documento donde estaba este enlace y que además se diferencia con el link C) en que acaba en una barra.
- El link E) permanecería sin estilo, puesto que es una ruta en https:// siendo http:// la del documento actual.
- El link F) no tendría un estilo definido porque no empieza por las www.
- El estilo 6 no se aplica a ningún elemento.

3.- La pseudo-clase :checked

Otro ejemplo de selector, esta vez implementado desde CSS3 que recoge esta especificación es la pseudo-clase :checked http://dev.w3.org/csswg/selectors4/#checked-pseudo. Es útil para seleccionar los elementos checkbox que están seleccionados o como decimos a veces erróneamente, "checados".

Para los elementos "checkbox" de nuestro HTML, podemos aplicar estilos dependiendo de si lo tenemos activado o no. Esto es bien sencillo. Si queremos seleccionar aquellos campos checkbox que tenemos activados para darles estilo, escribimos:

```
input:checked{
   margin-left: 20px;
}
```

Como acabas de conocer el selector :not seguramente te hagas una idea de lo que tienes que hacer para conseguir aplicar estilos a aquellos checkbox que no están seleccionados:

```
:not(:checked)
```

4.- Selectores de atributos

Otros selectores que debes conocer son los selectores de atributos, que son una familia de selectores bastante grande y que daría seguro de que hablar para uno o varios artículos. La mayoría de estos selectores ya los venimos usando desde CSS2, algunos se incorporaron en CSS3 y otros todavía no se encuentran disponibles pero ya se están especificando en el W3C en el documento Selectors Level 4 en el epígrafe Attribute selectors http://dev.w3.org/csswg/selectors4/#attribute-selectors.

Veamos algunos ejemplos.

######### [att^="val"]

Selecciona un elemento cuyo atributo "att" tenga un valor comienza con "val". Si "val" fuera la cadena vacía, el selector no selecciona nada.

######### [att$="val"]

Selecciona un elemento con el atributo "att" cuyo valor termina con el sufijo "val". Si "val" es la cadena vacía, el selector no selecciona nada.

######### [att*="val"]

Selecciona un elemento con el atributo "att" cuyo valor contiene al menos una instancia de la subcadena "val". Si "val" es la cadena vacía, el selector no selecciona nada.

Ejemplos:

El selector siguiente selecciona un elemento INPUT cuyo valor en el atributo name comienza por "mi":

```
input[name^="mi"]
```

El selector siguiente representa un ancla HTML con un atributo "href" cuyo valor termina con ". html".

```
a[href$=".html"]
```

El selector siguiente representa un párrafo HTML con un atributo "title" cuyo valor contiene la subcadena "hola"

```
p[title*="hola"]
```

Hay otra serie de selectores de atributo que te pueden interesar y los cuales ya están disponibles para usar en todos los navegadores. Puedes ver la referencia en la especificación del W3C.

Como puedes ver, hay muchos selectores que puedes estar dejando pasar para atinar mejor en tus hojas de estilo.

Puedes ver más novedades en el borrador aquí: www.w3.org/TR/2011/WD-selectors4-20110929.

Este artículo es obra de *Juan R. Castro Lurita*
Fue publicado por primera vez en *17/02/2014*
Disponible online en *http://desarrolloweb.com/articulos/futuro-diseno-css4.html*

Degradados CSS3

Los degradados CSS3 tienen muchas posibilidades que debemos ver a lo largo de varios artículos.

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.

Por suerte, dentro de poco, el uso de imágenes en degradados habrá pasado a la historia, ya que CSS 3 dispone de un potente mecanismo para producirlos que resulta todavía más versátil que el propio uso de imágenes de fondo. En este artículo introduciremos el uso de degradados en páginas web, aportando un par de ejemplos.

> Nota: En el momento de escribir este artículo la especificación de CSS 3 está todavía en estado de borrador, por lo que la implementación final de los degradados puede sufrir ligeras variaciones con respecto a la que aquí se explica. Además, debemos tener en cuenta que no todos los navegadores disponen de los degradados, ya que las CSS 3 aun no se encuentran en su fase final. En el Manual de CSS 3, en regla general, ocurre siempre este problema, pues estamos explicando cosas que todavía falta un poco para que sean un estándar definitivo.

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

El navegador es el encargado de renderizar el degradado en función de las características escritas para definirlo. Podemos asignar degradados como fondo en cualquier elemento HTML donde se podía implementar una imagen de fondo.

La especificación que se baraja actualmente incluyen degradados de dos tipos principales:

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);
}
```

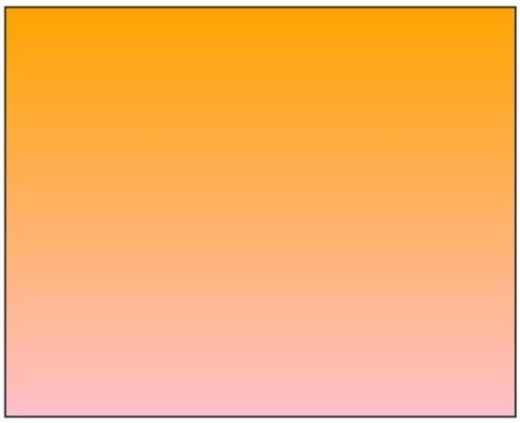


Imagen del renderizado del degradado lineal con CSS3, tal como aparecería en Firefox 4

linear-gradient(orange, pink);

Nota: Como decíamos, estos ejemplos no se ven perfectamente en todos los navegadores. De momento los podrás ver Webkit como Google Chrome o Safari, en Firefox (mejor la versión 4, aunque la 3 ya los implementa también) y Opera versión 11. De hecho, como habrás notado, todavía tenemos que escribirlos usando atributos propietarios de cada navegador, como -webkit-linear-gradient (para navegadores basados en Webkit) y -moz-linear-gradient (para navegadores de la fundación Mozilla) y -o-linear-gradient para Opera. Más adelante, estos atributos se podrán usar en sus versiones originales como linear-gradient (sin los prefijos de cada familia de navegadores) Con respecto a Internet Explorer 9, a pesar que implementa la mayoría de las características de las CSS 3, incluso con los atributos no propietarios, no parece que los degradados estén funcionando todavía. Aunque hay modos de hacer fondos con degradados de otra manera, las propiedades linear-gradient y radial-gradient no funcionan, incluso si intento meterle el prefijo -ms-.

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;
}
```

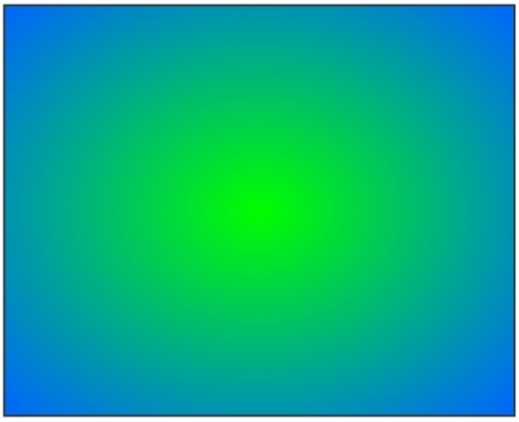


Imagen del renderizado del degradado radial con CSS3, tal como aparecería en Firefox 4

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

Nota: Volvemos a insistir en que se han utilizado los atributos propietarios -webkit-radial-gradient, -moz-radial-gradient y -o-radial-gradient, en lugar del atributo definitivo de las CSS 3 que será radial-gradient. Otra cosa destacable es que, en el momento de escribir este artículo, Opera no soporta degradados circulares. También, de momento, en Webkit parece que sólo podemos trabajar con degradados circulares y no con degradados en forma de elipse.

Puedes ver estos dos degradados, si utilizas Firefox, Opera, Safari o Chrome, en una página aparte.

Conclusión

De momento estamos ante características todavía en fase de borrador, ya que el W3C todavía no ha aclarado como va a ser el estándar definitivo. Los navegadores ya las implementan, aunque algunos solo parcialmente, y en general todo está realizado de manera experimental.

Para asegurarse que un degradado funciona en la mayoría de las plataformas, de momento estamos obligados a escribir las reglas de estilos con etiquetas propietarias para cada navegador y más adelante podremos escribirlos sólo con la etiqueta definitiva.

Osea, para definir un degradado deberíamos escribir todos estos estilos:

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

El primero para Webkit, segundo para Firefox, tercero para Opera y el último sería el atributo que se espera que sea definitivo, que lo podemos ir colocando para cuando Internet Explorer empiece a soportar los degradados CSS o para cuando los navegadores ya entiendan la etiqueta definitiva.

Por tanto, cuando CSS 3 se convierta en un estándar de implementación recomendada, podremos únicamente dejar la etiqueta del gradiente definitiva, linear-gradient y borrar todas las demás definiciones de estilos:

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

Aunque en este artículo hemos visto una simple introducción a los degradados CSS 3, el borrador actual del W3C incluye muchas otras cosas que servirán para hacer degradados bastante más complejos y configurables por infinidad de parámetros. No obstante, de momento lo dejamos por aquí y veremos más sobre este asunto ya cuando expliquemos el detalle de los degradados lineares y los degradados radiales.

Este artículo es obra de *Miguel Angel Alvarez*
Fue publicado por primera vez en *29/04/2011*
Disponible online en *http://desarrolloweb.com/articulos/degradado-css-3.html*

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.

En el pasado artículo del Manual de CSS 3 estuvimos haciendo una presentación inicial de los degradados CSS3, que nos servirá para conocer básicamente los tipos de degradados y ver un par de ejemplos. En este artículo veremos con detalle la construcción de los degradados lineales, que resultan bastante potentes y atractivos para decorar cualquiera de los elementos de un sitio web.

Veremos además diversos ejemplos que estamos seguros clarificarán las distintas posibilidades de construcción de degradados, pues, aunque no resulta difícil de conseguirlos, existen muchas variables que nos pueden ayudar a crear infinitos tipos de degradado con solo la definición de un atributo de estilos CSS.

Sintaxis de creación de degradados lineales

Así pues, después de haber leído la introducción a los degradados CSS3, veamos en líneas generales cuál sería la sintaxis para la construcción de un degradado lineal.

```
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.

Para poder entender mejor todo esto, lo más fácil es ver una una serie de ejemplos. Todos ellos los podemos visualizar en una página aparte.

> **Nota:** Por favor, tener en cuenta que el ejemplo se debe ver por el momento en las versiones más modernas de los navegadores. De momento lo podremos ver en Opera, o los browsers basados en Webkit, como Google Chrome o Safari, o navegadores basados en Mozilla, como Firefox.

```
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.

Nota: En realidad, tal como se explicó en el artículo anterior, para que los ejemplos funcionen en esto momento, tenemos que escribir las etiquetas de estilos propietarias de cada navegador. Así que el estilo anterior, en realidad en nuestro ejemplo está escrito con los atributos de estilos propios de cada navegador, que son los que funcionan actualmente:

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

También conviene colocar el atributo original linear-gradient, que es el que funcionará cuando CSS 3 sea un estándar de implementación recomendada por el W3C.

```
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.

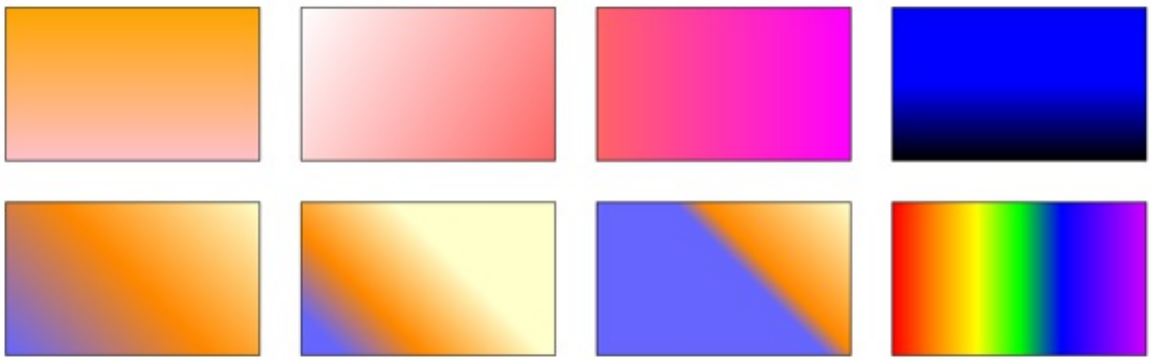
> **Nota:** Se aconseja no mezclar unidades CSS distintas en las paradas de color, como podría ser:
>
> ```
> background: linear-gradient(left, #66f 60%, #f80 50px)
> ```
>
> Porque en ese caso, dependiendo del anchura del elemento, la segunda parada de color, con 50px, podría estar en un punto anterior 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.

Será muy útil ver estos degradados en marcha en una página aparte. Los veremos bien siempre que dispongamos de un navegador compatible con esta nueva característica de las CSS3. Para

quien no disponga del navegador adecuado, puede ver la muestra en la siguiente imagen:



Paradas de color

Como hemos visto, las paradas de color, o "color stops" permiten alterar bastante los degradados incluso trabajando con los mismos colores. Para definirlas se debe tener en cuenta la línea imaginaria de distribución del gradiente de color. Al principio de esa línea tendríamos la distancia 0% y al final del elemento tendríamos la distancia 100%. El color inicial siempre comenzaría en el espacio del 0% y se completaría el degradado llegando al color final en el espacio del 100%.

> **Nota:** Si estamos trabajando con otras unidades, por ejemplo píxeles, que son perfectamente posibles en las paradas de color (siempre que no mezclemos unidades, pues no es recomendable), al principio tendríamos 0px y al final tendríamos Xpx siendo X el tamaño de esa línea de degradado desde el principio hasta el fin.

Aunque el degradado comience de 0% a 100%, las paradas de color las podemos poner en cualquier punto, incluso puntos que estén fuera de ese intervalo. Por ejemplo, nada nos impide poner una parada de color en 120%, lo que significaría que el color al que asignamos esa parada no llegaría a verse degradado completamente en el espacio del elemento. Es decir, si la parada de color está fuera del intervalo, no llegaríamos a ver el RGB exacto al que tendería el degradado, porque estaría fuera del espacio del elemento.

En el siguiente artículo comentaremos los degradados lineales con repetición.

> Este artículo es obra de *Miguel Angel Alvarez*
> Fue publicado por primera vez en *04/05/2011*
> Disponible online en *http://desarrolloweb.com/articulos/degradado-lineal-css3.html*

Degradados lineales con repetición con CSS3

Estudiaremos el atributo repeating-linear-gradient de CSS3, que permite realizar degradados lineales con múltiples repeticiones del mismo gradiente de color.

En este estudio profundo de los degradados de color en CSS3 ahora vamos a tratar de explicar los degradados con repetición, que se pueden hacer para conseguir un degradado entre dos o más colores que se repite varias veces a lo largo de todo el fondo del elemento al que se le coloque.

Los degradados con repetición se realizan de manera similar a los que vimos al tratar los degradados lineares normales. La diferencia es que utilizaremos el nombre de atributo repeating-linear-gradient y que, para que se produzcan las repeticiones tendremos que utilizar paradas de color.

La sintaxis es exactamente la misma que ya conocemos de los degradados lineales:

```
repeating-linear-gradient(origen, colores)
```

Siendo que en origen podremos colocar, tanto la posición inicial donde comienza el degradado, como el ángulo que debe formar el gradiente. Luego, los colores, como también vimos, se indican separados por comas. Sin embargo, ahora, para que realice la repetición, estamos obligados a señalar una parada de color.

> **Nota:** las paradas de color, que también conocemos como "color stop", se explicaron con detalle en el anterior artículo donde se trataron los degradados lineales con CSS3.

La parada de color la podremos hacer en cualquiera de los colores del degradado, pero claro, tendremos que asegurarnos que el último color tenga una parada de color menor que el tamaño del elemento, o si trabajamos con porcentajes, menor que el 100%. Así, en el espacio del elemento que sobre después de la última parada de color, comenzará la repetición del degradado.

Vamos ahora a mostrar varios ejemplos de degradados con repetición para que se puedan entender perfectamente. Si dispones de un navegador compatible con los degradados CSS podrías ver el ejemplo en marcha.

```
background: repeating-linear-gradient(#fff, #666 25%);
```

Este degradado comienza en blanco y termina en gris. Como no se indicó nada, irá de arriba a abajo, en un gradiente vertical en línea recta. Pero lo importante en este caso es que el segundo color tiene una parada al 25%. Eso quiere decir que se llegará al gris en el primer 25% del espacio del elemento y que a partir de ese punto comenzará de nuevo el degradado. El segundo degradado ocupará otro 25% de la imagen y entonces se volverá a repetir. Por tanto, en la práctica veremos que este degradado de blanco a gris se repetirá por 4 veces en el fondo del elemento donde lo coloquemos.

Nota: Volvemos a insistir en que las CSS 3 todavía, en el momento de escribir estas líneas, están en fase borrador, por lo que no todos los navegadores las implementan. Además, los que lo hacen, utilizan atributos propietarios. Por ello, para que se vea ese degradado en los navegadores, tendríamos que utilizar los atributos propios de cada familia de browsers y la declaración de estilos nos quedaría como sigue:

```
.midegradado{
   background: -webkit-repeating-linear-gradient(#fff, #666 25%);
   background: -moz-repeating-linear-gradient(#fff, #666 25%);
   background: -o-repeating-linear-gradient(#fff, #666 25%);
   background: repeating-linear-gradient(#fff, #666 25%);
}
```

La primera declaración (con el prefijo -webkit-) sería para navegadores como Chrome o Safari, la segunda, (-moz-) para Firefox, la tercera (-o-) para Opera y la última es para todos los navegadores, que empezarán a entenderla cundo esto de las CSS3 sea un estándar.

```
background: repeating-linear-gradient(left, #ffc, #f96 30%, #963 40%, #630 51%);
```

En este segundo ejemplo tenemos otro degradado, esta vez con 4 colores. Como se puede ver, se le han asignado varias paradas de color, en lugar de solo una al último elemento. Como el último color stop está al 51% del elemento, el degradado se verá solo dos veces.

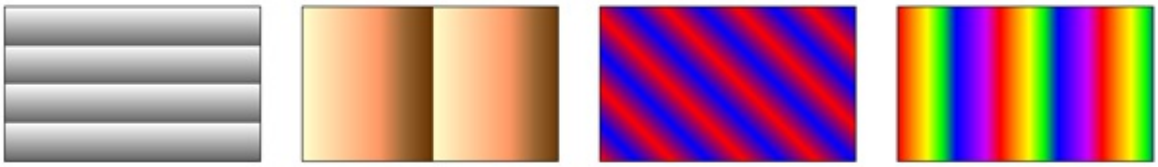
```
background: repeating-linear-gradient(45deg, red, blue, red 50px);
```

Esta es otra declaración de estilos, con un degradado en un gradiente oblicuo. Además, tiene la particularidad que se va de rojo a azul y luego de nuevo a rojo. Con ello conseguimos que las repeticiones del degradado siempre estén suavizadas y no se note cuando comienza y acaba una repetición, como nos ocurría en los dos ejemplos anteriores. Además, como se puede ver, la parada de color la hemos colocado en 50px, lo que quiere decir que el degradado se repetirá cada 50 píxeles, de modo que, el número de repeticiones variará dependiendo del tamaño del contenedor donde se asigne este fondo.

```
background: repeating-linear-gradient(left, #f00, #f80, #ff0, #0f0, #00f, #60f, #c0f, #f00 100px);
```

Este último ejemplo tiene los colores del arcoiris repetidos cada 100 píxeles.

Podemos ver el ejemplo en un navegador accediendo a este enlace, pero si tu cliente web aun no es compatible, puede ver los distintos degradados realizados en la siguiente imagen.



Como se ha podido ver, los degradados con repetición no tienen ningún misterio. Así que sin más tardar, podemos pasar a ver los degradados radiales.

Este artículo es obra de *Miguel Angel Alvarez*
Fue publicado por primera vez en *09/05/2011*
Disponible online en *http://desarrolloweb.com/articulos/degradados-lineales-repeticion-css3.html*

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.

Este es el cuarto artículo que estamos dedicando a los degradados con CSS3 y en esta ocasión vamos a ver los degradados radiales, que forman diseños circulares, con una distribución radial uniforme, o degradados en elipse, con una distribución radial variable.

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.

Nota: esta forma alternativa de especificar la forma no está implementada en ningún navegador totalmente. En el momento de escribir estas líneas, Firefox no la contempla, Chrome solo representa formas circulares y ni Opera ni Explorer soportan degradados radiales.

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.

Ahora, veamos una serie de ejemplos que esperamos aclaren las ideas con respecto a la declaración de fondos radiales.

Nota: Los fondos radiales son soportados en poca medida en estos momentos en los navegadores. Todos tienen al menos algún detalle que todavía tienen que pulir. De momento ni Opera ni IE9 los muestran en ningún caso. Chrome no tiene posibilidad de hacer degradados con forma de elipse y Firefox todavía no implementa la alternativa B), comentada anteriormente, para la definición del tamaño del gradiente por medio de los valores de anchura y altura. En los ejemplos que se mostrarán a continuación se presenta únicamente el código con el atributo de estilo definitivo, que estará disponible cuando las CSS3 sean un estandar.

```
background: radial-gradient(ellipse cover, #66f, #f80, #ffc);
```

Cuando en realidad, en estos momentos, para que funcione en Firefox y Chrome hay que colocar sus atributos con los nombres propietarios de cada navegador.

```
background: -webkit-radial-gradient(ellipse cover, #66f, #f80, #ffc);
background: -moz-radial-gradient(ellipse cover, #66f, #f80, #ffc);
```

La marca -webkit-radial-gradient sirve para navegadores basados en Webkit, como Chorme, y -moz-radial-gradient sirve para navegadores basados en Mozilla, como Firefox.

```
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.

Nota: las paradas de color, o "color stop", se explicaron cuando se trataron los degradados lineares.

```
background: radial-gradient(circle, #66f, #f80, #ffc);
```

Este es el primero de los ejemplos en el que definimos la forma del degradado, aunque solo indicamos "circle". Por tanto, el degradado comenzará en el centro y ocupará el 100% del espacio disponible en el elemento, aunque siempre con el mismo radio.

```
background: radial-gradient(ellipse cover, #66f, #f80, #ffc);
```

Este degradado es exactamente igual que el anterior, pero en vez de circular es de elipse, cubriendo el 100% del espacio disponible, y comenzando en el centro. Este es el comportamiento predeterminado del estilo.

> **Nota:** Sin embargo, en el momento de escribir este artículo la forma de elipse solo la implementa Firefox. Por lo que el comportamiento predeterminado en Chrome continua siendo el circular.

```
background: radial-gradient(10%, ellipse closest-side, #66f 60%, #f80 85%, #ffc);
```

Este ejemplo tiene definida la posición del degradado y la forma. Es el primero que especifica esos dos valores al mismo tiempo. En este coso, sobre la posición solo se declara 10%, así que aparecerá centrado en la vertical y en la horizontal aparecerá en el 10% del espacio del contenedor por la parte de la izquierda. Es de forma de elipse y closest-side significa que se expande en forma de elipse hasta completarse en el lado más cercano.

```
background: radial-gradient(10%, ellipse farthest-corner, #66f 60%, #f80 85%, #ffc);
```

Este degradado es igual que el anterior, en la misma posición y de forma de elipse, pero el tamaño se ha definido con farthest-corner, con lo que el degradado será mucho mayor, expandiéndose hasta la esquina más lejana.

```
background: radial-gradient(20px 100px, 30% 80%, #fff, #666, #66f);
```

En este caso hemos definido la posición inicial con las coordenadas en píxeles y, lo que resulta novedad, hemos definido tanto la forma como el tamaño del degradado en porcentaje. La anchura será el 30% del elemento y la altura el 80%.

> **Nota:** Esta alternativa de definir la forma y dimensiones del degradado, que vemos en este ejemplo y los tres siguientes, no funciona en Firefox en el momento de escribir el artículo, por lo que solo chrome mostrará un degradado en el elemento. Sin embargo, Chrome es incapaz por ahora de producir elipses, por lo que el degradado tendrá forma circular, a pesar de haber definido dimensiones menores en la anchura que para al altura. No obstante, lo más seguro es que pronto se mejore la compatibilidad de los navegadores con este tipo de definición del degradado.

```
background: radial-gradient(top left, 150px 100px, #ffc, #f96, #963, #630);
```

Hemos definido la posición inicial por medio de los valores top y left y el tamaño por medio de unidades en píxeles.

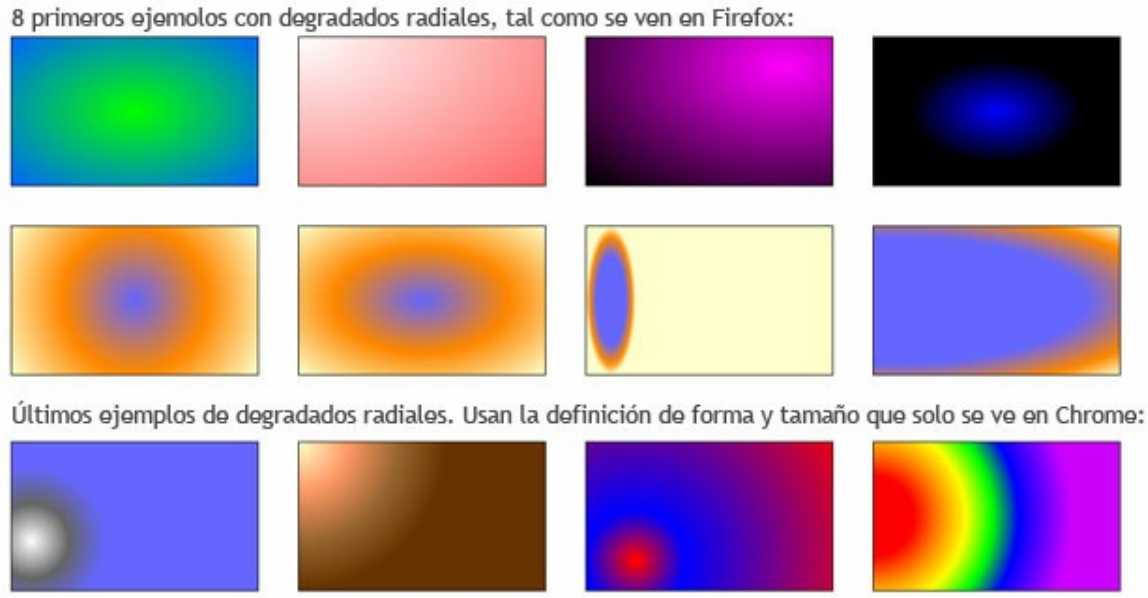
```
background: radial-gradient(20% 80%, 100% 50%, red, blue 50px, red);
```

La posición del centro del degradado está con porcentaje, así como el tamaño de la elipse, también con porcentajes.

```
background: radial-gradient(left, 200px 200px, #f00 20%, #f80, #ff0, #0f0, #00f, #60f, #c0f);
```

En este hemos definido el tamaño de la anchura y altura con dos valores en píxeles, pero como son iguales, en lugar de una elipse veríamos una forma circular, cuyo radio es siempre igual.

Recuerda que puedes ver el ejemplo en marcha, aunque solo en navegadores que soporten los degradados radiales. Los otros usuarios pueden ver los degradados producidos en los ejemplos en la siguiente imagen:



En el siguiente artículo hablaremos del último tipo de degradados CSS3 que nos queda por ver, los degradados radiales de repetición.

Este artículo es obra de *Miguel Angel Alvarez*
Fue publicado por primera vez en *10/05/2011*
Disponible online en *http://desarrolloweb.com/articulos/degradados-radiales-css3.html*

Degradados CSS 3 radiales de repetición

Los degradados CSS 3 radiales, en su versión con repeticiones, que nos permiten definir fondos con gradientes de color que serían muy difíciles de implementar con imágenes.

A lo largo de varios artículos venimos ofreciendo unas explicaciones completas sobre los degradados en CSS 3. Ahora, para finalizar esta serie, vamos a hablar de los degradados raliales con repetición.

Como veníamos diciendo, una de las ventajas de estas nuevas características de las CSS3 es que nos permiten realizar fondos para páginas web con degradados, pero sin tener que usar imágenes. Eso consigue ahorrarnos tiempo de desarrollo y diseño, a la vez que aligera el peso de las páginas web. Pero como curiosidad, en el caso de los degradados radiales con repeticiones, además habría que añadir que serían casi imposibles de realizar usando imágenes de fondo, o por lo menos tendríamos que usar archivos gráficos muy grandes y pesados.

Los degradados radiales con repetición son casi idénticos a los degradados radiales normales, explicados en el pasado artículo. De modo que, todo lo aprendido anteriormente, lo podemos aplicar aquí también.

> **Nota:** Algunas explicaciones sobre los degradados CSS3, que también podemos aplicar a los degradados radiales de repetición, la podemos ver en el artículo sobre los degradados lineales con repetición.

La sintaxis básica de este tipo de degradados es la siguiente:

```
repeating-radial-gradient(parámetros)
```

Los parámetros, tales como centro del degradado radial, forma y tamaño del motivo y los colores son exactamente los mismos que para los degradados radiales, por lo que no lo vamos a explicar. Las únicas diferencias es que tenemos que utilizar el atributo repeating-radial-gradient. Además, para que se produzca la repetición con el tamaño o intervalo que nosotros deseemos, tendremos que asignar alguna parada de color al último de los colores del degradado, que generalmente tendrá un valor menor del 100% del espacio del elemento.

> **Nota:** Las paradas de color, a las que también nos hemos referido repetidas veces como "color stops" las explicamos en el artículo que trataba los degradados lineales.

A continuación podemos ver una serie de ejemplos que ilustrarán el funcionamiento de los degradados radiales con repetición. Podemos verlos, si contamos con un navegador que soporte esta utilidad de las CSS3, en este enlace.

```
background: repeating-radial-gradient(circle, #fff, #666 25%);
```

Este ejemplo hace un degradado entre dos colores, cuyo segundo elemento tiene una parada de color en el 25%. Por ello el resultado producirá el mismo degradado repetido 4 veces, una en

cada 25% del espacio del elemento donde se coloque.

> **Nota:** Como venimos insistiendo, a pesar de poder llegar a parecer pesados, para que estos degradados funcionen, en el momento de escribir estas líneas, hay que colocar los prefijos propios de cada navegador. Es decir, hay que utilizar las etiquetas propietarias en lugar de la definitiva. De momento sólo los soportan los browsers basados en Webkit y los de la familia Mozilla. Por ello, tendremos que utilizar los atributos tal como se puede ver aquí:

```
background: -webkit-repeating-radial-gradient(circle, #fff, #666 25%);
background: -moz-repeating-radial-gradient(circle, #fff, #666 25%);
background: repeating-radial-gradient(circle, #fff, #666 25%);
```

El primero para Chrome o Safari, el segundo para Firefox y el tercero es para los navegadores que en el futuro soporten CSS3, pues la etiqueta definitiva comenzará a ser válida cuando CSS3 se convierta en un estándar.

```
background: repeating-radial-gradient(left, circle, #ffc, #f96 30%, #963 40%, #630 51%);
```

Este degradado hace un gradiente entre varios colores y el último de ellos tiene una parada en el 51%. Esto quiere decir que el degradado se repetirá dos veces.

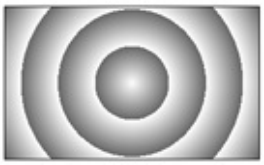
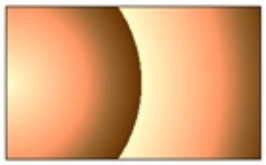
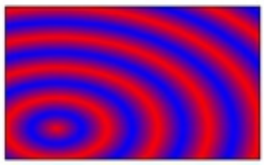
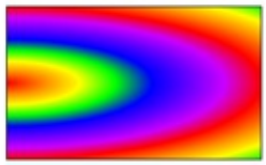
```
background: repeating-radial-gradient(20% 80%, ellipse closest-side, red, blue, red 50px);
```

En este ejemplo hacemos un degradado con forma de elipse y va desde rojo a azul y luego de nuevo a rojo. Al comenzar y acabar en rojo, se consigue que las repeticiones del degradado no tengan saltos bruscos de un color al otro. Como se puede ver, el último color tiene una parada de color en 50 píxeles, con lo que el degradado se repetirá de nuevo cada 50 puntos en la pantalla. El número de repeticiones dependerá del tamaño del elemento donde se coloque este degradado.

```
background: repeating-radial-gradient(left, ellipse farthest-side, #f00, #f80, #ff0, #0f0, #00f, #60f, #c0f, #f00);
```

Este otro caso produce una repetición del degradado, pero no hemos colocado ninguna parada de color en el último elemento. Esto quiere decir que el degradado se repetirá, pero el intervalo de la repetición en esta ocasión dependerá del tamaño y forma radial que se haya configurado. En este caso en concreto tenemos una elipse que se expande, desde la izquierda hasta el lado opuesto, con lo que sólo en una pequeña porción del fondo se verá la repetición de los colores.

Acabamos poniendo un enlace a estos degradados en marcha. Pero si tu navegador no te muestra nada, puedes ver su aspecto en la siguiente imagen.

Conclusión a los degradados CSS3

Con este artículo finalizamos el reportaje sobre los degradados de CSS3, una utilidad que sin duda se convertirá en un habitual del diseño web. Hasta ahora los degradados se utilizaban bastante, pero necesitábamos cargar la página con diversas imágenes, lo que requería esfuerzos adicionales en el diseño original de la página y también durante el mantenimiento.

A partir del momento en que CSS 3 sea un estándar para la implementación recomendada, estas características de las CSS 3 las veremos intensivamente, porque realmente tienen muchas ventajas. Sin embargo, si deseamos decorar nuestras webs ya desde este momento, podemos utilizar las etiquetas propietarias de cada navegador que soporte los degradados CSS. Cargaremos solo un poquito más nuestro archivo de estilos, pero nuestra página será visualmente más atractiva.

Este artículo es obra de *Miguel Angel Alvarez*
Fue publicado por primera vez en *08/06/2011*
Disponible online en *http://desarrolloweb.com/articulos/degradados-radiales-repeticion.html*

Animaciones CSS

Una de las características más impresionantes de CSS 3 es la posibilidad de crear animaciones de los elementos de la página. Todo ello nos abre infinitas posibilidades que antes solo estaban disponibles para los programadores Javascript o diseñadores con Flash. Explora las animaciones CSS que seguro te sorprenderás.

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.

> **Nota:** Hablando de Javascript y aunque quede fuera de lo que vamos a tratar en este artículo, existen algunos frameworks interesantes que permiten desarrollar animaciones de una manera bastante más sencilla de lo que sería si no los utilizásemos. Por ejemplo el popular jQuery permite animaciones simples pero que se hacen en cuestión de segundos. Tambien se pueden realizar animaciones CSS con Javascript incluso con frameworks especializados en este área como las librerias jsAnim.

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.

> **Nota:** Esta novedad puede resultar realmente interesante pero debemos decir que en el momento de escribir este artículo no funciona con todos los navegadores. Sólo admiten las animaciones CSS los navegadores Safari y Google Chrome. El resto esta trabajando duro para implementarlo y esperamos que en poco tiempo se puedan ver en todos los navegadores, sobre todo en Firefox y Internet Explorer que son los más utilizados.

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.

Inconvenientes de las animaciones CSS

Como todo en la vida, también existen algunas desventajas al trabajar con animaciones en CSS. Lo cierto es que la mayor que se podría destacar es sólo circunstancial, debido al poco soporte que existe actualmente a esta utilidad. Tenemos dos principales inconvenientes.

En el momento de escribir este artículo las animaciones CSS no son admitidas por los principales navegadores (Ninguna utilidad para animación con CSS 3 se puede utilizar en Internet Explorer y en Firefox algunas cosas ya podemos ver que funcionan a medias, pero aun le queda largo camino por recorrer). Consume bastantes recursos de máquina para producir las animaciones.

También podremos encontrar que existe alguna dificultad a la hora de la programación, pero no más de la que encontraríamos si tuviésemos que utilizar otros lenguajes o tecnologías distintos de CSS.

Finalmente, volvemos a remarcar que, debido a la imposibilidad de ver los resultados en todos los clientes web, al menos por el momento, deberemos utilizar navegadores basados en Webkit, como son Safari o Google Chrome (siempre en su versión mas actualizada).

Sin mucho más que añadir a nuestra pequeña introducción a las animaciones CSS, comenzaremos a ver cómo se realizan en el siguiente artículo, que nos muestra un poco la teoría sobre este tema y nos da los principios básicos para poder realizar animaciones con CSS 3.

Videotutorial de animaciones CSS3

Además, os presentamos la grabación de un evento en directo que emitimos el 6 de junio de 2012 en el que hablamos sobre animaciones CSS. Se trata de una estupenda presentación para quienes quieren comenzar a realizar sus animaciones en CSS, para que tengan una visión global sobre cómo se realizan y cuáles son sus posibilidades. Esperamos que os guste!

Para ver este vídeo es necesario visitar el artículo original en:
http://desarrolloweb.com/articulos/intro-animaciones-css.html

Este artículo es obra de *Sara Alvarez*
Fue publicado por primera vez en *11/10/2010*
Disponible online en *http://desarrolloweb.com/articulos/intro-animaciones-css.html*

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.

Estamos continuando con la serie de artículos sobre animaciones CSS, una utilidad que sin duda se convertirá en un gran aliado de los diseñadores para crear webs más impactantes y basadas en estándares web. Hay que decir que, aunque en el momento de escribir estos artículos las animaciones CSS están disponibles tan solo en navegadores basados en Webkit, como Google Chorme, en poco tiempo serán compatibles con todos los navegadores.

Así pues, una vez que sabemos lo que es la animación CSS, podemos pasar a ver los conceptos clave que necesitamos para poder crear una animación.

Nota: Otros conceptos importantes a tener en cuenta que hemos visto en otras entregas de contenido dentro de DesarrolloWeb.com es la programación HTML 5 y la especificación CSS 3.

Conceptos clave que tenemos que conocer bien:

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 seria 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.

> **Nota:** Este código está incompleto todavía, porque sólo es un fragmento del CSS y faltaría la parte del HTML. De momento por ahora está bien, pero aun tenemos que explicar otras cosas sobre las animaciones CSS que debes saber sobre el estado actual de los navegadores y sus particularidades de compatibilidad con esta nueva capacidad de CSS 3. Pero no preocupoarse, porque en breve veremos un códico completo y funcionando, en el siguiente

artículo.

Con estos conceptos podemos decir que estamos preparados para que en el siguiente artículo realicemos un ejemplo muy básico de animación dentro de una página web. Recordando siempre que no funciona actualmente en todos los navegadores y que trabajamos con CSS 3 y HTML 5.

Este artículo es obra de *Miguel Angel Alvarez*
Fue publicado por primera vez en *12/10/2010*
Disponible online en *http://desarrolloweb.com/articulos/conceptos-animacion-css.html*

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.

Antes de empezar, queremos dar un par de referencias sobre artículos tratados anteriormente en DesarrolloWeb.com donde explicamos algunos detalles a conocer previamente, para poder entender bien el presente ejemplo. En concreto os recomendamos empezar la lectura en el artículo que ofrece la Introducción a las Animaciones CSS y también por el articulo donde se explican los Conceptos Básicos sobre Animaciones CSS.

Para los que no lo sepan, remarcamos que para ver este ejemplo necesitamos un navegador basado en Webkit, que es el único que realmente funciona en el momento de escribir este artículo, para poder ver las animaciones CSS 3. Ejemplos de navegadores Webkit son los conocidos Google Chrome o Safari.

Además, tenemos que saber que, como en el momento de escribir este artículo las especificaciones de CSS 3 todavía están en fase de revisión, los atributos que necesitamos de Hojas de Estilo en Cascada para animación no están funcionando, sino que tendremos que utilizar unos atributos especiales que sólo entienden los navegadores basados en webkit. Estos atributos son exactamente iguales a los que comentábamos en el artículo de los conceptos sobre animaciones CSS, lo único que tenemos que añadir el prefijo -webkit- antes del nombre de cada uno de ellos.

Para que os quede más claro vamos a pasar a construir nuestro ejemplo más sencillo ya con el paquete webkit. Este ejemplo va a consistir en un texto dentro de un DIV, que se va a mover de izquierda a derecha y viceversa, por un tiempo infinito.

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>
```

Con todo esto funcionando nuestra animación quedaría así.

Como podéis ver este ejemplo de animación CSS 3 con Webkit es bien sencillo y no requiere unos conocimientos extensos en CSS. Os dejo el 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>
```

En el siguiente articulo realizaremos un ejemplo algo más complicado para crear un fondo en movimiento, del tipo de las pantallas de matrix.

Este artículo es obra de *Sara Alvarez*
Fue publicado por primera vez en *13/10/2010*
Disponible online en *http://desarrolloweb.com/articulos/animacion-texto-css3.html*

Fondo animado con CSS 3

Práctica sobre Animaciones CSS3, en la que construimos un fondo animado al estilo de la película Matrix, utilizando únicamente código CSS 3.

Estamos revisando las particularidades del sistema de animación que se ha creado a partir del lenguaje CSS, a partir de su tercera especificación. Todavía es todo un poco el preludio de lo que podremos hacer en el futuro con CSS y cómo nos ayudará a crear páginas dinámicas sin necesidad de Javascript.

Todo esto ya comenzamos a explicarlo en la introducción a las animaciones CSS, de modo que ahora vamos a hacer una práctica para asimilar los conocimientos adquiridos. Aunque hablando de ejemplos prácticos de animaciones CSS 3 queremos señalar que ya habíamos visto una práctica anteriormente, algo más sencilla que la que veremos a continuación, sobre animación de texto con CSS.

Para comenzar, podemos ver primero como quedaría nuestro ejemplo funcionando. Y recordar que para verlo, de momento, tenéis que utilizar Google Chrome o Safari.

Como podéis ver, es el típico fondo de letras que se mueven en la vertical, que conocemos si hemos visto la película Matrix, pero donde hemos colocado letras en vez de símbolos.

Lo primero que vamos a crear es nuestro archivo CSS con un fotograma clave y las propiedades de la animación.

```
@-webkit-keyframes fondo{
   from {top: -260%;}
   to {top: 100%;}
}
```

Con esto le estamos diciendo que vaya de arriba hacia abajo, con ello dará la sensación de que la animación comienza desde arriba de la parte superior de nuestra pantalla.

A continuación vemos los estilos y propiedades que vamos a necesitar para animar nuestro fondo.

```
#matrix{
   margin: 1em auto;
   width: 100%;
   height: 100%;
   overflow: hidden;
   background: #000;
   color: rgba(0, 255, 0, .7);
   text-shadow: rgba(255, 255, 255, .8) 0px 0px 4px;
   position: relative;
}
```

Aquí principalmente damos los estilos a nuestro fondo y decimos que el texto que aparezca sea verde.

```
#matrix DIV{
   position: absolute;
   top: 0;
   /* rotamos el texto*/
   -webkit-transform-origin: 0%;
   -webkit-transform: rotate(90deg);
   -webkit-animation-name: fondo;
   -webkit-animation-iteration-count: infinite;
   -webkit-animation-direction: normal;

}
```

Aquí ya damos las propiedades necesarias a nuestra animación. Lo primero que hacemos es rotarla 90 grados, esto lo hacemos para que las letras caigan y no vayan de izquierda a derecha. Ya solo nos queda lo que vimos en el artículo anterior, el nombre, las veces que se repite y la dirección.

Los estilos que vemos a continuación lo que van a hacer es que los textos bajen a diferentes velocidades y en distintos intervalos de tiempo.

```
.f1{
   font-size: 1.2em;
}
.f2{
   font-size: .9em;
}
.c1{
   color: rgba(0, 255, 0, .5);
}
.d1{
   -webkit-animation-duration: 4s;
}
```

```
.d2{
   -webkit-animation-duration: 6s;
}
.d3{
   -webkit-animation-duration: 8s;
}
.d4{
   -webkit-animation-duration: 10s;
}
.de{
   -webkit-animation-delay: 3s;
}
```

Con esto la parte de CSS esta lista, ahora pasamos a nuestro código HTML.

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

<html>
<head>
   <title>Fondo animado con CSS</title>

<link rel="stylesheet" href="matrix.css" type="text/css">
</head>

<body>

   <div id="matrix">
      <div class="d3 f1" style="left:30px;">hgd4ldhbc9kpugccsrq</div>
      <div class="d1 f2 c1" style="left:60px;">trzews0yfkldf4cvgbhjn</div>
      <div class="d2 f1" style="left:80px;">sodhr49whuyfbsrnlepjh</div>
      <div class="d4 c3 de" style="left:110px;">fue73sjf0tbkxpowfvn</div>
      <div class="d2 c1" style="left:140px;">sjdhfgueiwfgivecjowxkwpkpomvcjoeuur</div>
      <div class="d3 f2 c1" style="left:170px;">trzhfggh</div>
      <div class="d1 c1" style="left:185px;">thgpmsrthdvytfv09876tqfgv</div>
      <div class="d3 de" style="left:200px;">dhiwgfduesrjm</div>
      <div class="d4 f1" style="left:250px;">osadharshdyfeujm</div>
      <div class="d2 de" style="left:290px;">fwedjsdjhlgmrghftdercwewergjm</div>
      <div class="d3 f2" style="left:310px;">sodhr49whayfbsrnlepjh</div>
      <div class="d1 f1" style="left:350px;">p00oi0nf5sujhgdgbrjs36gdrgjpo</div>
      <div class="d4 c1" style="left:390px;">h68kgdetklbfbeswk</div>
      <div class="d2 de" style="left:410px;">dfrttvbscfsr</div>
      <div class="d1 c1 de" style="left:4430px;">sgfyb0hfrese4kc</div>
      <div class="d3 f1" style="left:530px;">hgd4ldhbc9kpugccsrq</div>
      <div class="d1 f2 c1" style="left:560px;">trzews0yfkldf4cvgbhjn</div>
      <div class="d2 f1" style="left:580px;">sodhr49whuyfbsrnlepjh</div>
      <div class="d4 c3 de" style="left:610px;">fue73sjf0tbkxpowfvn</div>
      <div class="d2 c1" style="left:740px;">sjdhfgueiwfgivecjowxkwpkpomvcjoeuur</div>
      <div class="d3 f2 c1" style="left:770px;">trzhfggh</div>
      <div class="d1 c1" style="left:785px;">thgpmsrthdvytfv09876tqfgv</div>
      <div class="d3 de" style="left:800px;">dhiwgfduesrjm</div>
      <div class="d4 f1" style="left:850px;">osadharshdyfeujm</div>
      <div class="d2 de" style="left:890px;">fwedjsdjhlgmrghftdercwewergjm</div>
      <div class="d3 f2" style="left:910px;">sodhr49whayfbsrnlepjh</div>
```

```
        <div class="d1 f1" style="left:950px;">p00oi0nf5sujhgdgbrjs36gdrgjpo</div>
        <div class="d4 c1" style="left:990px;">h68kgdetklbfbeswk</div>
        <div class="d2 de" style="left:1010px;">dfrttvbscfsr</div>
        <div class="d1 c1 de" style="left:1030px;">sgfyb0hfrese4kc</div>
    </div>
</body>
</html>
```

Si os fijáis tenemos varios DIV con las letras y la posición en la que se colocan, si no hacemos esto nos saldría todo junto y no se verían por toda la pantalla. Podemos poner tantos DIV como deseemos siempre viendo la posición en la que los colocamos y teniendo en cuenta las resoluciones de pantalla.

Como podéis ver, nuestro ejemplo resulta un poco laborioso y teniendo en cuenta que simplemente sería el fondo, al que deberíamos añadir el resto de nuestro código para la creación de nuestra web, la cosa se queda un poco más compleja de lo que desearíamos. De momento, hay que tomar este ejemplo como algo experimental, pues debido a su complejidad y que las animaciones CSS no funcionan en todos los navegadores, no sería muy recomendable utilizar la técnica para una web real. Por lo menos queremos que sirva para ver el tema de las animaciones CSS 3, pues el ejemplo es bastante interesante.

Por último os dejo el ejemplo funcionado en una página aparte.

En el siguiente artículo veremos un ejemplo parecido pero esta vez caerá nieve y con distintos movimientos en vez de la caída recta de este ejemplo.

Este artículo es obra de *Sara Alvarez*
Fue publicado por primera vez en *29/11/2010*
Disponible online en *http://desarrolloweb.com/articulos/fondo-animado.html*

Fondo nevando con CSS 3

Taller de animación CSS 3, en el que creamos un fondo donde simulamos que están cayendo copos de nieve, utilizando tan sólo con CSS 3.

Las características de las Hojas de Estilo en Cascada en su tercera especificación, incluyen diversas utilidades para crear animaciones sin necesidad de programar, solamente definiendo estilos CSS. Es algo que ya empezamos a explicar en el artículo de introducción a las animaciones CSS.

En estos momentos estamos ofreciendo varios ejemplos prácticos sobre cómo crear distintos tipos de animaciones CSS. En el presente artículo veremos un ejemplo de fondo con copos de nieve que caen por nuestra pantalla, con movimientos de forma ondulante. Es un ejemplo parecido al que hicimos anteriormente, en el artículo sobre el fondo de Matrix, pero en esta ocasión en vez de hacer movimientos rectos en la vertical, serán movimientos curvos.

De modo que, la complicación adicional en esta práctica será realizar esos movimientos circulares típicos de los copos de nieve, para los que vamos a enseñar una nueva propiedad de las animaciones CSS, transform, a la que se accede en el momento de escribir este artículo con la propiedad temporal -webkit-transform, que está disponible para navegadores como Chrome o Safari. Dicho sea de paso, aprender esta propiedad nos vendrá muy bien porque dará mucha opciones a la hora de animar objetos.

Como hacemos en estos ejercicios prácticos, lo primero que recomendamos hacer es ver el ejemplo funcionando en una página aparte, donde se podrá apreciar el resultado final de la práctica que pretendemos realizar.

Ahora vamos a ver el código CSS necesario para crear nuestro fondo. El cual será reducido, pero donde simplemente tenemos que aumentar los tamaños para sacar el mismo resultado en un fondo mayor.

```
#container{
   background: #666 url(images/fondo-nieve.jpg) no-repeat;
   width: 400px;
   height: 300px;
   position: relative;
}
```

A continuación damos propiedades al conjunto de la nieve en general y luego a cada copo en particular.

```
@-webkit-keyframes nieve{
   from {top: -10px;}
   to {top: 450px;}
}

@-webkit-keyframes copos{
    0% { -webkit-transform: rotate(-180deg) translate(0px, 0px);}
    100% { -webkit-transform: rotate(180deg) translate(10px, 75px);}
}
```

Con el primero decidimos que los copos vayan de la parte superior a la inferior, del tal forma que parezca que aparecen por arriba y desaparecen por abajo.

En el segundo lo que le decimos es que cada copo va ir girando haciendo una semicircunferencia, saliendo el copo desde 10px para finalizar en 75px.

Una vez que tenemos ya nuestros fotogramas clave nos queda darle propiedades a nuestras animaciones. Para ello creamos un estilo para las animaciones nieve y copos y a parte, una serie de estilos para cada copo en particular. Aquí, como en el ejercicio anterior, podemos crear todos los copos que necesitemos.

```
#snow div {
```

```
   position: absolute;
   top: -40px;
   -webkit-animation-name: nieve, copos;
   -webkit-animation-iteration-count: infinite;
   -webkit-animation-direction: normal;
   -webkit-animation-timing-function: ease-in;
}
.copos {
   color: #fff;
   font-size: 1em;
   position: absolute;
}
.copos.f1 {
   left: 40px;
   -webkit-animation-duration: 5s;
}
.copos.f2 {
   font-size: 1.8em;
   left: 120px;
   -webkit-animation-duration: 7s;
}
.copos.f3 {
   left: 200px;
   -webkit-animation-duration: 8s;
}
.copos.f4 {
   font-size: 1.5em;
   left: 280px;
   -webkit-animation-duration: 6s;
}
```

Considero que no hace falta explicar este código, que ya hemos visto en los ejemplos anteriores de nuestro manual, por lo que paso a poneros el código HTML necesario para que nuestro fondo funcione.

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

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

<body>
<div id="container">

<div id="snow" class="snow">
   <div class="copos f1">?</div>
   <div class="copos f2">?</div>
   <div class="copos f3">?</div>
   <div class="copos f4">?</div>
</div>
<div id="ground"></div>
```

```
</div>

</body>
</html>
```

De este código cabe destacar que no hemos utilizado imágenes para generar los copos de nieve, simplemente hemos colocado el código HTML que nos muestra el símbolo copo.

Podéis volver a ver el ejemplo funcionando en una pagina aparte.

En el siguiente artículo veremos cómo animar un muñeco para ir haciendo algo más que fondos animados.

Este artículo es obra de *Sara Alvarez*
Fue publicado por primera vez en *30/11/2010*
Disponible online en *http://desarrolloweb.com/articulos/fondo-nevado-css3.html*

Dibujo animado con CSS 3

Práctica general sobre diferentes técnicas de animación CSS, en el que realizamos un dibujo animado por varios elementos que tienen diferentes particualidades.

En el presente ejercicio vamos a ver cómo podemos animar un dibujo con CSS 3 en varios puntos, que nos darán una idea global de las posibilidades de las animaciones CSS y nos ofrecerán una práctica completa de las técnicas que que hemos visto hasta el momento.

En lo relacionado a las animaciones de objetos con CSS ya hemos visto algunos ejemplos interesantes con técnicas que vamos a utilizar en el presente artículo. Los ejemplos vistos hasta ahora son la animación de texto, el fondo de tipo Matrix y el fondo con los copos de nieve. En este artículo vamos a mezclarlo todo y realizar un dibujo con distintas animaciones.

Si lo deseas, puedes comenzar viendo el resultado final de este ejemplo. Pero ten en cuenta que de momento funciona sólo con Chorme o Safari.

Para la realización de esta práctica, lo primero que vamos a hacer es crear un fondo simple para nuestra animación. Algo como esto:

A continuación vamos a crear una nube, que se vaya moviendo por el cielo de nuestro dibujo.

Una vez que tenemos la nube, colocamos en nuestro archivo CSS el siguiente código:

```
@-webkit-keyframes nube {

from {
left: 100px;
opacity: 1;

}

to {
left: 300px;
opacity: 0;
}

}

#nube{
position:relative;
-webkit-animation-name: nube;
-webkit-animation-duration: 3s;
-webkit-animation-iteration-count: infinite;
-webkit-animation-direction: alternate;
-webkit-animation-timing-function: ease-in;
}
```

Del fotograma clave no hay mucho que comentar, simplemente va de 100px a 300px y va

perdiendo opacidad, es decir, va haciéndose más transparente a medida que se mueve hacia la derecha, con lo que va desapareciendo de nuestro dibujo. En cuanto a la animación pues es más de lo mismo, dura 3 segundos y va y viene a su posición inicial.

Ahora vamos a animar el árbol para que veamos como caen sus hojas, para ello nos creamos una hoja y colocamos el siguiente código en nuestro CSS:

```
@-webkit-keyframes hojas{
   from {
      top: 150px;
      opacity: 1;
   }
   to {
      top: 400px;
      opacity: 0;
   }
}
@-webkit-keyframes hoja{
   0% { -webkit-transform: rotate(-180deg) translate(0px, 0px);}
   100% { -webkit-transform: rotate(180deg) translate(10px, 75px);}
}

#hoja div {
   position: absolute;
   top: -40px;
   -webkit-animation-name: hoja, hojas;
   -webkit-animation-iteration-count: infinite;
   -webkit-animation-direction: normal;
   -webkit-animation-timing-function: ease-in;
}
.hojas {
   position: absolute;
}
.hojas.f1 {
   left: 40px;
   -webkit-animation-duration: 5s;
}
.hojas.f2 {
   font-size: 1.8em;
   left: 100px;
   -webkit-animation-duration: 7s;
}
```

Como podéis ver tenemos dos fotogramas clave, uno que nos dice cómo se van a mover todas las hojas que caen del árbol y el otro que nos dice el movimiento que va a realizar cada hoja por separado.

Después hemos ido dando propiedades a las hojas, con unas propiedades generales definidas en el DIV hoja y luego propiedades especificas para cada hoja, creo que no hace falta explicarlas porque las hemos visto todas en artículos anteriores de nuestro manual de CSS 3.

Y por último nos queda la animación del muñeco que hemos colocado. Para ello hemos realizado un dibujo con el tronco, las piernas y la cabeza y después hemos dibujado los brazos por separado que son los que se van a mover. Esta es la animación más complicada ya que hacer cuadrar los brazos con el tronco y conseguir el movimiento adecuado ha resultado un poco lioso y aun así no ha terminado de salir perfecto. Su código CSS es el siguiente:

```
@-webkit-keyframes mano2 {
   0% {
      -webkit-transform: rotate(0deg) translate(0px,0px);
   }
   100% {
      -webkit-transform: rotate(-60deg) translate(0px,0px);
   }
}

#manoderecha{
   position: absolute;
   top:250px;
   left:279px;
   width:-10px;
}

#manoizquierda{
   position: absolute;
   top:250px;
   left:340px;
   width:8px;
   -webkit-animation-name: mano2;
   -webkit-animation-duration: 3s;
   -webkit-animation-iteration-count: infinite;
   -webkit-animation-direction: alternate;
}
```

Esto lo que hace es dejarnos la mano derecha fija y la izquierda nos está saludando. Todo esto utilizando las propiedades ya vistas.

Con esto terminamos nuestro código CSS para las animaciones y nos falta un estilo más para nuestro ejemplo.

```
#container{
   background: #666 url(images/fondo-muneco2.jpg) no-repeat;
   width: 600px;
   height: 400px;
   position: relative;
}

#situacion {
   width: 200px;
   position: absolute;
   left: 300px;
   top: 200px;
```

```
}
```

Uno para las propiedades del dibujo en general y el de situación para colocar el muñeco en la posición deseada.

Ahora ya sólo nos queda el código HTML que es el siguiente:

```
<!DOCTYPE html>
<html>

<head>

   <title>Dibujo animado CSS 3</title>
<link rel="stylesheet" href="muneco.css" type="text/css">

</head>

<body>

   <div id=container>
      <div id="nube"><img src="images/nube2.png" alt="" border="0"></div>
      <div id="situacion"><img src="images/tronco-muneco2.png" alt="" border="0"></div>
      <div id="manoizquierda"><img src="images/mano-izquierda.png" alt="" border="0"></div>
      <div id="manoderecha"><img src="images/mano-derecha.png" alt="" border="0"></div>
      <div id="hoja" class="hoja">
         <div class="hojas f1"><img src="images/hoja.png" alt="" border="0"></div>
         <div class="hojas f2"><img src="images/hoja.png" alt="" border="0"></div>
      </div>
    </div>

   </body>
</html>
```

Todo va dentro de nuestro DIV container y luego cada animación dentro de un DIV. Otra cosa ha destacar es que hay tantos DIV hojas f1 como hojas queremos que se caigan, en nuestro caso 2, pero podría haber más si quisiéramos.

Ahora sólo nos queda ver nuevamente nuestro ejemplo funcionando.

Este artículo es obra de *Sara Alvarez*
Fue publicado por primera vez en *01/12/2010*
Disponible online en *http://desarrolloweb.com/articulos/dibujo-animado-css3.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.

A lo largo de varios artículos hemos presentado un avance de las posibilidades de CSS 3 para crear animaciones, que son sin duda una de las novedades más impactantes de esa nueva versión del lenguaje de estilos. En este artículo avanzamos un poco en dinámica de los anteriores del manual de CSS3 y pasamos a utilizar lo aprendido sobre las animaciones CSS para darle un mejor aspecto a nuestra web.

En este ejemplo comprobaremos como con CSS3 podemos, simplemente aplicando unas cuantas reglas de estilo, dar un formato totalmente distinto e impactante a algo tan simple como unas imágenes y un texto. Además, veremos cómo aplicarle dinamismo a nuestra presentación y una pequeña interactividad, definiendo unicamente unos estilos CSS.

Si hoy la práctica que nos ocupa puede parecer relativamente sencilla, hace años sería muy complicada de hacer y tendríamos que emplear (aparte de HTML y CSS) muchas líneas de código y conocimientos avanzados de Javascript o de alguna librería extra como jQuery. Es una muestra excelente de cómo CSS 3 sustituirá a Javascript en algunas parcelas de sus usos habituales. Estoy segura que os encantará.

> **Nota:** Vamos a darle estilo a una pequeña galería de imágenes con unas fotografías fijas, que tenemos en un código HTML que he escrito a mano. Pero podéis imaginar que, con un poco de programación del lado del servidor, podemos hacerla más dinámica y sacar las imágenes de una base de datos. Así mismo y en mi defensa debo decir que no soy diseñadora, así que si alguno piensa que esto mismo podría haber sido más bonito con otros criterios estéticos, espero sepa comprenderme y fijarse en lo que realmente importa, que es la implementación de las animaciones con CSS.

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.

> **Nota:** Aviso de antemano que este ejemplo está preparado para el navegador Google Chrome o Safari, ~~pero que pronto otros navegadores serán capaces de interpretar estas reglas de CSS 3~~ aunque otros navegadores ya serán capaces de entenderlo. No obstante, igual haya que cambiar algo el código del ejemplo, para incorporar los atributos y valores CSS definitivos, que se definan cuando el estándar termine de estar recomendado para implementación.

```
<!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 que 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 titulo 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;
}
```

Nota: Por favor, ten en cuenta que ahora que las animaciones CSS ya son un estándar, no

es necesario usar el prefijo "-webkit-". Quítalo y funcionará en todos los navegadores.

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.

Nota: Insistimos, hay que recordar que en el momento de redactar este artículo solamente es visible en Chrome y Safari, pero hoy ya se podría usar en otros navegadores si quitas el prefijo "webkit".

Bueno y el resultado final sería el siguiente.

En el siguiente artículo veremos otro ejemplo de álbum de fotos con CSS 3.

Este artículo es obra de *Sara Alvarez*
Fue publicado por primera vez en *21/03/2011*
Disponible online en *http://desarrolloweb.com/articulos/album-efectos-css3.html*

Segundo ejemplo de álbum fotográfico animado con CSS 3

Realizamos otro ejemplo de animaciones para tu galería de fotos solamente con CSS 3.

Continuamos explorando las posibilidades de las animaciones CSS3, con nuevos ejemplos prácticos que estamos seguros resultarán interesantes para los aficionados al desarrollo de páginas web.

En este artículo vamos a realizar una galería de fotos, en la que las imágenes aparecen desordenadas y al pasar por ellas pasan a primer plano y se ven a su tamaño real. En el artículo anterior vimos como realizar un ejemplo distinto de galería inspirado en las clásicas Polaroid. Ahora podemos ver otra idea de galería fotográfica con animaciones CSS 3, que tendrá un resultado bastante curioso, pero veremos que es más sencilla de realizar que la anterior.

Vamos a colocar primero nuestro código HTML para ir viendo poco a poco cómo construiríamos nuestro ejemplo.

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

<html>
<head>
```

```
   <title>Album de fotos con CSS</title>
<link rel="stylesheet" href="album2.css" type="text/css">
</head>

<body>
<div id="galeria">
   <div id="imagen1">
      <img src="images/image1.jpg">

   </div>
   <div id="imagen2">
      <img src="images/image2.jpg">

   </div>
   <div id="imagen3">
      <img src="images/image3.jpg">

   </div>
   <div id="imagen4">
      <img src="images/image4.jpg">

   </div>
   <div id="imagen5">
      <img src="images/image5.jpg">

   </div>
   <div id="imagen6">
      <img src="images/image6.jpg">

   </div>
</div>

</body>
</html>
```

En principio no hay nada que no sepamos, es simplemente un `<DIV id=galeria>` y dentro hemos ido colocando las imágenes dentro de otros `<DIV>`.

Ahora pasamos al código CSS que es el realmente importante:

Primero unos estilos generales para la galería:

```
#galeria {
      width: 800px;
      height: 600px;
      margin: 20px auto;
      padding: 40px;
      position: relative;
}
```

Luego unos estilos específicos para cada imagen:

```
#galeria div {
   position: absolute;
   overflow: hidden;
   opacity: 0.9;
   -webkit-transition: all 0.5s linear;
   -webkit-border-radius: 15px;
   border-radius: 15px;
   -webkit-box-shadow: -3px -3px 50px #666;
   box-shadow: -3px -3px 50px #666;
   -webkit-transform: scale(0.60);
}
```

Aquí lo que hacemos es crear un borde alrededor de cada imagen, dejarla un poco transparente y bajarla el tamaño, para que luego cuando pasemos el ratón crezca a su tamaño real.

> **Nota:** Como venimos advirtiendo, las animaciones CSS 3 todavía no se encuentran disponibles en todos los navegadores. En estos ejemplos estamos trabajando a modo de demostración, con los atributos de CSS 3 propios del navegador Chrome o Safari. En breve podremos usar los atributos propios de CSS 3, que serían simplemente los mismos, pero quitando el prefijo -webkit. Pero de momento, para ver los ejemplos en funcionamiento y que se vean todas las animaciones CSS, hay que ejecutarlos en Google Chrome o Safari.

```
#galeria div:hover {
   z-index: 999;
   -webkit-transform: rotate(0deg);
   opacity: 1;

}
```

Con este código conseguimos que la imagen vaya al primer plano de la pantalla y se vea sin transparencia, al pasar el ratón por encima de la imagen.

```
#imagen1 {
   top: 50px;
   left: 130px;
   z-index:1;
}

#imagen2 {
   top: 300px;
   left:100px;
   z-index:7;
}

#imagen3 {
   top: 420px;
   left:350px;
   z-index:3;
```

```
}

#imagen4 {
   top: 135px;
   left: 450px;
   z-index:4;
}

#imagen5 {
   top: -100px;
   right: 200px;
   z-index:5;
}

#imagen6 {
   top: 220px;
   right: 120px;
   z-index:6;
}
```

Y finalmente con estos estilos lo que hacemos es colocar las imágenes donde nos de la gana. Si tuviéramos un sistema automático que nos mostrara más imágenes estos estilos los podríamos poner directamente en nuestro código.

Otra opción posible para este álbum es la de rotar las imágenes para que parezcan más desordenadas, para ello tan solo tendríamos que añadir la propiedad -webkit-transform: rotate(xdeg); a cada imagen, siendo x los grados a rotar.

Podemos ver como se queda nuestro album en el siguiente enlace.

En el siguiente artículo veremos como crear un menú dinámico sin Javascript ni jQuery ni programación en otros lenguajes que no sean HTML y CSS 3.

Este artículo es obra de *Sara Alvarez*
Fue publicado por primera vez en *29/03/2011*
Disponible online en *http://desarrolloweb.com/articulos/album-fotos-css3-2.html*

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.

En este nuevo artículo de nuestro manual sobre animación con CSS 3 vamos a realizar un menú dinámico con tan solo HTML y CSS 3. Personalmente me ha parecido sencillo y puede darnos muchas posibilidades a la hora de maquetar nuestra web, ya que no necesitamos tener conocimientos de Javascript ni de jQuery.

Antes de ponernos manos a la obra, sugiero ver el resultado final en este enlace. Pero recordar

que solamente los navegadores que soporten animaciones CSS lo podrán ver correctamente. En el momento de escribir este artículo los navegadores con los que verás bien el ejemplo son: Google Chrome y Safari. Aunque cabe destacar que Opera y Mozilla Firefox estan en base beta y algunas propiedades como transition y box-shadow funcioan.

> **Nota:** Como la especificación de las animaciones CSS todavía se encuentra en fase de borrador, a todas las propiedades asociadas con ellas se les añade el prefijo "-moz-" para usarse en Gecko(Firefox 4). Para la compatibilidad con WebKit, se aconseja usar también el prefijo "-webkit-" y para la compatibilidad con Opera, el prefijo "-o-". Es decir, por ejemplo, especificarías la propiedad de transición como -moz-transition, -webkit-transition y -o-transition.

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="http://www.desarrolloweb.com/html/">HTML</a></li>
<li><a href="http://www.desarrolloweb.com/css/">CSS</a></li>
<li><a href="http://www.desarrolloweb.com/javascript/">Javascript</a></li>
<li><a href="http://www.desarrolloweb.com/asp/">ASP</a></li>
<li><a href="http://www.desarrolloweb.com/php/">PHP</a></li>

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

Como podéis ver no tiene nada del otro mundo, es simplemente un menú en el que utilizamos un listado.

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 fono 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;
}
```

Y 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.

Podemos ver el ejemplo en marcha en una página aparte.

Este artículo es obra de *Sara Alvarez*
Fue publicado por primera vez en *14/04/2011*
Disponible online en *http://desarrolloweb.com/articulos/menu-animado-css3.html*

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>
```

Cómo podéis ver tenemos la clase boton pequeno azul, pero podría ser boton mediano rojo, o similar, siempre manteniendo fija la clase boton que es la principal y la que nos va a dibujar el botón.

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;
}
```

El código no vamos a pasar a comentarlo linea a linea ya que damos por sentado que habéis leído nuestro manual de CSS 3 y comprendéis dicha nomenclatura.

Sólo destacar que hemos utilizado los prefijos moz- y webkit- para que sea compatible con varios navegadores, como son Firefox, Chrome y Safari, y que además hemos creado el efecto de cambiar un poco el color de fondo al pasar el ratón sobre el botón. Esto lo hemos realizado en el estilo .boton:hover, .boton.hover

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;
}
```

Con esto tenemos todo lo necesario para mostrar nuestros botones. Podemos ver el ejemplo funcionando en una página aparte.

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;
}
```

Este artículo es obra de *Sara Alvarez*
Fue publicado por primera vez en *08/09/2011*
Disponible online en *http://desarrolloweb.com/articulos/botones-css3.html*

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.

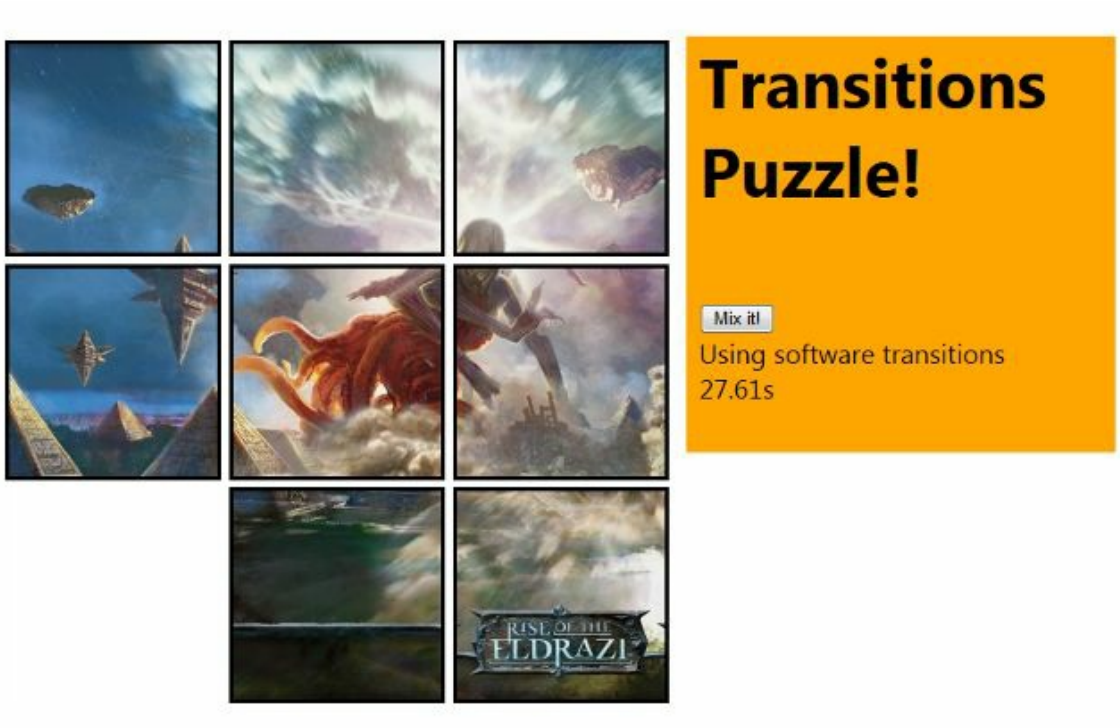
Según se indica en la especificación "CSS Transitions Module Level 3" que podemos leer en el sitio web del W3C, las transiciones CSS3 permiten realizar cambios en los valores CSS de manera progresiva durante un tiempo determinado.

El objetivo de este artículo consiste, en primer lugar, en describir el concepto de transición y después ver cómo funciona CSS3 Transitions y cómo podemos manejar el resultado en los navegadores que no soportan esta funcionalidad.

Además, sugiero que os leáis el artículo "Introducción a las animaciones con CSS3" (escrito por mí mismo) que os puede servir de complemento ideal para este otro.

Para saber cómo se utilizan las transiciones con CSS3 he desarrollado un ejemplo de un juego que utiliza CSS3 Transitions para animar las fichas de un puzzle (y que incluye un *fallback* hecho con JavaScript si el navegador no soporta CSS3 Transitions):

Para disponer de la versión ejecutable de este juego, visita mi blog aquí. El código del juego lo puedes encontrar aquí.



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)

SVG

Las propiedades de los objetos SVG son susceptibles de animación si se definen con la cláusula animatable:true en la especificación SVG: http://www.w3.org/TR/SVG/struct.html.

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;
```

Cuando actualicemos la opacidad en nuestra etiqueta, se inicia una animación que va alterando progresivamente el valor desde el actual al nuevo, durante 0,5 segundos, con una función de ajuste no lineal a lo largo de ese intervalo (easing), que nos proporciona el efecto de una animación suave.

Transiciones no lineales

La línea "transition-timing-function" indica que la transición no va a ser lineal, sino que utilizará una función de tiempo que genera una animación no lineal. Básicamente, las transiciones CSS3 utilizan la función de curva de Bezier cúbica para suavizar las transiciones mediante el cálculo de distintos valores de velocidad durante el intervalo de ejecución.

Hay otras funciones de transición también soportadas:

1. Linear: transición a velocidad constante
2. Cubic-bezier: la velocidad se calcula de acuerdo con la función de curva Bezier cúbica definida por dos puntos de control, P0 y P1 (así que tenemos que definir 4 valores en este caso: P0x-P0y y P1x- P1y)
3. Ease: La velocidad se calcula según la función de curva Bezier cúbica con los parámetros (0.25, 0.1, 0.25, 1)
4. Ease-in: La velocidad se calcula con la función de curva Bezier con los parámetros (0.42, 0, 1, 1)
5. Ease-inout: La velocidad se calcula con la función de curva Bezier con los parámetros (0.42, 0, 0.58, 1)
6. Ease-out: La velocidad se calcula con la función de curva Bezier con los parámetros (0, 0, 0.58, 1)

En http://www.catuhe.com/msdn/transitions/easingfunctions.htm tenemos una herramienta

de simulación (utilizando SVG por supuesto) para mostrar el impacto de cada una de estas funciones de ajuste.

Este simulador está escrito totalmente en JavaScript para facilitar la comprensión de la función:

```
TRANSITIONSHELPER.computeCubicBezierCurveInterpolation = function (t, x1, y1, x2, y2) {
// Extrae X (que aquí equivale al tiempo)
var f0 = 1 - 3 * x2 + 3 * x1;
var f1 = 3 * x2 - 6 * x1;
var f2 = 3 * x1;

var refinedT = t;
for (var i = 0; i < 5; i++) {
var refinedT2 = refinedT * refinedT;
var refinedT3 = refinedT2 * refinedT;

var x = f0 * refinedT3 + f1 * refinedT2 + f2 * refinedT;
var slope = 1.0 / (3.0 * f0 * refinedT2 + 2.0 * f1 * refinedT + f2);
refinedT -= (x - t) * slope;
refinedT = Math.min(1, Math.max(0, refinedT));
}

// Resuelve la curba Bezier cúbica para el valor x dado
return 3 * Math.pow(1 - refinedT, 2) * refinedT * y1 +
3 * (1 - refinedT) * Math.pow(refinedT, 2) * y2 +
Math.pow(refinedT, 3);
};
```

Este código es la implementación de la curva Bezier cúbica basada en esta definición y puedes encontrar el código fuente del simulador aquí.

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

Este sería un ejemplo de uso en IE10:

```
block.addEventListener("MSTransitionEnd", onTransitionEvent);
```

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:

Informe obtenido desde http://caniuse.com/#search=CSS3 transitions.

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.

Transiciones sin CSS3 Transitions

Así pues, para incluir un fallback para los navegadores no compatibles con CSS3 Transitions, vamos a desarrollar un pequeño kit de herramientas que harán lo mismo por programación.

Lo primero, vamos a crear un objeto contenedor para nuestro espacio de nombres:

```
var TRANSITIONSHELPER = TRANSITIONSHELPER || {};

TRANSITIONSHELPER.tickIntervalID = 0;

TRANSITIONSHELPER.easingFunctions = {
linear:0,
ease:1,
easein:2,
easeout:3,
easeinout:4,
custom:5
};

TRANSITIONSHELPER.currentTransitions = [];
```

Para dar soporte al mismo nivel de funciones de velocidad de transición, tenemos que declarar un "enum" con todos los campos necesarios.

El kit de herramientas se basa en una función a la que se llama cada 17ms (para conseguir animaciones a 60fps). La función irá pasando a través de una colección de transiciones activas. Para cada transición, el código evalúa el siguiente valor calculado a partir del valor actual y el indicado como valor final.

Necesitamos una serie de funciones muy útiles para extraer el valor de las propiedades y las unidades de medida utilizadas:

```
TRANSITIONSHELPER.extractValue = function (string) {
try {
var result = parseFloat(string);

if (isNaN(result)) {
return 0;
}

return result;
```

```
} catch (e) {
return 0;
}
};

TRANSITIONSHELPER.extractUnit = function (string) {

// si el valor está vacío, suponemos que se expresa en px
if (string == "") {
return "px";
}

var value = TRANSITIONSHELPER.extractValue(string);
var unit = string.replace(value, "");

return unit;
};
```

La función principal procesará las transiciones activas y llamará a la función cúbica Bezier para calcular los valores actuales:

```
TRANSITIONSHELPER.tick = function () {
// Procesando transiciones
for (var index = 0; index < TRANSITIONSHELPER.currentTransitions.length; index++) {
var transition = TRANSITIONSHELPER.currentTransitions[index];

// calcula el nuevo valor
var currentDate = (new Date).getTime();
var diff = currentDate - transition.startDate;

var step = diff / transition.duration;
var offset = 1;

// Función de timing
switch (transition.ease) {
case TRANSITIONSHELPER.easingFunctions.linear:
offset = TRANSITIONSHELPER.computeCubicBezierCurveInterpolation(step, 0, 0, 1.0, 1.0);
break;
case TRANSITIONSHELPER.easingFunctions.ease:
offset = TRANSITIONSHELPER.computeCubicBezierCurveInterpolation(step, 0.25, 0.1, 0.25, 1.0);
break;
case TRANSITIONSHELPER.easingFunctions.easein:
offset = TRANSITIONSHELPER.computeCubicBezierCurveInterpolation(step, 0.42, 0, 1.0, 1.0);
break;
case TRANSITIONSHELPER.easingFunctions.easeout:
offset = TRANSITIONSHELPER.computeCubicBezierCurveInterpolation(step, 0, 0, 0.58, 1.0);
break;
case TRANSITIONSHELPER.easingFunctions.easeinout:
offset = TRANSITIONSHELPER.computeCubicBezierCurveInterpolation(step, 0.42, 0, 0.58, 1.0);
break;
case TRANSITIONSHELPER.easingFunctions.custom:
offset = TRANSITIONSHELPER.computeCubicBezierCurveInterpolation(step, transition.customEaseP1X, transition.customEaseP1Y,
transition.customEaseP2X, transition.customEaseP2Y);
```

```
break;
}

offset *= (transition.finalValue - transition.originalValue);

var unit = TRANSITIONSHELPER.extractUnit(transition.target.style[transition.property]);
var currentValue = transition.originalValue + offset;

transition.currentDate = currentDate;

// ¿Transición muerta?
if (currentDate >= transition.startDate + transition.duration) {
currentValue = transition.finalValue; // Clamping
TRANSITIONSHELPER.currentTransitions.splice(index, 1); // Removing transition
index--;

// Evento final
if (transition.onCompletion) {
transition.onCompletion({propertyName:transition.property, elapsedTime:transition.duration});
}
}

// Valor final
transition.target.style[transition.property] = currentValue + unit;
}
};
```

La versión actual del kit de herramientas solo soporta valores numéricos, pero si quieres animar valores complejos (como el color), simplemente tienes que descomponerlos en valores individuales.

El registro de una transición en el sistema se haría utilizando este código:

```
TRANSITIONSHELPER.transition = function (target, property, newValue, duration, ease, customEaseP1X, customEaseP1Y, customEaseP2X, customEaseP2Y,
onCompletion) {

// Creamos una nueva transición
var transition = {
target: target,
property: property,
finalValue: newValue,
originalValue: TRANSITIONSHELPER.extractValue(target.style[property]),
duration: duration,
startDate: (new Date).getTime(),
currentDate: (new Date).getTime(),
ease:ease,
customEaseP1X:customEaseP1X,
customEaseP2X:customEaseP2X,
customEaseP1Y: customEaseP1Y,
customEaseP2Y: customEaseP2Y,
onCompletion: onCompletion
};
```

```
// Arranca el sevicio de tick si es preciso
if (TRANSITIONSHELPER.tickIntervalID == 0) {
TRANSITIONSHELPER.tickIntervalID = setInterval(TRANSITIONSHELPER.tick, 17);
}

// Elimina las transiciones anteriores para la misma propiedad y objeto
for (var index = 0; index < TRANSITIONSHELPER.currentTransitions.length; index++) {
var temp = TRANSITIONSHELPER.currentTransitions[index];

if (temp.target === transition.target && temp.property === transition.property) {
TRANSITIONSHELPER.currentTransitions.splice(index, 1);
index--;
}
}

// Registramos
if (transition.originalValue != transition.finalValue) {
TRANSITIONSHELPER.currentTransitions.push(transition);
}
};
```

La función "tick" arranca la primera vez que se activa la transición.

Finalmente, tendremos que utilizar modernizr para saber si el navegador soporta CSS3 Transitions. Si no, podemos dejar nuestro kit como fallback.

El código para el TransitionsHelper se puede descargar desde aquí: http://www.catuhe.com/msdn/transitions/transitionshelper.js

Por ejemplo, en mi juego de puzzle, utilizo este código para animar las piezas:

```
if (!PUZZLE.isTransitionsSupported) {
TRANSITIONSHELPER.transition(block.div, "top", block.x * totalSize + offset, 500, TRANSITIONSHELPER.easingFunctions.ease);
TRANSITIONSHELPER.transition(block.div, "left", block.y * totalSize + offset, 500, TRANSITIONSHELPER.easingFunctions.ease);
}
else {
block.div.style.top = (block.x * totalSize + offset) + "px";
block.div.style.left = (block.y * totalSize + offset) + "px";
}
```

Se puede ver que podría haber empleado otra solución para animar las cuadrículas en el caso de que CSS3 Transitions estuviera soportado: podría haber definido una colección de clases CSS3 con valores predefinidos left y top (uno por cada celda) para aplicarlos a las piezas correspondientes.

Ya existen algunos entornos y kits de desarrollo que soportan las transiciones por software:

- jQuery.transition.js
- jQUery-Animate-Enhanced

Y por supuesto, siempre podemos utilizar el conocido y potente método animate() de jQuery.

Conclusión

Como has podido comprobar, CSS3 Transitions es sin duda una forma muy sencilla de incorporar animaciones a nuestros proyectos. Podemos crear una aplicación más reactiva simplemente añadiendo algunas transiciones cuando vayamos a modificar ciertos valores.

Pero además tenemos dos alternativas para implementar un fallback con JavaScript:

1. Podemos hacerlo todo con JavaScript, y si detectamos que el navegador soporta las transiciones CSS3, inyectar las declaraciones CSS3 en la página.
2. O bien podemos utilizar la manera más ortodoxa (utilizando declaraciones CSS3 auténticas en los archivos CSS) y detectar si necesitamos o no aplicar un fallback desde JavaScript. Para mí esta es la mejor opción, puesto que el fallback tiene que considerarse una opción alternativa, no la principal. De cara al futuro próximo, todos los navegadores van a soportar CSS3 Transitions y en este caso solo tendríamos que eliminar el código de fallback. Pero además es que es la mejor (o si no, la única) manera de dejar todo el bloque CSS bajo el control del equipo de diseñadores, retirándolo en lo posible de la parte de programación.

Para saber más

- Blog sobre CSS3 Animations de David Rousset
- Especificaciones CSS3 Transitions
- Ejemplos de IE Test Drive sobre transiciones con CSS3

Otros artículos de interés:

- http://addyosmani.com/blog/css3transitions-jquery/
- http://css3.bradshawenterprises.com/
- http://webdesignerwall.com/trends/47-amazing-css3-animation-demos

Este artículo es obra de *David Rousset*
Fue publicado por primera vez en *28/06/2012*
Disponible online en *http://desarrolloweb.com/articulos/transiciones-css3.html*

Ejemplos de animaciones CSS3

Ejercicios sobre animaciones CSS3 realizados en un evento en directo emitido en DesarrolloWeb.com con distintas demostraciones de las posibilidades de esta técnica del HTML5.

El pasado día 6 de junio retransmitimos para DesarrolloWeb.com una charla en donde estuve hablando de las animaciones CSS3. Una interesante introducción a las posiblidades que existen ya mismo para las personas, que como yo, hemos salido del mundo Flash para

incorporarnos en los estándares basados en HTML5.

Durante ese evento que compartí con otras personas del equipo de DesarrolloWeb.com, realicé diversos ejemplos que quiero ahora publicar en este artículo. Realmente los ejemplos no son nada complicados, sino que sirven para ilustrar las posibilidades de las animaciones CSS3 a personas que no han trabajado nunca con esta utilidad relativamente nueva.

> **Nota:** Al final de este artículo podrás encontrar la grabación de la cita de animaciones CSS3, en donde se ofrecen muchas explicaciones interesantes que os complementarán los códigos que estoy compartiendo con vosotros.

Ejemplo básico: mi primera animación CSS

En este ejemplo realizamos nuestra primera animación. Es muy simple, pero ya demuestra bien las posibilidades que tendremos a poco que dominemos este aspecto de las CSS3.

Tenemos una capa "div" que tiene una imagen de fondo. Esa capa se mueve por la pantalla con tres "Keyframes". Los *frames* clave sirven para expresar los distintos estados de la animación y corresponden al mismo concepto que podrán haber aprendido otros desarrolladores en Flash o ex-flasheros como yo.

El código del ejemplo es el siguiente:

```
<!DOCTYPE html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>GIRAR</title>

<style type="text/css">
#circulo {
background-image: url(circulo.png);
height: 100px;
position: relative;
width: 100px;
-webkit-animation:gira 3s alternate infinite;
}
```

```
@-webkit-keyframes gira
{
0% {top: 0px; left: 600px;-webkit-transform: rotate(0deg);}
50% {top: 130px; left: 200px; -webkit-transform: rotate(180deg);}
100% {top: 400px; left:600px; -webkit-transform: rotate(360deg);}
}
</style>
</head>
<body>
<div id="circulo"></div>
</body>
</html>
```

> **Nota:** fíjate que estamos utilizando el prefijo "-webkit" por lo que sólo estará funcionando en Chrome o Safari.

El enlace para ver este ejemplo en marcha es el siguiente: desarrolloweb.com/articulos/ejemplos/css/animaciones/girar.html

Ejemplo 2: efectos hover y diferentes bordes

En este segundo ejemplo ya hacemos algo un poco más complejo, y cargamos otros prefijos de navegadores distintos de Webkit, para que la animación sea compatible con más *browsers*.

Como se puede ver, tenemos varias capas, con distintas clases. Todas tienen efectos de animación aplicados a los enlaces (etiqueta A), pero con las clases conseguimos que esos efectos varíen de una capa a otra.

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8" />
<title></title>
<style>
body {
padding: 10px 0 0 10px;
}
#contenedor {margin: 30px auto; width:960px;}
a {
display:inline-block;
-webkit-box-sizing:border-box;
-moz-box-sizing:border-box;
box-sizing:border-box;
width:100px;
height:100px;
-webkit-border-radius:50px;
-moz-border-radius:50px;
```

```
border-radius:50px;

background-image:url(circulo.png);
background-repeat:no-repeat;
background-origin:border-box;
background-position:50% 50%;

border-width:50px;
border-color:rgba(0,0,0,0);

-webkit-transition:0.5s ease;
transition:0.5s ease;
margin:20px;
}

a:hover {
border-width:0;
border-color:rgba(0,0,0,0.5);
}

a.one {border-style:solid;}
a.two {border-style:dashed;}
a.san {border-style:dotted;}

a.yon {border:50px solid rgba(0,0,0,0.7);}
a.goo {border:50px dashed rgba(0,0,0,0.7);}
a.rok {border:50px dotted rgba(0,0,0,0.7);}
a.ryk {border:50px double rgba(0,0,0,0.7);}
a.yon:hover {border: 1px solid rgba(0,0,0,0.7);}
a.goo:hover {border: 1px dashed rgba(0,0,0,0.7);}
a.rok:hover {border: 1px dotted rgba(0,0,0,0.7);}
a.ryk:hover {border: 1px double rgba(0,0,0,0.7);}

a.x7,
a.x8,
a.x9 {
-webkit-border-radius:0;
-moz-border-radius:0;
border-radius:0;
}
a.x7 {border:50px solid rgba(0,0,0,0.7);}
a.x8 {border:50px dashed rgba(0,0,0,0.7);}
a.x9 {border:50px dotted rgba(0,0,0,0.7);}
a.x7:hover {border: 1px double rgba(0,0,0,0.7);}
a.x8:hover {border: 1px dashed rgba(0,0,0,0.7);}
a.x9:hover {border: 1px dotted rgba(0,0,0,0.7);}
</style>
</head>
<body>
<div id="contenedor">
<a class="one" href="#"></a>
<a class="two" href="#"></a>
<a class="san" href="#"></a>
<br>
<a class="yon" href="#"></a>
```

```
<a class="goo" href="#"></a>
<a class="rok" href="#"></a>
<a class="ryk" href="#"></a>
<br>
<a class="x7" href="#"></a>
<a class="x8" href="#"></a>
<a class="x9" href="#"></a>
</div>
</body>
</html>
```

Podemos ver este segundo ejemplo en el siguiente enlace: desarrolloweb.com/articulos/ejemplos/css/animaciones/hover.html

Esto es todo, espero que haya servido de utilidad a quien quiera que desee aprender esta faceta del HTML5. Recordar que en DesarrolloWeb.com hay muchos otros ejemplos y explicaciones de Animaciones CSS en el Manual de CSS3.

Este artículo es obra de *Jorge Vargas Vega*
Fue publicado por primera vez en *14/08/2012*
Disponible online en *http://desarrolloweb.com/articulos/ejemplos-animaciones-css3.html*

Animaciones CSS3 vs Animaciones jQuery

Comprobaremos in situ cómo una simple animación afecta al rendimiento del navegador en cuanto al uso de la memoria se refiere. Una vez acabado el artículo extraeremos una conclusión con un ganador.

Originalmente Flash nos pavimentó el camino a la hora de caracterizar con más que texto o imágenes las páginas web. Permitió a los desarrolladores incluir animaciones y otros ricos efectos, guiándonos hacia una experiencia de usuario más colorida y variada. Sin embargo, Flash estaba plagado de fallos, como los problemas de seguridad, largos tiempos de carga en redes mediocres, etc. Entonces vinieron las librerías JavaScript como jQuery, Prototype y MooTools, las cuales desplazaron muchos de los problemas de Flash corriendo nativamente en el navegador, además de hacer que fuera más sencillo para el desarrollador medio usar Javascript a la hora de crear efectos y animaciones. Han pasado ya unos cuantos años y tenemos con nosotros las habilidades de CSS3 para crear animaciones, las cuales ofrecen ventajas adicionales, como el incremento potencial de velocidad, debido a que son renderizadas directamente por el navegador.

Pero, ¿cuál es realmente la mejor solución para nosotros a la hora de usar animaciones? En este artículo echaremos un vistazo a cómo se crean las animaciones en jQuery y en CSS3, y a cómo se desempeña cada uno.

Introducción a la animación en jQuery

La librería jQuery reduce mucho la complejidad para el desarrollador. Por ejemplo, aquí tenemos un caso en el que crearemos un simple `<div>`, el cual será animado posteriormente al hacer clic en un botón:

######### 1. Incluimos la librería jQuery en la página, por ejemplo así:

```
// Se recomienda que uses un CDN para invocar la librería jQuery. Aquí hay un enlace al CDN de Google:
<script src="http://ajax.googleapis.com/ajax/libs/jquery/1.8.1/jquery.min.js?ver=1.8.1"></script>
```

######### 2. Creamos el elemento `<div>`, y le damos un estilo básico:

```
<div></div>
div {
margin-left:10px;
margin-bottom:10px;
background-color:red;
opacity:1;
float:left;
width:100px;
height:1px;
}
```

######### 3. Creamos el botón que al ser pulsado accionará la función que anime el `<div>`:

```
<button id="inicio">Comenzar Animación</button>
```

######### 4. Escribimos un poco de código jQuery para seleccionar el elemento `<div>` y aplicar los efectos una vez que el botón haya sido accionado. Una vez hecho, la altura del `<div>` se incrementará a 25px y la opacidad se reducirá de 1 a 0.5 en un periodo de 2.000 milisegundos (2 segundos):

```
$("#inicio").click(function(){
$("div").animate({
opacity: 0.5,
height:"25px",
} , 2000);
});
```

Es muy bonito ver lo fácil que es animar un elemento con jQuery usando nada más que unas pocas líneas de código, y lo mejor sobre jQuery es que tu código funcionará igual de bien en todos los navegadores, ¡incluyendo versiones tan vetustas como IE6!

Introducción a la animación en CSS3

Para crear una animación en CSS3 necesitamos especificar dos diferentes construcciones en nuestro CSS. Antes que nada, necesitamos especificar qué forma tomará la animación usando la regla @keyframes, la cual se asemeja a ésto:

```
@keyframes mi-animacion {
0% {height:0px; opacity:1; }
100% {height:25px; opacity:0.5; }
}
```

"mi-animacion" es el nombre por el cual nuestra animación será identificada, y las líneas diferentes son diferentes *keyframes* (fotogramas clave). En cada caso, las propiedades que hay entre llaves indican las propiedades de animación que se tendrán en cuenta en cada fase de ésta, y los porcentajes dictan la duración de cada fase —en este caso particular nuestra animación es muy simple, así que únicamente definimos el comienzo (0%) y el final (100%) de la animación. Para aplicar la animación a un elemento de la página necesitas referirla a éste usando la propiedad "animation":

```
div {
margin-left:10px;
margin-bottom:10px;
background-color:red;
opacity:0.5;
float:left;
width:100px;
height:25px;
animation: mi-animacion 2s;
}
```

Es funcionalmente idéntico al ejemplo que hemos visto antes de jQuery: el `<div>` se anima en un periodo de 2 segundos, con su altura incrementándose a 25px y su opacidad reduciéndose de 1 a 0.5. Suena sencillo, ¿eh? Desafortunadamente, el soporte de los navegadores no es tan bueno en el caso de las animaciones CSS3 — las versiones de IE por debajo de la 10 no soportan esta propiedad-, y aunque la gran mayoría de los otros grandes navegadores sí lo hacen, y la especificación de animación es ya estable en ellos, la mayoría de ellos no han

eliminado los requerimientos de prefijos particulares de cada navegador, haciendo que para que se de soporte a la propiedad haya que incluirlos, siendo éstos -webkit-, -moz-, -ms- y -o-. Por ejemplo, para el navegador Opera sería así:

```
@-o-keyframes mi-animacion {
0% {height:0px; opacity:1; }
100% {height:25px; opacity:0.5; }
}
```

y

```
div {
...
-o-animation: mi-animacion 2s;
}
```

Esto hace que la base del código comience a ser desalentadora, aunque si lo que quieres es reducir todas las líneas de prefijos a una sola, puedes usar la solución del preprocesador. Aquí hay un par de ellos:

1. **SASS** — Un procesador CSS que te permite incluir variables, funciones y otras características logrando que tu CSS sea más rápido y eficiente en algunos casos. Usar SASS no debería afectar al rendimiento de tu sitio.
2. **Prefixfree** — Una librería JavaScript que simplemente añade los prefijos correctos a tu CSS cuando el navegador accede a él, en tiempo de ejecución. Esto significa la ejecución de más JavaScript en la máquina del cliente, lo cual podría añadir un poco de carga en el rendimiento, pero relativamente pequeño. La pega es que el diseño de la web se quedará a cuadros si el usuario tiene desactivado Javascript.

Así que por el momento parece que jQuery es el camino a seguir, especialmente en términos de compatibilidad de navegadores. Si quieres un sitio que permanezca usable en los viejos navegadores que no soportan la animación, es aconsejable que el ajuste por defecto de las propiedades que son animadas se indiquen en el estado final de la animación. Por ejemplo recuerda que hemos ajustado la altura a 25px y la opacidad a 0.5, así que si la animación no está disponible el navegador ajusta los valores por defecto a los del estado final, es decir, que en un navegador antiguo los valores que se verían serían los de 25px y 0.5, no los iniciales.

> **Nota:** para más detalles sobre las animaciones CSS, ojea este link de Chris Mills.

Guerra de animaciones: CSS3 vs jQuery

Para testar el rendimiento de las animaciones CSS3 frente a las animaciones jQuery, probemos la siguiente prueba. Usaremos el código que hemos enseñado arriba, pero en cada caso animaremos 300 `<div>`s simultáneamente. Así será fácil medir el tiempo que le lleva a cada

animación el ser ejecutada.

Animaciones CSS3

El gráfico de ejecución para el test de animaciones CSS3 se muestra en la Imagen 1. Este gráfico fue creado usando a herramienta Opera Dragonfly, y el navegador usado fue Opera 12 en Mac OS X.

(Leyenda Imagen 1: el tiempo que lleva animar 300 `<div>s` con animación CSS).

Como se puede ver en la imagen, la animación se completa en unos 2.9 segundos.

Vamos a analizar ahora el uso de memoria. Este gráfico fue creado usando la opción de Memoria dentro de la pestaña Timeline en las Herramientas de Desarrollador de Chrome 21.

(Leyenda Imagen 2: la memoria usada al animar 300 `<div>`s con animación CSS.

La memoria usada durante la animación CSS3 es muy pequeña -alrededor de 1.5MB, con solo 100 acciones requeridas. Los datos finales del test son:

- Número de acciones realizadas para acabar la animación: 100
- Tiempo tomado en ejecutar la animación: 2.9 segundos
- Memoria consumida al final de la animación: 1.5 MB

Ahora vamos a proceder realizando el mismo experimento con animaciones jQuery.

Animaciones jQuery

El gráfico de ejecución para el test de animación jQuery sale en la Imagen 3. El gráfico fue creado usando la herramientas de Opera Dragonfly, y el navegador usado fue Opera 12 en Mac OS X.

(Leyenda Imagen 3: El tiempo que tardan en animarse 300 `<div>s` con jQuery).

La operación completa lleva unos 5 segundos – ¿demasiado tiempo? La animación actual no lleva demasiado tiempo, pero tiene un tiempo extra con respecto a la animación cargada con JavaScript (nótese también que hay una pequeña fracción de tiempo entre que el botón se pulsa y la animación comienza, efectivamente, a proceder). También el número de acciones

realizadas por el navegador es mayor, siendo más de 2.000, enorme cifra comparada con las únicamente 100 en la misma animación hecha con CSS3.

Incluso sin usar herramientas de desarrollo, notarás que una vez que el botón “Empezar Animación” es pulsado, hay un breve intervalo de tiempo antes de que la animación comience.

En cuanto al uso de memoria, fijémonos en la Imagen 4. El gráfico fue generado usando la misma opción de Memoria de las Herramientas de Desarrollo de Chrome 21 que en el caso anterior.

(Leyenda Imagen 4: La memoria usada animando 300 `<div>s` con jQuery).

Cuando se trata de memoria, la animación muestra un hambre brutal, devorando cerca de ¡6 MB! Los datos finales del test son:

- Número de acciones realizadas para ejecutar la animación: 2119
- Tiempo que tarda la animación en finalizar: 5 segundos
- Memoria consumida al final de la animación: 6 MB

Negativamente hay que informar que los tests de animaciones de arriba darán diferentes resultados en los diferentes navegadores, pero al menos se nos da una comparación razonable. Actualmente Chrome tiene el más veloz procesador de JavaScript y ejecuta las animaciones jQuery unos cuantos cientos de milisegundos más rápido que sus competidores. Sin embargo es una historia completamente diferente en cuanto a las animaciones CSS3. Opera 12 se lleva la palma creando suaves animaciones, liderando en cuanto a manipulaciones del DOM y procesado de CSS se refiere. Firefox 14 y Safari 6 hacen un buen trabajo en ambas áreas. La pesadilla del desarrollador, IE (siendo la última versión estable en el momento de escritura del artículo la 9) no soporta, directamente, animaciones CSS3, pero ejecuta las animaciones jQuery decentemente.

Y el ganador es...

¡CSS3! Claramente gana CSS3 a mucha distancia. La gigantesca diferencia en rendimiento es debida a que el procesador de CSS del navegador está escrito en C++ y el código nativo se ejecuta muy rápido, mientras que jQuery (JavaScript) es un lenguaje interpretado y el navegador no puede predecir por adelantado su comportamiento, intentando adivinar qué evento ocurrirá después.

Aunque los resultados de arriba indican que tienes que usar animaciones CSS3, deberías tener en cuenta las ventajas y desventajas que hemos discutido anteriormente en el artículo. Necesitas priorizar que aún una importante cantidad de gente usa Internet Explorer 7 y 8, así que deberías usar jQuery si es absolutamente necesario que tus animaciones funciones igual de bien que en todos los navegadores antiguos.

Alternativamente, puede que quieras contemplar cómo tus animaciones funcionan

defectuosamente en los navegadores sin soporte, en cuyo caso las animaciones CSS3 son la mejor opción.

Nótese que para animaciones simples, como la trivial mostrada en el ejemplo del test, probablemente podrías usar menos CSS si lo hicieras como transición en vez de como animación. Ya depende de la particular situación que vivas. Las transiciones son más rápidas pero dependen de los cambios de estado, mientras que las animaciones son más flexibles y poderosas. Elije la opción que más te convenga.

Siddharth Rao Fuente

Más características de las CSS3

Otra serie de características interesantes y útiles para el trabajo con hojas de estilo en cascada, en su especificación CSS3.

CSS3 Flexbox

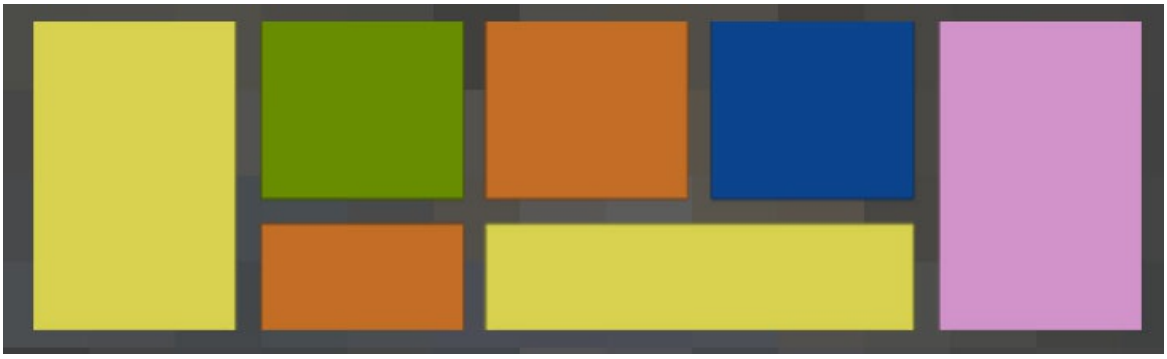
Qué es Flexbox y por qué tienes que comenzar a usar este potente mecanismo de CSS3 para componer layouts de una manera rápida, sencilla y poderosa.

Este es un artículo que nos sirve de introducción a Flexbox. Nos vamos a quedar en un conocimiento un tanto teórico, pero al final de este texto encontrarás un vídeo con poco más de dos horas, donde también lo conocerás por la práctica.

Antes de comenzar con Flexbox queremos que nos respondas una pregunta ¿Cuántas veces has sentido que CSS no era suficiente para resolver las necesidades de un proyecto? o quizás, ¿Cuántas veces has tenido que modificar el HTML para hacer posible la maquetación de unos elementos de una manera determinada? O peor, recurrir a Javascript para conseguir que se coloquen como deseas.

Si llevas un tiempo en el mundo del desarrollo para la web habrás observado que, a pesar que CSS ofrece muchas características para maquetación de contenidos y creación de "layouts", determinados comportamientos eran difíciles de producir y te obligaban a ingeniártelas de diversos modos para conseguir tus objetivos.

En resumen, si lo que necesitas es "clavar un diseño", para que se vea muy bien, tenías que sufrir con las herramientas disponibles hasta ahora. Esas y otras cuestiones se pretenden solucionar con Flexbox.



Qué es Flexbox

Flexbox es un módulo completo de layout disponible en la especificación de CSS3. Define cómo se muestran los elementos y cómo se relacionan con el resto. Como concepto, puedes entender

Flexbox como un modelo para la creación de layouts, que pretende mejorar los anteriores, aunque sin ser excluyente. Todas las técnicas disponibles antes de Flexbox tenían diversos problemas y limitaciones que se pretenden solucionar con esta especificación de CSS3. Flexbox es una herramienta muy avanzada para poder crear layouts de características avanzadas y necesarias en el día de hoy, donde es tan importante una estética cuidada y una gran adaptabilidad a distintos formatos de pantalla.

> **Nota:** hasta la fecha hemos maquetado con técnicas como float o display table, pero estos mecanismos no fueron pensados para utilizarse como los utilizamos. Los diseñadores los usaban y conseguían sus objetivos, muchas veces usando lo que se concen como "hacks css". Flexbox sí es un estándar pensado para hacer layouts y por tanto soluciona la mayoría de las necesidades de los desarrolladores sin tener que emplear técnicas rebuscadas.

En la práctica, Flexbox agrega un nuevo tipo de "display CSS", con una completa gama de nuevas propiedades aplicables a ese tipo de display, a partir de los que puedes conseguir cosas extraordinarias. Por aclararnos, igual que tienes en CSS el display "block", "inline", "inline-block", etc. y sabías todos las propiedades que te acepta ese tipo de elementos, ahora dispones de "flex" e "inline-flex", siendo que los elementos que tienen ese nuevo display aceptan una cantidad enorme de nuevas propiedades de gran utilidad.

En resumen, lo que antes tenías que conseguir con una docena de reglas y estilos CSS, cálculos, etc. ahora lo vas a poder implementar mucho más fácilmente, a veces incluso con una única propiedad.

En Flexbox diferenciamos dos elementos principales: la caja contenedora y los elementos que situamos dentro. Al aplicar un display flex o display inline-flex hacemos que una caja se comporte mediante este nuevo estándar y eso produce que los elementos que tiene como contenido se puedan distribuir con las propiedades de este estándar de maquetación.

El contenedor va a poder modificar las dimensiones y el orden de los items, para acomodarlos de distintas maneras controladas por el desarrollador. Podremos repartir el espacio entre ellos de diversas formas, para distribuirlo a nuestro antojo, permitir que los items se estiren para ocupar todo el contenedor, o se encojan para que quepan en él sin desbordar, de colocarse distribuidos en filas o en columnas, etc.

> **Nota:** para aclarar posibles dudas tenemos que advertir que Flexbox es una especificación de CSS3. No se trata de una librería de estilos como podría ser Bootstrap, sino de un estándar de la web. No necesitas instalar nada para que lo entiendan los navegadores, igual como tampoco necesitas instalar nada para para trabajar con cualquier característica actual de CSS. Usar Flexbox no es excluyente de usar cualquier cosa que ya estás usando de CSS. Es decir, si decides usar Flexbox en un momento dado, nada te impide seguir utilizando comportamientos anteriores como podría ser float o display table. Simplemente con Flexbox podrás hacer las cosas más rápidamente y llegar donde antes no era sencillo con otros atributos de CSS.

Qué soluciona Flexbox

Flexbox permite que se puedan posicionar elementos de una manera más concisa y distribuir los espacios entre ellos de forma más flexible. Permite gran cantidad de comportamientos, pero este sería un rápido resumen, si nos ceñimos a las cosas que antes eran muy difíciles de hacer mediante CSS anterior:

Alineación vertical:

Seguro que has sudado para conseguir alinear elementos en la vertical. A veces dado por imposible, hoy es algo que es muy sencillo con Flexbox.

Columnas de igual altura:

Conseguir que todos los items de un listado tengan la misma altura, independientemente de su contenido. También difícil de conseguir con CSS anterior, sobre todo cuando el contenido era variable e impredecible. Por supuesto, Flexbox también nos permite elementos con igual anchura, y aunque esto es algo que ya teníamos antes fácilmente, ahora resulta mucho más sencillo.

Cambiar el orden de los elementos:

Sin tener que cambiar el orden de los elementos en el HTML, con Flexbox conseguimos que se ordenen de maneras distintas al visualizarse en la página.

Con esto se consigue que los diseñadores, maquetadores y desarrolladores frontend en general tengan mucho mayor control sobre los elementos en la página y puedan, de una manera más detallada, especificar su apariencia y colocación, limitándose solamente a modificar los atributos CSS.

Flexbox y compatibilidad con navegadores

Flexbox hoy ya es una realidad, puesto que los navegadores modernos lo soportan y lo interpretan correctamente. Por tanto, está listo para usar en cualquier tipo de proyecto.

El único navegador que no lo soporta es Internet Explorer en versiones antiguas, como IE8 o IE9 (Navegadores que a día de hoy no están soportados ni por el propio fabricante). Si es requisito indispensable que tu web se vea igual en estos navegadores, tendrás que usar las técnicas de toda la vida, o fallbacks que enseguida comentamos. Pero antes de ello conviene ver en la siguiente tabla de compatibilidad, extraida de Caniuse:

Nota: ¿Por qué no debes preocuparte por IE8 o IE9? porque tienes cosas más importantes en las que pensar y decidir soportarlos producirá que tu web no pueda usar las herramientas más nuevas disponibles para hacer tu trabajo. La web probablemente acabe siendo peor en calidad y menos adaptada a los tiempos que corren y a ti te lleva mucho más tiempo el desarrollarla. Generalmente a los clientes hay que convencerles que no necesitan dar la mejor imagen para navegadores anticuados y uno de los mejores argumentos es cobrar más por hacer el trabajo. De todos modos, usar Flexbox no significa que el contenido no va a estar ahí. Simplemente que no va a lucir de la mejor manera. Hay técnicas, englobadas en el conocimiento de Resposive Web Design, que nos ofrecen la vía para que una web se vea correctamente en todos los navegadores y, a medida que el navegador es más avanzado, luzca mejor porque se usen nuevas y más potentes características. Si quieres profundizar lee el artículo Sobre Progressive Enhancement Vs Graceful Degradation.

Realmente IE8 o IE9 no son tan importantes en la actualidad, pues un pequeño porcentaje de usuarios (menor del 1% cada) lo usa. Lo que sí es más importante son las versiones antiguas de móviles, con Android 4.3 o anteriores. Esas versiones tienen navegadores antiguos que daban un soporte parcial a Flexbox, donde estamos obligados a usar los prefijos de las propiedades CSS ("vendor prefixes"). Por eso te vendría bien contar con algún tipo de preprocesador, postCSS y autoprefixer.

Aunque no suele ser la mejor solución, existen fallbacks o polyfills que aportan un soporte parcial a Flexbox, instalando una librería Javascript adicional. Son una opción cuando necesitamos soportar Flexbox en algunos navegadores antiguos, aunque también hay que advertir que tendrá un coste en términos de rendimiento / peso, y quizás no todo se vea exactamente igual que en navegadores con soporte nativo. Un ejemplo de polyfill lo encuentras en Flexibility.

Conoce Flexbox en la primera clase del Taller Profesional de Flexbox

Si quieres aprender más y además ver cómo se comportan las propiedades principales de Flexbox en una serie de ejemplos prácticos, te recomendamos continuar viendo el siguiente vídeo.

Es una clase impartida en octubre de 2016, la primera del Taller de Flexbox de EscuelaIT. Es

un taller en el que se va abordar el desarrollo con Flexbox por la práctica y con ejemplos reales.

Para ver este vídeo es necesario visitar el artículo original en:
http://desarrolloweb.com/articulos/css3-flexbox.html

Este artículo es obra de *Diana Aceves*
Fue publicado por primera vez en *14/10/2016*
Disponible online en *http://desarrolloweb.com/articulos/css3-flexbox.html*

Propiedad text-overflow CSS3

En qué consiste la propiedad text-overflow de CSS3, casos en los que la podemos usar y compatibilidad con navegadores.

En el Manual de CSS3 todavía no habíamos hablado de una propiedad bastante útil, denominada text-overflow. Nos permite indicar qué aspecto tendrá un texto cuando sobrepasa el espacio disponible dentro de una caja. Más concretamente, cómo queremos que se señale al usuario que un texto no se está pudiendo visualizar completamente dentro del elemento donde se encuentra.

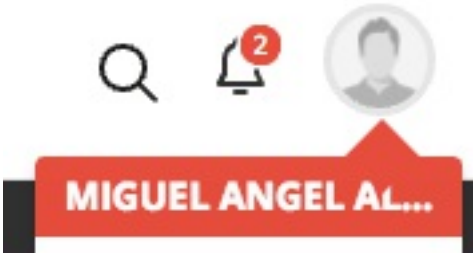
Antes de CSS3, cuando queríamos que se señale al usuario que un texto no se muestra por completo, teníamos que introducir algo de programación, para detectar los casos en los que era muy largo y entonces cortarlo programáticamente. Esa situación puede seguir siendo necesaria en muchos casos, pero realmente en muchos otros es un gasto de tiempo innecesario, porque el propio CSS puede cortarla en el lugar adecuado y mostrarlo al usuario convenientemente.

Es una propiedad sencilla de usar, por lo que no requerirá muchas explicaciones, pero no todo el mundo la conoce y la usa habitualmente. Sobre todo es interesante un valor en concreto de entre los posibles de la propiedad: "ellipsis", que nos introduce tres puntos suspensivos al final del texto que se está cortando.

```
.texto-potencialmente-largo {
    text-overflow: ellipsis;
}
```

Esto lo que quiere decir es que, si el texto potencialmente largo no cabe en la anchura del elemento, le coloque tres puntos suspensivos para que el usuario sea consciente de ello.

Puedes ver el efecto en la imagen siguiente, en la que mi nombre no cabe por completo en la caja donde está situado:



Nota: Ten en cuenta que con otras propiedades de CSS puedes indicar qué quieres que pase con ese texto potencialmente largo, por ejemplo que desborde la caja o que se muestre hasta donde pueda según su tamaño, que se recorten las palabras con saltos de línea o que se muestren todas en la misma línea. La propiedad text-overflow lo que nos indica es cómo se debe representar visualmente ese recorte en el texto.

Posibles valores de text-overflow

Como posibles valores de esta propiedad encontramos:

- clip: Es el valor predeterminado y quiere decir que el texto simplemente se recortará, sin mostrar ningún efecto además del propio recorte donde toque.
- ellipsis: Coloca tres puntos suspensivos al lado del texto, indicando que el texto no cabe completamente.
- string: Se prevé que se pueda indicar una cadena para colocar en vez de los puntos suspensivos. Su valor sería por ejemplo "--" y colocaría esa cadena al final del texto recortado. Sin embargo, esta funcionalidad es solo experimental y no funciona realmente en todos los navegadores, por lo que no se recomienda usar.
- fade: es otro valor experimental, que sirve para generar un efecto "fade-out" su valor puede ser una longitud y un porcentaje, pero de momento no se puede usar. Además puedes colocar los valores "initial" e "inherit" para restaurar la propiedad a su valor inicial o que herede del contenedor donde está situado.

Compatibilidad de text-overflow

Esta propiedad se puede usar sin problema alguno, ya que es compatible con todos los navegadores, incluso muchos que son realmente antiguos.

La soportan IE desde la versión 6 y prácticamente todos los demás, incluidos por supuesto Firefox, Chrome, etc.

Solo ten en cuenta que los valores que realmente te funcionarán son "clip" el predeterminado (que no muestra nada) y "ellipsis" que permite colocar esos tres puntos suspensivos que hemos explicado ya.

Este artículo es obra de *Miguel Angel Alvarez*
Fue publicado por primera vez en *06/04/2017*
Disponible online en *http://desarrolloweb.com/articulos/propiedades-textoverflow-css3.html*