Unidad Didáctica

CSS 3

Índice de contenido

1. Border-radius

Nuevocss3_2.css

```
body {
	text-align: center;
}
#principal {
	display: block;
	width: 500px;
	margin: 50px auto;
```

```
        padding: 15px;
        text-align: center;
        border: 1px solid #999999;
        background: #DDDDDD;
        -moz-border-radius: 20px;
        -webkit-border-radius: 20px;
        border-radius: 20px;
}
#titulo {
        font: bold 36px verdana, sans-serif;
}
```

La propiedad border-radius en este momento es experimental por lo que debemos usar los prefijos **–moz-** y **-webkit-** para que funcionen en navegadores basados en motores Gecko y WebKit, como Firefox, Safari y Google Chrome . Si todas las esquinas tienen la misma curvatura podemos utilizar un solo valor. Sin embargo, como ocurre con las propiedades margin y padding, podemos también declarar un valor diferente por cada una:

Nuevocss3_3.css

```
body {
text-align: center;
}
#principal {
display: block;
width: 500px;
margin: 50px auto;
padding: 15px;
text-align: center;
border: 1px solid #999999;
background: #DDDDDD;
-moz-border-radius: 20px 10px 30px 50px;
-webkit-border-radius: 20px 10px 30px 50px;
border-radius: 20px 10px 30px 50px;
```

```
}
#titulo {
font: bold 36px verdana, sans-serif;
}
```

Recorriendo la caja en dirección de las agujas del reloj, los valores se aplicarán en el siguiente orden: esquina superior izquierda, esquina superior derecha, esquina inferior derecha y esquina inferior izquierda. Los valores son siempre dados en dirección de las agujas del reloj, comenzando por la esquina superior izquierda.

Al igual que con margin o padding, border-radius puede también trabajar solo con dos valores. El primer valor será asignado a la primera y tercera equina (superior izquierda, inferior derecha), y el segundo valor a la segunda y cuarta esquina (superior derecha, inferior izquierda).

También podemos dar forma a las esquinas declarando un segundo grupo de valores separados por una barra. Los valores a la izquierda de la barra representarán el radio horizontal mientras que los valores a la derecha representan el radio vertical. La combinación de estos valores genera una elipsis:

```
body {
text-align: center;
}
#principal {
display: block;
width: 500px;
margin: 50px auto;
padding: 15px;
text-align: center;
border: 1px solid #999999;
background: #DDDDDD;
-moz-border-radius: 20px / 10px;
-webkit-border-radius: 20px / 10px;
border-radius: 20px / 10px;
}
#titulo {
font: bold 36px verdana, sans-serif;
}
```

2. Box-shadow

Gracias a CSS3 y a la nueva propiedad box-shadow podremos aplicar sombras a nuestras cajas con sólo una simple línea de código:

```
body {
text-align: center;
}
#principal {
display: block;
width: 500px;
margin: 50px auto;
padding: 15px;
text-align: center;
border: 1px solid #999999;
background: #DDDDDD;
-moz-border-radius: 20px;
-webkit-border-radius: 20px;
border-radius: 20px;
-moz-box-shadow: rgb(150,150,150) 5px 5px;
-webkit-box-shadow: rgb(150,150,150) 5px 5px;
box-shadow: rgb(150,150,150) 5px 5px;
}
#titulo {
font: bold 36px verdana, sans-serif;
}
```

La propiedad box-shadow necesita al menos tres valores.

- El primero es el color. Este valor fue construido utilizando la función rgb() y números decimales, pero podemos escribirlo en números hexadecimales también.

- Los siguientes dos valores, expresados en pixeles, establecen el desplazamiento de la sombra. Este desplazamiento puede ser positivo o negativo. Los valores indican, respectivamente, la distancia horizontal y vertical desde la sombra al elemento. Valores negativos posicionarán la sombra a la izquierda y arriba del elemento, mientras que valores positivos crearán la sombra a la derecha y debajo del elemento. Valores de 0 o nulos posicionarán la sombra exactamente detrás del elemento, permitiendo la posibilidad de crear un efecto difuminado a todo su alrededor.

La sombra que obtuvimos hasta el momento es sólida, sin gradientes o transparencias (no realmente como una sombra suele aparecer). Existen algunos parámetros más y cambios que podemos implementar para mejorar la apariencia de la sombra.

- Un cuarto valor que se puede agregar a la propiedad ya estudiada es la *distancia de difuminación*. Con este efecto ahora la sombra lucirá real. Puede intentar utilizar este nuevo parámetro declarando un valor de 10 pixeles , como en el siguiente ejemplo:

```
box-shadow: rgb(150,150,150) 5px 5px 10px;
```

- Agregando otro valor más en pixeles al final de la propiedad desparramará la sombra. Este efecto cambia un poco la naturaleza de la sombra expandiendo el área que cubre. A pesar de que no recomendamos utilizar este efecto, puede ser aplicable en algunos diseños. Intente agregar un valor de 20 pixeles al final del estilo.

IMPORTANTE: Siempre recuerde que en este momento las propiedades estudiadas son experimentales. Para usarlas, debe declarar cada una agregando los prefijos correspondientes, como –moz- o -webkit-, de acuerdo al navegador que usa (en este ejemplo, Firefox o Google Chrome).

- El último valor posible para box-shadow no es un número sino más bien una palabra clave: **inset**. Esta palabra clave convierte a la sombra externa en una sombra interna, lo cual provee un efecto de profundidad al elemento afectado.

```
box-shadow: rgb(150,150,150) 5px 5px 10px inset;
```

Mostrará una sombra interna alejada del borde de la caja por unos 5 pixeles y con un efecto de difuminación de 10 pixeles.

IMPORTANTE: Las sombras no expanden el elemento o incrementan su tamaño, por lo que tendrá que controlar cuidadosamente que el espacio disponible es suficiente para que la sombra sea expuesta y correctamente dibujada en la pantalla.

3. Text-shadow

La propiedad box-shadow fue diseñada especialmente para ser aplicada en cajas. Si intenta aplicar este efecto a un elemento , por ejemplo, la caja invisible ocupada por este elemento en la pantalla tendrá una sombra, pero no el contenido del elemento. Para crear sombras para figuras irregulares como textos, existe una propiedad especial llamada **text-shadow**:

```
body {
text-align: center;
}
#principal {
display: block;
width: 500px;
margin: 50px auto;
padding: 15px;
text-align: center;
border: 1px solid #999999;
background: #DDDDDD;
-moz-border-radius: 20px;
-webkit-border-radius: 20px;
border-radius: 20px;
-moz-box-shadow: rgb(150,150,150) 5px 5px 10px;
-webkit-box-shadow: rgb(150,150,150) 5px 5px 10px;
box-shadow: rgb(150,150,150) 5px 5px 10px;
}
#titulo {
```

```
font: bold 36px verdana, sans-serif;
text-shadow: rgb(0,0,150) 3px 3px 5px;
}
```

Ejemplo **nuevocss3_8.css**. Generando una sombra para el título.

Los valores para text-shadow son similares a los usados para box-shadow. Podemos declarar el color de la sombra, la distancia horizontal y vertical de la sombra con respecto al objeto y el radio de difuminación.

En el ejemplo una sombra azul fue aplicada al título de nuestra plantilla con una distancia de 3 pixeles y un radio de difuminación de 5.

4. @font-face

El problema con las fuentes o tipos de letra es tan viejo como la web. Usuarios regulares de la web a menudo tienen un número limitado de fuentes instaladas en sus ordenadores, usualmente estas fuentes son diferentes de un usuario a otro, y la mayoría de las veces muchos usuarios tendrán fuentes que otros no. Por años, los sitios webs sólo pudieron utilizar un limitado grupo de fuentes confiables (un grupo básico que prácticamente todos los usuarios tienen instalados) y así presentar la información en pantalla.

La propiedad @font-face permite a los diseñadores proveer un archivo conteniendo una fuente específica para mostrar sus textos en la página. Ahora podemos incluir cualquier fuente que necesitemos con sólo proveer el archivo adecuado:

```
body {
text-align: center;
}
#principal {
display: block;
width: 500px;
margin: 50px auto;
padding: 15px;
```

```
text-align: center;
border: 1px solid #999999;
background: #DDDDDD;
-moz-border-radius: 20px;
-webkit-border-radius: 20px;
border-radius: 20px;
-moz-box-shadow: rgb(150,150,150) 5px 5px 10px;
-webkit-box-shadow: rgb(150,150,150) 5px 5px 10px;
box-shadow: rgb(150,150,150) 5px 5px 10px;
}
#titulo {
font: bold 36px MiNuevaFuente, verdana, sans-serif;
text-shadow: rgb(0,0,150) 3px 3px 5px;
}
@font-face {
font-family: 'MiNuevaFuente';
src: url('letras/font.ttf');
}
```

Ejemplo **nuevocss3_9.css**.. Nueva fuente para el título.

Puede obtener más fuentes de forma gratuita en www.moorstation.org/typoasis/designers/steffmann/.

IMPORTANTE: El archivo conteniendo la fuente debe encontrarse en el mismo dominio que la página web (o en el mismo ordenador, en este caso). Esta es una restricción de algunos navegadores como Firefox, por ejemplo.

La propiedad @font-face necesita al menos dos estilos para declarar la fuente y cargar el archivo.

- El estilo construido con la propiedad **font-family** especifica el nombre que queremos otorgar a esta fuente en particular,
- y la propiedad **src** indica la URL del archivo con el código correspondiente a esa fuente. Una vez que la fuente es cargada, podemos comenzar a usarla en cualquier elemento del documento simplemente escribiendo su nombre (MiNuevaFuente). En el estilo font en la regla , especificamos que el título será mostrado con la nueva fuente o

las alternativas verdana y sans-serif en caso de que la fuente incorporada no sea cargada apropiadamente.

5. Gradiente lineal

Los gradientes son uno de los efectos más atractivos entre aquellos incorporados en CSS3. Este efecto era prácticamente imposible de implementar usando técnicas anteriores pero ahora es realmente fácil de hacer usando CSS. Una propiedad background con algunos pocos parámetros es suficiente para convertir su documento en una página web con aspecto profesional:

```
body {
text-align: center;
}
#principal {
display: block;
width: 500px;
margin: 50px auto;
padding: 15px;
text-align: center;
border: 1px solid #999999;
background: #DDDDDD;
-moz-border-radius: 20px;
-webkit-border-radius: 20px;
border-radius: 20px;
-moz-box-shadow: rgb(150,150,150) 5px 5px 10px;
-webkit-box-shadow: rgb(150,150,150) 5px 5px 10px;
box-shadow: rgb(150,150,150) 5px 5px 10px;
background: -webkit-linear-gradient(top, #FFFFFF, #006699);
background: -moz-linear-gradient(top, #FFFFFF, #006699);
}
#titulo {
font: bold 36px MiNuevaFuente, verdana, sans-serif;
text-shadow: rgb(0,0,150) 3px 3px 5px;
}
@font-face {
```

```
font-family: 'MiNuevaFuente';
src: url('letras/font.ttf');
}
```

nuevocss3_10.css . Agregando un gradiente de fondo a nuestra caja.

Los gradientes son configurados como fondos, por lo que podemos usar las propiedades **background o background- image** para declararlos. La sintaxis para los valores declarados en estas propiedades es:

linear-gradient(posición inicio, color inicial, color final).

Los atributos de la función linear-gradient() indican el punto de comienzo y los colores usados para crear el gradiente. El primer valor puede ser especificado en pixeles, porcentaje o usando las palabras clave top, bottom, left y right (como hicimos en nuestro ejemplo). El punto de comienzo puede ser reemplazado por un ángulo para declarar una dirección específica del gradiente:

```
background: linear-gradient(30deg, #FFFFFF, #006699);
```

Gradiente con un ángulo de dirección de 30 grados.

También podemos declarar los puntos de terminación para cada color:

```
background: linear-gradient(top, #FFFFFF 50%, #006699 90%);
```

Declarando puntos de terminación.

6. Gradiente radial

La sintaxis estándar para los gradientes radiales solo difiere en unos pocos aspectos con respecto a la anterior. Debemos usar la función radial-gradient() y un nuevo atributo para la forma:

```
background: radial-gradient(center, circle, #FFFFFF 0%, #006699 200%);
```

La posición de comienzo es el origen y puede ser declarada en pixeles, porcentaje o una combinación de las palabras clave center, top, bottom, left y right. Existen dos posibles valores para la forma (circle y ellipse) y la terminación para el color indica el color y la posición donde las transiciones comienzan.

IMPORTANTE: En este momento el efecto de gradientes ha sido implementado por los navegadores en diferentes formas.

7. RGBA

Hasta este momento los colores fueron declarados como sólidos utilizando valores hexadecimales o la función rgb() para decimales. CSS3 ha agregado una nueva función llamada rgba() que simplifica la asignación de colores y transparencias.
La función rgba() tiene cuatro atributos. Los primeros tres son similares a los usados en rgb() y simplemente declaran los valores para los colores rojo, verde y azul en números decimales del 0 al 255. El último, en cambio, corresponde a la nueva capacidad de opacidad. Este valor se debe encontrar dentro de un rango que va de 0 a 1, con 0 como totalmente transparente y 1 como totalmente opaco.

```
#titulo {
font: bold 36px MiNuevaFuente, verdana, sans-serif;
text-shadow: rgba(0,0,0,0.5) 3px 3px 5px;
}
```

nuevocss3_12.css Mejora la sombra del texto con transparencia.

Reemplazamos la función rgb() por rgba() en la sombra del título y agregamos un valor de opacidad/transparencia de 0.5. Ahora la sombra de nuestro título se mezclará con el fondo, creando un efecto mucho más natural.
En previas versiones de CSS teníamos que usar diferentes técnicas en diferentes navegadores para hacer un elemento transparente. Todas presentaban el mismo problema: el valor de opacidad de un elemento era heredado por sus hijos. Ese problema fue resuelto por rgba() y ahora podemos asignar un valor de opacidad al fondo de una caja sin afectar su contenido.

8. HSLA

Del mismo modo que la función rgba() agrega un valor de opacidad a rgb(), la función hsla() hace lo mismo para la función hsl().
La función hsla() es simplemente un función diferente para generar colores, pero es más intuitiva que rgba(). Algunos diseñadores encontrarán más fácil generar un set de colores personalizado utilizando hsla(). La sintaxis de esta función es: **hsla(tono, saturación, luminosidad, opacidad)**.

```
#titulo {
font: bold 36px MiNuevaFuente, verdana, sans-serif;
text-shadow: rgba(0,0,0,0.5) 3px 3px 5px;
color: hsla(120, 100%, 50%, 0.5);
}
```

nuevocss3_13.css. Nuevo color para el título usando hsla().

Siguiendo la sintaxis, tono representa el color extraído de una rueda imaginaria y es expresado en grados desde 0 a 360. Cerca de 0 y 360 están los colores rojos, cerca de 120 los verdes y cerca de 240 los azules. El valor saturación es representado en porcentaje, desde 0% (escala de grises) a 100% (todo color o completamente saturado). La luminosidad es también un valor en porcentaje desde 0% (completamente oscuro) a 100% (completamente iluminado). El valor 50% representa luminosidad normal o promedio. El último valor, así como en rgba(), representa la opacidad.

9. Outline

La propiedad outline es una vieja propiedad CSS que ha sido expandida en CSS3 para incluir un valor de desplazamiento.
Esta propiedad era usada para crear un segundo borde, y ahora ese borde puede ser mostrado alejado del borde real del elemento.

```
#principal {
display: block;
width: 500px;
margin: 50px auto;
padding: 15px;
text-align: center;
border: 1px solid #999999;
background: #DDDDDD;
outline: 2px dashed #000099;
outline-offset: 15px;
}
```

nuevocss3_14.css. Agregando un segundo borde a la cabecera.
Agregamos a los estilos originalmente aplicados a la caja de nuestra plantilla un segundo borde de 2 pixeles con un desplazamiento de 15 pixeles. La propiedad outline tiene similares características y usa los mismos parámetros que border. La propiedad outline-offset solo necesita un valor en pixeles.

10. Border-image

Los posibles efectos logrados por las propiedades border y outline están limitados a líneas simples y solo algunas opciones de configuración. La nueva propiedad border-image fue incorporada para superar estas limitaciones y dejar en manos del diseñador la calidad y variedad de bordes disponibles ofreciendo la alternativa de utilizar imágenes propias.
Vamos a utilizar una imagen PNG que incluye diamantes para probar esta propiedad.

La propiedad border-image toma una imagen y la utiliza como patrón. De acuerdo a los valores otorgados, la imagen es cortada como un pastel, las partes obtenidas son luego ubicadas alrededor del objeto para construir el borde.

Cada pieza es de 29 pixeles de ancho, como indica la figura. Para hacer el trabajo, necesitamos especificar tres atributos: el nombre del archivo de la imagen, el tamaño de las piezas que queremos obtener del patrón y algunas palabras clave para declarar cómo las piezas serán distribuidas alrededor del objeto.

```
#principal {
display: block;
width: 500px;
margin: 50px auto;
padding: 15px;
text-align: center;
border: 29px;
-moz-border-image: url("diamonds.png") 29 stretch;
-webkit-border-image: url("diamonds.png") 29 stretch;
border-image: url("diamonds.png") 29 stretch;
}
```

nuevocss3_15.css. Un borde personalizado para la cabecera.

Estamos definiendo un borde de 29 pixeles para la caja de nuestra cabecera y luego cargando la imagen diamonds.png para construir ese borde. El valor 29 en la propiedad border-image declara el tamaño de las piezas y stretch es uno de los métodos disponibles para distribuir estas piezas alrededor de la caja.
Existen tres valores posibles para el último atributo. La palabra clave repeat repetirá las piezas tomadas de la imagen todas las veces que sea necesario para cubrir el lado del elemento. En este caso, el tamaño de las piezas es preservado y la imagen será cortada si no existe más espacio para ubicarla. La palabra clave round considerará qué tan largo es el lado a ser cubierto y ajustará el tamaño de las piezas para asegurarse que cubren todo el lado y ninguna pieza es cortada. Finalmente, la palabra clave stretch estira sólo una pieza para cubrir el lado completo.

En nuestro ejemplo utilizamos la propiedad border para definir el tamaño del borde, pero se puede también usar border-with para especificar diferentes tamaños para cada lado del elemento (la propiedad border-with usa cuatro parámetros, con una sintaxis similar a margin y padding). Lo mismo ocurre con el tamaño de cada pieza, hasta cuatro valores pueden ser declarados para obtener diferentes imágenes de diferentes tamaños desde el patrón.

11. Transform y transition

Los elementos HTML, cuando son creados, son como bloques sólidos e inamovibles. Pueden ser movidos usando código Javascript o aprovechando librerías populares como jQuery (www.jquery.com), por ejemplo, pero no existía un procedimiento estándar para este propósito hasta que CSS3 presentó las propiedades transform y transition.

La propiedad transform puede operar cuatro transformaciones básicas en un elemento: scale (escalar), rotate (rotar), skew (inclinar) y translate (trasladar o mover). Veamos cómo funcionan:

11.1 Transform: scale

```
#principal {
display: block;
width: 500px;
margin: 50px auto;
padding: 15px;
text-align: center;
border: 1px solid #999999;
background: #DDDDDD;
-moz-transform: scale(2);
-webkit-transform: scale(2);
}
```

nuevocss3_16.css. Cambiando la escala de la caja de la cabecera.

En el ejemplo partimos de los estilos básicos utilizados para la cabecera y aplicamos transformación duplicando la escala del elemento. La función scale recibe dos parámetros: el valor X para la escala horizontal y el valor Y para la escala vertical. Si solo un valor es provisto el mismo valor es aplicado a ambos parámetros.
Números enteros y decimales pueden ser declarados para la escala. Esta escala es calculada por medio de una matriz.
Los valores entre 0 y 1 reducirán el elemento, un valor de 1 mantendrá las proporciones originales y valores mayores que 1 aumentarán las dimensiones del elemento de manera incremental.
Un efecto atractivo puede ser logrado con esta función otorgando valores negativos:

```
#principal {
display: block;
width: 500px;
margin: 50px auto;
padding: 15px;
text-align: center;
border: 1px solid #999999;
background: #DDDDDD;
-moz-transform: scale(1,-1);
-webkit-transform: scale(1,-1);
}
```

nuevocss3_17.css. Creando una imagen espejo con scale.

En el ejemplo, dos parámetros han sido declarados para cambiar la escala de la caja principal. El primer valor, 1, mantiene la proporción original para la dimensión horizontal de la caja. El segundo valor también mantiene la proporción original, pero invierte el elemento verticalmente para producir el efecto espejo.
Existen también otras dos funciones similares a scale pero restringidas a la dimensión horizontal o vertical: scaleX y scaleY. Estas funciones, por supuesto, utilizan un solo parámetro.

11.2 Transform: rotate

La función rotate rota el elemento en la dirección de las agujas de un reloj. El valor debe ser especificado en grados usando la unidad “deg”:

```
#principal {
display: block;
width: 500px;
margin: 50px auto;
padding: 15px;
text-align: center;
border: 1px solid #999999;
background: #DDDDDD;
-moz-transform: rotate(30deg);
-webkit-transform: rotate(30deg);
}
```

nuevocss3_18.css. Rotando la caja.
Si un valor negativo es declarado, solo cambiará la dirección en la cual el elemento es rotado.

11.3 Transform: skew

Esta función cambia la simetría del elemento en grados y en ambas dimensiones.

```
#principal {
display: block;
width: 500px;
margin: 50px auto;
padding: 15px;
text-align: center;
border: 1px solid #999999;
background: #DDDDDD;
-moz-transform: skew(20deg);
```

```
-webkit-transform: skew(20deg);
}
```

nuevocss3_19.css. Inclinar horizontalmente.
La función skew usa dos parámetros, pero a diferencia de otras funciones, cada parámetro de esta función solo afecta una dimensión (los parámetros actúan de forma independiente). En el ejemplo, realizamos una operación transforma la caja de la cabecera para inclinarla. Solo declaramos el primer parámetro, por lo que solo la dimensión horizontal de la caja será modificada. Si usáramos los dos parámetros, podríamos alterar ambas dimensiones del objeto. Como alternativa podemos utilizar funciones diferentes para cada una de ellas: skewX y skewY.

11.4 Transform: translate

Similar a las viejas propiedades top y left, la función translate mueve o desplaza el elemento en la pantalla a una nueva posición.

```
#principal {
display: block;
width: 500px;
margin: 50px auto;
padding: 15px;
text-align: center;
border: 1px solid #999999;
background: #DDDDDD;
-moz-transform: translate(100px);
-webkit-transform: translate(100px);
}
```

nuevocss3_20.css. Moviendo la caja de la cab ecera hacia la derecha.
La función translate considera la pantalla como una grilla de pixeles, con la posición original del elemento usada como un punto de referencia. La esquina superior izquierda del elemento es la posición 0,0, por lo que valores negativos moverán al objeto hacia la izquierda o hacia arriba de la posición original, y valores positivos lo harán hacia la derecha o hacia abajo.

En el ejemplo, movimos la caja de la cabecera hacia la derecha unos 100 pixeles desde su posición original. Dos valores pueden ser declarados en esta función si queremos mover el elemento horizontal y verticalmente, o podemos usar funciones independientes llamadas translateX y translateY.

11.5 Transformando todo al mismo tiempo

A veces podría resultar útil realizar sobre un elemento varias transformaciones al mismo tiempo. Para obtener una propiedad transform combinada, solo tenemos que separar cada función a aplicar con un espacio:

```
#principal {
display: block;
width: 500px;
margin: 50px auto;
padding: 15px;
text-align: center;
border: 1px solid #999999;
background: #DDDDDD;
-moz-transform: translateY(100px) rotate(45deg) scaleX(0.3);
-webkit-transform: translateY(100px) rotate(45deg) scaleX(0.3);
}
```

nuevocss3_21.css. Moviendo, escalando y rotando el elemento con solo una línea de código.
Una de las cosas que debe recordar en este caso es que el orden es importante. Esto es debido a que algunas funciones mueven el punto original y el centro del objeto, cambiando de este modo los parámetros que el resto de las funciones utilizarán para operar.

11.6 Transformaciones dinámicas

Lo que hemos aprendido hasta el momento en este capítulo cambiará la forma de la web, pero la mantendrá tan estática como siempre. Sin embargo, podemos aprovecharnos de la combinación de transformaciones y pseudo clases para convertir nuestra página en una aplicación dinámica:

```
#principal {
display: block;
width: 500px;
margin: 50px auto;
padding: 15px;
text-align: center;
border: 1px solid #999999;
background: #DDDDDD;
}
#principal:hover{
-moz-transform: rotate(5deg);
-webkit-transform: rotate(5deg);
}
```

nuevocss3_22.css.. Respondiendo a la actividad del usuario.
En el ejemplo, la regla original para la caja de la cabecera fue conservada intacta, pero una nueva regla fue agregada para aplicar efectos de transformación usando la vieja pseudo clase :hover. El resultado obtenido es que cada vez que el puntero del ratón pasa sobre esta caja, la propiedad transform rota la caja en 5 grados, y cuando el puntero se aleja la caja vuelve a rotar de regreso a su posición original. Este efecto produce una animación básica pero útil con nada más que propiedades CSS.

12. Transiciones

La propiedad transition fue incluida para suavizar los cambios y generar una transición suave desde un estado al otro.

```
#principal {
display: block;
width: 500px;
margin: 50px auto;
padding: 15px;
text-align: center;
border: 1px solid #999999;
```

```
background: #DDDDDD;
-moz-transition: -moz-transform 1s ease-in-out 0.5s;
-webkit-transition: -webkit-transform 1s ease-in-out 0.5s;
}
#principal:hover{
-moz-transform: rotate(5deg);
-webkit-transform: rotate(5deg);
}
```

nuevocss3_23.css. Una rotación usando transiciones.

Como puede ver en el ejemplo, la propiedad transition puede tomar hasta cuatro parámetros separados por un espacio. El primer valor es la propiedad que será considerada para hacer la transición (en nuestro ejemplo elegimos transform). Esto es necesario debido a que varias propiedades pueden cambiar al mismo tiempo y probablemente necesitemos crear los pasos del proceso de transición solo para una de ellas. El segundo parámetro especifica el tiempo que la transición se tomará para ir de la posición inicial a la final. El tercer parámetro puede ser cualquiera de las siguientes palabras clave: ease, linear, ease-in, ease-out o ease-in-out. Estas palabras clave determinan cómo se realizará el proceso de transición basado en una curva Bézier. Cada una de ellas representa diferentes tipos de curva Bézier, y la mejor forma de saber cómo trabajan es viéndolas funcionar en pantalla. El último parámetro para la propiedad transition es el retardo. Éste indica cuánto tiempo tardará la transición en comenzar.
Para producir una transición para todas las propiedades que están cambiando en un objeto, la palabra clave all debe ser especificada. También podemos declarar varias propiedades a la vez listándolas separadas por coma.