Especialidad: Técnico en Programación

Módulo III. Submódulo1: Construye páginas Web
Prof: Hilda Lucía Rodríguez Gómez
Competencia Profesional: Utiliza CSS para dar formato a páginas web

Ejercicios con propiedades de CSS3

1.- Crear el siguiente documento HTML que solo contiene algunas etiquetas básicas, nombrarlo como: **plantilla.html**

```
<!DOCTYPE html>
<html lang="es">
<head>
  <title>Nuevos Estilos CSS3</title>
  <link rel="stylesheet" href="nuevocss3.css">
</head>
<body>
<header id="principal">
  <span id="titulo">Estilos CSS Web 2.0</span>
</header>
</body>
</html>
```

Listado 3-1. Una plantilla simple para probar nuevas propiedades.

Nuestro documento solo tiene una sección con un texto breve en su interior. El elemento <header> usado en la plantilla podría ser reemplazado por , <nav>,<section> o cualquier otro elemento estructural de acuerdo a la ubicación en el diseño y a su función. Luego de aplicar los estilos, la caja generada con el código del ejemplo del Listado 3-1 lucirá como una cabecera, por consiguiente, decidimos usar <header> en este caso.

Debido a que en HTML5, los elementos usados para mostrar texto son normalmente para líneas cortas y <p> para párrafos, el texto en nuestra plantilla fue insertado usando etiquetas .

2.- Crear otro archivo, dentro del cual se estarán probando algunas propiedades de CSS3, llamarlo: **nuevocss3.css** y copiarle el siguiente código:

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

Listado 3-2. Reglas básicas CSS con las que comenzar.

No hay nada nuevo en las reglas del Listado 3-2, solo los estilos necesarios para dar forma a la plantilla y crear una caja ancha, posicionada en el centro de la ventana, con un fondo gris, un borde y un texto grande en su interior que dice “Estilos CSS Web 2.0”.

Una de las cosas que notará sobre esta caja cuando sea mostrada en pantalla es que sus esquinas son rectas. Esto no es algo que nos agrade, ¿verdad? Puede ser un factor psicológico o no, lo cierto es que a casi nadie en este negocio le agradan las esquinas rectas. Por lo tanto, lo primero que haremos será cambiar este aspecto.

3.- A partir de este momento, ir probando las propiedades que siguen, modificando el archivo de css, según los ejemplos dados. Dejar todo el código que se vaya poniendo, y encerrarlo entre líneas de comentario: */* código ya no utilizado */* para aquellas líneas que ya no se utilicen.

Border-radius

Por muchos años diseñadores han sufrido intentando lograr el efecto de esquinas redondeadas en las cajas de sus páginas web. El proceso era casi siempre frustrante y extenuante. Todos lo padecieron alguna vez. Si mira cualquier presentación en video de las nuevas características incorporadas en HTML5, cada vez que alguien habla sobre las propiedades de CSS3 que hacen posible generar fácilmente esquinas redondeadas, la audiencia enloquece. Esquinas redondeadas eran esa clase de cosas que nos hacían pensar: "debería ser fácil hacerlo". Sin embargo, nunca lo fue. Esta es la razón por la que, exploraremos en primera instancia la propiedad: **borderradius**:

```
  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;
}
```

Agregar estas líneas

Listado 3-3. *Generando esquinas redondeadas.*

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:

```
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;
}
```

Listado 3-4. *Diferentes valores para cada esquina.*

Como puede ver en el Listado 3-4, los cuatro valores asignados a la propiedad **border-radius** representan diferentes ubicaciones. 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;
}
```

***Listado 3-5.** Esquinas elípticas.*

Box-shadow

Ahora que finalmente contamos con la posibilidad de generar bonitas esquinas para nuestras cajas podemos arriesgarnos con algo más. Otro muy buen efecto, que había sido extremadamente complicado de lograr hasta este momento, es sombras. Por años diseñadores han combinado imágenes, elementos y algunas propiedades CSS para generar sombras. Gracias a CSS3 y a la nueva propiedad **box-shadow** podremos aplicar sombras a nuestras cajas con solo 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;
}
```

***Listado 3-6.** Aplicando sombra a nuestra caja.*

La propiedad **box-shadow** necesita al menos tres valores. El primero, que puede ver en la regla del Listado 3-6, es el color. Este valor fue construido aquí 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 a la regla del Listado 3-6, 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.

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. Con la siguiente línea, se mostrará una sombra interna alejada del borde de la caja por unos 5 pixeles y con un efecto de difuminación de 10 pixeles.

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

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.

Text-shadow

Ahora que conoce todo acerca de sombras probablemente estará pensando en generar una para cada elemento de su documento. 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**, cuyos valores 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.

```
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;
}
```

Listado 3-9. *Generando una sombra para el título.*

En el Listado 3-9 una sombra azul fue aplicada al título de nuestra plantilla con una distancia de 3 pixeles y un radio de difuminación de 5.

@font-face

Obtener un texto con sombra es realmente un muy buen truco de diseño, imposible de lograr con métodos previos, pero más que cambiar el texto en sí mismo solo provee un efecto tridimensional. Una sombra, en este caso, es como pintar un viejo coche, al final será el mismo coche. En este caso, será el mismo tipo de letra.

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 solo 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 solo proveer el archivo adecuado.

Utilice el archivo **font.ttf proporcionado,** o use uno que ya posea y cópielo en el mismo directorio (carpeta) donde está trabajando. Puede obtener más fuentes similares de forma gratuita en: *www.moorstation.org/typoasis/designers/steffmann/*.

La propiedad **@font-face** necesita al menos dos estilos para declarar la fuente y cargar el archivo, (en el ejemplo utilizamos verdana y sans-serif a parte de nuestra nueva fuente).

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. En el Listado

3-10, el nombre **MiNuevaFuente** fue asignado a nuestro nuevo tipo de letra y el archivo **font.ttf** fue indicado como el archivo correspondiente a esta fuente.

```
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('font.ttf');
}
```

Listado 3-10. Nueva fuente para el título.

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 del Listado 3-10, 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.

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.

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);

También podemos declarar los puntos de terminación para cada color:

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

```
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('font.ttf');
}
```

***Listado 3-11**. Agregando un hermoso gradiente de fondo a nuestra caja.*

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.

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. Esta función además resuelve un problema previo provocado por la propiedad **opacity**.

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;
}
```

***Listado 3-15**. Mejorando la sombra del texto con transparencia.*

El Listado 3-15 ofrece un simple ejemplo que demuestra cómo los efectos son mejorados aplicando 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.

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);
}
```

Listado 3-16. 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.

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;
}
```

Listado 3-17. Agregando un segundo borde a la cabecera.

En el Listado 3-17 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.

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**.

Ahora ya no tenemos que pensar en cómo hacerlo. En su lugar, solo tenemos que conocer cómo ajustar unos pocos parámetros y nuestro sitio web puede ser tan flexible y dinámico como lo imaginamos. 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:

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);
}
```

***Listado 3-19.** Cambiando la escala de la caja de la cabecera.*

En el ejemplo del Listado 3-19 partimos de los estilos básicos utilizados para la cabecera generada en el Listado 3-2 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);
}
```

***Listado 3-20.** Creando una imagen espejo con `scale`.*

En el Listado 3-20, 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.

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**". Si un valor negativo es declarado, solo cambiará la dirección en la cual el elemento es rotado.

```
#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);
}
```

***Listado 3-21**. Rotando la caja.*

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);
}
```

***Listado 3-22**. 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 Listado 3-22, realizamos una operación **transform** a 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**.

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);
}
```

***Listado 3-23**. Moviendo la caja de la cabecera 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 Listado 3-23, 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**.

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);
}
```

***Listado 3-24.** 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.

Transformaciones dinámicas

Podemos aprovecharnos de la combinación de transformaciones y pseudo clases de la siguiente manera:

```
#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);
}
```

***Listado 3-25.** Respondiendo a la actividad del usuario.*

En el Listado 3-25, la regla original del Listado 3-2 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.

Transiciones

De ahora en más, hermosos efectos usando transformaciones dinámicas son accesibles y fáciles de implementar. Sin embargo, una animación real requiere de un proceso de más de dos pasos.

La propiedad **transition** fue incluida para suavizar los cambios, creando mágicamente el resto de los pasos que se encuentran implícitos en el movimiento. Solo agregando esta propiedad forzamos al navegador a tomar cartas en el asunto, crear para nosotros todos esos pasos invisibles, 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);
}
```

***Listado 3-26.** Una hermosa rotación usando transiciones.*

Como puede ver en el Listado 3-26, 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.