

Tema 7. Nuevas opciones con CSS3

Nuevas opciones con CSS3	1
Introducción.....	1
Transformaciones 2D en CSS3.....	2
Movimiento de traslación	4
Movimiento de escalado	5
Movimiento de rotación.....	6
Movimiento de inclinación.....	7
Cambiando el origen de la transformación.....	8
Transformaciones 3D en CSS3.....	9
Definiendo la perspectiva para las transformaciones 3D	11
Movimiento de traslación 3D	13
Movimiento de rotación 3D.....	14
Cambiando el estilo de la transformación	15
Cambiando la forma de ver la parte trasera de nuestros elementos.....	17
Animaciones en CSS3	20
Ejemplo de animación	23
Transiciones en CSS3	25
Ejemplo de transición	28
Ejercicios	32
Ejercicio 1: Crear una galería de imágenes animada.....	32
Pasos a seguir.....	33
Ejercicio 2: Crear un menú desplegable animado	34
Pasos a seguir.....	34

Nuevas opciones con CSS3

Introducción

Todos sabemos que CSS es el fiel compañero de HTML a la hora de crear páginas Web. Digamos que CSS (*Cascading Style Sheets* u *Hojas de Estilo en Cascada*) es el encargado de dar formato y color a nuestra página. Tanto es así, que a partir de HTML5, las etiquetas de HTML que tenían relación directa con el formato de la página o los elementos de ella, dejan de estar en

funcionamiento, o mejor dicho se desaconseja su uso, para fijar sobre CSS todos los aspectos relacionados con el formato.

La última versión de la que disponemos es CSS3, y al igual que HTML5, tiene grandes novedades respecto a sus anteriores versiones. El avance en las tecnologías Web y la necesidad de crear contenidos animados e interactivos tienen mucho que ver en esto. Aparte de mejorar y posibilitar la creación de páginas con formatos mucho más espectaculares mediante sombras, gradientes de color, degradados o redondeo de elementos, CSS3 presenta la parte más novedosa en la **posibilidad de crear animaciones y transformaciones** sobre los elementos de nuestra página, y es este apartado el que vamos a tratar en esta unidad.

Hasta ahora, las animaciones en las páginas web siempre se tenían que realizar utilizando tecnologías más allá de HTML o CS, la tecnología *Flash* era quien se llevaba la palma en este aspecto. Aunque han ido apareciendo otras de características similares, no han conseguido arrebatarle el protagonismo a esta última. Todas ellas son tecnologías propietarias, que necesitan de la instalación de un plugin para funcionar en el navegador, lo que impide que sean universales, por mucha aceptación que hayan tenido. Esto, unido a que algunos de los principales desarrolladores se han negado a utilizar esta tecnología *Flash*, ha llevado a buscar otras alternativas.

La API de Canvas en HTML5 junto a JavaScript, ofrece infinitas posibilidades para crear todo tipo de animaciones, sin embargo, CSS3 ofrece una nueva forma de realizar animaciones totalmente novedosa y que resultará más sencilla de utilizar. Soporta transformaciones como rotaciones, ampliaciones y reducciones del tamaño, inclinaciones o perspectivas, y además dispone de ciertas interacciones con el usuario y que se consiguen únicamente con CSS3. Todo ello sin tener que programar, lo que puede resultar mucho más agradable y al alcance de los desarrolladores menos experimentados.

Transformaciones 2D en CSS3

Vamos a ver primero las propiedades que CSS3 tiene disponibles para crear alguna de las transformaciones 2D que veremos a continuación.

Anotación

Las propiedades de transformación que vamos a ver a continuación, no están todavía estandarizadas totalmente, sin embargo, cada navegador o grupo de navegadores dispone ya de su propia librería para hacerlas funcionar a su manera. Por lo tanto, no podemos utilizar la propiedad estandarizada para que funcione en todos los principales navegadores, sino que deberemos utilizar las propiedades que cada uno de ellos dispone. De manera que para hacerlas funcionar en todos los navegadores en las que están disponibles, deberemos añadir delante de la propiedad la extensión para cada grupo de navegadores de la siguiente manera:

```
transform:funcion(); Propiedad estandarizada
-ms-transform:funcion(); Propiedad para IE
-moz-transform:funcion(); Propiedad para mozilla
-webkit-transform:funcion(); Propiedad para Chrome y Safari
-o-transform:funcion(); Propiedad para Opera
```

De esta manera la propiedad *transform*, funcionaría en todos los principales navegadores.

Las propiedades son las siguientes:

transform: Aplica una función de transformación espacial a un objeto de nuestra página.

transform-origin: Define el origen de la transformación.

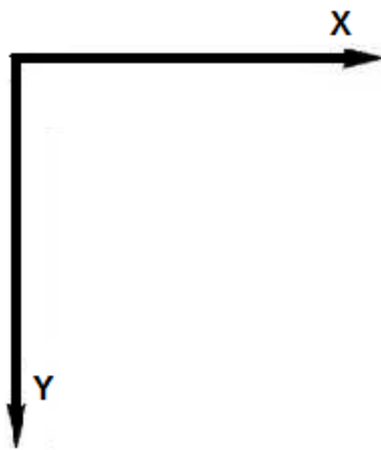
Una vez conocemos las propiedades de que disponemos para crear transformaciones 2D mediante CSS3, vamos a ver cuáles son las funciones de transformación que podemos aplicar a los elementos de nuestra página, y como funcionan.

¡IMPORTANTE!

Estas propiedades están en funcionamiento en los siguientes navegadores:



Vamos a ver las funciones de transformación disponible sobre el plano, es decir, en dos dimensiones. El eje de coordenadas según miramos a la pantalla sería el siguiente:



Las funciones son las siguientes:

matrix(matriz de 3x2, 6 valores): Define la matriz de transformación de 2D.

translate(x, y): Define un movimiento de traslación de x píxeles a la derecha e y píxeles hacia abajo.

translateX(x): Define un movimiento de traslación de x píxeles a la derecha.

translateY(y): Define un movimiento de traslación de y píxeles hacia abajo.

scale(x, y): Define un escalado multiplicando el tamaño por el factor x en horizontal y por el factor y en vertical.

scaleX(x): Define un escalado multiplicando el tamaño por el factor x en horizontal.

scaleY(y): Define un escalado multiplicando el tamaño por el factor y en vertical.

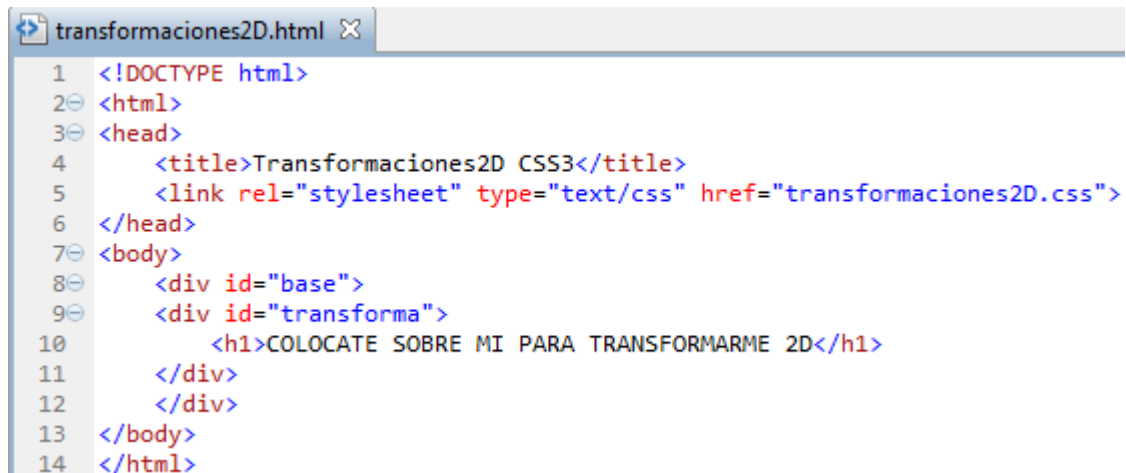
rotate(angle): Define una rotación del ángulo definido en la dirección de las agujas del reloj.

skew(angleX, angleY): Define una inclinación sobre el eje horizontal y otra sobre el eje vertical con los ángulos definidos en la dirección de las agujas del reloj.

skewX(angle): Define una inclinación sobre el eje horizontal con el ángulo definido en la dirección de las agujas del reloj.

skewY(angle): Define una inclinación sobre el eje vertical con el ángulo definido en la dirección de las agujas del reloj.

Vamos a ver en los siguientes ejemplos el funcionamiento de cada una de las transformaciones disponibles. En todos ellos, disponemos de un elemento *DIV* dentro de otro, uno nos servirá de referencia y el único formato que tiene es un borde negro, y el otro será el que transformemos al colocarnos sobre él. Por lo tanto, en todos los casos el código HTML utilizado va a ser el siguiente:



```
1 <!DOCTYPE html>
2 <html>
3 <head>
4   <title>Transformaciones2D CSS3</title>
5   <link rel="stylesheet" type="text/css" href="transformaciones2D.css">
6 </head>
7 <body>
8   <div id="base">
9     <div id="transforma">
10      <h1>COLOCATE SOBRE MI PARA TRANSFORMARME 2D</h1>
11    </div>
12  </div>
13 </body>
14 </html>
```

El *DIV* con identificador "base", será el que nos sirva como referencia para ver mejor la transformación que apliquemos sobre el *DIV* con identificador "transforma". Todo el contenido CSS lo vamos a tener en un fichero aparte, que como vemos está referenciado en el cabecero de la página. Por lo tanto, vamos a ir cambiando en el fichero CSS la transformación que le hacemos al *DIV* y viendo el resultado.

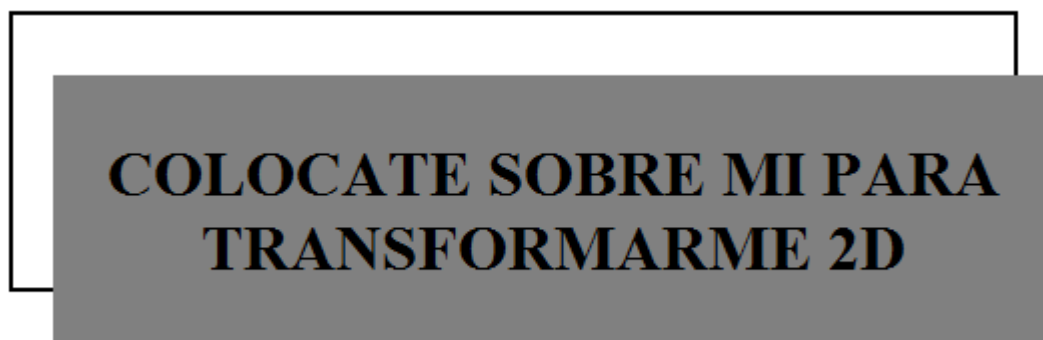
Movimiento de traslación

Vamos a empezar con un movimiento de **traslación**, el código completo del fichero CSS será el siguiente:

```
transformaciones2D.css
1 body{
2     width:900px;
3     margin:auto;
4 }
5
6 #base{
7     width:500px;
8     margin:10px auto;
9     border:2px solid black;
10 }
11 #transforma{
12     width:480px;
13     padding:10px;
14     background:grey;
15     text-align:center;
16 }
17
18 #transforma:hover{
19     transform:translate(20px,30px);
20     -ms-transform:translate(20px,30px);
21     -moz-transform:translate(20px,30px);
22     -webkit-transform:translate(20px,30px);
23     -o-transform:translate(20px,30px);
24 }
```

En primer lugar damos un formato a los elementos de la página, y después, vemos como asignamos una transformación de tipo *translate()* cuando nos colocamos sobre el elemento indicado anteriormente. El movimiento será de 20px en el eje horizontal hacia la derecha, y de 30px en el eje vertical hacia abajo. Aplicamos todas las propiedades necesarias para que funcione en cualquiera de los principales navegadores como hemos dicho al principio.

El resultado cuando estamos sobre el elemento será el siguiente:



Antes de colocarnos sobre él, se encontraba en el mismo lugar que está el elemento con borde de color negro, por lo tanto, se ve claramente el desplazamiento.

La utilización de las funciones *translateX()* y *translateY()* es idéntica a las que hemos visto, solo que generan el movimiento sobre un solo eje.

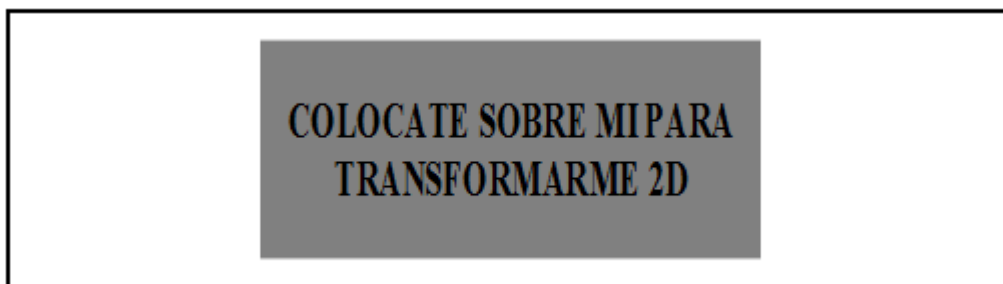
Movimiento de escalado

Vamos a seguir con un movimiento de **escalado**, el código que hemos cambiado del fichero CSS será el siguiente:

```
18 #transforma:hover{
19     transform:scale(0.5,0.8);
20     -ms-transform:scale(0.5,0.8);
21     -moz-transform:scale(0.5,0.8);
22     -webkit-transform:scale(0.5,0.8);
23     -o-transform:scale(0.5,0.8);
24 }
```

Vemos como asignamos una transformación de tipo *scale()* cuando nos colocamos sobre el elemento indicado anteriormente. El tamaño se disminuirá con un factor de 0.5 en anchura, es decir, se reducirá a la mitad, y con un factor de 0.8 en altura. Aplicamos todas las propiedades necesarias para que funcione en cualquiera de los principales navegadores como hemos dicho al principio.

El resultado cuando estamos sobre el elemento será el siguiente:



Antes de colocarnos sobre él, era del mismo tamaño que el elemento con borde de color negro, por lo tanto, se ve claramente el escalado.

La utilización de las funciones *scaleX()* y *scaleY()* es idéntica a las que hemos visto, solo que generan el escalado sobre un solo eje.

Movimiento de rotación

El siguiente será un movimiento de **rotación**, el código que hemos cambiado del fichero CSS será el siguiente:

```
18 #transforma:hover{
19     transform:rotate(30deg);
20     -ms-transform:rotate(30deg);
21     -moz-transform:rotate(30deg);
22     -webkit-transform:rotate(30deg);
23     -o-transform:rotate(30deg);
24 }
```

Vemos como asignamos una transformación de tipo *rotate()* cuando nos colocamos sobre el elemento indicado anteriormente. La rotación será de 30 grados en el sentido de las agujas del reloj. Aplicamos todas las propiedades necesarias para que funcione en cualquiera de los principales navegadores como hemos dicho al principio.

El resultado cuando estamos sobre el elemento será el siguiente:



Antes de colocarnos sobre él, se encontraba en el mismo lugar que está el elemento con borde de color negro, por lo tanto, se ve claramente la rotación de 30 grados en el sentido de las agujas del reloj.

Movimiento de inclinación

Terminamos con un movimiento de **inclinación**, el código que hemos cambiado del fichero CSS será el siguiente:

```
18 #transforma:hover{  
19     transform:skew(10deg,30deg);  
20     -ms-transform:skew(10deg,30deg);  
21     -moz-transform:skew(10deg,30deg);  
22     -webkit-transform:skew(10deg,30deg);  
23     -o-transform:skew(10deg,30deg);  
24 }
```

Vemos como asignamos una transformación de tipo *skew()* cuando nos colocamos sobre el elemento indicado anteriormente. La inclinación será de 10 grados en el eje horizontal, y de 30 grados en el eje vertical. Aplicamos todas las propiedades necesarias para que funcione en cualquiera de los principales navegadores como hemos dicho al principio.

El resultado cuando estamos sobre el elemento será el siguiente:



Antes de colocarnos sobre él, se encontraba en el mismo lugar que está el elemento con borde de color negro, por lo tanto, se ve claramente la inclinación en ambos ejes.

La utilización de las funciones `skewX()` y `skewY()` es idéntica a las que hemos visto, solo que generan la inclinación sobre un solo eje.

Cambiando el origen de la transformación

Si aplicamos la propiedad **transform-origin**, lo que estamos haciendo es cambiar el punto de referencia desde donde aplicamos la transformación, vamos a verlo con el ejemplo del movimiento de **rotación**:

El código que hemos cambiado del fichero CSS será el siguiente:

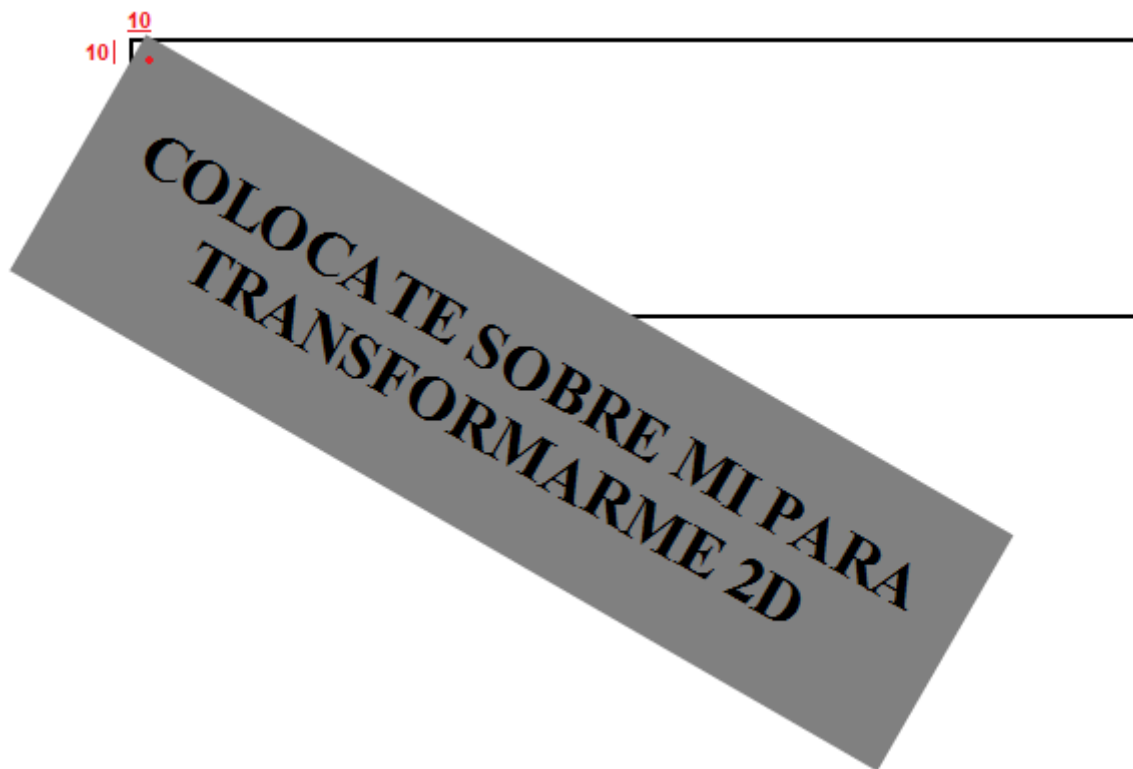

```

18 #transforma:hover{
19     -ms-transform-origin:10px 10px;
20     -moz-transform-origin:10px 10px;
21     -webkit-transform-origin:10px 10px;
22     -o-transform-origin:10px 10px;
23
24     transform:rotate(30deg);
25     -ms-transform:rotate(30deg);
26     -moz-transform:rotate(30deg);
27     -webkit-transform:rotate(30deg);
28     -o-transform:rotate(30deg);
29 }

```

Vemos como ahora, antes de asignar el movimiento de rotación al elemento indicado anteriormente, definimos el eje de rotación 10px a la derecha y 10px hacia abajo desde la esquina superior izquierda de nuestro elemento a rotar. Aplicamos en ambos casos, todas las propiedades necesarias para que funcione en cualquiera de los principales navegadores como hemos dicho al principio.

En resultado cuando estamos sobre el elemento será el siguiente:



Antes de colocarnos sobre él, se encontraba en el mismo lugar que está el elemento con borde de color negro, por lo tanto, se ve claramente la rotación sobre el punto marcado en rojo, y que se encuentra a 10px a la derecha y 10px hacia abajo respecto a la esquina superior izquierda del elemento a rotar. Si no definimos este origen de la transformación, por defecto está en el centro del elemento como hemos visto en ejemplos anteriores.

Transformaciones 3D en CSS3

Vamos a ver primero las propiedades que CSS3 tiene disponibles para crear alguna de las transformaciones 3D que veremos a continuación.

La propiedades son las siguientes:

transform: Aplica una función de transformación espacial a un objeto de nuestra página.

transform-origin: Define el origen de la transformación.

transform-style: Define la manera de mostrar el elemento en el espacio. Puede ser: *flat* y *preserve-3d*.

perspective: Aplica la función de transformación *perspectiva* a un objeto de nuestra página.

perspective-origin: Define el origen de la transformación de perspectiva.

backface-visibility: Define si la cara trasera de un elemento es visible o no.

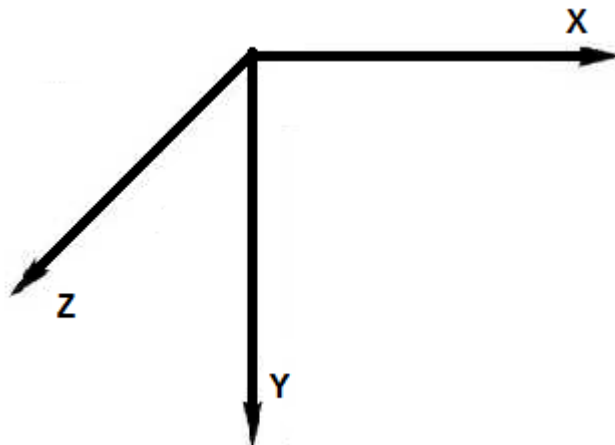
Una vez conocemos las propiedades de que disponemos para crear transformaciones 3D mediante CSS3, vamos a ver cuáles son las funciones de transformación que podemos aplicar a los elementos de nuestra página, y cómo funcionan.

¡IMPORTANTE!

Estas propiedades están en funcionamiento en los siguientes navegadores:



Vamos a ver las funciones de transformación disponible sobre el espacio, es decir, en tres dimensiones. El eje de coordenadas según miramos a la pantalla sería el siguiente:



Las funciones son las siguientes:

matrix3d(matriz de 4x4, 16 valores): Define la matriz de transformación de 3D.

translate3d(x, y, z): Define un movimiento de traslación de x píxeles a la derecha, y píxeles hacia abajo y z píxeles en el eje Z.

translateZ(z): Define un movimiento de traslación de z píxeles en el eje Z.

scale3d(x, y, z): Define un escalado multiplicando el tamaño por el factor x en horizontal, por el factor y en vertical y por el factor z en el eje Z.

scaleZ(z): Define un escalado multiplicando el tamaño por el factor z en el eje Z.

rotate3d(x, y, z, angle): Define una rotación del ángulo definido en la dirección que define el vector (x, y, z).

rotateX(angle): Lo mismo que *rotate3d(1, 0, 0, angle)*. Define una rotación sobre el eje X.

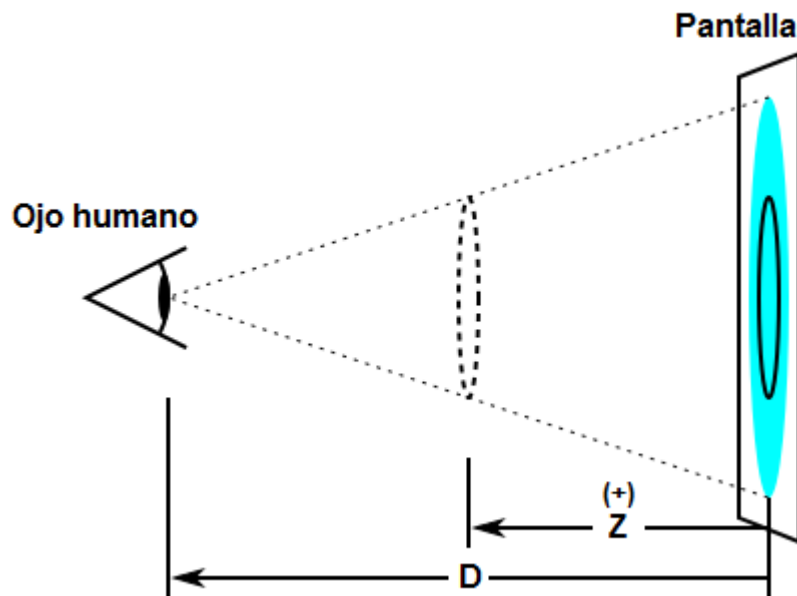
rotateY(angle): Lo mismo que *rotate3d(0, 1, 0, angle)*. Define una rotación sobre el eje Y.

rotateZ(angle): Lo mismo que *rotate3d(0, 0, 1, angle)*, o lo mismo que *rotate(angle)*. Define una rotación sobre el eje Z.

perspective(longitud): Define la intensidad de los efectos 3D aplicados, cuanto mayor es la perspectiva, menor es la intensidad de los efectos aplicados.

Definiendo la perspectiva para las transformaciones 3D

Las transformaciones producidas por estas funciones, son iguales que las que hemos visto en el apartado anterior, pero en este caso introducimos una dimensión más, la dimensión Z. Esta dimensión, es positiva de la pantalla hacia nosotros, y es negativa de la pantalla hacia atrás. Por eso, necesitamos establecer una perspectiva mediante el atributo **perspective**. Vamos a explicar en qué consiste:

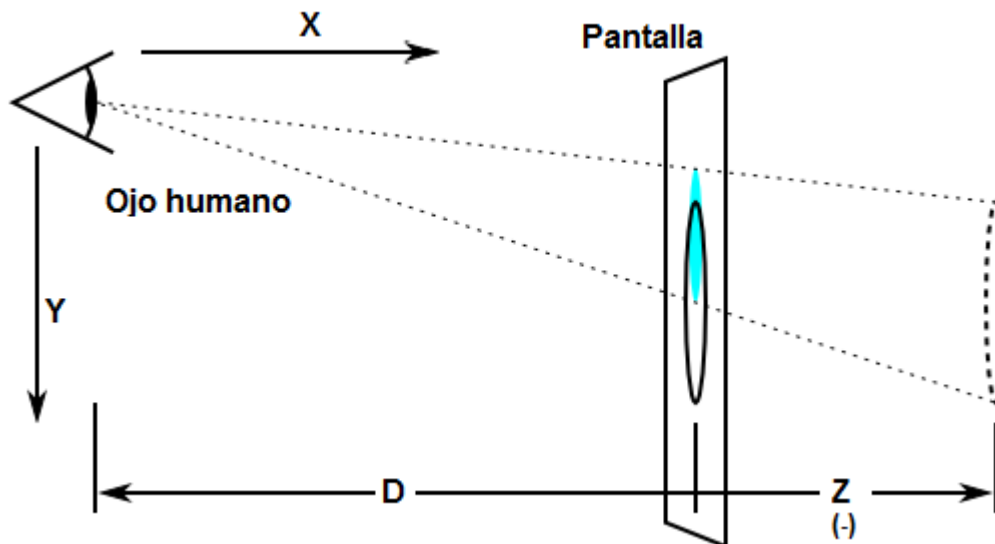


Estando nosotros a una distancia **D** desde la pantalla, y el objeto a una distancia **Z**, el valor del atributo **perspective** será la diferencia entre ambas, es decir, **D-Z**. Por tanto, este valor define de qué manera vamos a apreciar nosotros una transformación en el eje Z, cuanto menor sea ese valor, más lo apreciaremos, y cuanto mayor sea, menos.

Por ejemplo, si nosotros desplazamos un elemento de nuestra página con un valor de 20px sobre el eje Z, y el valor de la perspectiva es 20, apreciaremos muchísimo ese movimiento, ya que la sensación sería la de que el objeto está junto en la mitad entre el ojo y la pantalla, pero si el valor de

la perspectiva es 200, casi ni lo apreciaremos, ya que la sensación sería la de que el objeto está a 180px de nuestro ojo y muy cerquita de la pantalla a solo 20px.

Mediante la propiedad **perspective-origin**, cambiaríamos el posicionamiento de nuestro ojo por así decirlo. Por defecto está justo delante del objeto en línea recta con la pantalla, pero mediante esta propiedad podemos asignar la posición tanto en el eje horizontal como vertical, y la sensación que se presentaría en pantalla sería la de estar mirando a la pantalla desde otro ángulo diferente.



Teniendo en cuenta todo esto, vamos a ver sobre el ejemplo anterior estas transformaciones. Hay que tener en cuenta, que en este caso las probaremos sobre uno de los navegadores del *Webkit*.

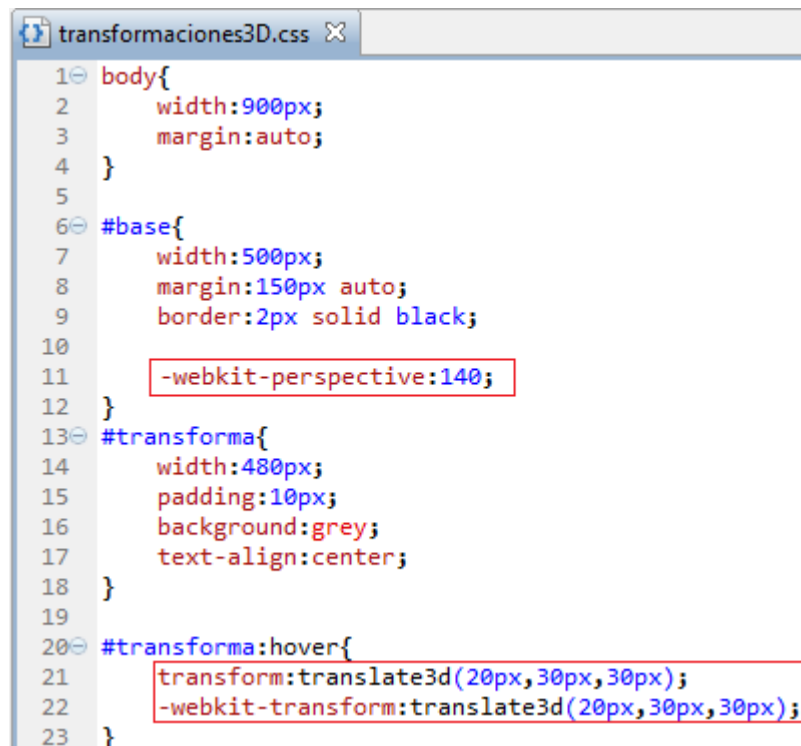
El código HTML a utilizar será prácticamente el mismo que hemos utilizado en el apartado anterior:

```
transformaciones3D.html X
1 <!DOCTYPE html>
2 <html>
3 <head>
4   <title>Transformaciones3D CSS3</title>
5   <link rel="stylesheet" type="text/css" href="transformaciones3D.css">
6 </head>
7 <body>
8   <div id="base">
9     <div id="transforma">
10      <h1>COLOCATE SOBRE MI PARA TRANSFORMARME 3D</h1>
11    </div>
12  </div>
13 </body>
14 </html>
```

El *DIV* con identificador "base", será el que nos sirva como referencia para ver mejor la transformación que apliquemos sobre el *DIV* con identificador "transforma". Todo el contenido CSS lo vamos a tener en un fichero aparte, que como vemos está referenciado en el cabecero de la página. Por lo tanto, vamos a ir cambiando en el fichero CSS la transformación que le hacemos al *DIV* y viendo el resultado.

Movimiento de traslación 3D

Vamos a empezar con un movimiento de **traslación en 3D**, el código completo del fichero CSS será el siguiente:



```
1 body{
2     width:900px;
3     margin:auto;
4 }
5
6 #base{
7     width:500px;
8     margin:150px auto;
9     border:2px solid black;
10
11     -webkit-perspective:140;
12 }
13 #transforma{
14     width:480px;
15     padding:10px;
16     background:grey;
17     text-align:center;
18 }
19
20 #transforma:hover{
21     transform:translate3d(20px,30px,30px);
22     -webkit-transform:translate3d(20px,30px,30px);
23 }
```

En primer lugar damos un formato a los elementos de la página, y después, vemos como asignamos una transformación de tipo *translate3d()* cuando nos colocamos sobre el elemento indicado anteriormente. El movimiento será de 20px en el eje horizontal hacia la derecha, de 30px en el eje vertical hacia abajo y de 30px en el eje Z hacia nosotros. La perspectiva se la aplicamos al *DIV* que dejamos como referencia, y en este caso será de 140.

El resultado cuando estamos sobre el elemento será el siguiente:



**COLOCATE SOBRE MI PARA
TRANSFORMARME 3D**

Vemos claramente que se ha desplazado hacia la derecha y hacia abajo como en el ejemplo de 2D, pero en este caso tenemos también la traslación en el eje Z, que se aprecia ya que al acercar el elemento hacia nosotros lo vemos más grande. Si hiciéramos más grande la propiedad *perspective*, lo veríamos más pequeño, y por tanto apreciaríamos menos ese movimiento hacia nosotros, si dicha propiedad fuese más pequeña, lo veríamos más grande, y apreciaríamos todavía mejor el movimiento.

La utilización de la función *translateZ()* es idéntica a las que hemos visto, solo que genera el movimiento sobre un solo eje, el eje Z.

Movimiento de rotación 3D

Vamos a seguir con un movimiento de **rotación en 3D**, el código que hemos cambiado del fichero CSS será el siguiente:

```
20 #transforma:hover{
21     transform: rotate3d(1,1,0,10deg);
22     -webkit-transform: rotate3d(1,1,0,10deg);
23 }
```

Vemos como asignamos una transformación de tipo *rotate3d()* cuando nos colocamos sobre el elemento indicado anteriormente. La rotación se hará respecto al vector indicado, en este caso es un vector (1,1,0). El giro será de 10 grados sobre ese eje y en el sentido de las agujas del reloj.

El resultado cuando estamos sobre el elemento será el siguiente:



Si colocamos la imagen sobre el eje de coordenadas, vemos como la rotación se ha dado sobre el eje marcado por el vector indicado. Apreciamos también la perspectiva, ya que cuando el elemento se acerca hacia nosotros se amplía.

La utilización de las funciones *rotateX()*, *rotateY()* y *rotateZ()* es idéntica a las que hemos visto, solo que generan la rotación sobre un solo eje.

Anotación

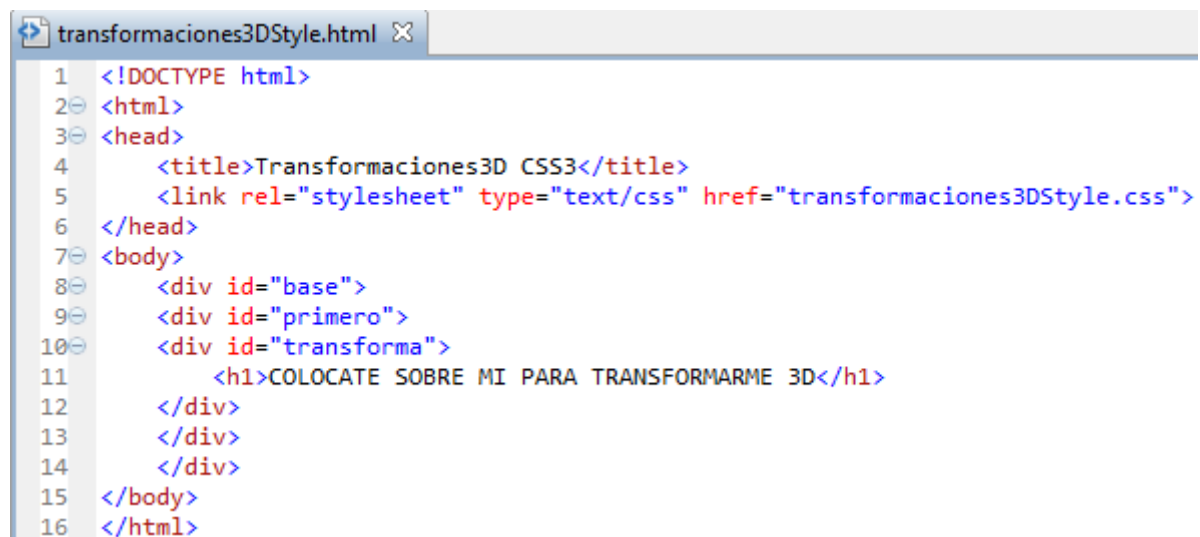
El ejemplo para un movimiento de **escalado en 3D** no lo hemos realizado, ya que es complejo ver un escalado en el eje Z en elementos que definimos en un plano, y el escalado en el plano ya lo hemos visto en el apartado anterior.

La función ***perspective()*** es la función que ejecutamos cuando definimos el valor para la propiedad ***perspective***.

Cambiando el estilo de la transformación

La propiedad ***transform-style*** como hemos dicho al comienzo, acepta dos valores que son: *flat* y *preserve-3d*. Con la primera opción, los elementos secundarios no conservarán su posición en 3D, sin embargo, con la segunda opción si. Hay que tener en cuenta, que en este caso lo probaremos sobre uno de los navegadores del *Webkit*, ya que en los demás no está disponible esta propiedad. Vamos a ver un ejemplo:

El código HTML a utilizar será el siguiente:



```
1 <!DOCTYPE html>
2 <html>
3 <head>
4     <title>Transformaciones3D CSS3</title>
5     <link rel="stylesheet" type="text/css" href="transformaciones3DStyle.css">
6 </head>
7 <body>
8     <div id="base">
9         <div id="primero">
10            <div id="transforma">
11                <h1>COLOCATE SOBRE MI PARA TRANSFORMARME 3D</h1>
12            </div>
13        </div>
14    </div>
15 </body>
16 </html>
```

El *DIV* con identificador "base", será el que nos sirva como referencia para ver mejor la transformación que apliquemos sobre los *DIVs* con identificadores "primero" y "transforma". El primero será el elemento primario sobre el que apliquemos una transformación, y el segundo el elemento secundario sobre el que apliquemos otra transformación. Todo el contenido CSS lo vamos a tener en un fichero aparte, que como vemos está referenciado en el cabecero de la página.

El código completo del fichero CSS será el siguiente:

```
transformaciones3DStyle.css X
1 body{
2     width:900px;
3     margin:auto;
4 }
5
6 #base{
7     width:500px;
8     margin:150px auto;
9     border:2px solid black;
10
11     -webkit-perspective:140;
12 }
13
14 #transforma{
15     width:480px;
16     padding:10px;
17     background:grey;
18     text-align:center;
19 }
20
21 #primero:hover{
22     border:2px solid green;
23     transform:rotate3d(1,1,0,10deg);
24     -webkit-transform:rotate3d(1,1,0,10deg);
25     -webkit-transform-style:preserve-3d;
26 }
27
28 #transforma:hover{
29     transform:rotate3d(0,1,0,5deg);
30     -webkit-transform:rotate3d(0,1,0,5deg);
31 }
```

En primer lugar damos formato a los elementos de nuestra página, y a continuación vamos a aplicar dos movimientos de rotación diferentes al *DIV* primario y al *DIV* secundario al colocarnos sobre ellos. En el primer caso será una rotación de 10 grados sobre el vector (1,1,0), y en el segundo caso, una rotación de 5 grados sobre el eje Y. Vemos como al primero de ellos, le definimos el *transform-style* como **preserve-3d**, por lo tanto en este caso se respetará la rotación del elemento secundario.

El resultado cuando estamos sobre el elemento será el siguiente:



Vemos como el *DIV* con borde verde es el primario, que ha sufrido una rotación de 10 grados sobre el vector indicado anteriormente, y el *DIV* gris sin borde, es el elemento secundario que ha sufrido una rotación de 5 grados sobre el eje Y, desde el punto en el que se encontraba una vez girado el *DIV* principal.

Si cambiamos el valor de *transform-style* por el de **flat**, el resultado cuando estamos sobre el elemento será el siguiente:



Vemos como en este caso la rotación del *DIV* primario si se ha ejecutado correctamente, pero la rotación del *DIV* secundario no se ha producido. Ahí está la diferencia entre ambos estilos.

Cambiando la forma de ver la parte trasera de nuestros elementos

La propiedad **backface-visibility** define si la parte de atrás de nuestros elementos será visible o no. Por defecto será visible, pero si le decimos lo contrario a través de esta propiedad la cosa cambia. Hay que tener en cuenta, que en este caso lo probaremos sobre uno de los navegadores del *Webkit*, ya que en los demás no está disponible esta propiedad. Vamos a ver un ejemplo:

El código HTML a utilizar será el siguiente:

```
transformaciones3DBV.html X
1 <!DOCTYPE html>
2 <html>
3 <head>
4     <title>Transformaciones3D CSS3</title>
5     <link rel="stylesheet" type="text/css" href="transformaciones3DBV.css">
6 </head>
7 <body>
8     <div id="base">
9         <div id="primero">
10            <div id="transforma">
11                <h1>COLOCATE SOBRE MI PARA TRANSFORMARME 3D</h1>
12            </div>
13        </div>
14    </div>
15 </body>
16 </html>
```

Los *DIVs* con identificadores "base" y "primero", serán los que nos sirvan como referencia para ver mejor la transformación que apliquemos sobre el *DIV* con identificador "transforma". Todo el contenido CSS lo vamos a tener en un fichero aparte, que como vemos está referenciado en el encabezado de la página.

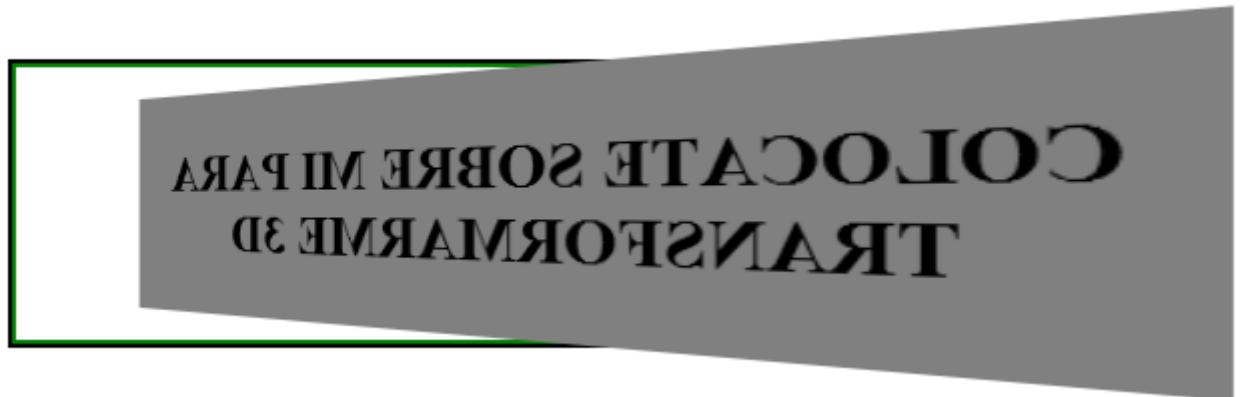
El código completo del fichero CSS será el siguiente:

```
transformaciones3DBV.css X
1 body{
2     width:900px;
3     margin:auto;
4 }
5
6 #base{
7     width:500px;
8     margin:150px auto;
9     border:2px solid black;
10
11     -webkit-perspective:140;
12 }
13
14 #transforma{
15     width:480px;
16     padding:10px;
17     background:grey;
18     text-align:center;
19 }
20
21 #primero:hover{
22     border:2px solid green;
23 }
24
25 #transforma:hover{
26     transform:rotateY(170deg);
27     -webkit-transform:rotateY(170deg);
28     -webkit-backface-visibility:visible;
29 }
```

En primer lugar damos formato a los elementos de nuestra página, y a continuación vamos a aplicar un movimiento de rotación de 170 grados sobre el eje Y al *DIV* con identificador "transforma". De

esta manera podremos ver la parte trasera de dicho elemento. Hemos definido la propiedad *backface-visibility* como **visible**.

El resultado cuando estamos sobre el elemento será el siguiente:



Vemos como el elemento ha sufrido una rotación sobre el eje Y de 160 grados, y por tanto vemos la parte de atrás del *DIV*, puesto que hemos definido que sea visible.

Si cambiamos el valor de *backface-visibility* por el de **hidden**, el resultado cuando estamos sobre el elemento será el siguiente:



Aunque no vemos el elemento, ha sufrido una rotación sobre el eje Y de 160 grados, pero como hemos definido que la parte de atrás no sea visible, no podemos ver nada.

¡IMPORTANTE!

Cuando hacemos una transformación modificamos el elemento, pero no estamos cambiando sus propiedades. Por lo tanto, no tiene ningún efecto sobre los demás elementos de la página.

Por ejemplo, si hacemos un escalado, no estamos cambiando la propiedad de anchura que ese elemento tiene en la página, y por lo tanto no ocupa más espacio en el flujo de la página, por eso los demás elementos permanecerán en su sitio sin moverse. Sin embargo, si modificamos la propiedad de anchura directamente, el elemento ocupa más y por lo tanto los demás elementos serán desplazados, pero eso no sucede en las transformaciones.

¡IMPORTANTE!

Podemos consultar todos los aspectos relacionados con las transformaciones de CSS3 en la recomendación W3C desde este enlace: [**Transformaciones CSS3**](#)

Animaciones en CSS3

Hemos visto ya las transformaciones disponibles en CSS3 y que podemos aplicar sobre los elementos de nuestra página. Pero ahora, necesitamos animar de alguna manera esas transformaciones, y que se vayan ejecutando durante un tiempo determinado de manera automática. Vamos a ver a continuación los pasos a seguir para conseguir estas animaciones mediante el uso de CSS3, y veremos algunos ejemplos para ir familiarizándonos con ellas.

A la hora de crear una animación, lo primero es crear los llamados fotogramas clave mediante la propiedad **keyframes**. En estos fotogramas clave, definiremos la posición o el estado en que van a estar los elementos que participen en la animación en momentos clave. Por ejemplo, sabemos a ciencia cierta dónde y cómo debe de empezar la animación, y dónde y cómo debe de terminar, pero aparte de los fotogramas clave de inicio y fin, también podemos definir fotogramas clave para momentos intermedios de la animación. Hay que aclarar, que una vez terminada una animación el elemento siempre vuelve a su estado inicial.

La forma de definir estos fotogramas clave la podemos ver en el siguiente ejemplo:

```
1  @keyframes movimiento_circular
2  {
3      0% {left: 400px; top: 0px;}
4      25% {left: 800px; top: 200px;}
5      50% {left: 400px; top: 400px;}
6      75% {left: 0px; top: 200px;}
7      100% {left: 400px; top: 0px;}
8  }
```

Definimos en primer lugar el nombre de la animación, que en este caso será "movimiento circular", y después vamos definiendo cada uno de los fotogramas clave tal y como vemos en la imagen. Entre llaves vamos definiendo la posición y las características que tendrá el elemento animado en el momento de la animación indicado por el porcentaje que tiene al lado. Por lo tanto, de alguna manera estamos definiendo varios puntos y situaciones por las que debe de pasar nuestra animación. Luego, el navegador será quien lo interprete y ejecute la animación.

Podemos definir únicamente los fotogramas clave de inicio y fin de la siguiente manera:

```
1  @keyframes movimiento_lineal
2  {
3      from {left: 400px; top: 0px;}
4      to {left: 800px; top: 200px;}
5  }
```

De esta manera solo definimos las situaciones de inicio y fin, sin marcar ningún punto intermedio.

Anotación

La propiedad **keyframes**, al igual que las **propiedades que vamos a ver a continuación**, tampoco están todavía estandarizadas totalmente. De manera que para hacerlas funcionar en todos los principales navegadores, deberemos añadir delante de la propiedad la extensión para cada grupo de navegadores de la siguiente manera:

```
@-webkit-keyframes nombre_de_la_animacion {}
@-ms-keyframes nombre_de_la_animacion {}
@-moz-keyframes nombre_de_la_animacion {}
@-o-keyframes nombre_de_la_animacion {}
```

Por lo tanto, mediante la propiedad *keyframes*, definimos de alguna manera la animación como tal, es decir, dónde y cómo debe de estar en cada momento el elemento al que se le aplique dicha animación. En la situación de estos fotogramas, podemos aplicar las transformaciones que hemos visto con anterioridad, y por lo tanto animarlas como veremos en los ejemplos.

En un segundo paso, deberemos aplicarle al elemento que queramos dicha animación, y definir algunos parámetros que personalizan la animación para ese elemento en concreto. Estos parámetros o propiedades son los siguientes:

animation-name: Indicamos cuál es el nombre de la aplicación a aplicar.

animation-duration: Indicamos la duración de la animación.

animation-iteration-count: Indicamos el número de repeticiones que queremos.

animation-timing-function: Indicamos la curva de velocidad de la animación, es decir, si se ejecutará a un ritmo constante, más rápido al principio que al final...

animation-direction: Indicamos la dirección de la animación.

animation-delay: Indicamos el momento en el que comenzará la animación. Si el valor es 0 se ejecuta en cuanto se carga la página.

animation: Esta propiedad combina las anteriores de una forma resumida.

¡IMPORTANTE!

Estas propiedades están en funcionamiento en los siguientes navegadores:



Vamos a ver a continuación, un ejemplo de una animación sencilla paso a paso. Dispondremos de un elemento *DIV*, que ira moviéndose trazando un rectángulo.

El código HTML será el siguiente:

```
animacionescss3.html
1 <!DOCTYPE html>
2 <html>
3 <head>
4   <title>Animacion CSS3</title>
5   <link rel="stylesheet" href="animacionescss3.css" type="text/css">
6 </head>
7 <body>
8   <div id="anima">ESTE ELEMENTO SE MUEVE</div>
9 </body>
10 </html>
```

Vemos como únicamente tenemos un elemento *DIV* que contiene un texto, y que será el elemento que animemos. Todo el código CSS está en un fichero aparte y está referenciado en el cabecero de nuestra página.

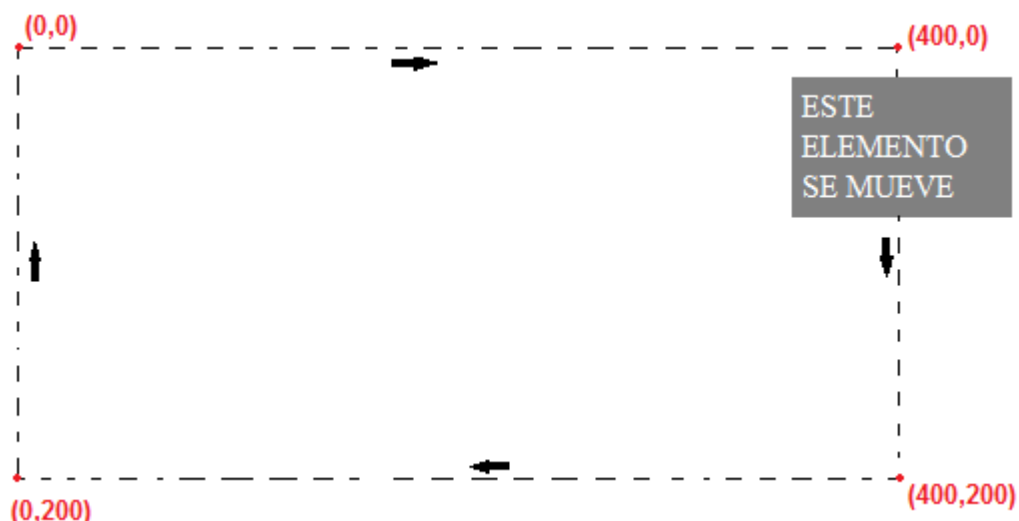
El código CSS será el siguiente:

```
animacionescss3.css X
1  @-webkit-keyframes movimiento-rectangular
2  {
3      0% {left: 0px; top: 0px;}
4      25% {left: 400px; top: 0px;}
5      50% {left: 400px; top: 200px;}
6      75% {left: 0px; top: 200px;}
7      100% {left: 0px; top: 0px;}
8  }
9
10 #anima {
11     -webkit-animation-name: movimiento-rectangular;
12     -webkit-animation-duration: 5s;
13     -webkit-animation-iteration-count: infinite;
14     -webkit-animation-timing-function: linear;
15     -webkit-animation-direction: normal;
16     -webkit-animation-delay: 2s;
17
18     position: relative;
19     width: 100px;
20     background-color: grey;
21     color: #fff;
22     padding: 5px;
23 }
```

En primer lugar hemos creado los fotogramas principales mediante la propiedad *keyframes*. Hemos definido 5 puntos diferentes, y en cada uno de ellos las características definidas han sido únicamente las posiciones, es decir, en cada momento de la animación en que punto debe de estar el elemento animado. Y vemos que lo que hacemos es describir un rectángulo con dicho movimiento.

Una vez creada la animación, vamos a definir las propiedades del elemento que animaremos. En este caso es el *DIV* con identificador "anima". Vemos que aparte de las propiedades habituales, hemos ido definiendo todas las propiedades relacionadas con la animación. Asignamos la animación creada que tendrá una duración de 5s, estará realizando la animación de manera infinita y con una velocidad constante, la dirección será la indicada a la hora de definir la animación y comenzará a ejecutarse 2 segundos después de cargar la página.

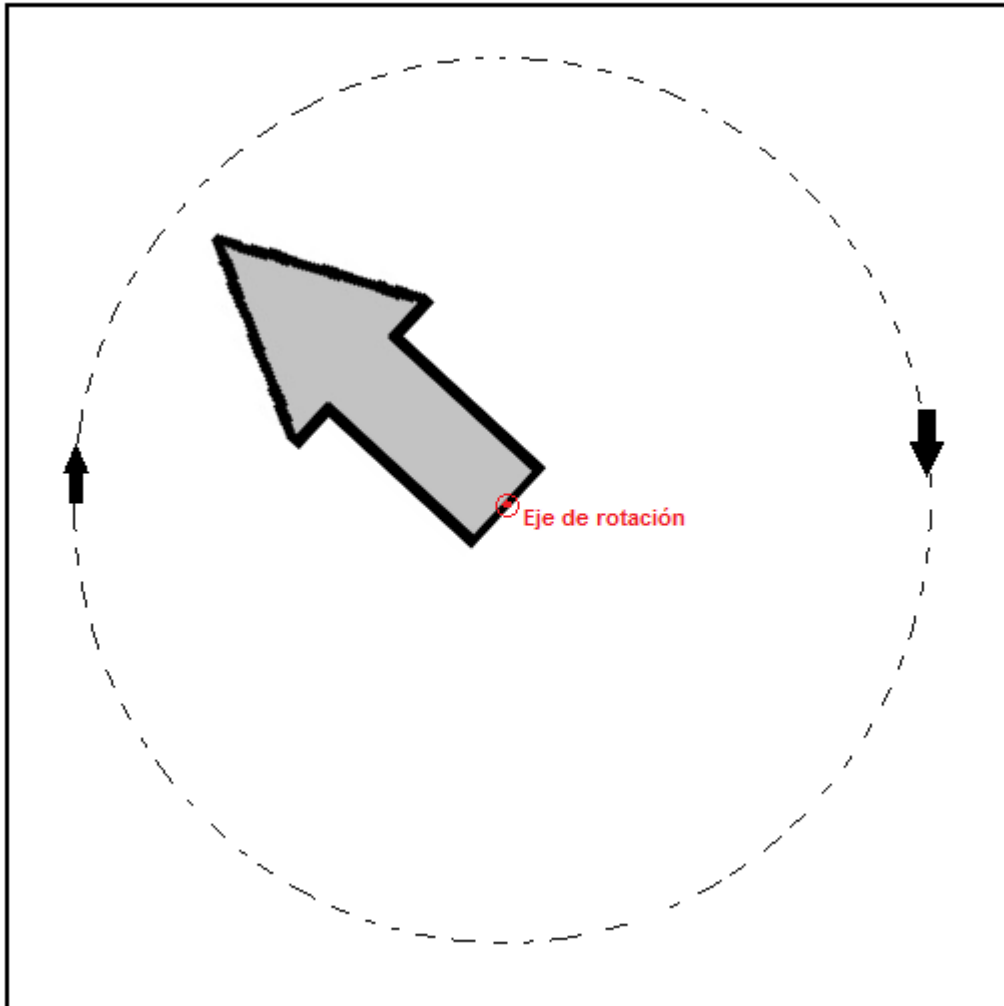
La respuesta en el navegador por tanto será la siguiente:



El *DIV* animado será el elemento de color gris, e irá trazando el recorrido indicado por la línea de puntos en la dirección indicada y con una velocidad constante. La animación no parará, ya que el número de iteraciones es infinito.

Ejemplo de animación

En el siguiente ejemplo, vamos a hacer girar la imagen de una flecha como si fuera el segundero de un reloj. El objetivo es conseguir lo que aparece en la siguiente imagen:



Para ello lo primero será definir el *DIV* con borde negro, y así nos hará de referencia para posicionar nuestra imagen y ejecutar la animación. El código HTML será sencillo por tanto:

```
EjemploAnimacion.html | EjemplosU7 X
1 <!DOCTYPE html>
2 <html>
3 <head>
4   <title>Ejemplo Animacion</title>
5   <link rel="stylesheet" href="EjemploAnimacion.css" type="text/css">
6 </head>
7 <body>
8   <div id="fondo"></div>
9 </body>
10 </html>
```

Vemos que únicamente tenemos un *DIV* con identificador "fondo", y dentro de él la imagen con nombre *flecha.jpg*. Todas las propiedades se las daremos por tanto en el fichero CSS que tenemos referenciado en el cabecero.

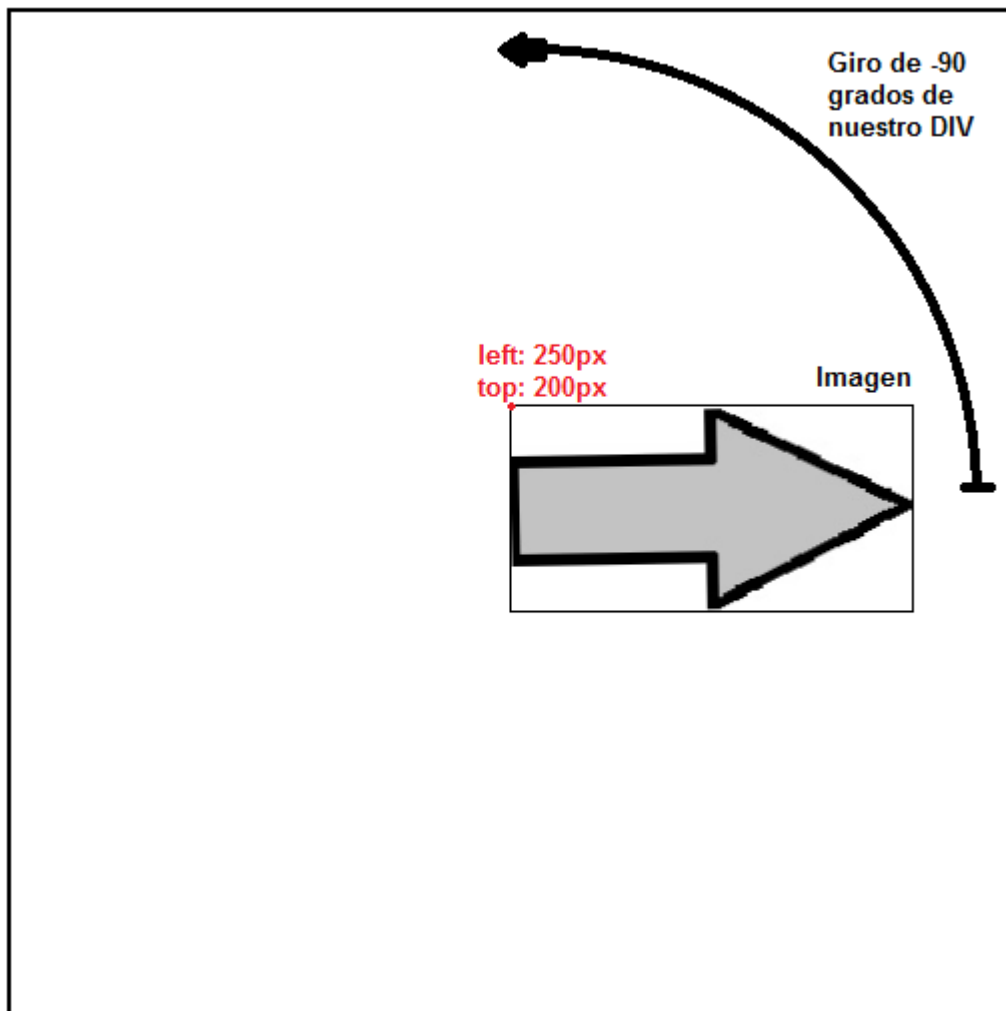
El código de dicho fichero CSS será el siguiente:

```
EjemploAnimacion.css | EjemplosU7 X
1 @-webkit-keyframes movimiento-rotatorio
2 {
3   25% {-webkit-transform: rotate(90deg);}
4   50% {-webkit-transform: rotate(180deg);}
5   75% {-webkit-transform: rotate(270deg);}
6   100% {-webkit-transform: rotate(360deg);}
7 }
8
9 #fondo{
10   width:500px;
11   height:500px;
12   border:2px solid black;
13   -webkit-transform: rotate(-90deg);
14 }
15
16 img{
17   position: relative;
18   left: 250px;
19   top: 200px;
20   -webkit-transform-origin: 0% 50%;
21
22   -webkit-animation-name: movimiento-rotatorio;
23   -webkit-animation-duration: 60s;
24   -webkit-animation-iteration-count: infinite;
25   -webkit-animation-timing-function: linear;
26 }
```

En primer lugar creamos los fotogramas principales de la animación con nombre "movimiento-rotatorio", vemos que lo que hace dicha animación es rotar 360 grados en total, es decir, dar una vuelta sobre el eje de rotación. Le hemos marcado unos puntos intermedios, con las posiciones de giro que debe de tener en cada punto de la animación.

Si pasamos ver las características que le hemos aplicado a nuestro *DIV*, simplemente le hemos dado un alto y un ancho de 500px, y un borde negro. Pero también le hemos aplicado un giro de -90 grados. Este giro lo hemos ejecutado, para que el movimiento de nuestra imagen comience en el punto superior y no en el punto de la derecha de nuestro *DIV*. Lo vemos mejor en la imagen.

Por último, vamos a ver las características que le hemos aplicado a nuestra imagen. En primer lugar, la hemos posicionado dentro de nuestro *DIV* de la siguiente manera ya que tiene un posicionamiento relativo:



Una vez posicionada, hemos cambiado el origen de la transformación a la parte derecha de la imagen y al centro en altura, ya que queremos que la imagen gire sobre el parte trasera de la flecha, y no sobre el centro de la misma. De esta manera, ya tenemos colocada la imagen y el punto de rotación, y solo nos queda aplicarle la animación definida con anterioridad. Esta animación durará 60 segundos, ya que simula el segundero de un reloj, mantendrá una velocidad constante, y se ejecutará de manera infinita.

¡IMPORTANTE!

En este caso, hemos creado una animación que funcionará en navegadores *Webkit*, si quisiéramos que funcionase en todos, tendríamos que repetirlo con cada una de las extensiones anteriormente expuestas.

¡IMPORTANTE!

Podemos consultar todos los aspectos relacionados con las animaciones de CSS3 en la recomendación W3C desde este enlace: [Animaciones CSS3](#)

Transiciones en CSS3

Las transiciones, son otro de los aspectos destacados a la hora de animar nuestra página mediante CSS. Hoy en día, cuando navegamos, en función de nuestros movimientos sobre la página, se ejecutan varios cambios de las distintas propiedades de los elementos que en ella tenemos. Por ejemplo, cuando pasamos sobre una imagen con el ratón, esta cambia de tamaño, o cuando pasamos sobre un botón, este cambia de color. Pues hablar de transiciones, es hablar de esos cambios de propiedades, y mediante las nuevas opciones que ofrece CSS3 entorno a ellas, podemos hacer que esos cambios no se ejecuten de manera inmediata, sino que se animen de diferentes maneras durante un espacio de tiempo.

Anotación

Las **propiedades que vamos a ver a continuación**, tampoco están todavía estandarizadas totalmente. De manera que para hacerlas funcionar en todos los principales navegadores, deberemos añadir delante de la propiedad la extensión para cada grupo de navegadores de la siguiente manera:

```
-webkit-transition-property: propiedad;  
-ms-transition-property: propiedad;  
-moz-transition-property: propiedad;  
-o-transition-property: propiedad;
```

Las propiedades disponibles a la hora de definir estas transiciones son las siguientes:

transition-property: Indicamos cuál es el nombre de la propiedad CSS que animaremos cuando cambie.

transition-duration: Indicamos la duración de la transición.

transition-timing-function: Indicamos la curva de velocidad de la transición, es decir, si se ejecutará a un ritmo constante, más rápido al principio que al final...

transition-delay: Indicamos el momento en el que comenzará la transición. Si el valor es 0 se ejecuta en cuanto cambiamos el valor de la propiedad.

transition: Esta propiedad combina las anteriores de una forma resumida.

¡IMPORTANTE!

Estas propiedades están en funcionamiento en los siguientes navegadores:



Vamos a ver a continuación, un ejemplo de una serie de transiciones sencillas paso a paso. Dispondremos de un elemento *DIV*, al que le iremos cambiando propiedades, y animando dichos cambios.

El código HTML será el siguiente:

```
transicionescss3.html X
1 <!DOCTYPE html>
2 <html>
3 <head>
4   <title>Transicion CSS3</title>
5   <link rel="stylesheet" href="transicionescss3.css" type="text/css">
6 </head>
7 <body>
8   <div id="cambia">ESTE ELEMENTO VA A CAMBIAR</div>
9 </body>
10 </html>
```

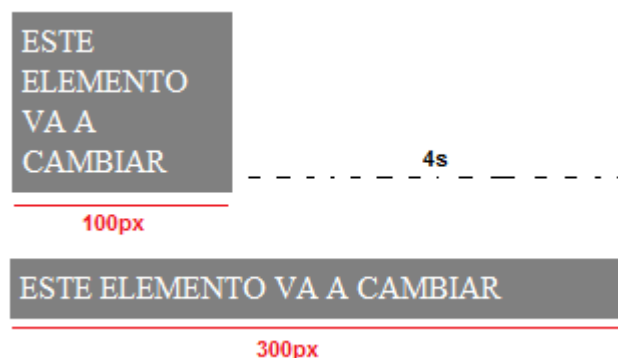
Vemos como únicamente tenemos un elemento *DIV* que contiene un texto, y que será el elemento sobre el que vayamos a ejecutar los cambios de propiedad y las transiciones sobre esos cambios. Todo el código CSS está en un fichero aparte y está referenciado en el cabecero de nuestra página.

El código CSS será el siguiente:

```
transicionescss3.css X
1 #cambia {
2   -webkit-transition-property: width;
3   -webkit-transition-duration: 4s;
4   -webkit-transition-timing-function: linear;
5   -webkit-transition-delay: 1s;
6
7   width: 100px;
8   background-color: grey;
9   color: #fff;
10  padding: 5px;
11 }
12
13 #cambia:hover{
14   width: 300px;
15 }
```

Vemos como aparte de asignarle las propiedades habituales al elemento sobre el que aplicaremos los cambios, hemos definido una transición para el cambio de propiedad *width*. Concretamente, cuando cambie la anchura del elemento tardará 1 segundo en iniciarse la transición, dicho cambio durará 4 segundos y se ejecutará a una velocidad constante. El cambio de anchura, se dará cuando nos coloquemos sobre el elemento, como hemos definido en el fichero CSS.

La respuesta en el navegador por tanto será la siguiente:



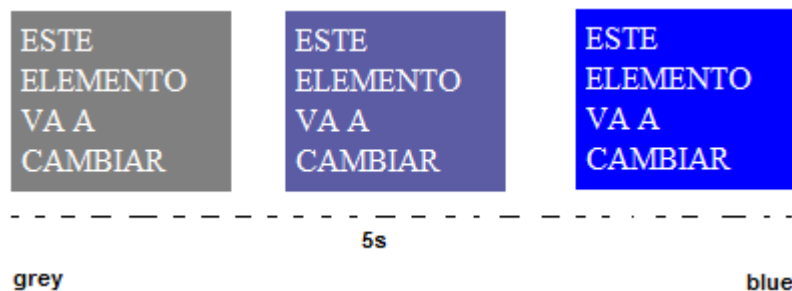
Al colocarnos sobre el *DIV*, esperará 1 segundo sin hacer nada, y a partir de ese momento iniciará el cambio de anchura que durará 4 segundos, hasta conseguir la anchura indicada de 300px. Una vez estemos fuera, el retroceso a la anchura inicial, se hará de la misma manera, ya que es también un cambio de anchura.

Y si el código CSS lo cambiamos de la siguiente manera:

```
transicionescss3.css X
1 #cambia {
2   -webkit-transition-property: background-color;
3   -webkit-transition-duration: 5s;
4   -webkit-transition-timing-function: linear;
5   -webkit-transition-delay: 0s;
6
7   width: 100px;
8   background-color: grey;
9   color: #fff;
10  padding: 5px;
11 }
12
13 #cambia:hover{
14   background-color: blue;
15 }
```

En este caso hemos definido una transición para el cambio de propiedad *background-color*. Concretamente, cuando cambie el color de fondo del elemento se iniciará la transición, dicho cambio durará 5 segundos y se ejecutará a una velocidad constante. El cambio de color de fondo, se dará cuando nos coloquemos sobre el elemento, como hemos definido en el fichero CSS.

La respuesta en el navegador por tanto será la siguiente:



Al colocarnos sobre el *DIV*, iniciará el cambio de color de fondo, que durará 5 segundos, hasta conseguir al final el color azul indicado. Una vez estemos fuera, el retroceso al color gris inicial, se hará de la misma manera, ya que es también un cambio de color de fondo.

Ejemplo de transición

En el siguiente ejemplo, vamos a aplicar diferentes cambios de propiedades sobre un *DIV* que contiene una imagen dentro, en función del evento que suceda. Para cada uno de estos cambios, definiremos transiciones, para que los cambios se den en un espacio de tiempo y no de manera inmediata. La *DIV* y la imagen serán los que vemos a continuación:



Para ello lo primero será definir el *DIV* con borde negro y el fondo gris, y después colocar la imagen en su interior. El código HTML será sencillo por tanto:

```
EjemploTransicion.html X
1  <!DOCTYPE html>
2  <html>
3  <head>
4      <title>Ejemplo Transicion</title>
5      <link rel="stylesheet" href="EjemploTransicion.css" type="text/css">
6  </head>
7  <body>
8      <div id="trans">
9          
10     </div>
11 </body>
12 </html>
```

Vemos que únicamente tenemos un *DIV* con identificador "trans", y dentro de él la imagen con nombre *paisaje.jpg*. Todas las propiedades se las daremos por tanto en el fichero CSS que tenemos referenciado en el cabecero.

El código de dicho fichero CSS será el siguiente:

```
EjemploTransicion.css X
1  #trans{
2      width:300px;
3      height:200px;
4      padding:20px;
5      border:2px solid black;
6      background:grey;
7
8      -webkit-transition-property: width, height, background-color;
9      -webkit-transition-duration: 4s;
10     -webkit-transition-timing-function: linear;
11 }
12
13 img{
14     width:100%;
15     height:100%;
16 }
```

En primer lugar vemos que a nuestra imagen las únicas características que le hemos aplicado son la altura y la anchura, y que en ambos casos son relativas al *DIV* en el que está colocada. Por lo tanto, según cambiemos las propiedades de tamaño de nuestro *DIV*, estaremos cambiando también las de nuestra imagen.

Si pasamos a ver las características que le hemos aplicado a nuestro *DIV* con identificador "trans", vemos que le hemos definido un tamaño, una distancia entre el borde y el contenido que es el trozo gris que vemos alrededor de la imagen, un borde y color de fondo gris.

Aparte, hemos definido las transiciones para los cambios de propiedades de anchura, altura y color de fondo. En todos los casos la transición durará 4 segundos y se ejecutará a una velocidad constante.

Por lo tanto, ahora solo nos queda definir los cambios de propiedades en función del eventos que sucedan, para así ver en funcionamiento esas transiciones que hemos definido. Esto lo definiremos desde JavaScript, y el código será el siguiente:

```
EjemploTransicion.html X
1 <!DOCTYPE html>
2 <html>
3 <head>
4   <title>Ejemplo Transicion</title>
5   <link rel="stylesheet" href="EjemploTransicion.css" type="text/css">
6
7   <script language="JavaScript">
8     window.onload = function (){
9       document.getElementById("trans").onmouseover = cambiaColor;
10      document.getElementById("trans").onclick = grande;
11      document.getElementById("trans").ondblclick = pequeno;
12    }
13
14    function cambiaColor(){
15      document.getElementById("trans").style.background = "#58ACFA";
16    }
17
18    function grande(){
19      document.getElementById("trans").style.width = "150px";
20      document.getElementById("trans").style.height = "100px";
21    }
22
23    function pequeno(){
24      document.getElementById("trans").style.width = "500px";
25      document.getElementById("trans").style.height = "500px";
26    }
27  </script>
28
29 </head>
30 <body>
31   <div id="trans">
32     
33   </div>
34 </body>
35 </html>
```

En el momento que carga por completo la página, definimos las funciones a ejecutarse cuando suceda un evento u otro sobre nuestro *DIV*. En este caso, cuando suceda el evento *mouseover*, es decir, cuando nos pongamos sobre el elemento con el ratón, se ejecutará la función *cambiaColor()*, que si nos fijamos, lo único que hace es cambiar el color de fondo de nuestro *DIV*. Cuando sucede el evento *click*, es decir, cuando hacemos clic sobre el elemento, se ejecuta la función *grande()*, que como vemos lo único que hace es cambiar la altura y la anchura de nuestro *DIV* a tamaños mayores. Cuando sucede el evento *dblclick*, es decir, cuando hacemos doble clic sobre el elemento, se ejecuta la función *pequeno()*, que como vemos lo único que hace es cambiar la altura y la anchura de nuestro *DIV* a tamaños menores.

Por lo tanto, cuando nos ponemos sobre el elemento vemos una transición del color de fondo del gris al color azul, cuando hacemos clic vemos una ampliación tanto del *DIV* como de la imagen, ya que el tamaño de la imagen es relativo al del *DIV*, y cuando hacemos doble clic una disminución del tamaño, también de ambos elementos. Todas estas transiciones duran 4 segundos, y se ejecutan a una velocidad constante como hemos dicho.

¡IMPORTANTE!

En este caso, hemos creado una animación que funcionará en navegadores *Webkit*, si quisiéramos que funcionase en todos, tendríamos que repetirlo con cada una de las extensiones anteriormente expuestas.

¡IMPORTANTE!

Podemos consultar todos los aspectos relacionados con las transiciones de CSS3 en la recomendación W3C desde este enlace: [Transiciones CSS3](#)

Ejercicios

Ejercicio 1: Crear una galería de imágenes animada

Vamos a crear una galería de fotos animada utilizando las opciones que ofrece CSS3 en cuanto a transformaciones, animaciones y transiciones. La galería tendrá la forma que vemos en la imagen. En la parte superior colocaremos las fotos disponibles en miniatura, y al hacer clic sobre la que queramos ver, dicha imagen se desplegará a un tamaño mayor en el recuadro gris de debajo. La forma de desplegarse será animada, en un primer momento dará un giro sobre si misma como vemos en la imagen, y una vez terminado comenzará a agrandarse, primero en horizontal, y luego en vertical hasta conseguir el tamaño deseado.



Pasos a seguir

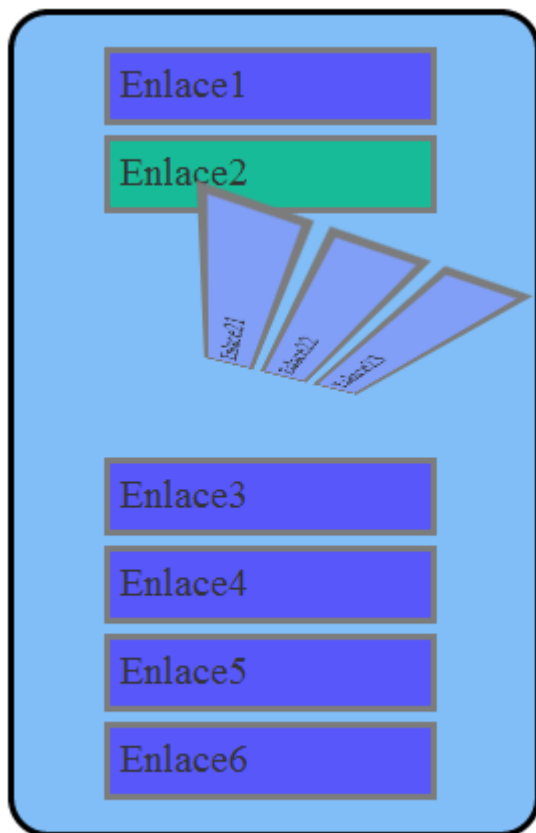
1. En primer lugar, estructuraremos la página con dos elementos *DIV* uno dentro del otro. El más grande contendrá las imágenes en miniatura y el segundo será donde se amplíen las imágenes.
2. A continuación, colocaremos nuestras imágenes disponibles en miniatura dentro del *DIV* más grande.
3. Una vez estructurado, deberemos definir una animación que será la que rote la imagen, y dos transiciones de anchura y altura que se ejecutarán sobre la imagen seleccionada. Hay

que tener en cuenta, que se ejecuta primero la animación, luego la transición de anchura, y luego la de altura.

4. Y hecho esto, solo nos queda generar una función en JavaScript que se ejecutará cuando hagamos clic sobre alguna de las imagenes, y que será la encargada de poner en marcha todo el proceso.

Ejercicio 2: Crear un menú desplegable animado

Vamos a crear un menú animado utilizando las opciones que nos ofrece CSS3 en cuanto a transformaciones, animaciones y transiciones. El menú tendrá la forma que vemos en pantalla, constará de enlaces y subenlaces, aunque estos últimos estarán ocultos hasta que no hagamos clic sobre alguno de los enlaces principales. En ese momento, se desplegará el submenú correspondiente al enlace pulsado y cambiará de color del propio enlace principal. Como vemos, el cambio de color no se hará de manera inmediata, sino que será una transición, y la forma de mostrar el submenú, será una animación como vemos en la imagen. Al hacer clic sobre cualquier otro enlace, todos los demás volverán al estado inicial, es decir, solo se mantendrá uno activo.



Pasos a seguir

1. En primer lugar crearemos el DIV que contiene todo el menú.

2. El siguiente paso será crear el menú, y lo haremos mediante listas. Los enlaces serán elementos de una lista, y los subenlaces serán sublistas de los elementos de esta lista principal. Todas esas sublistas deberán estar ocultas de inicio.
3. A continuación, crearemos la transición para el color de los enlaces principales, y la animación para las sublistas, que será una animación 3D.
4. Y el último paso, será crear la función de JavaScript que se ejecutará al pulsar cualquier enlace. Esta función, aparte de activar todo el proceso, deberá devolver los elementos activos anteriormente a su estado inicial.