UD3 HTML Y CSS.

DISEÑO DE INTERFACES WEB

3.3 Multimedia en HTML5. Transformaciones y transiciones CSS3

Contenidos

1. Objetivos

En este tercer tema se pretenden conseguir los siguientes objetivos:

- Conocer los diferentes **códecs de vídeo y audio** utilizados en la web por los distintos navegadores.
- Utilizar los **nuevos elementos HTML5** para audio y vídeo.
- **Insertar JavaScript** en nuestros documentos.
- Crear **manejadores de eventos JavaScript** de forma estándar.
- Seleccionar elementos de la página con los nuevos **selectores JavaScript**. Utilizar el **nuevo api** de manejo de elementos multimedia que incorpora HTML5.
- Crear **efectos web** utilizando las transformaciones y transiciones de CSS3.

2. Introducción

Ya en el punto anterior vimos algunas de las nuevas características que nos aportan tanto HTML5 como CSS3. En este tercer punto, nos vamos a centrar en tres aspectos fundamentales:

- La importancia que tiene JavaScript para la web en general y para HTML5 en particular.
- Cómo se manejan los elementos multimedia (principalmente audio y vídeo) con HTML5.
- Cómo realizar pequeñas animaciones sin utilizar flash, ni JavaScript, ni librerías externas como jQuery.

3. HTML5

Multimedia en HTML5

Reproduciendo vídeo en HTML5

Hoy en día lo más común descargar y ver vídeo dentro del navegador web. En un principio, la tecnología Flash de Adobe ascendió velozmente al trono de la contención de vídeo incrustado. Hasta hace pocos años esto no era un gran problema, ya que la mayoría del contenido de vídeo de la web se encontraba en FLV o F4V1. Sin embargo, varios factores hicieron que el panorama cambiara:

- La llegada de dispositivos móviles, ligeros, de bajo consumo de energía y gran poder gráfico.
- La necesidad de contar con un lenguaje web mejor y más expresivo.
- Conflictos de interés de grandes jugadores de la industria tecnológica.

Con todo esto, la hegemonía de Flash como contenedor de vídeo comenzó a verse amenazada

3. HTML5

El elemento `<vídeo>`

El elemento `<video>` ofrece varios **atributos** para establecer su comportamiento y configuración:

- Los atributos `width` y `height` , declaran las **dimensiones** para el elemento o ventana del reproductor. El tamaño del vídeo **se ajusta automáticamente para entrar dentro** de estos valores, pero **no fueron considerados para redimensionar** el vídeo, sino para limitar el área ocupada por el mismo y mantener la consistencia en el diseño.
- El atributo `src` indica la **fuente** del vídeo. Este atributo puede ser reemplazado por el elemento `<source>` y su propio atributo `src` para declarar varias fuentes con diferentes formatos, como en el siguiente ejemplo:

3. HTML5

El elemento <vídeo>

```
<video id="medio" width="720" height="400" controls>
   <source src="video.mp4">
   <source src="video.ogg">
</video>
```

En el ejemplo anterior, el elemento <video> fue expandido. Ahora, dentro de las etiquetas del elemento hay dos elementos <source>. Estos nuevos elementos proveen **diferentes fuentes de vídeo para que los navegadores puedan elegir**. El navegador leerá la etiqueta **<source>** y decidirá qué archivo reproducir de acuerdo a los formatos que soporte.

3. HTML5

Atributos para `<vídeo>`

En el ejemplo anterior vemos que se ha añadido el atributo `controls`. Este atributo **muestra controles de vídeo** provistos por el navegador.

Junto con controls, también podemos usar los atributos:

- `autoplay`: cuando este atributo está presente, el navegador comenzará a **reproducir el vídeo automáticamente**, tan pronto como pueda.
- `loop`: si este atributo es especificado, el navegador comenzará a **reproducir el vídeo nuevamente** cuando llegue al final.
- `poster`: este atributo es utilizado para **proveer una imagen** que será mostrada **mientras esperamos** que el vídeo comience a ser reproducido.

3. HTML5

Atributos para <vídeo>

- preload: este atributo puede recibir tres valores distintos:
  - none: indica que el vídeo **no debería ser cacheado**, por lo general, con el propósito de minimizar tráfico innecesario.
  - metadata: recomendará al navegador que trate de capturar **información acerca de la fuente** (por ejemplo, dimensiones, duración, primer cuadro, etc.).
  - auto: es el valor configurado por defecto, que le sugerirá al navegador **descargar el archivo** tan pronto como sea posible.

```
<video id="reproductor" width="720" height="400" preload="none" controls loop poster="poster.jpg">
   <source src="video.mp4">
   <source src="video.ogg">
</video>
```

3. HTML5

El elemento `<audio>`

Trabaja del mismo modo y comparte varios atributos con el elemento <video>:

- src: **Ruta** del archivo a reproducir. Será reemplazado por <source> para ofrecer diferentes formatos de audio entre los que el navegador pueda elegir.
- controls: Activa la **interfaz** que cada navegador provee por defecto para controlar la reproducción del audio.
- autoplay: **Reproducción automática** cuando sea posible.
- loop: Reproducción en **bucle** automática.
- preload:
  - none: No cargar en caché.
  - metadata: Obtiene información sobre el medio.
  - auto: Valor por defecto. Descargar cuando sea posible.

3. HTML5

El elemento <audio>

```
<audio id="medio" controls>
  <source src="cancion.mp3">
  <source src="cancion.ogg">
</audio>
```

El código anterior reproducirá música en todos los navegadores utilizando los **controles por defecto**. Aquellos que **no puedan reproducir MP3 reproducirán OGG** y viceversa. Debemos recordar que MP3, al igual que MP4 para vídeo, tienen uso restringido por **licencias comerciales**, por lo que sólo podemos usarlos en circunstancias especiales, de acuerdo con lo determinado por cada licencia.

3. HTML5

Tipos MIME (Internet Media Types)

Los tipos MIME son una forma de definir formatos de archivos para que nuestro sistema sepa cómo manejarlos. 2 niveles:

Desde el Servidor: Configurar el servidor para servir correctamente los tipos MIME. En el caso de tener un servidor Apache, debemos incluir lo siguiente en el archivo .htaccess:

```
#AddType TYPE/SUBTYPE EXTENSION
AddType audio/aac .aac
AddType audio/mp4 .mp4 .m4a
AddType audio/mpeg .mp1 .mp2 .mp3 .mpg .mpeg
AddType audio/ogg .oga .ogg
AddType audio/wav .wav
AddType audio/webm .webm
AddType video/mp4 .mp4 .m4v
AddType video/ogg .ogv
AddType video/webm .webm
```

3. HTML5

Tipos MIME (Internet Media Types)

Desde el Cliente: Cuando definimos fuentes en nuestro código, podemos ayudar a nuestro navegador en la identificación del contenido especificando el tipo MIME utilizado.

```
<audio>
   <source src="elvis.mp3" type='audio/mpeg; codecs="mp3”’>
   <source src="elvis.ogg" type='audio/ogg; codecs="vorbis"'>
</audio>
<video poster="star.png" autoplay loop controls tabindex="0">
   <source src="movie.webm" type='video/webm; codecs="vp8,
        vorbis"' />
   <source src="movie.ogv" type='video/ogg; codecs="theora,
        vorbis"' />
</video>
```

3. HTML5

Tipos MIME (Internet Media Types)

En el ejemplo anterior definimos el elemento y las fuentes. El **navegador elegirá solamente una** de ellas.

Al especificar el atributo type (no es obligatorio), permitimos que el navegador conozca el tipo MIME y los tipos de codecs que debe utilizar antes de descargar la canción.

Si no indicamos dicho atributo, el navegador intentará averiguar, mediante **prueba y error**, cuál es el tipo adecuado.

4. CSS3

Posicionamiento de los elementos

Los elementos pueden ser colocados utilizando las **propiedades** `top`, `bottom`, `left` y `right`. Sin embargo, estas propiedades no funcionarán a menos que la propiedad position se establezca en primer lugar.

En función del método de posicionamiento que utilicemos, estas propiedades funcionaran de una manera u otra. Disponemos de **cuatro métodos de posicionamiento** diferentes variando el valor de la propiedad `position`:

- **static** (posicionamiento estático): Este es el **valor por defecto** de la propiedad. Los elementos estáticos se encuentran posicionados de acuerdo al **flujo normal de la página**, es decir, a continuación del anterior elemento con posicionamiento estático. Los elementos con este posicionamiento no se ven afectados por las propiedades `top`, `bottom`, `left` y `right`.

4. CSS3

- **fixed**: Un elemento con la posición fija se posiciona **relativo a la ventana del navegador**, por lo tanto, **no se moverá aunque se haga scroll** en la página. Además, estos elementos se eliminan del flujo normal del documento y el resto de elementos se comportarán como si estos elementos no existieran. Esto hace que puedan **superponerse** a otros elementos o entre sí.

```
footer { position:fixed; bottom:0px; left:0px; }
```

En el ejemplo, el <footer>, estará colocado siempre en el extremo inferior izquierdo de la pantalla.

- **relative**: Los elementos con este posicionamiento se encuentran **relativos a la posición que debería ocupar dentro del flujo** normal del documento. Así, su contenido puede ser movido y/o superpuesto a otros elementos, pero el espacio reservado para el elemento se mantendrá reservado dentro del flujo normal del documento. A menudo, se utilizan elementos con posicionamiento relativo para contener otros elementos con posicionamiento absoluto.

4. CSS3

- **absolute**: Un elemento con posicionamiento absoluto se posiciona de forma **relativa al primer elemento padre que tiene un posicionamiento distinto de estático**, normalmente relativo y, en ocasiones, absoluto o fijo. Si no se encuentra dentro de ningún elemento con estos posicionamientos, se colocará relativo al elemento <html>. En este caso, será muy probable que cuando movamos horizontalmente la ventana del navegador o -todavía más grave- que según tenga el usuario una anchura u otra del navegador, el elemento este posicionado en un lugar u otro horizontalmente. Para evitar este grave problema, o bien introducimos el elemento con posición absoluta dentro de otro elemento con posición relativa, o bien le damos una posición relativa al body, que es el elemento que contiene al resto de elementos. Evidentemente, los elementos con posicionamiento absoluto se eliminarán del flujo normal del documento, por lo que el resto de elementos se posicionarán como si estos elementos no existieran.

4. CSS3

Elementos superpuestos

Cuando los elementos se posicionan fuera del flujo normal del documento, puede ser que se superpongan. En estas circunstancias, debemos indicar de alguna forma **qué elementos se colocan delante y cuáles se colocan detrás**. Para ello disponemos de la propiedad **`z-index`**.

La propiedad `z-index` indica el **orden de apilamiento** de un elemento (qué elemento debe situarse delante o detrás del resto). Un elemento con un orden de apilamiento **mayor, está siempre delante** de un elemento con un orden menor. Si dos elementos se solapan sin tener especificada la propiedad `z-index`, el que haya sido insertado más tarde en el código `html` estará sobre el otro.

Veamos el siguiente ejemplo:

```
    <div id="chincheta1"></div>
    <div id="chincheta2"></div>
    <h1>Título de la página</h1>
  </header>
  <style>
    header { position: relative; padding: 20px 100px; }
    #chincheta1 {
      position:absolute;
      left:320px; top:-5px;
      width: 35px; height: 49px;
      background-image: url("imgs/chincheta.png");
      z-index: 2;
      border: 1px solid black;
    }
    #chincheta2 {
    position:absolute;
    left:350px; top:-5px;
    width: 35px; height: 49px;
    background-image: url("imgs/chincheta.png");
    z-index: 1;
    }
  </style>
```

Título de la página

4. CSS3

Transformaciones (`transform`)

Con HTML5 y CSS3 podemos utilizar la propiedad `transform` para modificar características de un elemento como la **posición** (translate), la **rotación** (rotation), la **escala** (scale) o el **sesgo**[1] (skew).

Este cambio de posición, tamaño o rotación se produce al instante (**no hay animación** en este cambio, para ello habría que utilizar otras propiedades que veremos más adelante).

Para poder comprobar estas propiedades vamos a utilizar un que contiene el dibujo sencillo de un martillo creado con Scalable Vector Graphics (SVG).

[1]Sesgo: orientación que toma un elemento

```
<div id="martillo1">
   <svg width="250" height="250" xmlns="http://www.w3.org/2000/svg">
      <!-- Mango (marrón oscuro) -->
         <rect x="43" y="10" width="14" height="160" rx="6" ry="5" fill="#895111"/>
      <!-- Brillo mango (marrón claro) -->
         <rect x="45" y="15" width="4" height="150" rx="6" ry="6" fill="#A68661"/>
      <!-- Cabeza hierro -->
         <rect x="10" y="22" width="80" height="45" rx="14" ry="12" fill="#6E6E5F"/>
      <!-- Brillo cabeza -->
         <rect x="20" y="37" width="62" height="24" rx="6" ry="6" fill="#898982"/>
         <rect x="20" y="25" width="62" height="6" rx="4" ry="4" fill="#C2C2B6"/>
      <!-- Sombra de la cabeza en el mango -->
         <rect x="43" y="67" width="14" height="9" fill="#482907"/>
   </svg>
</div>
#martillo {
   position:relative;
   width:110px;
   height:190px;
   padding:16px;
   border:1px dotted gray;
   border-radius:9px;
   float:left;
}
```

4. CSS3

Movimiento (`translate`)

El movimiento se basa en la **suma o resta de píxeles** en su posición horizontal y vertical respecto a la posición en la que se encuentra el objeto en este momento. Con números **positivos** se desplaza el objeto hacia la **derecha y hacia abajo** y con números **negativos** se desplaza hacia arriba y hacia la izquierda. En este ejemplo, el martillo se desplaza 200 píxeles hacia la derecha y 5 píxeles hacia abajo respecto a su posición original.

```
#martillo { transform:translate(200px,5px); }
```

También podemos utilizar funciones independientes para cada dimensión: `translateX` y `translateY`.

Si queremos colocar este martillo en la mitad horizontal de la pantalla, es decir, centrar el martillo horizontalmente, podemos indicar un porcentaje (50%), en lugar de utilizar píxeles.

```
#martillo { transform:translate(50%,0px); }
```

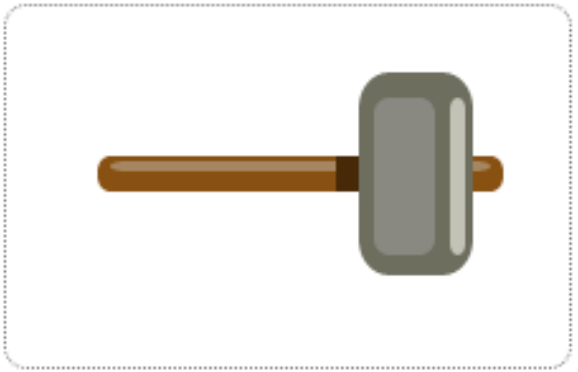
4. CSS3

Rotación (`rotate`)

La rotación se basa en los **grados** de rotación (de 0º a 360º). El eje de rotación siempre está situado en el **centro del elemento** y los grados se especifican añadiendo deg después del valor (por ejemplo: 90deg).

Si se indican valores positivos en los grados de rotación, el objeto rota hacia la derecha. Con los grados especificados en números negativos, el objeto rota hacia la izquierda.

```
#martillo{ transform: rotate(90deg); }
```



En este caso, el elemento del martillo rota 90º hacia la derecha tomando como **eje de rotación el centro del ** donde se encuentra el martillo (que no siempre coincide con el contenido de dicho).

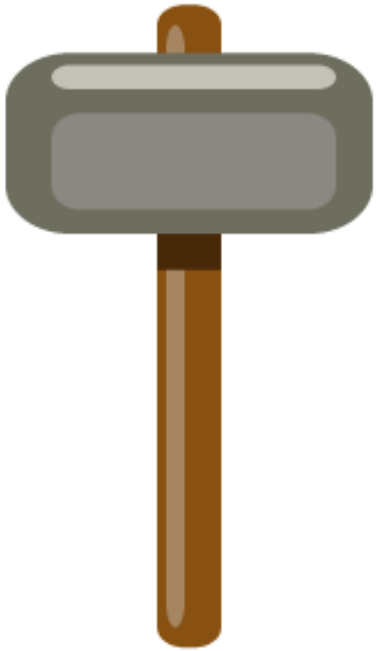
4. CSS3

Escala (`scale`)

Esta propiedad se basa en una **escala horizontal, seguida de otra vertical**, siendo 1 la escala original. Para obtener un porcentaje se puede multiplicar el valor indicado por 100, así un valor de '2' (2x100=200) representaría un 200%, es decir, el doble de tamaño. El elemento aumenta o disminuye de tamaño tomando como eje su centro.

```
#martillo { transform:scale(2, 1.75); }
```

En el ejemplo, no aumenta de tamaño de forma proporcional. Aumenta el doble de anchura (2x100= 200%) y soló un 175% de altura (1.75x100= 175%).



4. CSS3

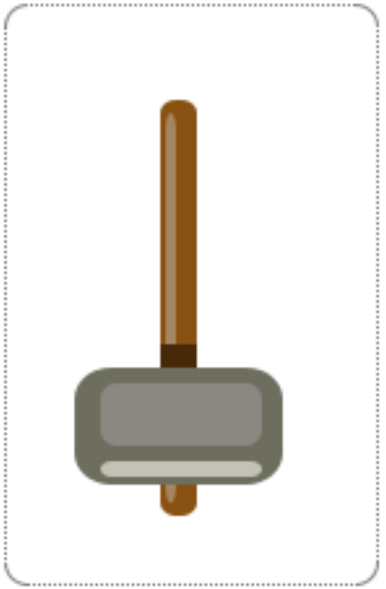
Escala (`scale`)

Podemos conseguir el **efecto del espejo** utilizando números negativos para la escala.

```
#martillo { transform:scale(1, -1); }
```

En el ejemplo, se mantiene la escala en el eje x. Sin embargo, el valor **-1** en el eje y, aunque también mantiene la proporción original, **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 único parámetro.



4. CSS3

Escala (`scale`)

Podemos entender el sesgo como la **rotación del elemento en los ejes x e y**. Para definirlo se indica primero la rotación en el eje x en grados, y posteriormente, la rotación en el eje y, también en grados. Se pueden utilizar valores positivos o negativos, según se requiera para la deformación.

```
#martillo { transform:skew(10deg, 25deg); }
```

Si sólo indicamos el primer parámetro, únicamente se rotará el elemento sobre el eje x. Al igual que ocurría con la función `rotate` podemos utilizar funciones diferentes para cada dimensión: `skewX` y `skewY`.

4. CSS3

Múltiples transformaciones simultáneas

Se pueden indicar, **de manera simultánea**, tantas transformaciones como se deseen, separando cada una de ellas por un espacio.

```
#martillo { transform:translate(200px, 0px) rotate(33deg) scale(0.5,0.5); }
```

En este ejemplo, al del martillo se le aplica un desplazamiento hacia la derecha de 200 píxeles, seguidamente una rotación hacia la derecha de 33 grados y, finalmente, una redacción de tamaño proporcional del 50%.

4. CSS3

Compatibilidad con otros navegadores

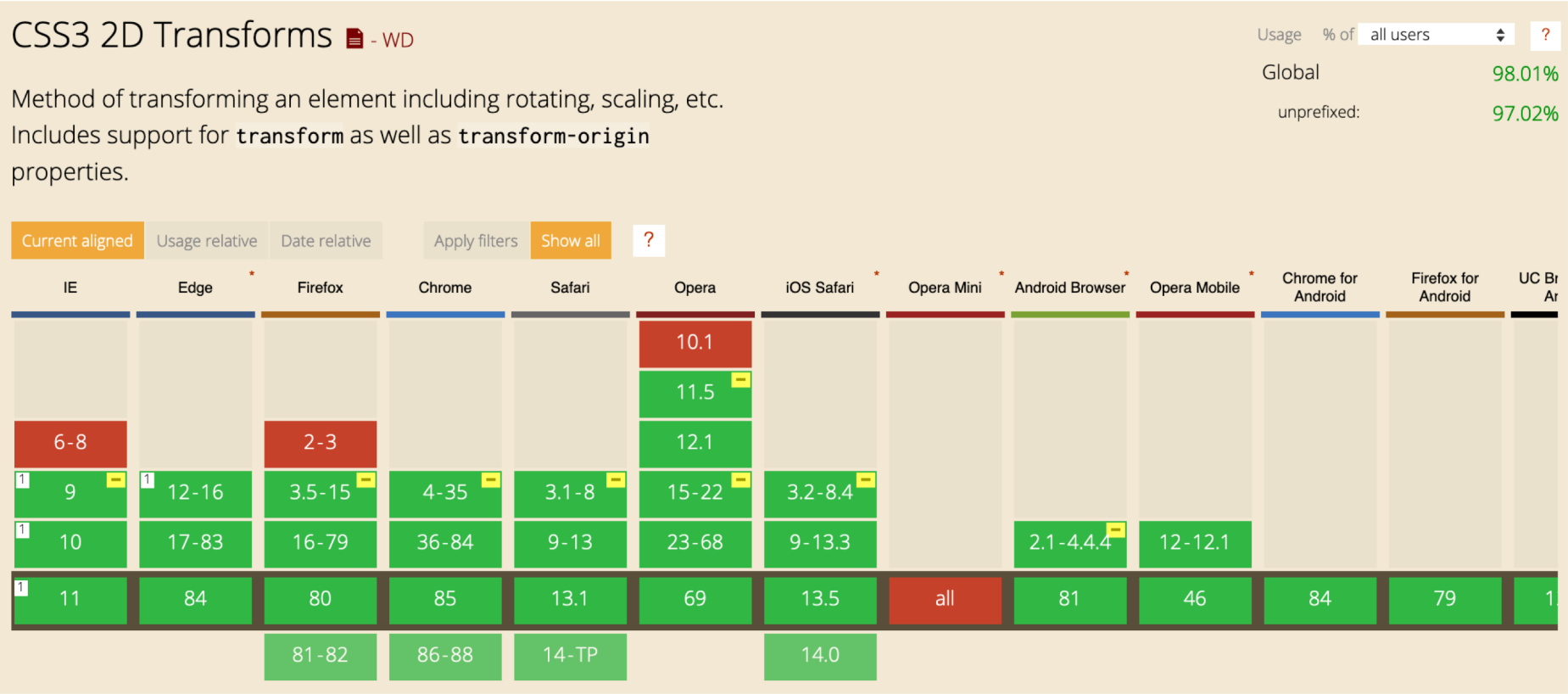
Es importante tener en cuenta que para que transform funcione en todos los **navegadores** deberemos repetir la propiedad anteponiendo los **prefijos** de cada uno de los navegadores:

- `-webkit-` (para Chrome y Safari)
- `-moz-` (para Firefox)
- `-o-` (para Opera)
- `-ms-` (para Internet Explorer)

```
#martillo{
   transform:translate(200px, 0px) rotate(33deg) scale(0.5,0.5);
   -webkit-transform:translate(200px, 0px) rotate(33deg) scale(0.5,0.5);
   -moz-transform:translate(200px, 0px) rotate(33deg) scale(0.5,0.5);
   -o-transform:translate(200px, 0px) rotate(33deg) scale(0.5,0.5);
   -ms-transform:translate(200px, 0px) rotate(33deg) scale(0.5,0.5);
}
```

4. CSS3

Veamos la tabla de compatibilidad que nos ofrecía `Can I use?` para esta propiedad en el momento de crear estos apuntes. (URL: http://caniuse.com/#feat=transforms2d):



4. CSS3

Transformaciones dinámicas

Las transformaciones vistas hasta ahora cambiarán la forma de la web, pero la mantendrán tan estática como siempre. Sin embargo, podemos aprovecharnos de la combinación de transformaciones y pseudo-clases para **convertir nuestra página en una aplicación dinámica**:

```
#martillo {
  position:relative;
  width:110px;
  height:190px;
  padding:16px;
  border:1px dotted gray;
  border-radius:9px;
  float:left;
}
#martillo:hover { transform: scale(1,-1); }}
```

4. CSS3

Transformaciones dinámicas

En el ejemplo anterior, la regla original del elemento martillo se ha conservado intacta, pero hemos añadido una nueva regla para aplicar efectos de transformación usando la **pseudo-clase :hover**.

El resultado obtenido es que cada vez que el puntero del **ratón pase sobre este elemento**, la propiedad transform lo invertirá sobre el eje vertical, y cuando el puntero salga del elemento, volverá a rotar para volver a su posición original.

Esta técnica ya se utilizaba con versiones anteriores de CSS para crear rollovers, por ejemplo, pero con la nueva propiedad transform se pueden conseguir efectos más logrados.

4. CSS3

Transiciones (transition)

Funcionan como pequeñas animaciones que siguen las mismas características que las transformaciones (cambio de posición, tamaño, rotación,…), aunque con la capacidad de poder **especificar un tiempo** para que se produzca dicha transformación, y así poder utilizarla como una animación sencilla.

Aunque no es obligatorio, tenemos la posibilidad de especificar el **tipo** de movimiento y si debe existir algún **retardo** para que empiece la animación.

Además, es interesante poder especificar todos estos parámetros en una sola línea.

```
transition:[propiedad a modificar] [Duración] [Tipo de animación] [Retardo]; transition: width 2s ease-out 0.5s;
```

4. CSS3

Transiciones (transition)

- [Propiedades a modificar]:

all	bottom	margin	text-shadow
background-	left	opacity	vertical-align
color	right	word-spacing	visibility
border	box-shadow	letter-spacing	z-index
border-radius	width	fill	
color	height	padding	
top	line-height	stroke	

- [Duración en segundos]: Por ejemplo 3s
- [Tipo de animación]: Aquí se indica un tipo de `easing`, es decir, si la animación empezará suavemente e irá **acelerándose** progresivamente, si pasará todo lo contrario o si el movimiento tendrá una **velocidad uniforme** durante todo su recorrido entre otras. Algunas de las posibilidades son las siguientes:

4. CSS3

Transiciones (transition)

- `[Tipos de animación]`
  - `linear`: La velocidad es uniforme.
  - `ease`: La velocidad se acelera al inicio, luego se retarda un poco y se acelera al final de nuevo.
  - `ease-in`: La animación empieza lentamente y va aumentando progresivamente.
  - `ease-out`: La animación empieza muy rápida y va descendiendo progresivamente.
  - `ease-in-out`: La animación empieza y acaba lentamente, y es en la parte central donde va más rápida.

4. CSS3

Transiciones (transition)

- [Retardo]: Tiempo (en segundos) que el navegador esperará antes de poner en marcha la animación. Por ejemplo 1s
- Al igual que ocurría con las transformaciones, para que `transition` funcione en todos los navegadores deberemos repetir la propiedad anteponiendo los prefijos de cada uno de los navegadores:
  - `-webkit-` (para Chrome y Safari)
  - `-moz-` (para Firefox)
  - `-o-` (para Opera)
  - `-ms-` (para Internet Explorer)

4. CSS3

Transiciones con una única propiedad

Ejemplo de transición que cambia el color de fondo con una duración de 1s, tipo ease-in-out y retardo de 0,5s.

```
#martillo {
   position:relative;
   width:110px; height:190px; padding:16px;
   border:1px dotted gray;
   border-radius:9px; float:left;
   transition: background-color 1s ease-in-out 0.5s;
}
```

A continuación, utilizaremos la pseudo-clase :hover para cambiar el color de fondo del elemento:

```
#martillo:hover { background-color: black; }
```

El efecto obtenido será que al pasar con el ratón sobre el martillo se producirá la transición.

4. CSS3

Transiciones con propiedades CSS3

Cuando realizamos transiciones que requieren el uso de **prefijos**, estos debemos indicarlos, tanto en la **propiedad transition** como en la **propiedad a modificar**.

```
#martillo
{
  position:relative;
  width:110px;
  height:190px;
  padding:16px;
  border:1px dotted gray;
  border-radius:9px;
  float:left;
  transition: transform 1s ease-in-out 0.5s;
  -moz-transition: -moz-transform 1s ease-in-out 0.5s;
  -webkit-transition: -webkit-transform 1s ease-in-out 0.5s;
  -o-transition: -o-transform 1s ease-in-out 0.5s;
  -ms-transition: -ms-transform 1s ease-in-out 0.5s;
}
```

```
#martillo:hover {
  transform: scale(1,-1);
  -moz-transform:scale(1,-1);
  -webkit-transform:scale(1,-1);
  -o-transform:scale(1,-1);
  -ms-transform:scale(1,-1);
}
```