Transformaciones

Las transformaciones, me permite rotar, escalar y trasladar los elementos de mi **HTML**, para luego hacer un buen complemento con animaciones y transiciones de **CSS3**. Veamos un ejemplo de sintaxis para comprender mejor las **transformaciones**,

```
<style>
div {
   transform: tipoTransformacion(parametro);
    }


                                    </style>
```

Rotate

Rotamos el objeto que puede ser de línea o de bloque tanto a favor como en el sentido contrario a las agujas del reloj, es decir que los valores posibles son, **0 a 360deg** o en el sentido contrario de las agujas del reloj **0 a -360deg. Veamos un ejemplo,**

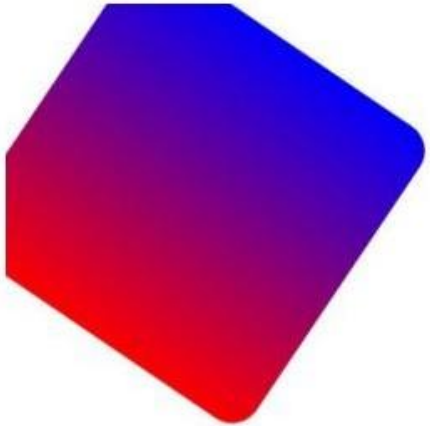
```
<style>
div {
   width: 200px;
   height: 200px;
   border-radius: 20px;
   padding: 20px;
   background: linear-gradient(blue,red);
   transform: rotate(34deg);
    }

                              </style>



    <div>
     mi contenedor

       </div>
```



Transform-origin

Se pueden utilizar, medidas de **longitud , px, em, %** (en referencia al alto o al ancho del elemento que estoy transformando), así como palabras (**center, top,bottom,etc**), es decir aquellos valores que por ejemplo utilizamos en la propiedad **background-position.**

```
<style>
div {
   width: 200px;
   height: 200px;
   border-radius: 20px;
   padding: 20px;
   background: linear-gradient(blue,red);
   transform: rotate(34deg);
   transform-origin: 20px top;
    }

                                </style>
```

Rotación 3D

La rotación **3D** va desde **0 a 180deg 0 a -180deg,** y se puede realizar tanto con **x, y así como z,** como se ve en las imágenes debajo,

Rotación 3D

El ejemplo del **código anterior** se verá de la siguiente forma,

También se puede trabajar con **shorthand o declaración de una sóla línea,**

```
div {
    width: 200px;
    height: 200px;
    border-radius: 20px;
    padding: 20px;
    background: linear-gradient(blue,red);
    transform: rotateY(160deg) rotateX(-34deg);


     }
```

Rotación: valores negativos

También podemos **trabajar con valores negativos** como en el ejemplo anterior. Por supuesto también podemos sumar una **rotación en 2D** veamos un ejemplo,

```
<style>
div {
    width: 200px;
    height: 200px;
    border-radius: 20px;
    padding: 20px;
    background: linear-gradient(blue,red);
    transform: rotate(34deg) rotateY(-30deg);

    }


                                </style>
```

Perspective

La propiedad **perspective** se aplica al **padre del elemento** que contiene la **rotación en 3D,** veamos un ejemplo,

```
<padre>

<hijo> </hijo> <!--elemento es el que va a tener la transformación en 3D-->

</padre> <!--elemento que va a tener la perspectiva-->
```

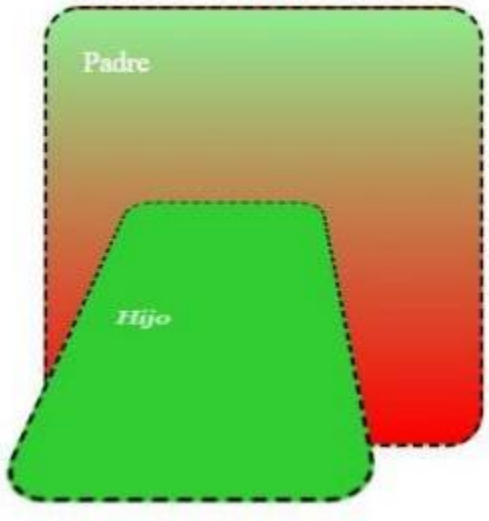
```
<style>
div {
   width: 200px;
   height: 200px;
   border-radius: 20px;
   padding: 20px;
   background: linear-gradient(lightgreen,red);
   color: white;
   border: 2px dashed black;    }
#padre {   margin-left: 100px; perspective: 200px;
   perspective-origin: center bottom;}
 #hijo { background: limegreen; width: 100px;  transform: rotateX(34deg); line-height: 230px;}

                              </style>


<div id="padre">Padre  <div id="hijo">  Hijo </div> </div>
```

Perspective

El ejemplo **anterior** se verá en nuestro **navegador** de la siguiente forma,



Scale

Me permite **escalar un elemento** tanto haciéndolo más **pequeño o más grande**. Veamos un ejemplo,

```
<style>
div {
   width: 200px;
   height: 200px;
   border-radius: 20px;
   padding: 20px;
   background: linear-gradient(lightgreen,red);
   color: white;
   border: 2px dashed black;
   transform: scale(2);    }

                              </style>


<div>contenido</div>
```

Scale

La **alternativas para trabajar** varían, veamos todas las posibles,

```
transform: scale(width,height)
/*escalamos tanto el ancho como el alto*/
transform: scaleX(n)
/*ancho del elemento*/
transform: scaleY(n)
/*alto del elemento*/
```

Scale

Veamos un ejemplo para trabajar con **scale,**

```
<style>
div {
    width: 200px;
    height: 200px;
    border-radius: 20px;
    padding: 20px;
    background: linear-gradient(lightgreen,red);
    color: white;
    border: 2px dashed black;
    transform: scale(0.5, 2);    }


                                </style>



<div>contenido</div>
```

Translate

La diferencia **entre trasladar** un elemento en una animación en una **transición** con translate a hacerlo con **position** , es que es mucho **más fluido** el movimiento. Te permite **trasladar un elemento** a través de las **transformaciones**, las posibles variantes son,

```
transform: translate(x,y);
transform: translateX();
transform: translateY();
```

```
<!DOCTYPE html>
<head>
<style>
div {
   width: 200px;
   height: 200px;
   border-radius: 20px;
   padding: 20px;
   background: linear-gradient(lightgreen,red);
   color: white;
   border: 2px dashed black;
   transform: translate(100px, 200px);    }


                                    </style>
```

Skew

Esto sería lo que por ejemplo en **PHOTOSHOP** llamamos sesgar, las posibles variantes son:

```
<style>
div {
   width: 200px;
   height: 200px;
   border-radius: 20px;
   padding: 20px;
   background: linear-gradient(lightgreen,red);
   color: white;
   border: 2px dashed black;
   transform: skewX(20deg)    }

                              </style>

<div> transformaciones </div>
```

```
transform: skew(x,y)
transform: skewX()
transform: skewY()
```

Skew

No tiene sentido trabajar ni con 0 ni con 180deg porque ese sería el valor predeterminado, el elemento no tiene cambios es decir no está sesgado, la idea es siempre trabajar con valores entre estos dos. En todos los casos anteriores, es decir en el **skew** y en el translate podemos hacerlo también sobre el eje Z, **por ejemplo skewZ() o translateZ(),**