

Creando tus primeras animaciones con SVG



Por:
Jean Paul Yepes

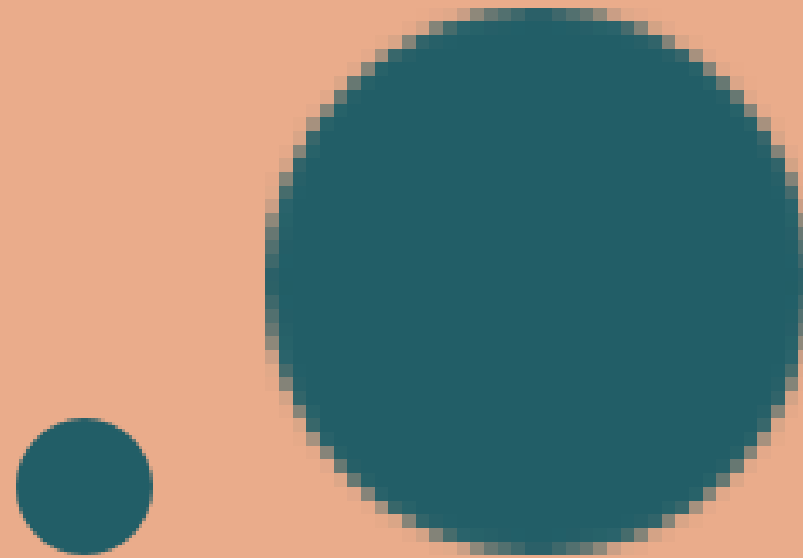
¿Qué son los SVG?



Son el estandar de W3C para los vectores en la web

Comparación SVG vs JPG

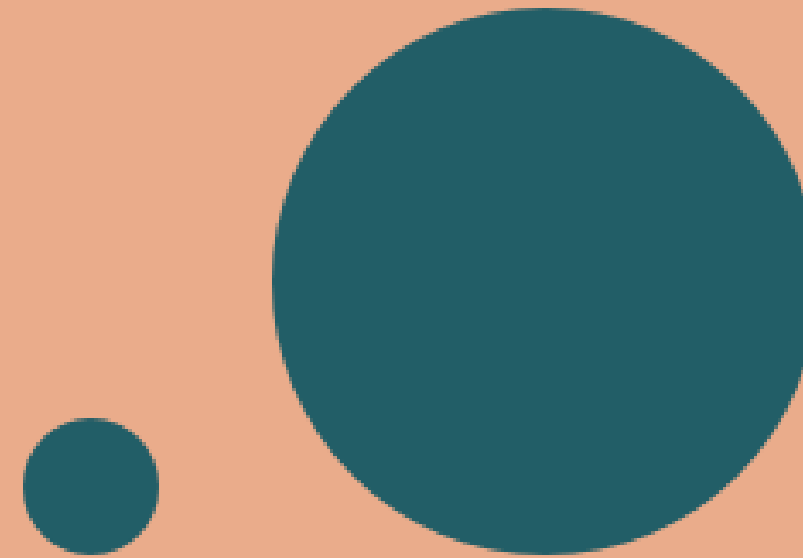
Raster



1x

4x

Vector



1x

4x

Los SVG se pueden redimensionar sin perder calidad

Comparación SVG vs JPG

SVG



JPG



Pueden ser transparentes

Comparación SVG vs JPG

SVG



JPG



Menor tamaño de archivo

Comparación SVG vs JPG

SVG



JPG



Se pueden animar partes dentro del SVG

Cómo luce un SVG

```

<svg width="737" height="741" viewBox="0 0 737 741" fill="none">
  <g id="SmileFace">
    <rect width="737" height="741" fill="white" />
    <circle id="Face" cx="368" cy="371" r="213" fill="#FBB040" />
    <g id="eyesAndMouth">
      <circle id="Ellipse 32" cx="294" cy="320" r="34" fill="#2F2E41" />
      <circle id="Ellipse 33" cx="443" cy="320" r="34" fill="#2F2E41" />
      <circle id="Ellipse 34" cx="368" cy="459" r="64" fill="#2F2E41" />
    </g>
  </g>
</svg>

```

Cómo luce un SVG



Trayectorias en SVG

M = moveto

L = lineto

H = horizontal lineto

V = vertical lineto

C = curveto

S = smooth curveto

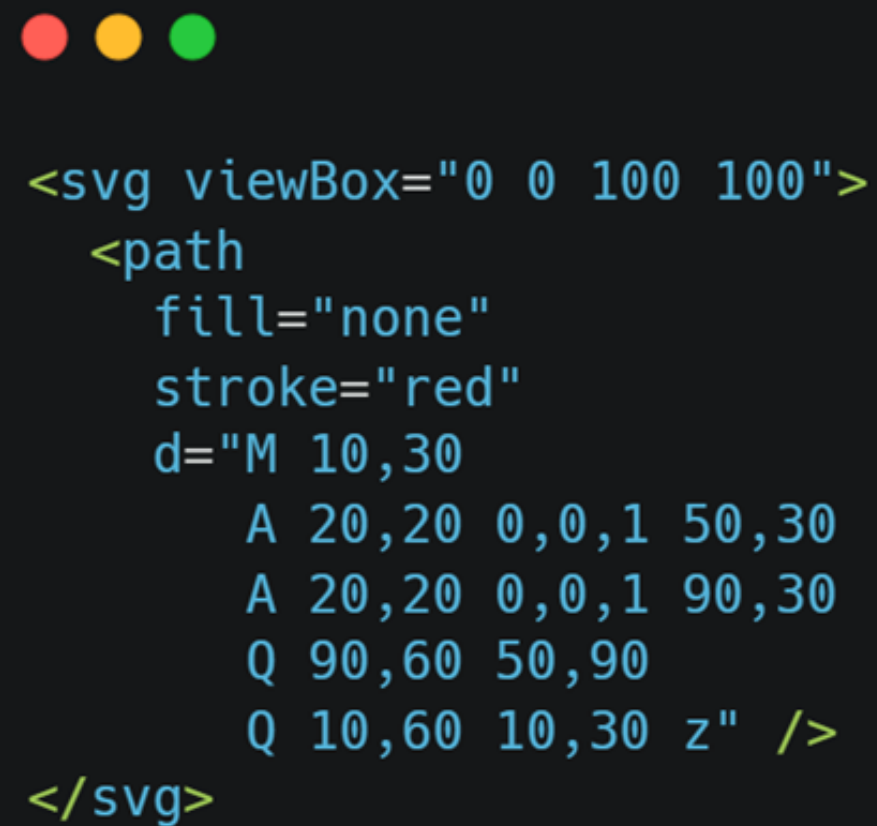
Q = quadratic Bézier curve

T = smooth quadratic Bézier curveto

A = elliptical Arc

Z = closepath

Trayectorias en SVG



```
<svg viewBox="0 0 100 100">  
  <path  
    fill="none"  
    stroke="red"  
    d="M 10,30  
      A 20,20 0,0,1 50,30  
      A 20,20 0,0,1 90,30  
      Q 90,60 50,90  
      Q 10,60 10,30 z" />  
</svg>
```

Trayectorias en SVG





Conceptos de animación

Learning the basics

Linea de tiempo

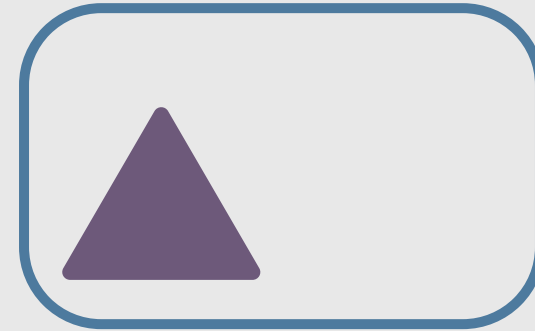


Linea de tiempo

Es una representación visual para ubicar en el tiempo los cambios que van a suceder en la animación

Key Frames

Move X



Move Y



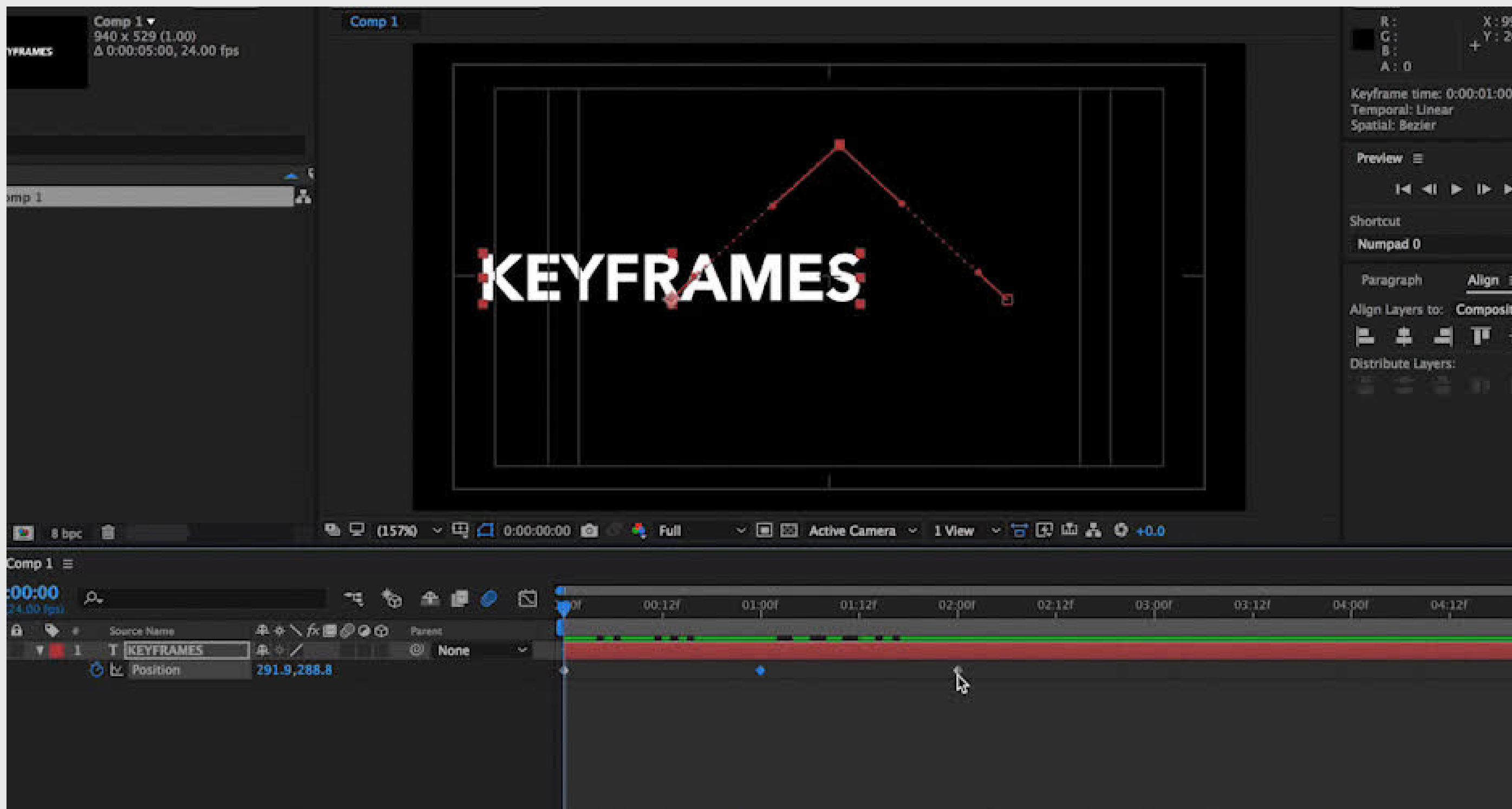
Rotate



Key Frames

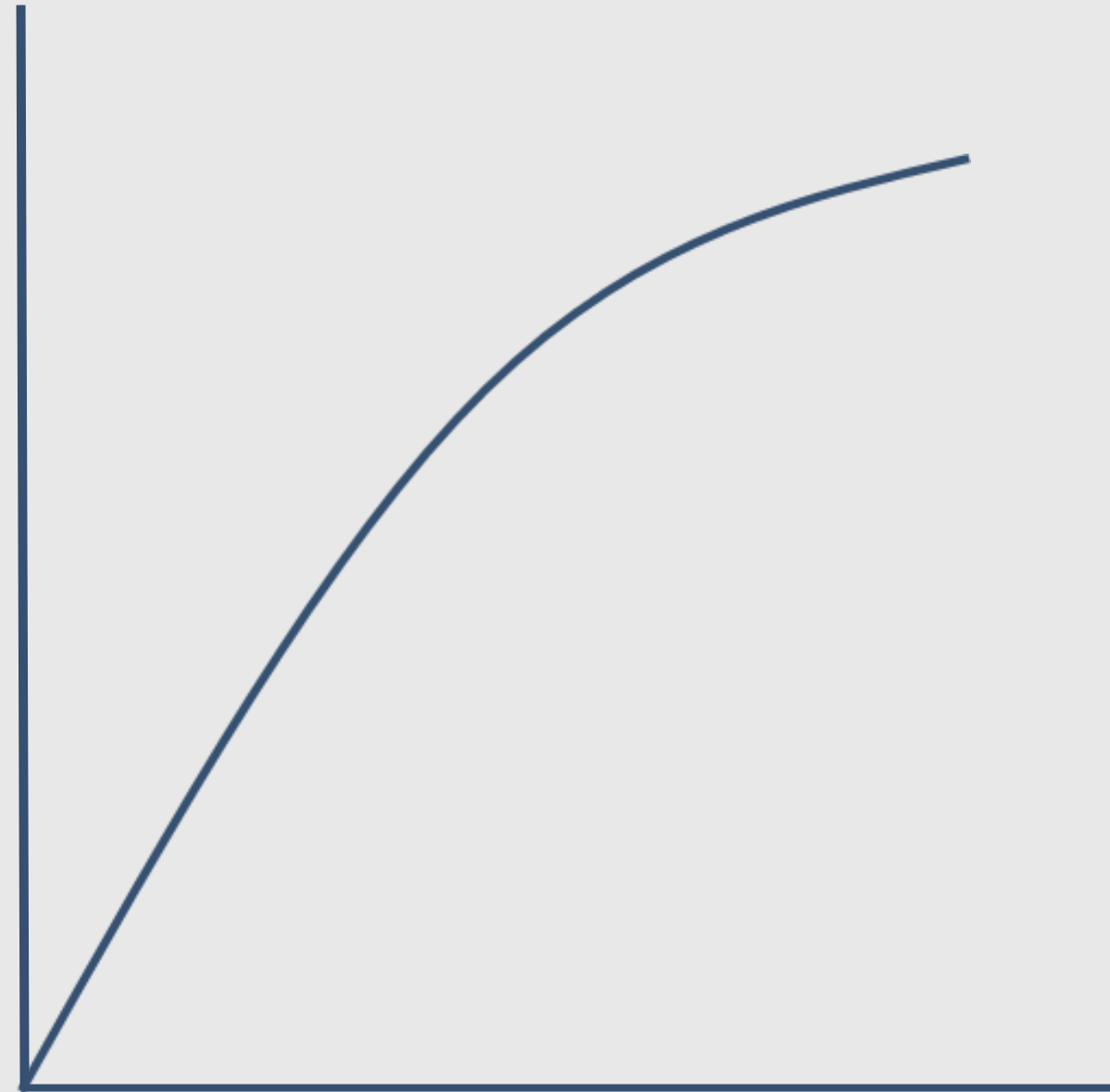
Es un cambio de una propiedad que se puede animar estableciendo duración y el momento de inicio

Key Frames



Curva aceleración

Progresión

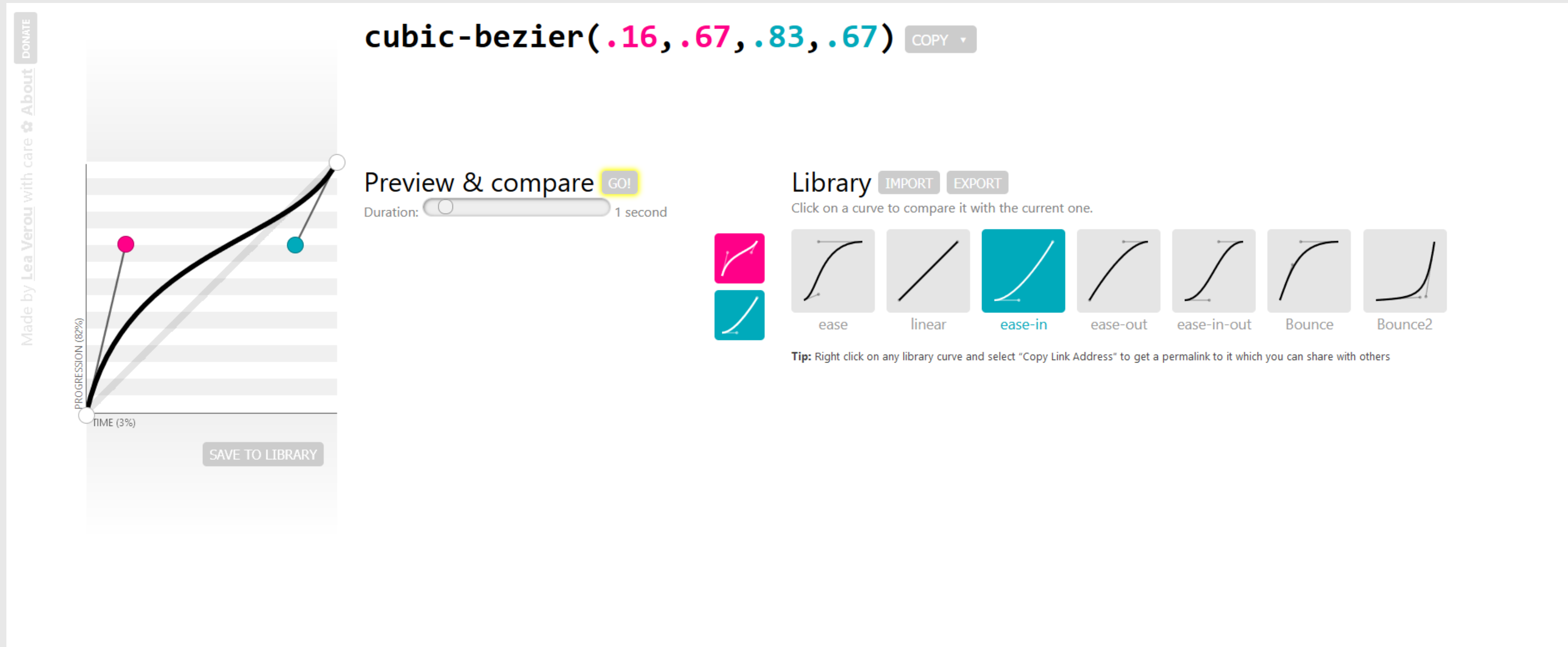


Tiempo

Curva aceleración

Es la progresión de una animación, aumentando y reduciendo la velocidad de la animación en ciertos puntos

Ejemplo Curva de aceleración





Animaciones en CSS

Viendo los detalles de implementación

Existen dos formas de animar en CSS



Solo veremos keyframes por simplicidad

Para usar las animaciones en un elemento tenemos las siguientes propiedades



animation-name



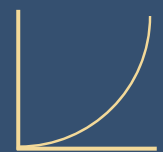
animation-iteration-count



animation-duration



animation-fill-mode



animation-timing-function



animation-play-state



animation-direction



animation-delay

Veamos algunas opciones de las
propiedades

animation-direction

animation-direction

The `animation-direction` [CSS](#) property sets whether an animation should play forward, backward, or alternate back and forth between playing the sequence forward and backward.

Try it

CSS Demo: animation-direction

RESET


animation-direction: `normal`;

animation-direction: `reverse`;

animation-direction: `alternate`;

animation-direction: `alternate-reverse`;

<div>



Pause

animation-fill-mode

animation-fill-mode

The `animation-fill-mode` [CSS](#) property sets how a CSS animation applies styles to its target before and after its execution.

Try it

CSS Demo: animation-fill-mode

RESET

```
animation-fill-mode: none;  
animation-delay: 1s;
```

```
animation-fill-mode: forwards;  
animation-delay: 1s;
```

```
animation-fill-mode: backwards;  
animation-delay: 1s;
```

```
animation-fill-mode: both;  
animation-delay: 1s;
```

Animation **finished**

Select a mode to
start!

animation-play-state

animation-play-state

The `animation-play-state` [CSS](#) property sets whether an animation is running or paused.

Try it

CSS Demo: animation-play-state

RESET

```
animation-play-state: paused;
```

```
animation-play-state: running;
```



Y es útil repasar las funciones que podemos
usar con transform en CSS



translate()



scale()



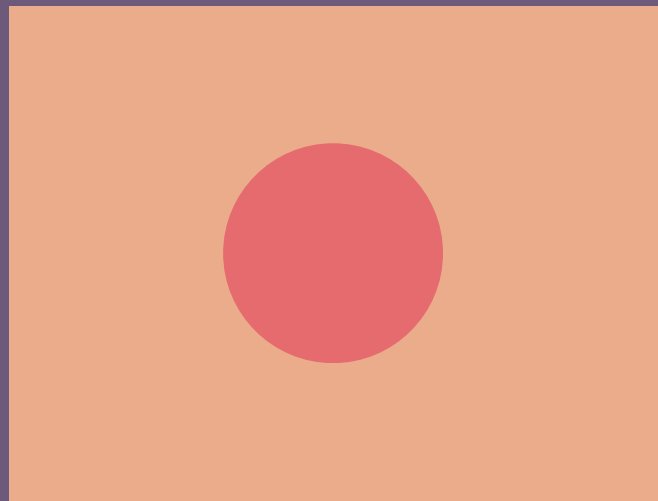
rotate()



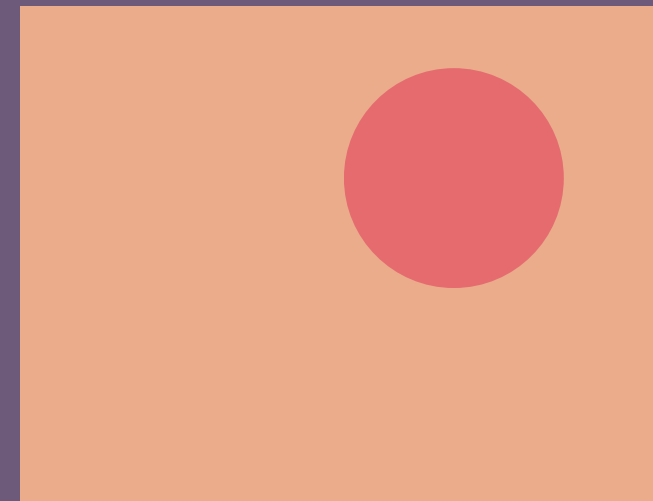
skew()

Translate

Permite mover elementos de su punto original



Original



Con transform

Translate

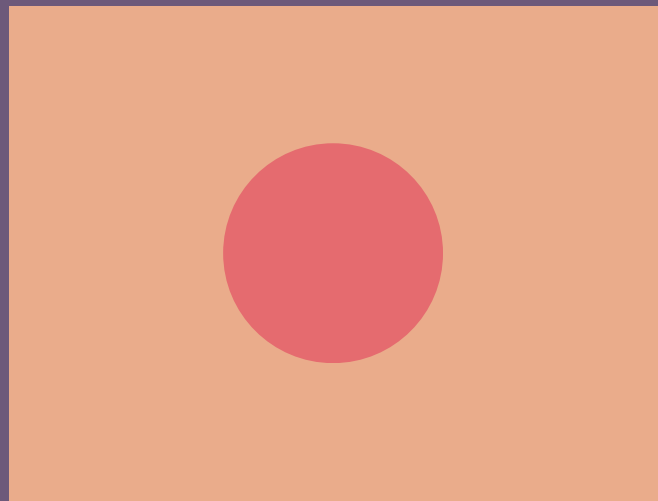
Diferentes opciones para translate

translateX(x) translateY(Y)

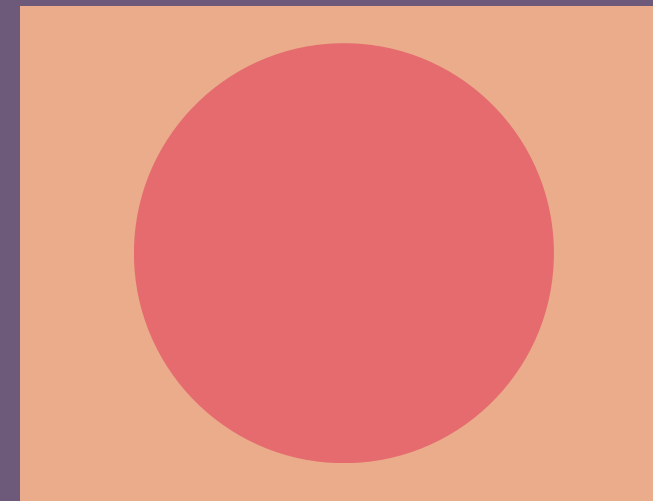
translate(x, y)

Scale

Hace los elementos mas grandes por un número dado



Original



Con scale

Scale

Diferentes opciones para translate

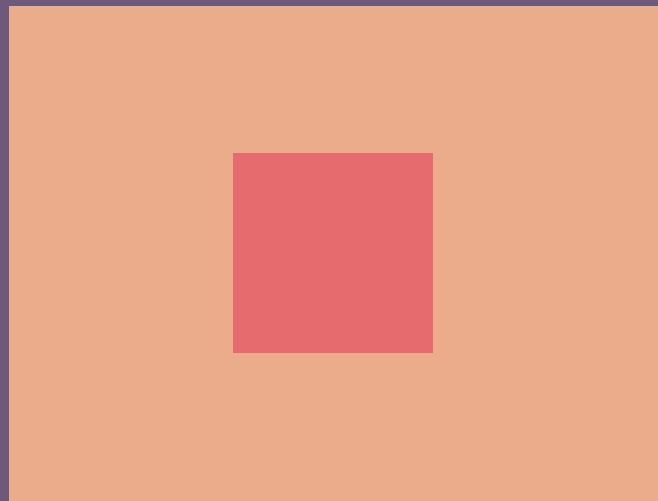
`scaleX(x)`

`scaleY(Y)`

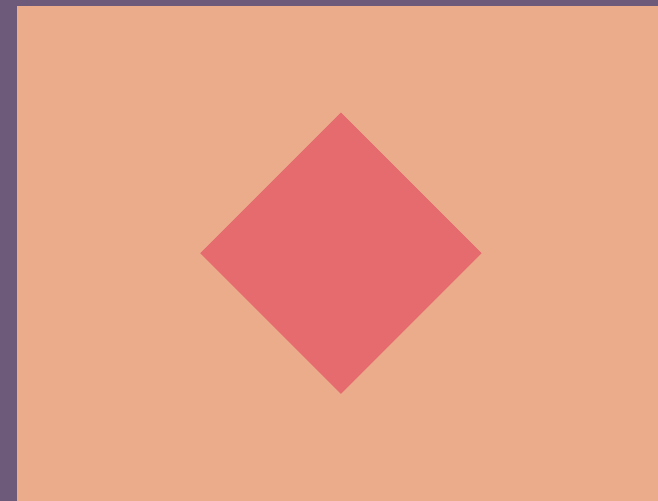
`scale(x, y)`

Rotate

Rota los elementos desde un punto de origen



Original



Con transform

Rotate

Diferentes opciones para rotate

rotateX(x)

rotateY(Y)

rotate(x, y)

**Otro elemento a tener en cuenta es
transform-origin**

Dependiendo del origen de las transformaciones se va a obtener resultados diferentes

References > CSS > transform-originEnglish (US)

Filter

top

touch-action

▼ transform-*

transform

transform-box

transform-origin

transform-style

► transition-*

translate

unicode-bidi

user-modify

user-select

vertical-align

► view-*

visibility

white-space

white-space-collapse

widows


width


will-change


A demonstration of various transform values


This example shows the effect of choosing different `transform-origin` values for a variety of transformation functions.

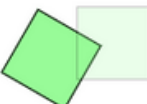
Play

`transform: none;`

`transform: rotate(30deg);`

`transform: rotate(30deg);`
`transform-origin: 0 0;`

`transform: rotate(30deg);`
`transform-origin: 100% 100%;`

`transform: rotate(30deg);`
`transform-origin: -1em -3em;`

In this article

Try it

Syntax

Formal definition

Formal syntax

Examples

Specifications

Browser compatibility

See also

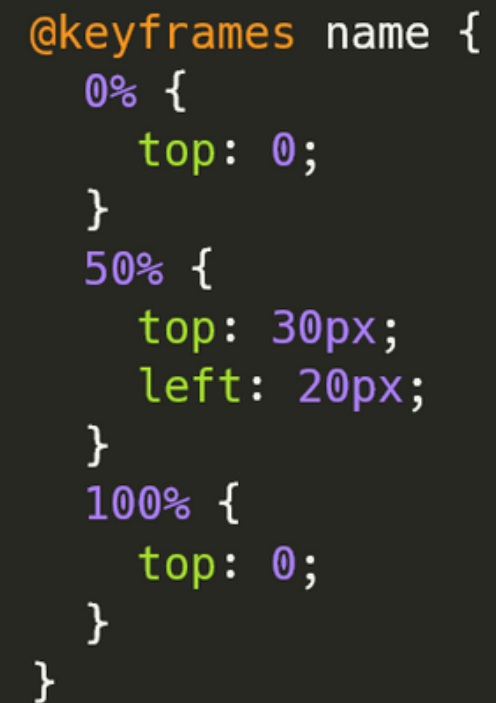
FrontendMastersEXPLORATIONSALE\$100 OFF

Find your perfect course or learning path — \$100 OFF Frontend Masters

Mozilla ads

Don't want to see ads?

Sintaxis keyframes



```
@keyframes name {  
  0% {  
    top: 0;  
  }  
  50% {  
    top: 30px;  
    left: 20px;  
  }  
  100% {  
    top: 0;  
  }  
}
```


Sintaxis uso animacion

```
.circle {  
  /** Estilos para crear el circulo **/  
  width: 200px;  
  height: 200px;  
  border-radius: 50% 50%;  
  background-color: salmon;  
  margin: 0 auto;  
  
  /** Estilos para animaciones **/  
  animation-name: moveX;  
  animation-iteration-count: infinite;  
  animation-duration: 2s;  
  animation-direction: alternate;  
}  
  
/** Estilos para crear animación **/  
@keyframes moveX {  
  from {  
    transform: translateX(-200px);  
  }  
  to {  
    transform: translateX(200px);  
  }  
}
```

Sintaxis uso animacion

```
.boxAnimated {  
  animation-name: skewAnimation;  
  animation-duration: 3s;  
  animation-iteration-count: infinite;  
}  
  
@keyframes skewAnimation {  
  40% {  
    transform: skew(10deg);  
  }  
  80% {  
    transform: skew(0);  
  }  
}
```


Ahora vamos a practicar



The screenshot shows a GitHub repository interface for 'crea-tu-primer-svg'. The repository is public and has 1 branch (main) and 0 tags. The README file is selected, showing its content. The README describes a workshop for creating SVG animations, specifically focusing on rotating stars. It includes a list of requirements for the animation and mentions the IDs of the stars to be animated.

crea-tu-primer-svg

En este workshop haremos tres ejercicios para empezar a desarrollar tus habilidades con animaciones SVG. Vamos a trabajar sobre tres animaciones: estrellas, tarjetas de crédito y un avión.

¡Así que vamos con toda crear tus primeras animaciones con SVG!

Ejercicio 1 - Estrella

Para este primer ejercicio vamos a hacer rotar unas estrellas con bordes redondeados.

La secuencia de animación debe cumplir con los siguientes requerimientos:

- Ambas estrellas deben rotar 360° en el tiempo que dura la animación.
- La animación debe durar 5 segundos.
- Las estrellas deben rotar desde el centro de la imagen.
- La función de aceleración puede ser la predeterminada.

Adicionalmente para completar la tarea: la primera estrella tiene el id `estrellaAzul` y la segunda estrella tiene el id



Sígueme

@JeanPaulYps



yjeanpaul@gmail.com