MANUAL DE USO ANIME.JS

Hecho por: David Zapata González
Centro: IES Emili Darder
Año y curso: ASIX 2 2021

ÍNDICE

Descripción de la librería

Anime.js es una biblioteca de animación JavaScript. Funciona con propiedades CSS, SVG, atributos DOM y objetos JavaScript.

En este manual veremos cómo usar la librería de animaciones para JavaScript; en general es una librería que nos recordará bastante a cómo trabajan las animaciones y transiciones en CSS.

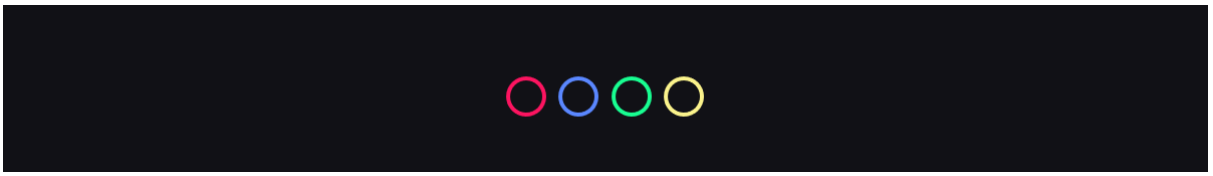
La ventaja que tenemos es la posibilidad de crear animaciones dinámicas que pueden depender de muchos escenarios que podemos manejar fácilmente con JavaScript y sin la necesidad de agregar/remover clases cada cierto tiempo para variar el comportamiento de cualquier elemento animable; además de ser más extensible lo que nos trae más posibilidades de uso.

¿Por qué utilizar la animación?

CSS, es capaz de proporcionar animaciones y transiciones para agregar movimiento dentro de una web, sin embargo, JavaScript sigue siendo un gran reproductor para esto.

Podemos encontrar diferentes tipos de animación para nuestra web:

- **Animación tranquila:** La animación del principio se ejecuta hasta el final de la lógica sin tomar una decisión.
- **Animación de estado:** Estas són las que se utilizan con la barra de navegación, altera el estado visual de la página mediante el desplazamiento que hacemos con esta.
- **Animación dinámica:** Este tipo de animación cambia dependiendo de los factores, ya sea como la interacción del usuario, eventos DOM, animación de otros elementos del mismo documento, etc. No se puede crear un estado dinámico con solo animación CSS, en este caso, JavaScript es el que mejor nos funcionará para esto...



Además de la animación dinámica, si existe alguna de las siguientes situaciones, podemos ayudar a JS con su animación:

- Necesidad de escalonar más las animaciones a los elementos anteriores
- Los gráficos SVG son animados
- Si el proyecto necesita ser compatible con navegadores antiguos donde las animaciones CSS no están disponibles, se puede recurrir a una potente solución de JavaScript que hará que las animaciones estén más disponibles.

Al final… Anime JS es una biblioteca de animación de JavaScript ligera y flexible. Funciona con CSS, transformaciones individuales, SVG, atributos DOM y objetos JS que nos permite trabajar de una manera muy cómoda y nos permite una gran visibilidad dinámica dentro de nuestra web para el usuario.

Es importante el uso de animaciones además de que esta librería, es compatible con los siguientes navegadores:

- Chrome
- Safari
- ópera
- Firefox
- Navegador de Internet 10+

Esto nos deja vía libre para trabajar libremente con animaciones ya que se encuentran adaptadas a la gran mayoría de navegadores.

Cómo realizar la instalación

Mediante npm

Para instalar dicha librería, podemos realizar la instalación vía npm.
Entraremos dentro de nuestro terminal y escribiremos:

```
$ npm install animejs --save
```

Mediante CDN

Añadiremos a nuestro head un link con:

https://cdnjs.cloudflare.com/ajax/libs/animejs/3.2.1/anime.min.js

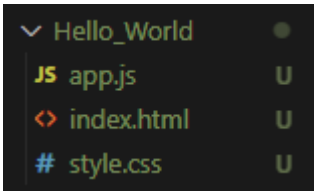
Mediante GitHub

https://github.com/juliangarnier/anime

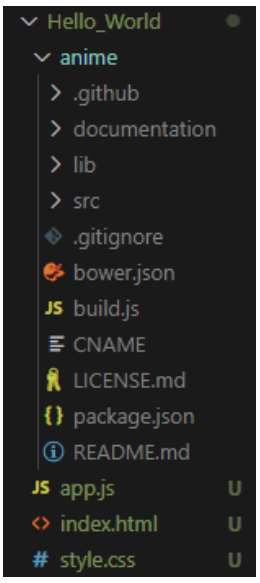
Hello World

Procederemos a realizar una pequeña práctica para entender brevemente conceptos básicos para entrar en razón con esta librería

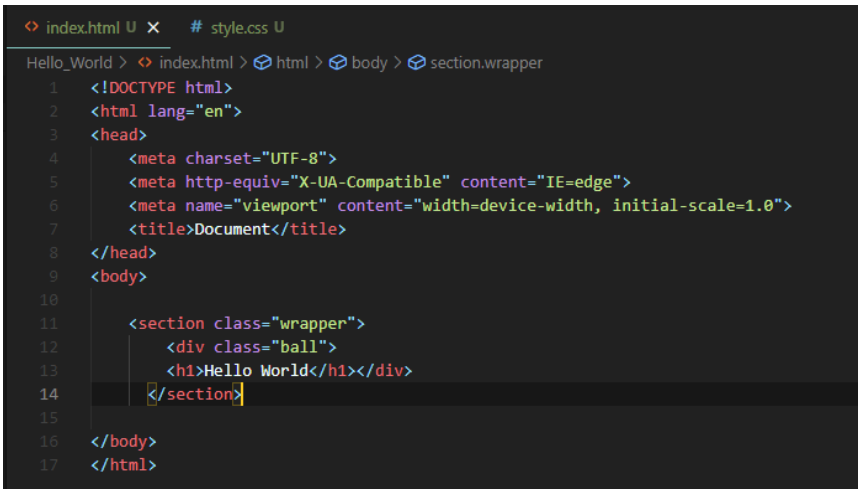
1. Vamos a crear nuestro proyecto llamado Hello-World con una estructura básica y rápida



2. Descargamos el archivo anime dentro de la carpeta ánime vista anteriormente en github y la añadiremos a nuestro documento.

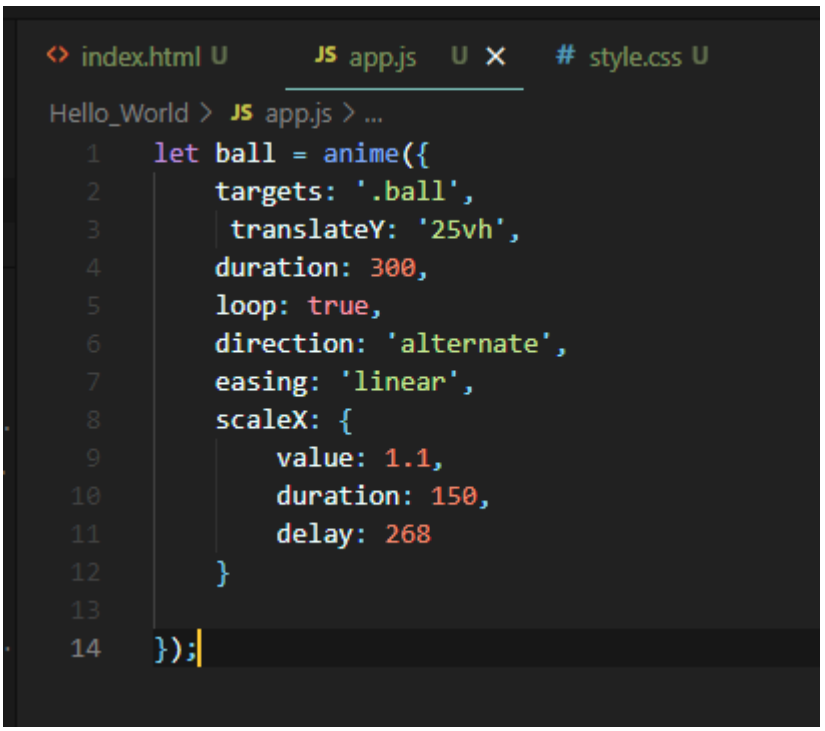


3. Comenzaremos creando nuestro fichero de html con un section con la clase wrapper y un div con clase ball, estas dos clases vienen implementadas en el paquete de anime.



4. Seguiremos con un css básico

5. Pondremos esto en nuestro JS (con anime({animations}) indicamos animaciones)



- Con translateY le indicamos el movimiento de la bola en Y
- duration será la duración de la bola en tocar el suelo
- loop indicará si se quiere repetir el movimiento o no
- direction será la dirección en la que irá la bola
- dentro de scaleX indicaremos qué tan ancha se vuelve la pelota cuando cae, la duración y su delay

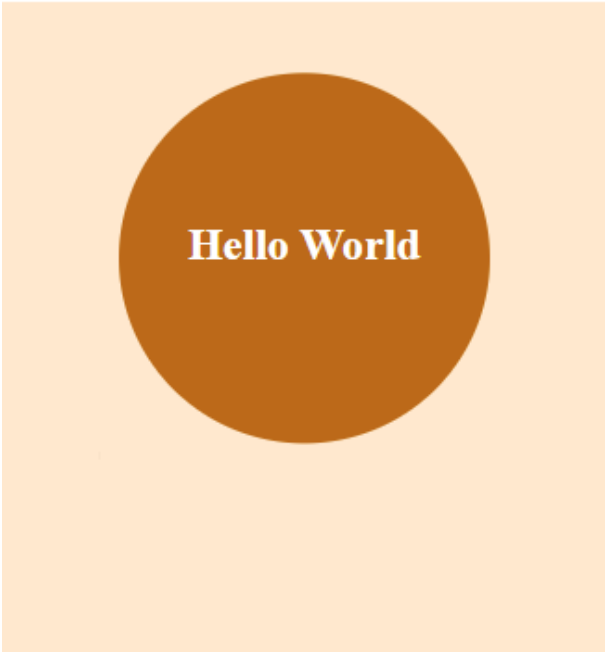
6. Por último nos encargaremos de linkear el html con el css y js

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
    <link rel="stylesheet" href="style.css">
</head>
<body>

    <section class="wrapper">
        <div class="ball">
        <h1>Hello World</h1></div>
      </section>


      <script src="anime/lib/anime.min.js"></script>
      <script src="app.js"></script>
```

7. Abriremos el navegador y ya veríamos nuestro primer Hello-World con Anime JS!!



(no se ve el efecto por que es una imagen, aquí dejo un link para verlo)
https://codepen.io/mariphkhakadze/pen/VwLOZNE

Documentación

5.1 - Targets

5.1.1 CSS Selector: Puede ser cualquier selector de CSS.

Los pseudo elementos no se pueden segmentar mediante JavaScript.

TYPE	DEFAULT	EXAMPLE
String	null	targets: '.item'

Ejemplo de código:

```
anime({
  targets: '.css-selector-demo .el',
  translateX: 250
});
```

5.1.2 NODELIST: Puede ser cualquier nodo DOM o NodeList.

ESCRIBE	DEFECTO	EJEMPLO
Nodo DOM	null	targets: el.querySelector('.item')
NodeList	null	targets: el.querySelectorAll('.item')

Ejemplo de código:

```
var elements = document.querySelectorAll('.dom-node-demo .el');

anime({
  targets: elements,
  translateX: 270
});
```

5.1.3 JS OBJECT: Un objeto JavaScript con al menos una propiedad que contiene un valor numérico.

ESCRIBE	DEFECTO	EJEMPLO
Objeto	null	targets: myObjectProp

Ejemplo de código:

```
var logEl = document.querySelector('.battery-log');

var battery = {
  charged: '0%',
  cycles: 120
}

anime({
  targets: battery,
  charged: '100%',
  cycles: 130,
  round: 1,
  easing: 'linear',
  update: function() {
    logEl.innerHTML = JSON.stringify(battery);
  }
});
```

5.1.4 Array: Un array que contiene varios objetos.
Acepta tipos mixtos. P.ej['.el', domNode, jsObject]

Ejemplo de código:

```
var el = document.querySelector('.mixed-array-demo .el-01');

anime({
     targets:  [el,  '.mixed-array-demo  .el-02',  '.mixed-array-demo
.el-03'],
  translateX: 250
});
```

5.2 Propiedades

5.2.1 Propiedades CSS: Se puede animar cualquier propiedad CSS.

La mayoría de las propiedades de CSS provocarán cambios de diseño o volverán a pintar, y darán como resultado una animación entrecortada. Priorice la opacidad y las transformaciones CSS tanto como sea posible.

EJEMPLO	VALOR
opacity	.5
left	'100px'

Ejemplo de código:

```
anime({
  targets: '.css-prop-demo .el',
  left: '240px',
  backgroundColor: '#FFF',
  borderRadius: ['0%', '50%'],
  easing: 'easeInOutQuad'
});
```

5.2.2 Transformaciones CSS: Transforma las propiedades individualmente.

Es posible especificar tiempos diferentes para cada propiedad de transformación.

PROPIEDADES VÁLIDAS	UNIDAD PREDETERMINADA
'translateX'	'px'
'translateY'	'px'
'translateZ'	'px'
'rotate'	'deg'
'rotateX'	'deg'
'rotateY'	'deg'
'rotateZ'	'deg'
'scale'	–
'scaleX'	–
'scaleY'	–
'scaleZ'	–
'skew'	'deg'
'skewX'	'deg'
'skewY'	'deg'
'perspective'	'px'

Ejemplo de codigo:

```
anime({
  targets: '.css-transforms-demo .el',
  translateX: 250,
  scale: 2,
  rotate: '1turn'
});
```

5.2.3 Propiedades del Objeto: Se puede animar cualquier propiedad de Objeto que contenga un valor numérico.

EJEMPLO	VALOR
prop1	50
'prop2'	'100%'

Ejemplo de código:

```
var objPropLogEl = document.querySelector('.js-object-log');

var myObject = {
  prop1: 0,
  prop2: '0%'
}

anime({
  targets: myObject,
  prop1: 50,
  prop2: '100%',
  easing: 'linear',
  round: 1,
  update: function() {
    objPropLogEl.innerHTML = JSON.stringify(myObject);
  }
});
```

5.2.4 Atributos DOM: Se puede animar cualquier atributo DOM que contenga un valor numérico.

EJEMPLO	VALOR
value	1000
volume	0
data-custom	'3 dogs'

Ejemplo de código:

```
anime({
  targets: '.dom-attribute-demo input',
  value: [0, 1000],
  round: 1,
  easing: 'easeInOutExpo'
});
```

5.2.5 Atributos SVG: Como cualquier otro atributo DOM , todos los atributos SVG que contienen al menos un valor numérico se pueden animar.

EJEMPLO	VALOR
points	'64 68.64 8.574 100 63.446 67.68 64 4 64.554 67.68 119.426 100'
scale	1
baseFrequency	0

Ejemplo de código:

```
anime({
        targets:     ['.svg-attributes-demo     polygon',     'feTurbulence',
'feDisplacementMap'],
  points: '64 128 8.574 96 8.574 32 64 0 119.426 32 119.426 96',
  baseFrequency: 0,
  scale: 1,
  loop: true,
  direction: 'alternate',
  easing: 'easeInOutExpo'
});
```

5.3 Parámetros de propiedad

5.3.1 Duración: Define la duración en milisegundos de la animación.

ESCRIBE	DEFECTO	EJEMPLO
Número	1000	3000
anime.stagger	Ver sección asombrosa	anime.stagger(150)
Función	Ver la sección de parámetros basados en funciones	(el, i) => i * 150

Ejemplo de código:

```
anime({
  targets: '.duration-demo .el',
  translateX: 250,
  duration: 3000
});
```

5.3.2 Delay: Define el retraso en milisegundos de la animación.

ESCRIBE	DEFECTO	EJEMPLO
Número	0	1000
anime.stagger	Ver sección asombrosa	anime.stagger(150)
Función	Ver la sección de parámetros basados en funciones	(el, i) => i * 150

Ejemplo de código:

```
anime({
  targets: '.delay-demo .el',
  translateX: 250,
  delay: 1000
});
```

5.3.3 End Delay: Agrega algo de tiempo extra en milisegundos al final de la animación.

ESCRIBE	DEFECTO	EJEMPLO
Número	`0`	`1000`
anime.stagger	Ver sección asombrosa	`anime.stagger(150)`
Función	Ver la sección de parámetros basados en funciones	`(el, i) => i * 150`

Ejemplo de código:

```
anime({
  targets: '.end-delay-demo .el',
  translateX: 250,
  endDelay: 1000,
  direction: 'alternate'
});
```

5.3.4 Easing: Define la función de sincronización de la animación.

ESCRIBE	DEFECTO	EJEMPLO
Cuerda	`'easeOutElastic(1, .5)'`	`easing: 'easeInOutQuad'`

Ejemplo de código:

```
anime({
  targets: '.easing-demo .el',
  translateX: 250,
  easing: 'easeInOutExpo'
});
```

5.3.5 Round: Redondea el valor a x decimales.

ESCRIBE	DEFECTO	EJEMPLO
Número	0	round: 10

Ejemplo de código:

```
var roundLogEl = document.querySelector('.round-log');

anime({
  targets: roundLogEl,
  innerHTML: [0, 10000],
  easing: 'linear',
  round: 10 // Will round the animated value to 1 decimal
});
```

5.3.6 Propiedades específicas de parámetros: Define parámetros específicos para cada propiedad de la animación usando un Objeto como valor. Otras propiedades que no se especifican en el objeto se heredan de la animación principal.

ESCRIBE	EJEMPLO

Objeto	rotate: {
	value: 360,
	duration: 1800,
	easing: 'easeInOutSine'
	}

Ejemplo de código:

```
anime({
  targets: '.specific-prop-params-demo .el',
  translateX: {
    value: 250,
    duration: 800
  },
  rotate: {
    value: 360,
    duration: 1800,
    easing: 'easeInOutSine'
  },
  scale: {
    value: 2,
    duration: 1600,
    delay: 800,
    easing: 'easeInOutQuart'
  },
  delay: 250 // All properties except 'scale' inherit 250ms delay
});
```

5.3.7 Parámetros basados en funciones: Obtienen diferentes valores para cada objetivo y propiedad de la animación.

Esta función acepta 3 argumentos:

ARGUMENTOS	INFOS
target	El elemento objetivo animado actual
index	El índice del elemento objetivo animado
targetsLength	El número total de objetivos animados.

Ejemplo de código:

```
anime({
  targets: '.function-based-params-demo .el',
  translateX: 270,
  direction: 'alternate',
  loop: true,
  delay: function(el, i, l) {
    return i * 100;
  },
  endDelay: function(el, i, l) {
    return (l - i) * 100;
  }
});
```

5.4 Parámetros de animación

5.4.1 Dirección: Define la dirección de la animación.

ACEPTA	INFOS
`'normal'`	El progreso de la animación va de 0 a 100%
`'reverse'`	El progreso de la animación va del 100% al 0%
`'alternate'`	El progreso de la animación va del 0% al 100% y luego vuelve al 0%

Ejemplo de código:

```
anime({
  targets: '.dir-normal',
  translateX: 250,
  easing: 'easeInOutSine'
});

anime({
  targets: '.dir-reverse',
  translateX: 250,
  direction: 'reverse',
  easing: 'easeInOutSine'
});

anime({
  targets: '.dir-alternate',
  translateX: 250,
  direction: 'alternate',
  easing: 'easeInOutSine'
});
```

5.4.2 Loop: Define el número de iteraciones de su animación.

ACEPTA	INFOS
Number	El número de iteraciones
true	Bucle indefinidamente

Ejemplo de código:

```
anime({
  targets: '.loop',
  translateX: 270,
  loop: 3,
  easing: 'easeInOutSine'
});

anime({
  targets: '.loop-infinity',
  translateX: 270,
  loop: true,
  easing: 'easeInOutSine'
});

anime({
  targets: '.loop-reverse',
  translateX: 270,
  loop: 3,
  direction: 'reverse',
  easing: 'easeInOutSine'
});

anime({
  targets: '.loop-reverse-infinity',
  translateX: 270,
  direction: 'reverse',
  loop: true,
  easing: 'easeInOutSine'
});
```

5.4.3 Autoplay: Define si la animación debe iniciarse automáticamente o no.

ACEPTA	INFOS
true	Inicia automáticamente la animación.
false	La animación está en pausa de forma predeterminada

Ejemplo de código:

```
anime({
  targets: '.autoplay-true',
  translateX: 250,
  autoplay: true,
  easing: 'easeInOutSine'
});

anime({
  targets: '.autoplay-false',
  translateX: 250,
  autoplay: false,
  easing: 'easeInOutSine'
});
```

5.5 Valores

5.5.1 Sin unidad: Si el valor original tiene una unidad, se agregará automáticamente al valor animado.

ESCRIBE	EJEMPLO
Número	translateX: 250

Ejemplo de código:

```
anime({
  targets: '.unitless-values-demo .el',
  translateX: 250, // -> '250px'
  rotate: 540 // -> '540deg'
});
```

5.5.2 Unidad específica: Obliga a la animación a utilizar una unidad determinada y convertirá automáticamente el valor objetivo inicial.
La precisión de conversión puede variar según la unidad utilizada.

TYPE	EXAMPLE
String	width: '100%'

Ejemplo de código:

```
anime({
  targets: '.specific-unit-values-demo .el',
  width: '100%', // -> from '28px' to '100%',
  easing: 'easeInOutQuad',
  direction: 'alternate',
  loop: true
});
```

5.5.3 Relative: Suma, resta o multiplica el valor original.

ACCEPTS	EFFECT	EXAMPLE
'+='	Add	'+=100'
'-='	Substract	'-=2turn'
'*='	Multiply	'*=10'

Ejemplo de código:

```
var relativeEl = document.querySelector('.el.relative-values');
relativeEl.style.transform = 'translateX(100px)';

anime({
  targets: '.el.relative-values',
  translateX: {
    value: '*=2.5', // 100px * 2.5 = '250px'
    duration: 1000
  },
  width: {
    value: '-=20px', // 28 - 20 = '8px'
    duration: 1800,
    easing: 'easeInOutSine'
  },
  rotate: {
    value: '+=2turn', // 0 + 2 = '2turn'
    duration: 1800,
    easing: 'easeInOutSine'
  },
  direction: 'alternate'
});
```

5.5.4 Colores: anime.js acepta y convierte valores de color hexadecimales, RGB, RGBA, HSL y HSLA.

Los códigos de color CSS (por ejemplo :) no son compatibles. 'red', 'yellow', 'aqua'

ACEPTA	EJEMPLO
Hexadecimal	`'#FFF'` o `'#FFFFFF'`
RGB	`'rgb(255, 255, 255)'`
RGBA	`'rgba(255, 255, 255, .2)'`
HSL	`'hsl(0, 100%, 100%)'`
HSLA	`'hsla(0, 100%, 100%, .2)'`

Ejemplo de código:

```
var colorsExamples = anime.timeline({
  endDelay: 1000,
  easing: 'easeInOutQuad',
  direction: 'alternate',
  loop: true
})
.add({ targets: '.color-hex',  background: '#FFF' }, 0)
.add({ targets: '.color-rgb',  background: 'rgb(255,255,255)' }, 0)
.add({ targets: '.color-hsl',  background: 'hsl(0, 100%, 100%)' }, 0)
.add({  targets:  '.color-rgba', background:  'rgba(255,255,255,  .2)'  },
0)
.add({ targets: '.color-hsla', background: 'hsla(0, 100%, 100%, .2)' },
0)
.add({ targets: '.colors-demo .el', translateX: 270 }, 0);
```

5.5.5 Desde hasta: Obliga a que la animación comience en un valor especificado.

ESCRIBE	EJEMPLO
Array	['50%', '100%']

Ejemplo de código:

```
anime({
  targets: '.el.from-to-values',
  translateX: [100, 250], // from 100 to 250
  delay: 500,
  direction: 'alternate',
  loop: true
});
```

5.5.6 Valores basados en funciones: Obtiene diferentes valores para cada objetivo y propiedad de la animación.

La función acepta 3 argumentos:

ARGUMENTOS	INFOS
target	El elemento objetivo actualmente animado
index	El índice del elemento objetivo animado
targetsLength	El número total de objetivos animados.

Ejemplo de código:

```
anime({
  targets: '.function-based-values-demo .el',
  translateX: function(el) {
    return el.getAttribute('data-x');
  },
  translateY: function(el, i) {
    return 50 + (-50 * i);
  },
  scale: function(el, i, l) {
    return (l - i) + .25;
  },
  rotate: function() { return anime.random(-360, 360); },
  borderRadius: function() { return ['50%', anime.random(10, 35) + '%']; },
  duration: function() { return anime.random(1200, 1800); },
  delay: function() { return anime.random(0, 400); },
  direction: 'alternate',
  loop: true
});
```

5.6 Keyframes

5.6.1 Keyframes de animación: Los fotogramas clave de animación se definen mediante un array, dentro de la keyframes property.

Si no se especifica una duración dentro de los fotogramas clave, la duración de cada fotograma clave será igual a la duración total de la animación dividida por el número de fotogramas clave.

ESCRIBE	EJEMPLO
Array	`[ {value: 100, easing: 'easeOutExpo'}, {value: 200, delay: 500}, {value: 300, duration: 1000} ]`

Ejemplo de código:

```
anime({
  targets: '.animation-keyframes-demo .el',
  keyframes: [
    {translateY: -40},
    {translateX: 250},
    {translateY: 40},
    {translateX: 0},
    {translateY: 0}
  ],
  duration: 4000,
  easing: 'easeOutElastic(1, .8)',
  loop: true
});
```

5.6.2 Keyframes de propiedad: De manera similar a los fotogramas clave de animación , los fotogramas clave de propiedad se definen mediante una matriz de objeto de propiedad. Los fotogramas clave de propiedad permiten animaciones superpuestas, ya que cada propiedad tiene su propia matriz de fotogramas clave.

Si no se especifica una duración dentro de los fotogramas clave, la duración de cada fotograma clave será igual a la duración total de la animación dividida por el número de fotogramas clave.

ESCRIBE	EJEMPLO
Array	`[ {value: 100, easing: 'easeOutExpo'}, {value: 200, delay: 500}, {value: 300, duration: 1000} ]`

Ejemplo de código:

```
anime({
  targets: '.property-keyframes-demo .el',
  translateX: [
    { value: 250, duration: 1000, delay: 500 },
    { value: 0, duration: 1000, delay: 500 }
  ],
  translateY: [
    { value: -40, duration: 500 },
    { value: 40, duration: 500, delay: 1000 },
    { value: 0, duration: 500, delay: 1000 }
  ],
  scaleX: [
    { value: 4, duration: 100, delay: 500, easing: 'easeOutExpo' },
    { value: 1, duration: 900 },
    { value: 4, duration: 100, delay: 500, easing: 'easeOutExpo' },
    { value: 1, duration: 900 }
  ],
  scaleY: [
    { value: [1.75, 1], duration: 500 },
    { value: 2, duration: 50, delay: 1000, easing: 'easeOutExpo' },
    { value: 1, duration: 450 },
    { value: 1.75, duration: 50, delay: 1000, easing: 'easeOutExpo' },
    { value: 1, duration: 450 }
  ],
  easing: 'easeOutElastic(1, .8)',
  loop: true
});
```

5.7 Staggering

5.7.1 Conceptos básicos: El escalonamiento le permite animar múltiples elementos con acciones de seguimiento y superposición .

ARGUMENTOS	ESCRIBE	INFORMACIÓN	REQUERIDO
Valor	Number, String, Array	Los valores manipulados	sí
Opciones	Object	Parámetros escalonados	No

Ejemplo de código:

```
anime({
  targets: '.basic-staggering-demo .el',
  translateX: 270,
    delay: anime.stagger(100) // increase delay by 100ms for each elements.
});
```

5.7.2 Valor inicial: Inicia el efecto de asombro a partir de un valor específico.

TIPOS	INFORMACIÓN
Number, String	Igual que los valores de propiedad, consulte la sección de valores

Ejemplos de código:

```
anime({
  targets: '.staggering-start-value-demo .el',
  translateX: 270,
  delay: anime.stagger(100, {start: 500}) // delay starts at 500ms then increase by 100ms for each elements.
});
```

5.7.3 Valor de rango: Distribuye uniformemente los valores entre dos números.

ESCRIBE	INFORMACIÓN
`'easingName'`	Se aceptan todos los nombres de suavizado válidos, consulte la sección de suavizado
`function(i)`	Función de aceleración personalizada, consulte la sección de aceleración personalizada

Ejemplo de código:

```
anime({
  targets: '.range-value-staggering-demo .el',
  translateX: 270,
   rotate: anime.stagger([-360, 360]), // rotation will be distributed from -360deg to 360deg evenly between all elements
  easing: 'easeInOutQuad'
});
```

5.7.4 From Value: Inicia el efecto de escalonamiento desde una posición específica.

OPCIONES	ESCRIBE	INFORMACIÓN
`'first'` (defecto)	`'string'`	Inicie el efecto desde el primer elemento
`'last'`	`'string'`	Iniciar el efecto desde el último elemento
`'center'`	`'string'`	Inicie el efecto desde el centro
`index`	`number`	Iniciar el efecto desde el índice especificado

Ejemplo de código:

```
anime({
  targets: '.staggering-from-demo .el',
  translateX: 270,
  delay: anime.stagger(100, {from: 'center'})
});
```

5.7.5 Dirección: Cambia el orden en el que opera el escalonamiento.

OPCIONES	INFORMACIÓN
`'normal'` (defecto)	Escalonamiento normal, desde el primer elemento hasta el último
`'reverse'`	Escalonamiento invertido, desde el último elemento hasta el primero

Ejemplo de código:

```
anime({
  targets: '.staggering-direction-demo .el',
  translateX: 270,
  delay: anime.stagger(100, {direction: 'reverse'})
});
```

5.7.6 Facilidades: Alterna los valores utilizando una función de aceleración.

ESCRIBE	INFORMACIÓN
`'string'`	Se aceptan todos los nombres de suavizado válidos
`function(i)`	Utilice su propia función de aceleración personalizada

Ejemplo de código:

```
anime({
  targets: '.staggering-easing-demo .el',
  translateX: 270,
  delay: anime.stagger(300, {easing: 'easeOutQuad'})
});
```

5.7.7 Grid: Valores asombrosos basados en una matriz 2D que permiten efectos de "onda".

ESCRIBE	INFORMACIÓN
array	Una matriz de 2 elementos, el primer valor es el número de filas, el segundo el número de columnas

Ejemplo de código:

```
anime({
  targets: '.staggering-grid-demo .el',
  scale: [
    {value: .1, easing: 'easeOutSine', duration: 500},
    {value: 1, easing: 'easeInOutQuad', duration: 1200}
  ],
  delay: anime.stagger(200, {grid: [14, 5], from: 'center'})
});
```

5.7.8 Axis: Fuerza la dirección de un efecto de escalonamiento de cuadrícula .

PARÁMETROS	INFORMACIÓN
'x'	Sigue el eje x
'y'	Sigue el eje y

Ejemplo de código

```
anime({
  targets: '.staggering-axis-grid-demo .el',
   translateX: anime.stagger(10, {grid: [14, 5], from: 'center', axis: 'x'}),
   translateY: anime.stagger(10, {grid: [14, 5], from: 'center', axis: 'y'}),
  rotateZ: anime.stagger([0, 90], {grid: [14, 5], from: 'center', axis: 'x'}),
  delay: anime.stagger(200, {grid: [14, 5], from: 'center'}),
  easing: 'easeInOutQuad'
});
```

5.8 Timeline

5.8.1 Conceptos básicos de la Timeline: Las líneas de tiempo le permiten sincronizar varias animaciones juntas. De forma predeterminada, cada animación agregada a la línea de tiempo comienza después de que finaliza la animación anterior.

ARGUMENTO	ESCRIBE	INFORMACIÓN	REQUERIDO
parameters	Object	Los parámetros predeterminados de la línea de tiempo heredados por los niños.	No

Agregar animaciones a una línea de tiempo:

ARGUMENTO	TIPOS	INFORMACIÓN	REQUERIDO
parameters	Object	Los parámetros de animación secundaria, anulan los parámetros predeterminados de la línea de tiempo	sí
time offset	String o Number	Consulte la sección de compensaciones de la línea de tiempo	No

Ejemplo de código:

```
var tl = anime.timeline({
  easing: 'easeOutExpo',
  duration: 750
});

// Add children
tl
.add({
  targets: '.basic-timeline-demo .el.square',
  translateX: 250,
})
.add({
  targets: '.basic-timeline-demo .el.circle',
  translateX: 250,
})
.add({
  targets: '.basic-timeline-demo .el.triangle',
  translateX: 250,
});
```

5.8.2 Compensación de tiempo: Las compensaciones de tiempo se pueden especificar con un segundo parámetro opcional usando la .add()función de línea de tiempo . Define cuándo comienza una animación en la línea de tiempo, si no se especifica ningún desplazamiento, la animación comenzará después de que termine la animación anterior. Un desplazamiento puede ser relativo a la última animación o absoluto a toda la línea de tiempo.

ESCRIBE	COMPENSAR	EJEMPLO	INFOS
Cuerda	`'+='`	`'+=200'`	Comienza 200 ms después de que finaliza la animación anterior
Cuerda	`'-='`	`'-=200'`	Comienza 200ms antes de que finalice la animación anterior
Número	`Number`	`100`	Comienza a los 100 ms, independientemente de la posición de la animación en la línea de tiempo.

Ejemplo de código:

```
var tl = anime.timeline({
  easing: 'easeOutExpo',
  duration: 750
});

tl
.add({
  targets: '.offsets-demo .el.square',
  translateX: 250,
})
.add({
  targets: '.offsets-demo .el.circle',
  translateX: 250,
}, '-=600') // relative offset
.add({
  targets: '.offsets-demo .el.triangle',
  translateX: 250,
}, 400); // absolute offset
```

5.8.3 Herencia de parámetros: Algunos de los parámetros establecidos en la instancia de la línea de tiempo principal pueden ser heredados por todos los elementos secundarios.

PARÁMETROS QUE SE PUEDEN HEREDAR
targets
easing
duration
delay
endDelay
round

Ejemplo de código:

```
var tl = anime.timeline({
  targets: '.params-inheritance-demo .el',
  delay: function(el, i) { return i * 200 },
  duration: 500, // Can be inherited
  easing: 'easeOutExpo', // Can be inherited
  direction: 'alternate', // Is not inherited
  loop: true // Is not inherited
});

tl
.add({
  translateX: 250,
  // override the easing parameter
  easing: 'spring',
})
.add({
  opacity: .5,
  scale: 2
})
.add({
  // override the targets parameter
  targets: '.params-inheritance-demo .el.triangle',
  rotate: 180
})
.add({
  translateX: 0,
  scale: 1
});
```

5.9 Controles

5.9.1 Reproducir y pausar: Reproduce una animación en pausa o inicia la animación si los autoplayparámetros están configurados en false. animation.play(); Pausa una animación en ejecución. animation.pause();

Ejemplo de código:

```
var animation = anime({
  targets: '.play-pause-demo .el',
  translateX: 270,
  delay: function(el, i) { return i * 100; },
  direction: 'alternate',
  loop: true,
  autoplay: false,
  easing: 'easeInOutSine'
});

document.querySelector('.play-pause-demo .play').onclick = animation.play;
document.querySelector('.play-pause-demo .pause').onclick = animation.pause;
```

5.9.2 Reiniciar: Reinicia una animación desde sus valores iniciales. animation.restart();

Ejemplo de código:

```
var animation = anime({
  targets: '.restart-demo .el',
  translateX: 250,
  delay: function(el, i) { return i * 100; },
  direction: 'alternate',
  loop: true,
  easing: 'easeInOutSine'
});

document.querySelector('.restart-demo .restart').onclick = animation.restart;
```

5.9.3 Marcha atrás: Invierte la dirección de una animación. animation.reverse();

Ejemplo de código:

```
var animation = anime({
  targets: '.reverse-anim-demo .el',
  translateX: 270,
  loop: true,
  delay: function(el, i) { return i * 200; },
  easing: 'easeInOutSine'
});

document.querySelector('.reverse-anim-demo .reverse').onclick = animation.reverse;
```

5.9.4 Buscar: Salta a un tiempo específico (en milisegundos). animation.seek(timeStamp); También se puede utilizar para controlar una animación mientras se desplaza. animation.seek((scrollPercent / 100) * animation.duration);

Ejemplo de código:

```
var animation = anime({
  targets: '.seek-anim-demo .el',
  translateX: 270,
  delay: function(el, i) { return i * 100; },
  elasticity: 200,
  easing: 'easeInOutSine',
  autoplay: false
});

var seekProgressEl = document.querySelector('.seek-anim-demo .progress');
seekProgressEl.oninput = function() {
  animation.seek(animation.duration * (seekProgressEl.value / 100));
};
```

5.9.5 Controles de línea de tiempo: Las líneas de tiempo se pueden controlar como cualquier otra instancia de anime.js .

```
timeline.play();
timeline.pause();
timeline.restart();
timeline.seek(timeStamp);
```

Ejemplo de código:

```
var controlsProgressEl = document.querySelector('.timeline-controls-demo .progress');

var tl = anime.timeline({
  direction: 'alternate',
  loop: true,
  duration: 500,
  easing: 'easeInOutSine',
  update: function(anim) {
    controlsProgressEl.value = tl.progress;
  }
});

tl
.add({
  targets: '.timeline-controls-demo .square.el',
  translateX: 270,
})
.add({
  targets: '.timeline-controls-demo .circle.el',
  translateX: 270,
}, '-=100')
.add({
  targets: '.timeline-controls-demo .triangle.el',
  translateX: 270,
}, '-=100');

document.querySelector('.timeline-controls-demo .play').onclick = tl.play;
document.querySelector('.timeline-controls-demo .pause').onclick = tl.pause;
document.querySelector('.timeline-controls-demo .restart').onclick = tl.restart;

controlsProgressEl.addEventListener('input', function() {
  tl.seek(tl.duration * (controlsProgressEl.value / 100));
});
```

5.10 Callbacks y promesas

5.10.1 Actualizar: La devolución de llamada se activa en cada fotograma tan pronto como la animación comienza a reproducirse.

ESCRIBE	PARÁMETROS	INFORMACIÓN
Función	animación	Devuelve el objeto de animación actual

Ejemplo de código:

```
var updates = 0;

anime({
  targets: '.update-demo .el',
  translateX: 270,
  delay: 1000,
  direction: 'alternate',
  loop: 3,
  easing: 'easeInOutCirc',
  update: function(anim) {
    updates++;
    progressLogEl.value = 'progress : '+Math.round(anim.progress)+'%';
    updateLogEl.value = 'updates : '+updates;
  }
});
```

5.10.2 Comenzar y completar: begin() la devolución de llamada se activa una vez, cuando la animación comienza a reproducirse.

complete() la devolución de llamada se activa una vez, cuando se completa la animación.

ESCRIBE	PARÁMETROS	INFORMACIÓN
Función	animación	Devuelve el objeto de animación actual

Ejemplo de código:

```
anime({
  targets: '.begin-complete-demo .el',
  translateX: 240,
  delay: 1000,
  easing: 'easeInOutCirc',
  update: function(anim) {
    progressLogEl.value = 'progress : ' + Math.round(anim.progress) +
'%';
    beginLogEl.value = 'began : ' + anim.began;
    completeLogEl.value = 'completed : ' + anim.completed;
  },
  begin: function(anim) {
    beginLogEl.value = 'began : ' + anim.began;
  },
  complete: function(anim) {
    completeLogEl.value = 'completed : ' + anim.completed;
  }
});
```

5.10.3 Loopbegin & Loopcomplete: loopBegin() La devolución de llamada se activa una vez cada vez que comienza un bucle.

loopComplete() La devolución de llamada se activa una vez cada vez que se completa un bucle.

ESCRIBE	PARÁMETROS	INFORMACIÓN
Función	animación	Devuelve el objeto de animación actual

Ejemplo de código:

```
var loopBegan = 0;
var loopCompleted = 0;

anime({
  targets: '.loopBegin-loopComplete-demo .el',
  translateX: 240,
  loop: true,
  direction: 'alternate',
  easing: 'easeInOutCirc',
  loopBegin: function(anim) {
    loopBegan++;
    beginLogEl.value = 'loop began : ' + loopBegan;
  },
  loopComplete: function(anim) {
    loopCompleted++;
    completeLogEl.value = 'loop completed : ' + loopCompleted;
  }
});
```

5.10.4 Cambio: Devolución de llamada activa en cada fotogramas intermedios de la animación de retardo y endDelay .

ESCRIBE	PARÁMETROS	INFORMACIÓN
Función	animación	Devuelve el objeto de animación actual

Ejemplo de código:

```
var changes = 0;

anime({
  targets: '.change-demo .el',
  translateX: 270,
  delay: 1000,
  endDelay: 1000,
  direction: 'alternate',
  loop: true,
  easing: 'easeInOutCirc',
  update: function(anim) {
    progressLogEl.value = 'progress : '+Math.round(anim.progress)+'%';
  },
  change: function() {
    changes++;
    changeLogEl.value = 'changes : ' + changes;
  }
});
```

5.10.5 Changebegin & Changecomplete: changeBegin() La devolución de llamada se activa cada vez que la animación comienza a cambiar.

changeComplete() La devolución de llamada se activa cada vez que la animación deja de cambiar.

ESCRIBE	PARÁMETROS	INFORMACIÓN
Función	animación	Devuelve el objeto de animación actual

Ejemplo de código:

```
var changeBegan = 0;
var changeCompleted = 0;

anime({
  targets: '.changeBegin-chnageComplete-demo .el',
  translateX: 240,
  delay: 1000,
  endDelay: 1000,
  loop: true,
  direction: 'alternate',
  easing: 'easeInOutCirc',
  update: function(anim) {
    progressLogEl.value = 'progress : '+Math.round(anim.progress)+'%';
  },
  changeBegin: function(anim) {
    changeBegan++;
    beginLogEl.value = 'change began : ' + changeBegan;
  },
  changeComplete: function(anim) {
    changeCompleted++;
    completeLogEl.value = 'change completed : ' + changeCompleted;
  }
});
```

5.10.6 Promesa terminada: odas las instancias de animación devuelven una finishedpromesa cuando la animación finaliza.

animation.finished.then(function() { // Do things... });

Ejemplo de código:

```
var progressLogEl = document.querySelector('.promise-demo .progress-log');
var promiseEl = document.querySelector('.promise-demo .el');
var finishedLogEl = document.querySelector('.promise-demo .finished-log');
var demoPromiseResetTimeout;

function logFinished() {
  anime.set(finishedLogEl, {value: 'Promise resolved'});
  anime.set(promiseEl, {backgroundColor: '#18FF92'});
}

var animation = anime.timeline({
  targets: promiseEl,
  delay: 400,
  duration: 500,
  endDelay: 400,
  easing: 'easeInOutSine',
  update: function(anim) {
    progressLogEl.value = 'progress : '+Math.round(anim.progress)+'%';
  }
}).add({
  translateX: 250
}).add({
  scale: 2
}).add({
  translateX: 0
});

animation.finished.then(logFinished);
```

5.11 SVG

5.11.1 Ruta de movimiento: Anima un elemento relativo a los valores x, yy angle de un elemento de ruta SVG.

var myPath = anime.path('svg path');

La función de ruta devuelve una nueva función que devuelve la propiedad especificada.

PARÁMETROS	EJEMPLO	INFORMACIÓN
`'x'`	`myPath('x')`	Devuelve el valor x actual `'px'` de la ruta SVG
`'y'`	`myPath('y')`	Devuelve el valor y actual `'px'` de la ruta SVG
`'angle'`	`myPath('angle')`	Devuelve el valor actual del ángulo `'degrees'` de la ruta SVG

Ejemplo de código:

```
var path = anime.path('.motion-path-demo path');

anime({
  targets: '.motion-path-demo .el',
  translateX: path('x'),
  translateY: path('y'),
  rotate: path('angle'),
  easing: 'linear',
  duration: 2000,
  loop: true
});
```

5.11.2 Morphing: Crea una transición entre dos formas svg.
¡Las formas deben tener el mismo número de puntos!

Ejemplo de código:

```
anime({
  targets: '.morphing-demo .polymorph',
  points: [
    { value: [
      '70 24 119.574 60.369 100.145 117.631 50.855 101.631 3.426 54.369',
      '70 41 118.574 59.369 111.145 132.631 60.855 84.631 20.426 60.369']
    },
    { value: '70 6 119.574 60.369 100.145 117.631 39.855 117.631 55.426 68.369'
},
    { value: '70 57 136.574 54.369 89.145 100.631 28.855 132.631 38.426 64.369'
},
    { value: '70 24 119.574 60.369 100.145 117.631 50.855 101.631 3.426 54.369'
}
  ],
  easing: 'easeOutQuad',
  duration: 2000,
  loop: true
});
```

5.11.3 Dibujo lineal: Crea una animación de dibujo de ruta utilizando la 'stroke-dashoffset'propiedad. Establezca el 'dash-offset' valor de la ruta con anime.setDashoffset() un valor formateado desde a .
strokeDashoffset: [anime.setDashoffset, 0]

Ejemplo de código:

```
anime({
  targets: '.line-drawing-demo .lines path',
  strokeDashoffset: [anime.setDashoffset, 0],
  easing: 'easeInOutSine',
  duration: 1500,
  delay: function(el, i) { return i * 250 },
  direction: 'alternate',
  loop: true
});
```

5.12 Easings

5.12.1 Lineal: No aplica ningún tiempo de relajación a su animación. Útil para transiciones de opacidad y colores.

easing: 'linear'

Ejemplo de código:

```
anime({
  targets: '.linear-easing-demo .el',
  translateX: 250,
  direction: 'alternate',
  loop: true,
  easing: 'linear'
});
```

5.12.2 Funciones de penner: Funciones de aceleración integradas de Robert Penner .

Facilidades disponibles:

EN	FUERA	EN FUERA	SALIR EN
'easeInQuad'	'easeOutQuad'	'easeInOutQuad'	'easeOutInQuad'
'easeInCubic'	'easeOutCubic'	'easeInOutCubic'	'easeOutInCubic'
'easeInQuart'	'easeOutQuart'	'easeInOutQuart'	'easeOutInQuart'
'easeInQuint'	'easeOutQuint'	'easeInOutQuint'	'easeOutInQuint'
'easeInSine'	'easeOutSine'	'easeInOutSine'	'easeOutInSine'
'easeInExpo'	'easeOutExpo'	'easeInOutExpo'	'easeOutInExpo'
'easeInCirc'	'easeOutCirc'	'easeInOutCirc'	'easeOutInCirc'
'easeInBack'	'easeOutBack'	'easeInOutBack'	'easeOutInBack'
'easeInBounce'	'easeOutBounce'	'easeInOutBounce'	'easeOutInBounce'

5.12.3 Curva cúbica de bézier: Utilice sus propias curvas Bézier cúbicas personalizadas cubicBezier(x1, y1, x2, y2). easing: 'cubicBezier(.5, .05, .1, .3)'

Ejemplo de código:

```
anime({
  targets: '.cubic-bezier-demo .el',
  translateX: 250,
  direction: 'alternate',
  loop: true,
  easing: 'cubicBezier(.5, .05, .1, .3)'
})
```

5.12.4 Elástico: Alivio elástico. easing: 'easeOutElastic(amplitude, period)'

EN	FUERA	EN FUERA	SALIR EN
'easeInElastic'	'easeOutElastic'	'easeInOutElastic'	'easeOutInElastic'

PARÁMETRO	DEFECTO	MIN	MAX	INFORMACIÓN
Amplitud	1	1	10	Controla el sobreimpulso de la curva. Cuanto mayor sea este número, más rebasamiento habrá.
Período	.5	0.1	2	Controla cuántas veces va y viene la curva. Cuanto menor sea este número, más veces se mueve el telón hacia adelante y hacia atrás.

Ejemplo de código:

```
anime({
  targets: '.elastic-easing-demo .line:nth-child(1) .el',
  translateX: 270,
  easing: 'easeInElastic(1, .6)'
});

anime({
  targets: '.elastic-easing-demo .line:nth-child(2) .el',
  translateX: 270,
  easing: 'easeOutElastic(1, .6)'
});

anime({
  targets: '.elastic-easing-demo .line:nth-child(3) .el',
```

```
  translateX: 270,
  easing: 'easeInOutElastic(1, .6)'
});

anime({
  targets: '.elastic-easing-demo .line:nth-child(4) .el',
  translateX: 270,
  easing: 'easeOutInElastic(1, .6)'
});
```

Bibliografía

https://github.com/juliangarnier/anime/

https://animejs.com/documentation/

https://codepen.io/collection/XLebem/

https://openbase.com/js/animejs/documentation

https://www.desarrollolibre.net/blog/javascript/animejs-para-realizar-animaciones-con-javascript

https://codepen.io/mariphkhakadze/pen/VwLOZNE