

# **DEEP DIVE INTO OCEANIC JAVASCRIPT ANIMATIONS WITH A GREENSOCK**

Deep dive into oceanic javascript animations with a greensock



# ALESSANDRO RIGOBELLO ⚡

💻 Freelance creative developer & interaction designer  
@sndrgb  
sndrgb.com

🤝 Co-founder THESIGNOF  
@thesignof

👨‍💼 Awwwards developer jury member  
awwwards.com

# PERCHÈ FARE ANIMAZIONI?



Deep dive into oceanic javascript animations with a greensock

- L'animazione è potente per trasmettere significato
- Può guidare gli utenti
- Se costruita correttamente, l'animazione può essere utile e performante
- Perché altrimenti non stiamo usando il web al suo pieno potenziale
- **MORE FUN!**

# **WHY JAVASCRIPT?**

## **(OR WHY NOT CSS)**

# **MYTH**

**CSS:** "Hardware accelerated" e "Mobile-friendly"

**JS:** "Dirty" e "oldy"

# **FACT**

**CSS:** "Hardware accelerated" e "Mobile-friendly"

**JS:** Performante, versatile, event-based...

# **CONTROLLO INDIPENDENTE DEI VARI TRANSFORM**

Codepen

# **PERFORMANCE**

Hardware Acceleration?

**Coinvolgimento GPU  
Thread diverso per calcoli**

Render Hell by Simon Schreibt

# **CONTROLLI RUNTIME ED EVENTI**

## **CURVE DI BEZIER**

### **FISICA - ES. SNAP-BACK BASATI SUL MOMENTUM ( $M \cdot V$ )**

snap-back

## **WORKFLOW COMPLESSI**

**CONVINTO. VOGLIO  
ANIMARE CON IL  
JAVASCRIPT, E ORA?**

# GREENSOCK ANIMATION PLATFORM!



Deep dive into oceanic javascript animations with a greensock

# PRINCIPALI LIBRERIE

- GSAP
  - Design Canada
  - Antoni
- Anime.js
  - Submit Button
  - Fireworks canvas
- MOJS
  - Dusty Burst
  - Love or Hate?

# TWEENING

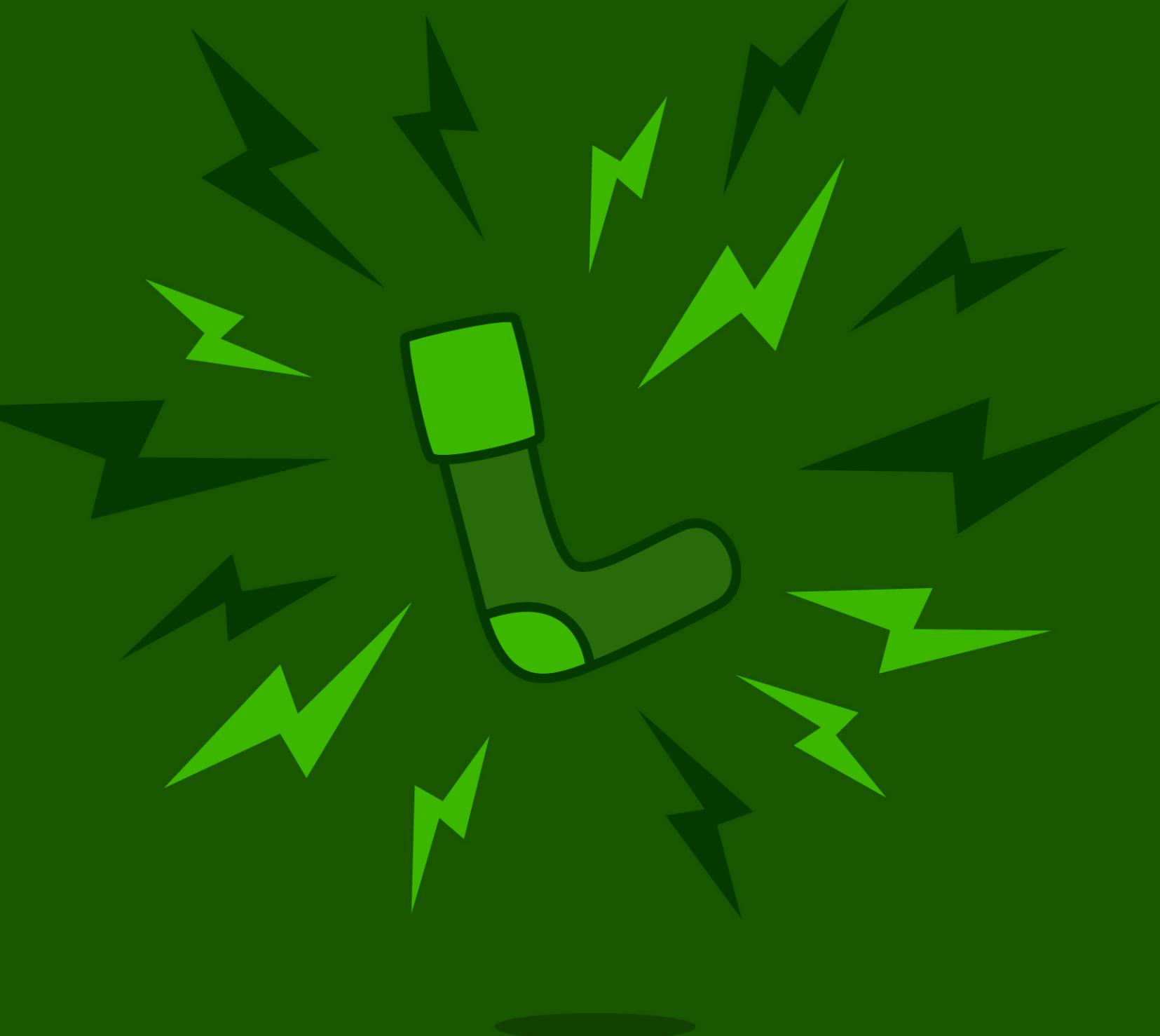
to tween, /twi:n/

Il tweening è il processo per il quale, durante la realizzazione di un'animazione, vengono creati una serie di fotogrammi (definiti tweens) tra un fotogramma di partenza e uno di arrivo (definiti keyframe).

[Wikipedia](#)

# GSAP

- Free eccetto plugin
- Espandibile e modulare
- Può animare qualsiasi cosa
- Veloce, robusta, API molto vasta e compatibile
- Logo e visual identity terribile



# **ESPANDIBILE E MODULARE**

- TweenLite, TweenMax
- TimelineLite, TimelineMax
- Plugins: Attr, DrawSVG, MorphSVG, Physics2D, Pixi, ScrollTo
- Draggable
- SplitText

# PUÒ ANIMARE QUALSIASI COSA

CSS, elementi DOM, colori, Canvas, oggetti e array...

```
const obj = { score: 0 }
const showScore = () => { document.body.innerHTML = obj.score }

const tween = TweenMax.to(obj, 20, { score:100, onUpdate: showScore })
```

# METODI (PIÙ COMUNI)

- `.add()` -> Aggiunge un tween, una timeline, una callback o una label
- `.set()` -> Tween di durata 0 con destinazione
- `.to()` -> Valore di destinazione
- `.from()` -> Valore di partenza
- `.fromTo()` -> Valore di partenza e destinazione
- `.staggerFromTo()` -> Stagger con partenza e destinazione

# PARAMETRI (PIÙ COMUNI)

targets: Object | Array | String selector | DOM element,  
duration: Number (second based),  
fromVars: Object,  
toVars: Object,  
stagger: Number,  
onCompleteAll | onComplete: Function

# TIMELINE EXAMPLE

Codepen Example

```
import { TimelineMax, Expo } from 'gsap/all'

const tl = new TimelineMax({
  onComplete: resolve
})

tl
  .add('start')
  .from('#numbers', 0.3, {
    autoAlpha: 0,
  }, 'start')
  .to($el, 1.2, {
    y: -100,
    autoAlpha: 0,
    ease: Expo.easeInOut,
  }, 'start')
  .add(() => {
    ctx.destroyScrollbar();
  })
  .to('#numbers', {
    yPercent: 100
})
```

# ARRAY TWEENING

Codepen Example



**delay:** Number - Delay in secondi prima che l'animazione parta.

**ease:** Ease (or Function or String).

**repeat:** Number - Numero di volte di ripetizione dell'animazione dopo la prima iterazione (**-1 per infinito**).

**repeatDelay:** Number - Tempo in secondi tra le varie ripetizioni.

**yoyo:** Boolean - Se true ogni ciclo di ripetizione alterna la direzione.

**yoyoEase:** Ease | Boolean - Un'ease per lo yoyo.

**paused:** Boolean - Se true l'animazione viene interrotta subito dopo la sua creazione.

**overwrite:** String (or integer) - Controlla come e se altri ween sullo stesso target vengono sovrascritti.

**onComplete:** Function - Funzione che viene chiamata quando l'animazione è completata.

# ANIMATION PRINCIPLES



I dodici principi fondamentali dell'animazione della Disney sono stati introdotti dagli animatori Ollie Johnston e Frank Thomas nel loro libro del 1981, *The Illusion of Life: Disney Animation*. Johnston e Thomas a loro volta hanno basato il loro libro sul lavoro dei principali animatori Disney dagli anni '30 in poi, e il loro sforzo per produrre animazioni più realistiche. Lo scopo principale dei principi era quello di produrre un'illusione di personaggi che aderivano alle leggi fondamentali della fisica, ma si occupavano anche di questioni più astratte, come i tempi emotivi e l'attrattiva del personaggio.

Tumblr

Vimeo Video

# EASING

Ease Visualizer



CustomEase



# COOL STUFF

Deep dive into oceanic javascript animations with a greensock

# STAGGER A PARTIRE DALL'ULTIMO ELEMENTO

```
// dal primo all'ultimo
TweenMax.staggerTo(".box", 0.8, {
  rotation:360,
  onComplete: tweenComplete,
  onCompleteParams:['{self}']
}, 1.2, myCompleteAll);

// dall'ultimo al primo
TweenMax.staggerTo(".box", 0.8, {
  rotation:360,
  onComplete: tweenComplete,
  onCompleteParams:['{self}']
}, -1.2, myCompleteAll);
```

# **DEFAULT EASE**

```
TweenLite.defaultEase = Expo.easeOut;
```

# **STEPPED ANIMATIONS**

```
TweenMax.to($cursor, duration, { ease: new  
SteppedEase(1), opacity: 0 });
```

# STORE TWEENS

```
const tl = new TimelineMax({paused: true});
```

```
tl  
  .to($manny, 0.4, { autoAlpha: 1 })  
  .to($meche, 0.6, { x: -3 }, 0.3);
```

e poi:

```
tl.restart();
```

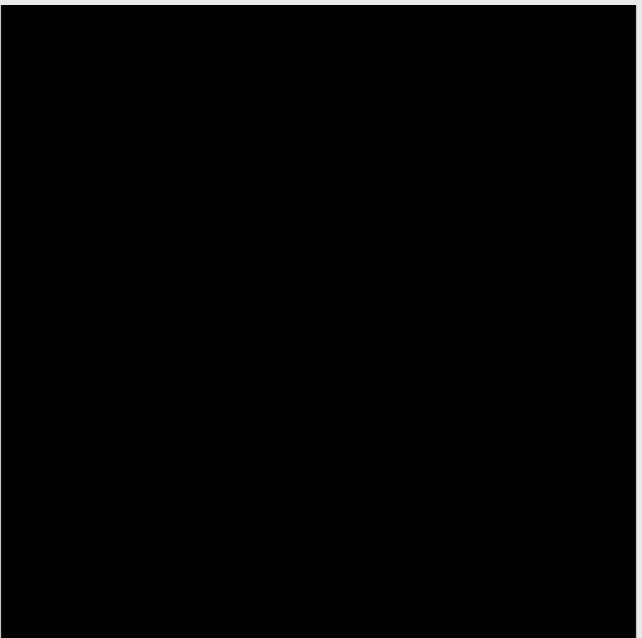
# RIMUOVERE CSS INLINE DOPO IL TWEEN

```
TweenMax.to($manny, 1, { autoAlpha: 0, clearProps: 'all' }));  
TweenMax.to($meche, 1, { y: 0, autoAlpha: 0, clearProps: 'y' }));
```

# DELAYEDCALL VERSUS SETTIMEOUT

```
TweenMax.delayedCall(4.5, myFunction)  
TweenMax.killTweensOf(myFunction)
```

Sinitra delayedCall, destra setTimeout



# SPLITTEXT

```
const chars = new SplitText('#quote', {type: 'words, chars'}).chars  
TweenMax.staggerFrom(chars, 0.8, { ... }, 0.01);
```

Codepen Example

# CYCLE

## Codepen Example

```
const tl = new TimelineMax({ paused: true })

tl.staggerTo(slides, 1, {
  cycle: {
    y: (loop) => index === loop ? 0 : windowHeight * direction
  },
  ease: Expo.easeInOut
}, 0, 0)

tl.restart()
```

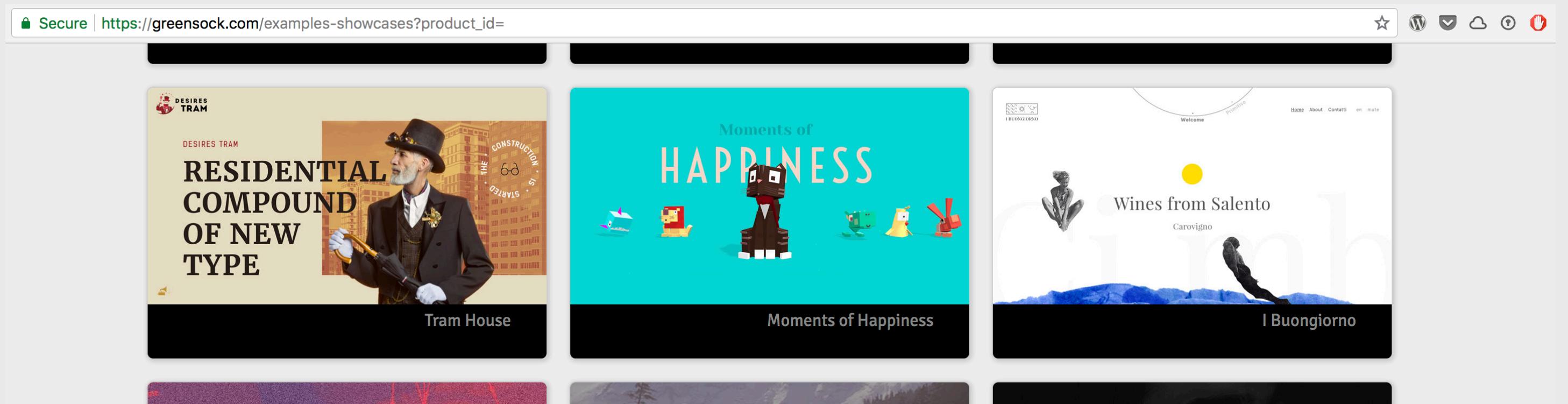
# DRAGGABLE

```
Draggable.create('.box', {  
    type: 'x,y',  
    edgeResistance:0.65,  
    bounds:'#container',  
    throwProps: true  
});
```

Draggable Example

# ANIMATING 2D CANVAS WITH PIXI

I Buongiorno



Deep dive into oceanic javascript animations with a greensock

# ANIMATING 2D CANVAS WITH PIXI

```
addSprites(el, cont) {
    // create a new Sprite from an image path
    const path = isMobile() ? `/assets/images/mobile/${el.url}.png` : `/assets/images/${el.url}.png`
    const sprite = PIXI.Sprite.fromImage(path)
    const tl = new TimelineMax()
    sprite.setupDimensions = el.setup

    this.setupSprite(sprite)

    sprite.alpha = 0
    sprite.parallaxFactor = el.factor / 10
    sprite.enterTimeline = tl

    tl.set(sprite, {
        alpha: 1,
    }).from(sprite.position, 1.5, {
        y: viewportSize().y + sprite.height,
        ease: Power4.easeOut,
    });
    cont.addChild(sprite);
}
```

I Buongiorno



# MORPHSVG

Temperature  
May the morph be with you

Deep dive into oceanic javascript animations with a greensock

# **TIMELINE WORKFLOW**

Deep dive into oceanic javascript animations with a greensock

```

import { TimelineMax, Expo, Power2 } from 'gsap/all';
import SplitText from '../assets/lib/SplitText';
import enterText from './timelines/glitch-text';

export const enterTransitions = new Map();
export const leaveTransitions = new Map();
const getEnterTl = done => new TimelineMax({ onComplete: done });
const getLeaveTl = done => new TimelineMax({ onComplete: done });

export function getTransition(type, page) {
    const transition = type === 'enter' ? enterTransitions : leaveTransitions;
    return transition.get(page);
}

enterTransitions.set('home', ctx => new Promise((resolve) => {
    const tl = getEnterTl(resolve);
    const split = new SplitText(ctx.$refs.introText, { type: 'lines, words' });
    const $titles = ctx.$el.querySelectorAll('[data-text]');

    ctx.$el.setAttribute('data-cloak', false);

    tl
        .add('start')
        .add(enterText($titles, 1))
        .staggerFrom('[data-slide]', 1.2, {
            autoAlpha: 0,
            ease: Expo.easeInOut,
            height: 0,
            clearProps: 'all',
        }, 0.07, 'start')
        .from('[data-anim="top"]', 0.5, {
            autoAlpha: 0,
            y: 20,
        })
        .staggerFrom(split.lines, 0.8, {
            skewX: -35,
            ease: Power2.easeOut,
            autoAlpha: 0,
        }, 0.05);
)));

```



# gsap-then

Make every GSAP Tween a promise

gzipped size 207 B build passing npm v2.1.1

Once loaded, every GSAP tween (TweenLite, TimelineLite, TweenMax, TimelineMax) will automatically be a promise. See the [usage examples](#) to see what this enables.

## Install

```
npm install --save gsap-then
```

```
• import 'gsap';
• import 'gsap-then';
```

Or include the file `dist/gsap-then.browser.js` after loading GreenSock.

## Usage

```
TweenLite.to('.title', 1, {opacity: 0}).then(function () {
  console.log('Done animating title');
})
```

# **OK. SONO PRONTO, E ORA?**

## **DIVERTITI**

# THANKS TO ALL



Deep dive into oceanic javascript animations with a greensock