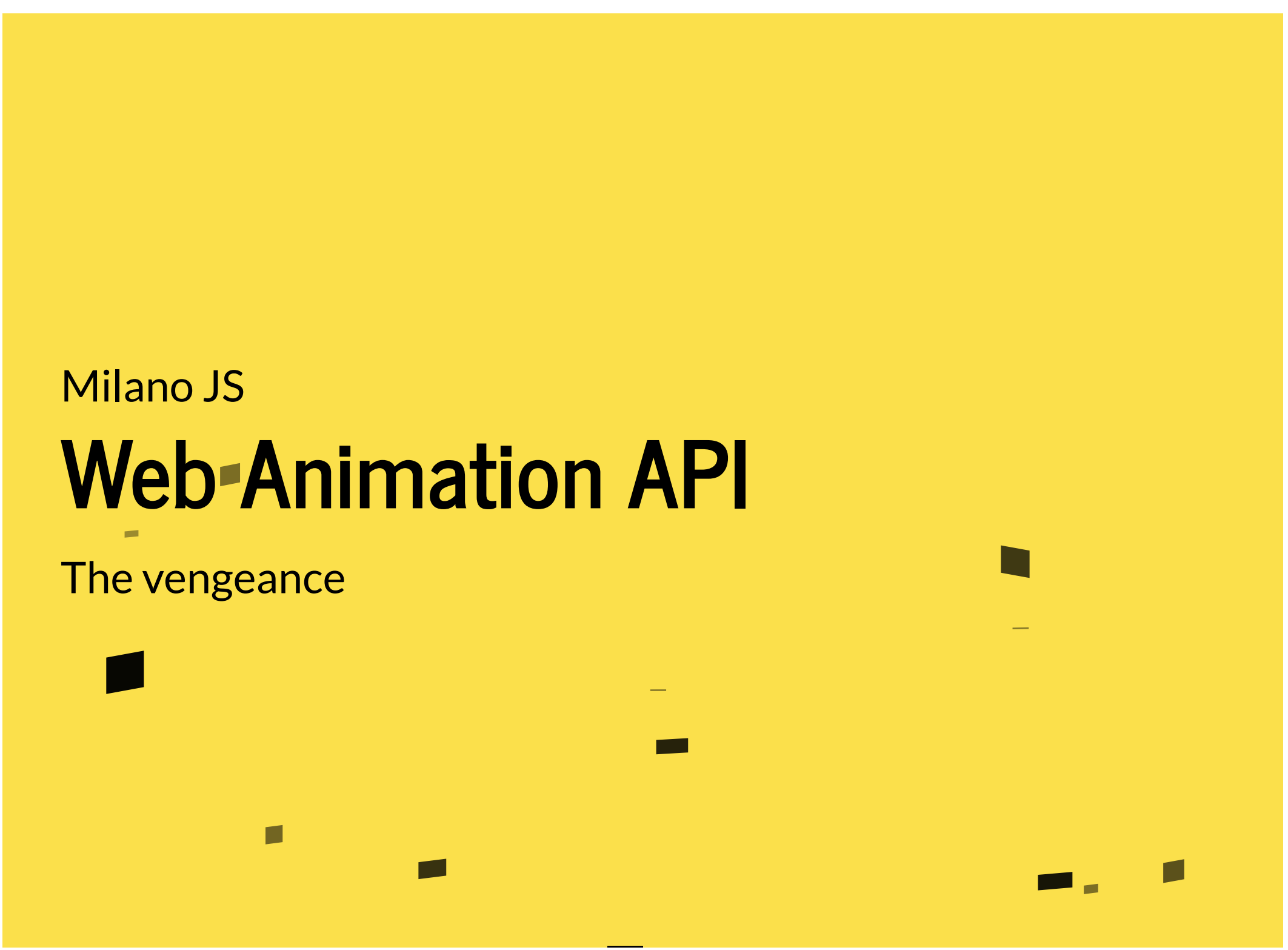
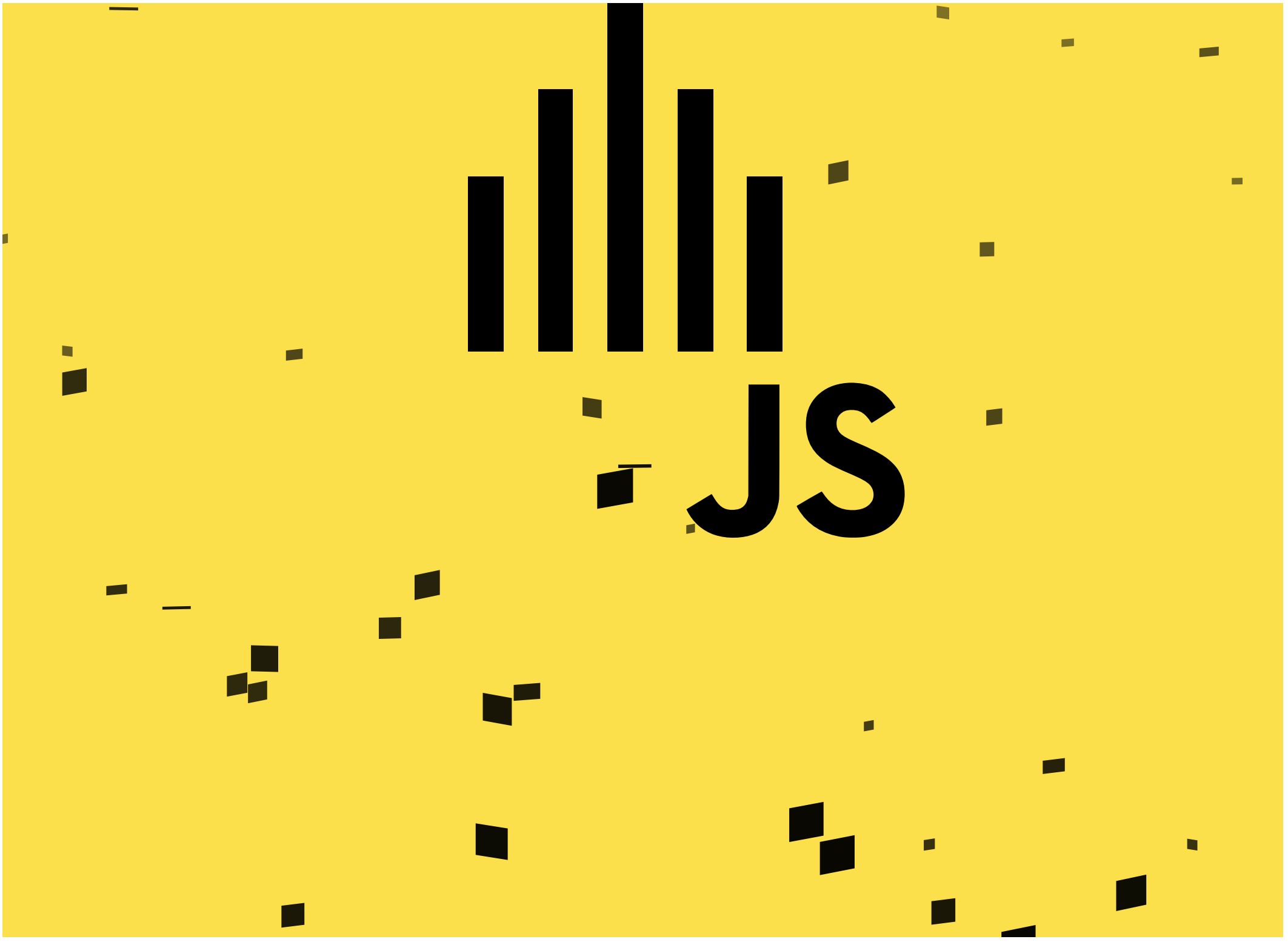
Milano JS

Web Animation API

The vengeance




JS

Davide Di Pumpo

- Full stack designer (?)
- Senior Frontend developer at Credimi
- Co-organizer Milano Frontend
- **MakhBeth** on: Twitter, Github, Internet
- I like Cats, Scotch, Comics and Videogames...


LET'S START MEOW

Cosa sono?

> The Web Animations API provides a common language for browsers and developers to describe animations on DOM elements.

MDN

Come animiamo oggi?

(In classifica)

- `setInterval`
- `jQuery.animate()`
- `requestAnimationFrame()`
- CSS animations o librerie come GSAP e Velocity

Set interval

```
var elem = document.getElementById("animatedElem");
var left = 0;
var lastFrame = +new Date;
var timer;
timer = setInterval(function() {
var now = +new Date,
        deltaT = now - lastFrame;
elem.style.left = ( left += 10 * deltaT / 16 ) + "px";
lastFrame = now;
if ( left > 400 ) {
        clearInterval( timer );
}
}, 16);
```

Set interval

seriamente?

- Impreciso
- Prestazionalmente osceno
- Comodo da usare come un registratore betamax

jQuery Animate

```
$( "#book" ).animate({
        opacity: 0.25,
        left: "+=50",
        height: "toggle"
}, 5000, function() {
        // Animation complete.
});
```

jQuery Animate

- < 3.0 Usa setInterval
- È utile solo se usate già jQuery
- Sì ok, ma state usando jQuery?

Request Animation Frame

```
var start = null;
var element = document.getElementById('SomeElementYouWantToAnimate');
element.style.position = 'absolute';

function step(timestamp) {
        if (!start) start = timestamp;
        var progress = timestamp - start;
        element.style.left = Math.max(progress / 10, 200) + 'px';
        if (progress < 2000) {
                window.requestAnimationFrame(step);
        }
}

window.requestAnimationFrame(step);
```

Request Animation Frame

- Ancora troppo boilerplate
- Il prossimo repaint non vuol dire avere 60fps
- L'animazione è finita? Voglio modificarla? Voglio cambiare easing?

CSS


I poteri forti hanno censurato questo snippet

CSS

Sono ok per la maggiorparte dei casi ma:

- Non possono essere composte sullo stesso elemento
- È difficile lanciarle in parallelo
- È impossibile animare valori che cambiano dinamicamente (provate un vertical accordion con testo ed height auto)!

Librerie esterne

Come se non avessimo già abbastanza dipendenze

Ok se fate un utilizzo massivo di animazioni del progetto, ma:

- È comunque una libreria esterna
- Probabilmente state sparando ad una mosca con un cannone
- Per quanto ne dica GSAP (140kB) non sono uno standard

Quindi?

ste uebbe animescion apiai?

vediamo un po' di codice

```
var player = document.getElementById('toAnimate').animate([
  { transform: 'scale(1)', opacity: 1, offset: 0 },
  { transform: 'scale(.5)', opacity: .5, offset: .3 },
  { transform: 'scale(.6)', opacity: .6, offset: 1 }
], {
  duration: 700, //milliseconds
  easing: 'ease-in-out', //'linear', a bezier curve, etc.
  delay: 10, //milliseconds
  iterations: Infinity, //or a number
  direction: 'alternate', //'normal', 'reverse', etc.
  fill: 'forwards' //'backwards', 'both', 'none', 'auto'
});
```

Un pelo più dichiarative, così dichiarative che potremmo scriverle in puro CSS

```
@keyframes emphasis {
  0% {
    transform: scale(1);
    opacity: 1;}
  30% {
    transform: scale(.5);
    opacity: .5;}
  100% {
    transform: scale(.6);
    opacity: .6;}
}
#toAnimate {
  animation: emphasis 700ms ease-in-out 10ms infinite alternate forwards;
}
```

Senza timore dei poteri forti

```
var player = document.getElementById('toAnimate').animate([
  { transform: 'scale(1)', opacity: 1, offset: 0 },
  { transform: 'scale(.5)', opacity: .5, offset: .3 },
  { transform: 'scale(.6)', opacity: .6, offset: 1 }
], {
  duration: 700, //milliseconds
  easing: 'ease-in-out', //'linear', a bezier curve, etc.
  delay: 10, //milliseconds
  iterations: Infinity, //or a number
  direction: 'alternate', //'normal', 'reverse', etc.
  fill: 'forwards' //'backwards', 'both', 'none', 'auto'
});
```

Andando un po’ più a fondo

```
var animation = element.animate(keyframes, options);
```

`keyframes`

È un oggetto che contiene tutte le proprietà animabili, può essere formattato in due modi:

```
element.animate({
  opacity: [ 0, 1 ],           // [ from, to ]
  color:   [ "#fff", "#000" ] // [ from, to ]
}, 2000);
```

oppure:

```
element.animate([
  { // from
    opacity: 0,
    color: "#fff"
  },
  { // to
    opacity: 1,
    color: "#000"
  }
], 2000);
```

`options`

Un **intero** che rappresenta la durata dell'animazione in millisecondi oppure un **oggetto** che contiene una o più delle seguenti proprietà:

- `id`
- `delay`
- `direction`
- `duration`
- `easing`
- `endDelay`
- `fill`
- `iterationStart`
- `iterations`

Sì, in pratica sono le proprietà **`animation-`** del CSS esposte su JS

E fin qua è quasi CSS in JS

La console di Chrome

```
> document.getElementsByTagName('body')[0].animate({})
<· ▼ Animation {startTime: null, currentTime: 0,
     playbackRate: 1, playState: "pending", id: ""…} i
     currentTime: 0
     id: ""
     oncancel: null
     onfinish: null
     playState: "finished"
     playbackRate: 1
     startTime: 63899.18399992166
   ▶ __proto__: Animation
```

Esatto, `animate()` ritorna un oggetto di tipo Animation con i propri metodi e proprietà. Ed è qui che arriva il bello!

Metodi

- `cancel()` torna allo stato iniziale
- `finish()` salta alla fine del dell'animazione
- `pause()` mette l'animazione in pausa
- `play()` mette l'animazione in esecuzione
- `reverse()` riproduce l'animazione al contrario

Proprietà

- `currentTime` mostra/setta il tempo di esecuzione in ms
- `id` ritorna l'id
- `oncancel` esegue una funzione quando l'animazione viene cancellata dall'apposito metodo
- `onfinish` esegue una funzione quando l'animazione termina
- `playbackRate` la "velocità" dell'animazione, può essere negativa
- `playState` mostra/setta lo stato di esecuzione
- `startTime`

Promise

```
//set up 1 second animation to fade box out
var box = document.getElementById('box');
var animation = box.animate([{ opacity: 1 }, { opacity: 0 }], 1000);

function finishedHandler() {
  console.log('animation finished: ' + Date.now());
}
function canceledHandler() {
  console.log('animation canceled: ' + Date.now());
}

//the Promise version, log the timestamp when the Animation finishes
animation.finished.then(finishedHandler, canceledHandler);

//the effective successful equivalent with the onfinish callback
animation.onfinish = finishedHandler;
```

Dan Wilson

Promise

- `finished`
- `ready`

Altre cosucce da sapere

Un elemento può ovviamente avere più animazioni

```
var animated = document.getElementById('cat');
var purr = animated.animate({}, 1000);
var rotate = animated.animate({}, 2000);
var jump = animated.animate({}, 3500);
```

Potete vedere tutte le animazioni presenti attraverso il metodo:

```
document.getAnimations()
```

Perchè e quando usarle?

#OpinionePersonale

Se l'animazione è funzionale va in JS, se è estetica nel CSS

Un **accordion**, un **tooltip** una **sidebar**, un **dropdown** fanno della loro comparsa un motivo d'esistere
L'effetto hover su un bottone è *enhancement*

Ci sono alcuni casi come l'*altezza* la *posizione* in base alla viewport o ad altri elementi in pagina, in cui JS è necessario

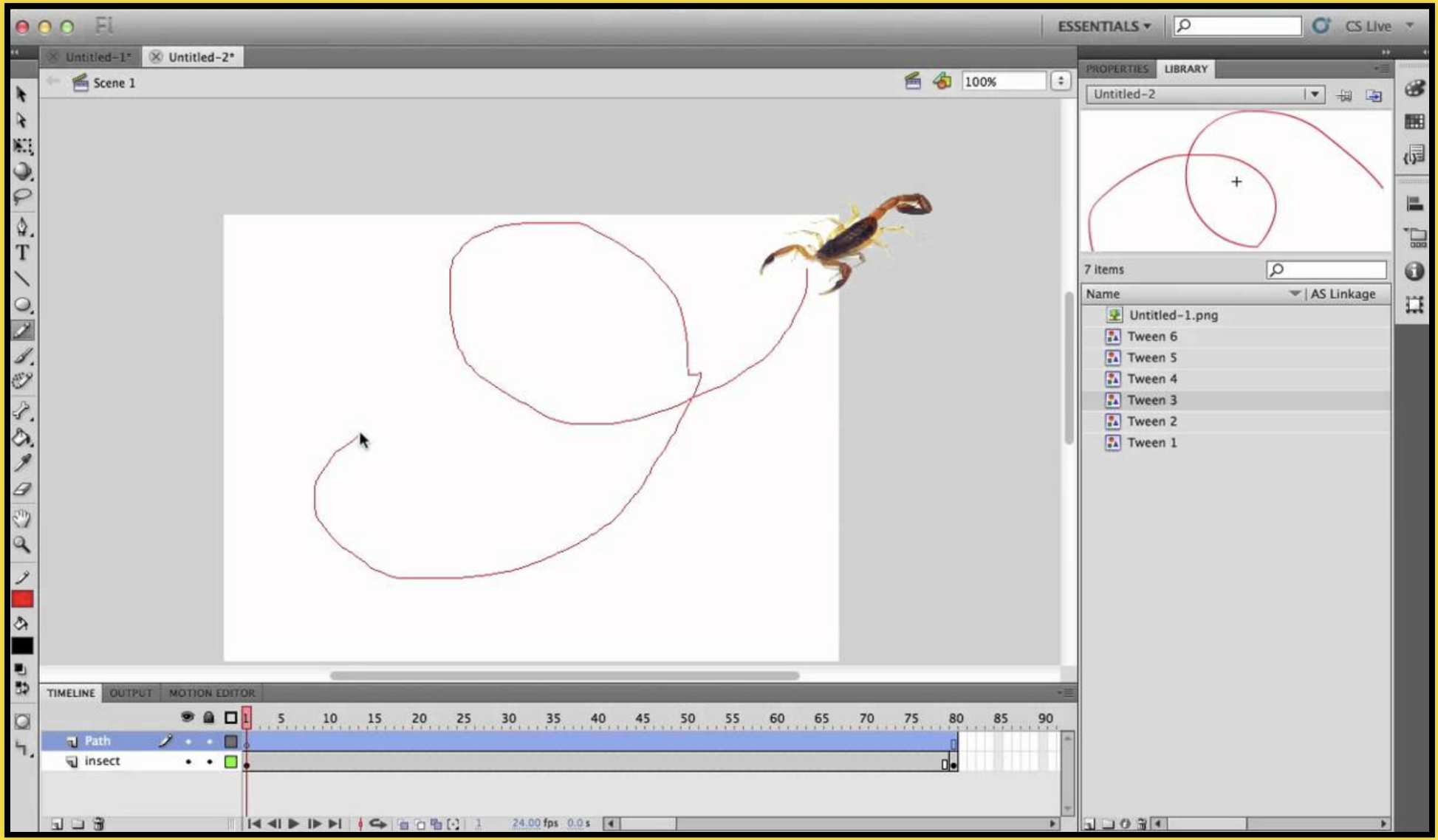
LA POTENZA È NULLA SENZA CONTROLLO

HTML CSS JS Result

EDIT ON

Ma veniamo al momento da tutti agognato!

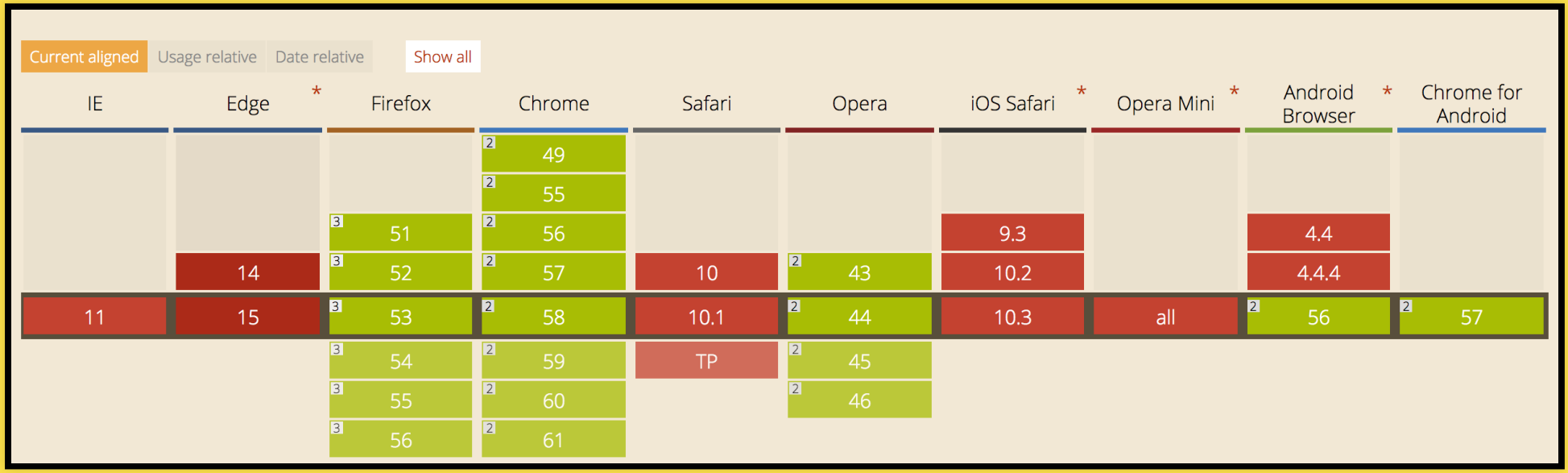
Il momento del ricordo


ESSENTIALS
CS Live
Untitled-1*
Untitled-2*
Scene 1
100%
PROPERTIES
LIBRARY
Untitled-2
7 items
Name
AS Linkage
Untitled-1.png
Tween 6
Tween 5
Tween 4
Tween 3
Tween 2
Tween 1
TIMELINE
OUTPUT
MOTION EDITOR
Path
insect
24.00 fps
0.0 s

È possibile animare su un path!!1!1

```
var path = `path("M300.358,297.675c55.955,57.158 83.03,107.698 141.392,107.698c58
var p = document.getElementById('cat2');
var animation = p.animate({
        offsetPath: [path, path ],
        offsetDistance: [0, "100%"]
}, {
        duration: 4000,
        iterations: Infinity
});
```

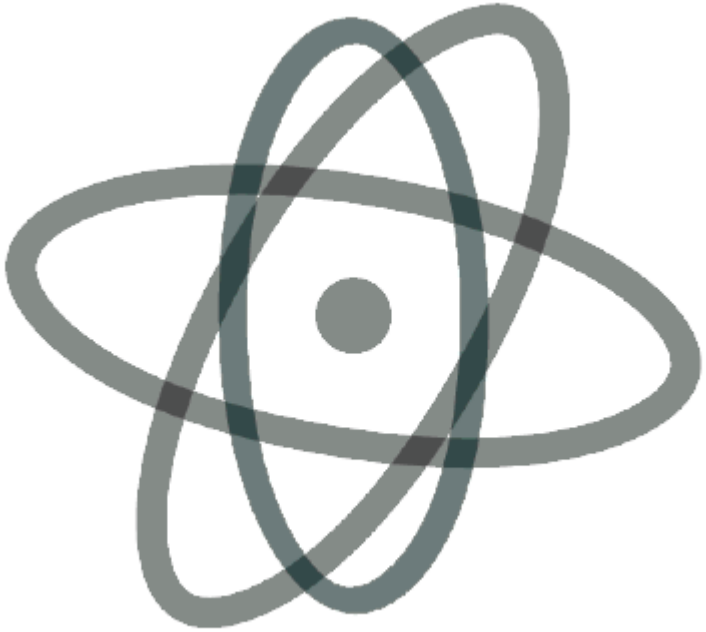
Utilizzarle oggi?


Current aligned
Usage relative
Date relative
Show all
IE
Edge *
Firefox
Chrome
Safari
Opera
iOS Safari *
Opera Mini *
Android Browser *
Chrome for Android
11
15
3 53
3 54
3 55
3 56
2 49
2 55
2 56
2 57
2 58
2 59
2 60
2 61
10
10.1
TP
2 43
2 44
2 45
2 46
9.3
10.2
10.3
all
4.4
4.4.4
2 56
2 57

Ma abbiamo un polyfill

e iniziano ad apparire progetti interessanti

react-web-animation



```
import { Component } from 'react';
import { Animated } from 'react-web-animation';
export default class Basic extends Component {
        getKeyFrames() {
                return [
                        { transform: 'scale(1)',    opacity: 1,     offset: 0 },
                        { transform: 'scale(.5)',   opacity: 0.5,   offset: 0.3 }
                        { transform: 'scale(.6)',   opacity: 0.6,   offset: 1 }
                ];
        }
        getTiming( duration ) {
                return {
                        duration,
                        easing: 'ease-in-out',
                        delay: 0,
                        iterations: 2,
                        direction: 'alternate'
```

Spe, parlando di performance?

CSS-based animations, and Web Animations where supported natively, are typically handled on a thread known as the "**compositor thread**". This is different from the browser's "main thread", where styling, layout, painting, and JavaScript are executed. This means that if the browser is running some expensive tasks on the main thread, these animations can keep going without being interrupted.

Google developers

Quindi?

Plz, possiamo usarle, facciamolo.

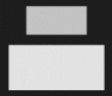
Il web non è un file .sketch o .psd, la user interface può trarre grandi vantaggi dalle animazioni

The 12 Principles of UX in Motion

CREATING USABILITY WITH MOTION

Easing

Offset & Delay

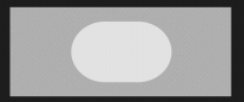


Parenting

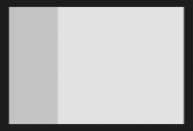
Transformation

7

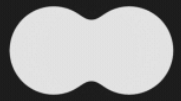
Value change



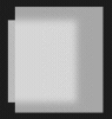
Masking



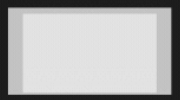
Overlay



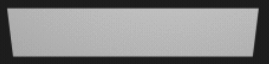
Cloning



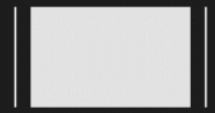
Obscuration



Parallax

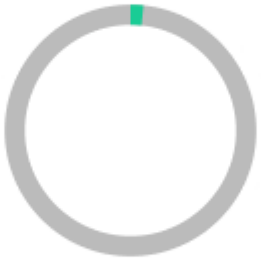


Dimensionality



Dolly & Zoom

uxinmotion.net



Credit: Colin Garven


THANKS
FOR YOUR ATTENTION
quickmeme.com

Bibliography

- Mozilla Developers Network
- Chrome platform status
- WAAPI Tutorial
- Intro to WAAPI
- Exploring WAAPI
- Motion in UX