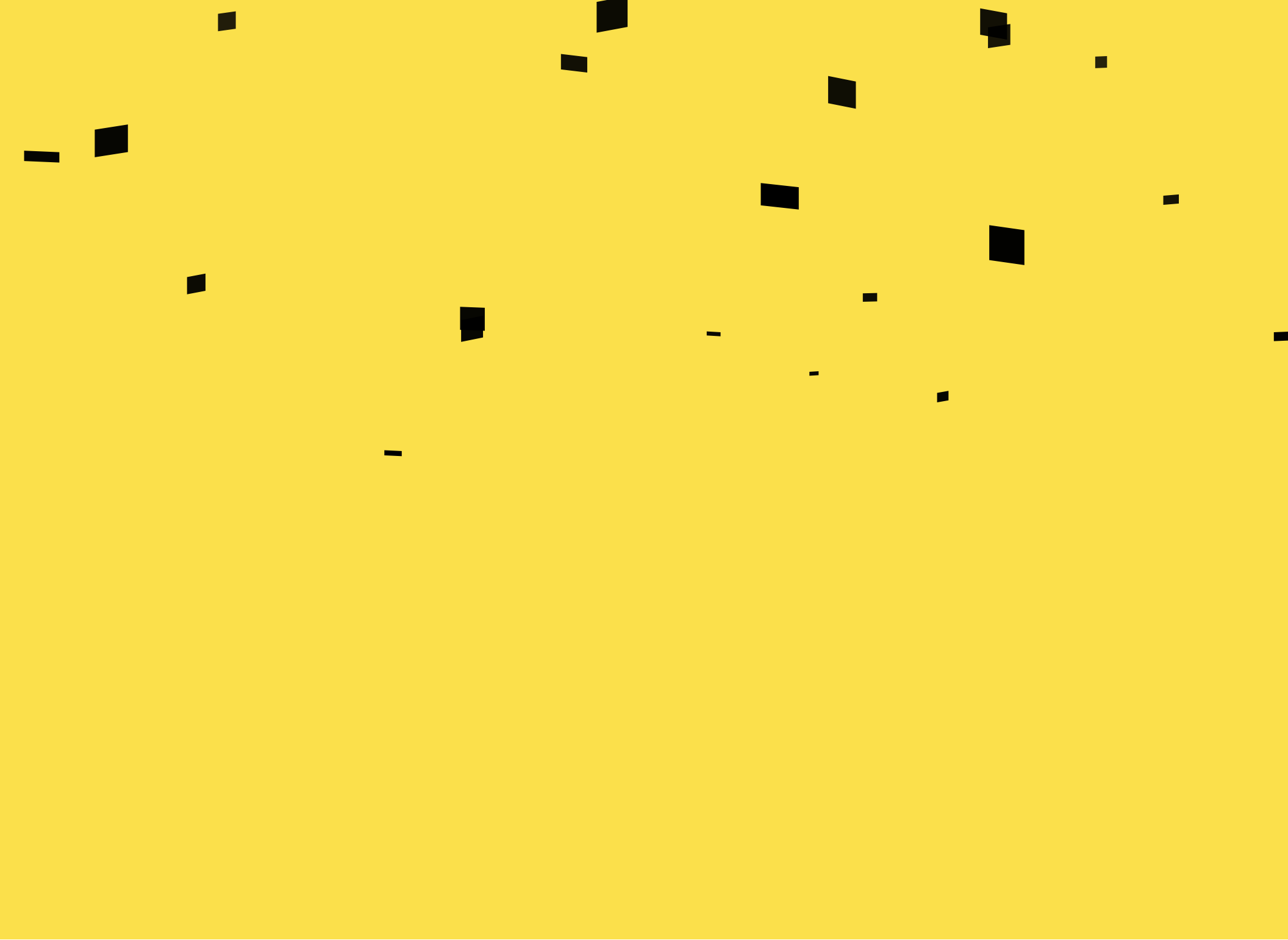


Milano JS

Web Animation API



JS



LET'S START MEOW

A black and white photograph of two cats acting as DJs. In the foreground, a black and white cat is scratching a vinyl record on a turntable. In the background, an orange cat is also on a turntable. Between them is a mixer with many knobs and sliders. The scene is set in a room with a bookshelf and a window with blinds.

Anche perché in un minuto vi ho fatto vedere n'animazione e ringraziato tutti, mancano solo 6 minuti, non vorrei andare lungo che poi chi te lo sente Granze. Mo' fa finta di ridere, ma guardategli gli occhi. Io c'ho paura.

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

seriamente?

È impreciso e le animazioni possono scattare parecchio

jQuery Animate

Oltre al fatto di avere di mezzo jQuery, ha seri problemi prestazionali

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);
```

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 non sono uno standard

Quindi?

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'
});
```

Oppure 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;  
}
```



Achievement unlocked
Full CSS slide at MilanoJS

```
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: ""...} ⓘ  
    currentTime: 0  
    id: ""  
    oncancel: null  
    onfinish: null  
    playState: "finished"  
    playbackRate: 1  
    startTime: 63899.183999992166  
    ▶ __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 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

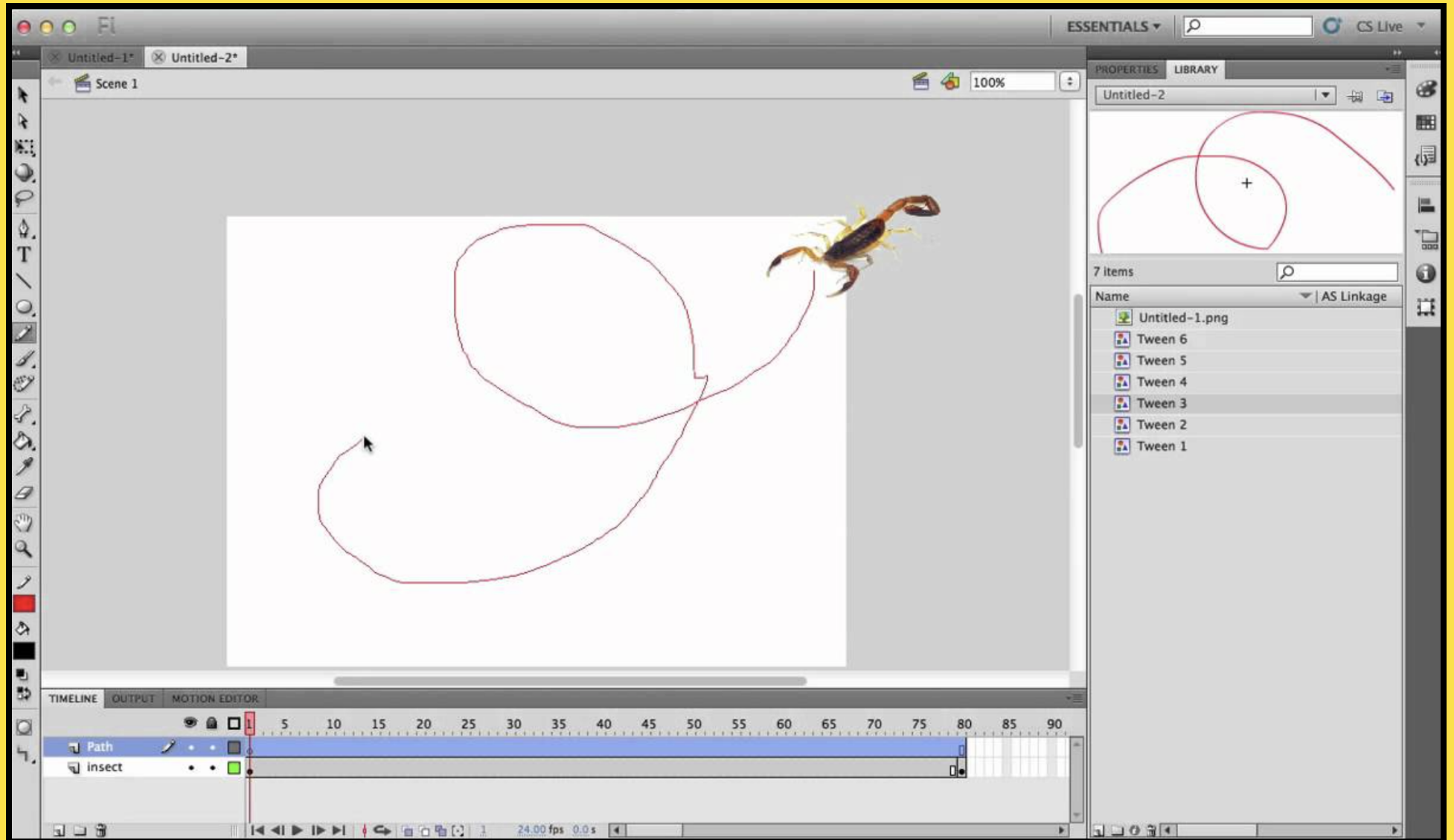
```
var animated = document.getElementById('cat');  
var purr = animated.animate({}, 1000);  
var rotate = animated.animate({}, 2000);  
var jump = animated.animate({}, 3500);
```

**Potete vedere tutte le animazioni presenti
attraverso il metodo:**

```
document.getAnimations( )
```

Ma veniamo al momento da tutti agognato!

Il momento del ricordo



**È possibile
animare su un
path!!1!1**

MillanoJS



```
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?

<div>Current alignedUsage relativeDate relativeShow all</div>									
IE	Edge*	Firefox	Chrome	Safari	Opera	iOS Safari*	Opera Mini*	Android Browser*	Chrome for Android
			² 49						
			² 55						
		³ 51	² 56			9.3		4.4	
		³ 52	² 57			10.2		4.4.4	
11	14			10	² 43				
		³ 53	² 58	10.1	² 44	10.3	all	² 56	² 57
		³ 54	² 59	TP	² 45				
		³ 55	² 60		² 46				
		³ 56	² 61						

Ma abbiamo un polyfill

e tranquilli che c'è già pronto per il vostro framework preferito

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



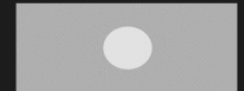
Parenting



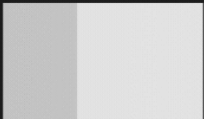
Transformation

10

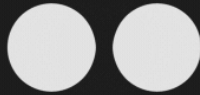
Value change



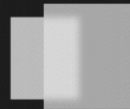
Masking



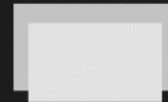
Overlay



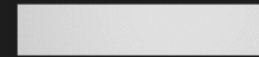
Cloning



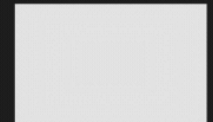
Obscuration



Parallax



Dimensionality



Dolly & Zoom

uxinmotion.net

THANKS



FOR YOUR ATTENTION

Bibliography

- [Mozilla Developers Network](#)
- [Chrome platform status](#)
- [WAAPI Tutorial](#)
- [Intro to WAAPI](#)
- [Exploring WAAPI](#)
- [Motion in UX](#)