

tt()

# Schedule

1. Storytelling
2. Transitions
3. CSS Keyframes
4. JS Animations



# Schedule

1. Storytelling
2. Transitions
3. CSS Keyframes
4. JS Animations



# Schedule

1. Storytelling
2. Transitions
3. CSS Keyframes
4. JS Animations



# Storytelling



# Storytelling

So why do we use storytelling?

# Storytelling

1. It is a way of connecting people.
2. It is ambiguous.
3. Stories *stick* with people.

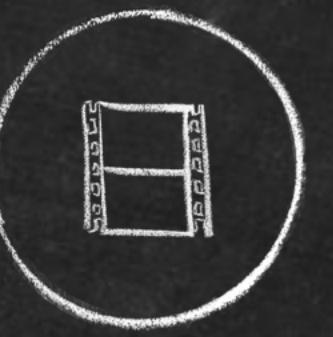
 English ▾

— THE —  
**Fallen**  
— OF —  
**World War II**



INTERACTIVE

Requires a modern computer.  
Experimental.

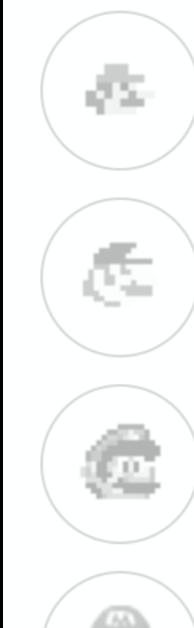


VIDEO

Better for mobile, Apple TV,  
Chromecast ...



START



FINISH

IGN Presents

# THE MUSEUM OF MARIO

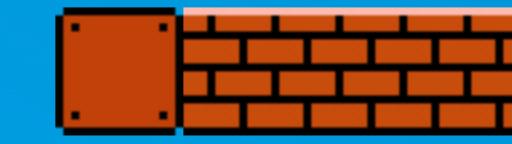
An interactive experience exploring the many eras of Mario. Turn on the audio and click around to find hidden interactions!

Built by Intel's HTML5 Hub in collaboration with IGN

<https://mario.ign.com/>



Share Your #MarioMemories



# Storytelling

So, how can we do it?

# Storytelling

Explain the why.

# Storytelling

Emotion is a big seller.

# Storytelling

Make it simple.

# Storytelling

Create anticipation

# Storytelling

Use the information gap.

# Schedule

1. Storytelling
- 2. Transitions**
3. CSS Keyframes
4. JS Animations

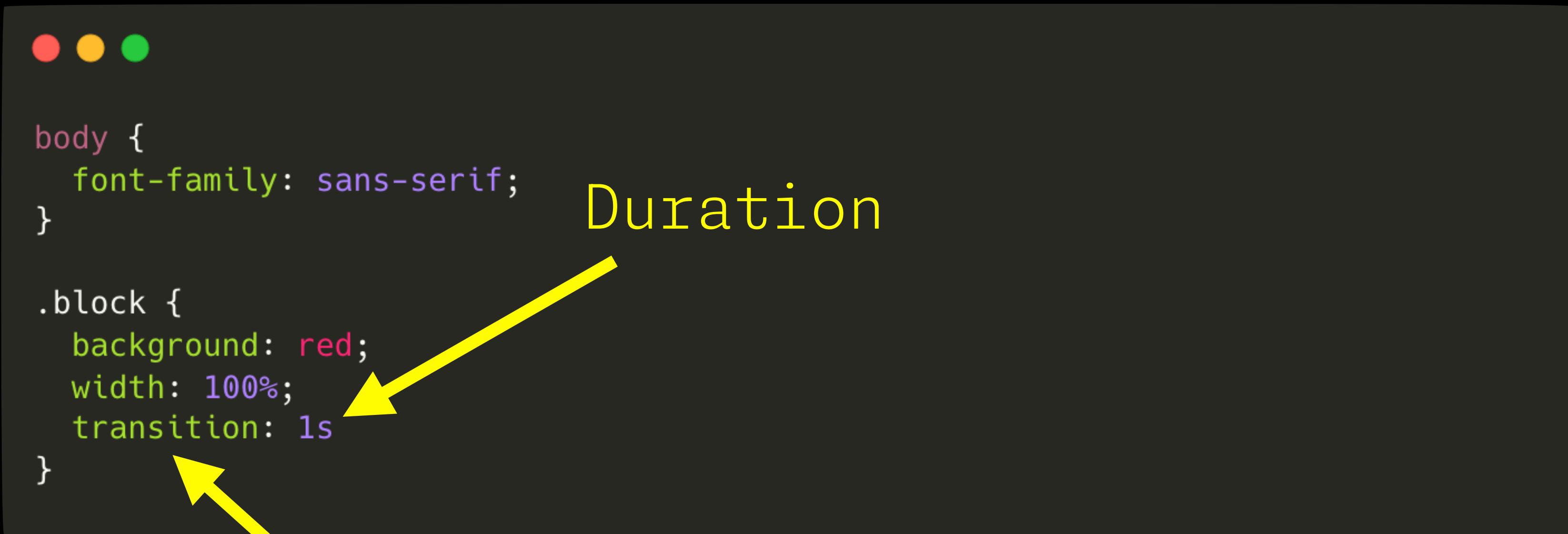


# Transitions



```
body {  
    font-family: sans-serif;  
}  
  
.block {  
    background: red;  
    width: 100%;  
    transition: 1s  
}
```

# Transitions



```
● ● ●  
body {  
    font-family: sans-serif;  
}  
  
.block {  
    background: red;  
    width: 100%;  
    transition: 1s  
}
```

Duration

Property

# Storytelling

1. We use it to transition between states, such as hover, focus, active etc.
2. It is applicable on many CSS properties
3. It is extremely memory efficient in comparison to the alternatives.

# Transitions in D3

<https://www.d3indepth.com/transitions/>

<https://d3js.org/d3-transition>

# Storytelling

However, transitions require interactivity and user input

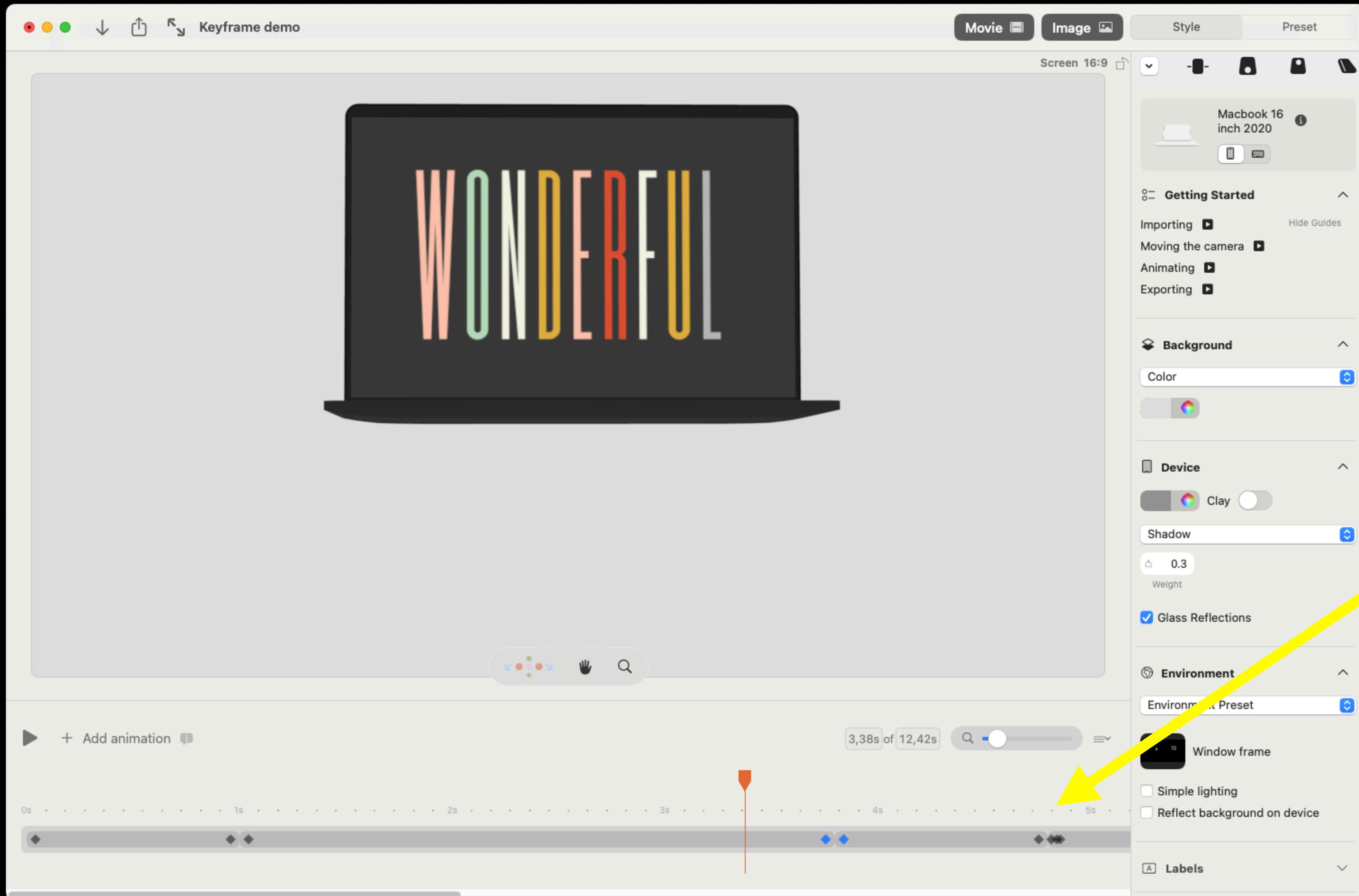
What if we don't want that?

# Schedule

1. Storytelling
2. Transitions
- 3. CSS Keyframes**
4. JS Animations



# Keyframes



This is a timeline that includes some keyframes

# Keyframes

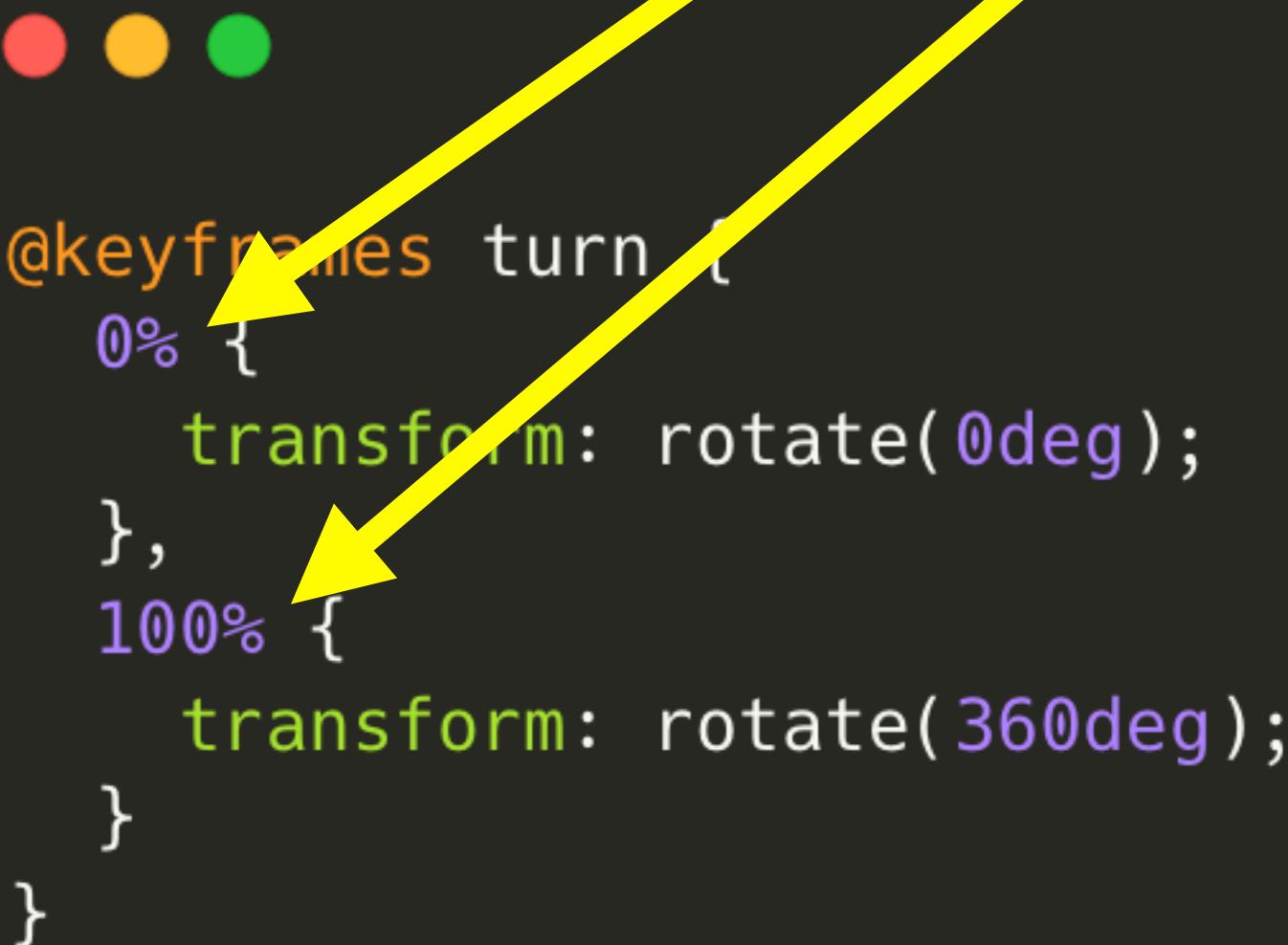
We can write them in CSS



```
@keyframes turn {  
  0% {  
    transform: rotate(0deg);  
  },  
  100% {  
    transform: rotate(360deg);  
  }  
}
```

# Keyframes

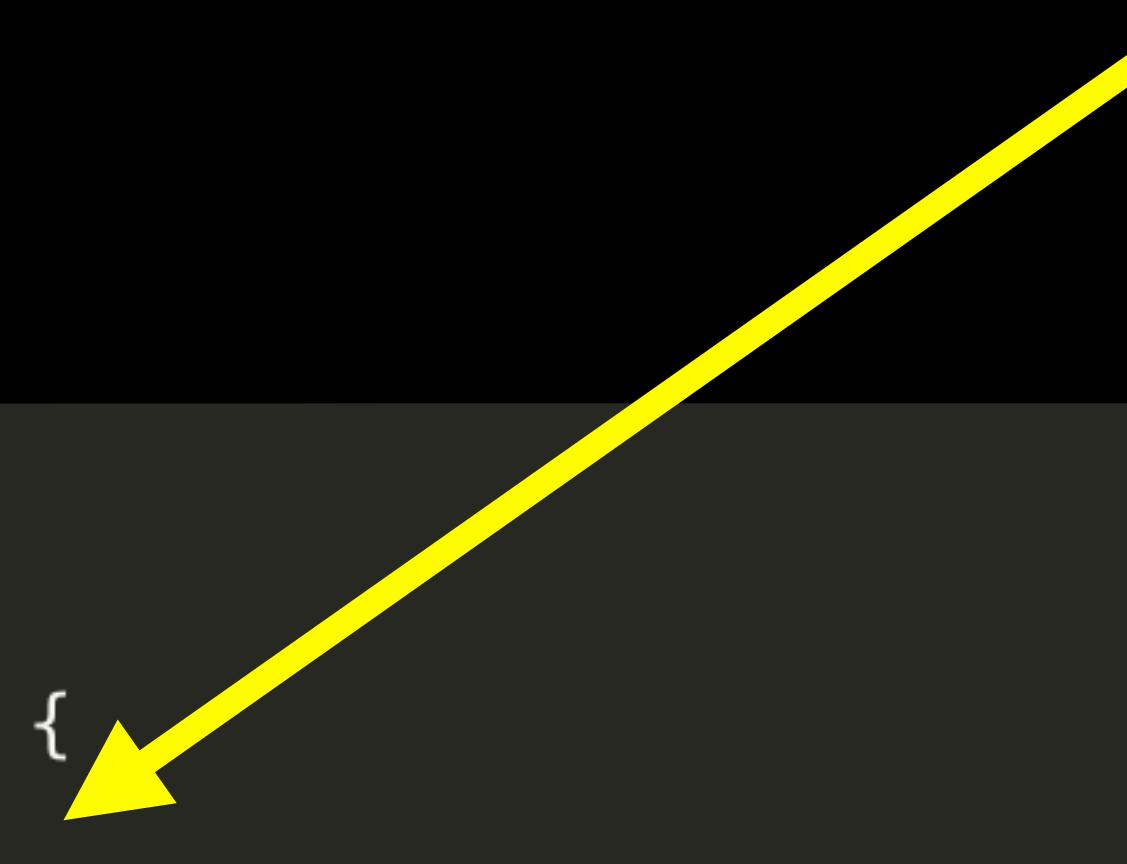
The percentage is the relative location on the invisible “timeline”



```
@keyframes turn {
  0% {
    transform: rotate(0deg);
  },
  100% {
    transform: rotate(360deg);
  }
}
```

# Keyframes

Inside here is what actually happens



```
● ○ ●  
  
@keyframes turn {  
  0% {  
    transform: rotate(0deg);  
  },  
  100% {  
    transform: rotate(360deg);  
  }  
}
```

# Keyframes

Let's fire up a codepen.

<https://codepen.io/vijnv/pen/oNmodWp>

# Schedule

1. Storytelling
2. Transitions
3. CSS Keyframes
- 4. JS Animations**



# JS Animations

1. It gives us an enormous amount of control and flexibility
2. Although that is not always needed.

# JS Animations



<https://greensock.com/>

# JS Animations

Your 2022  
Neighbourly  
Impact

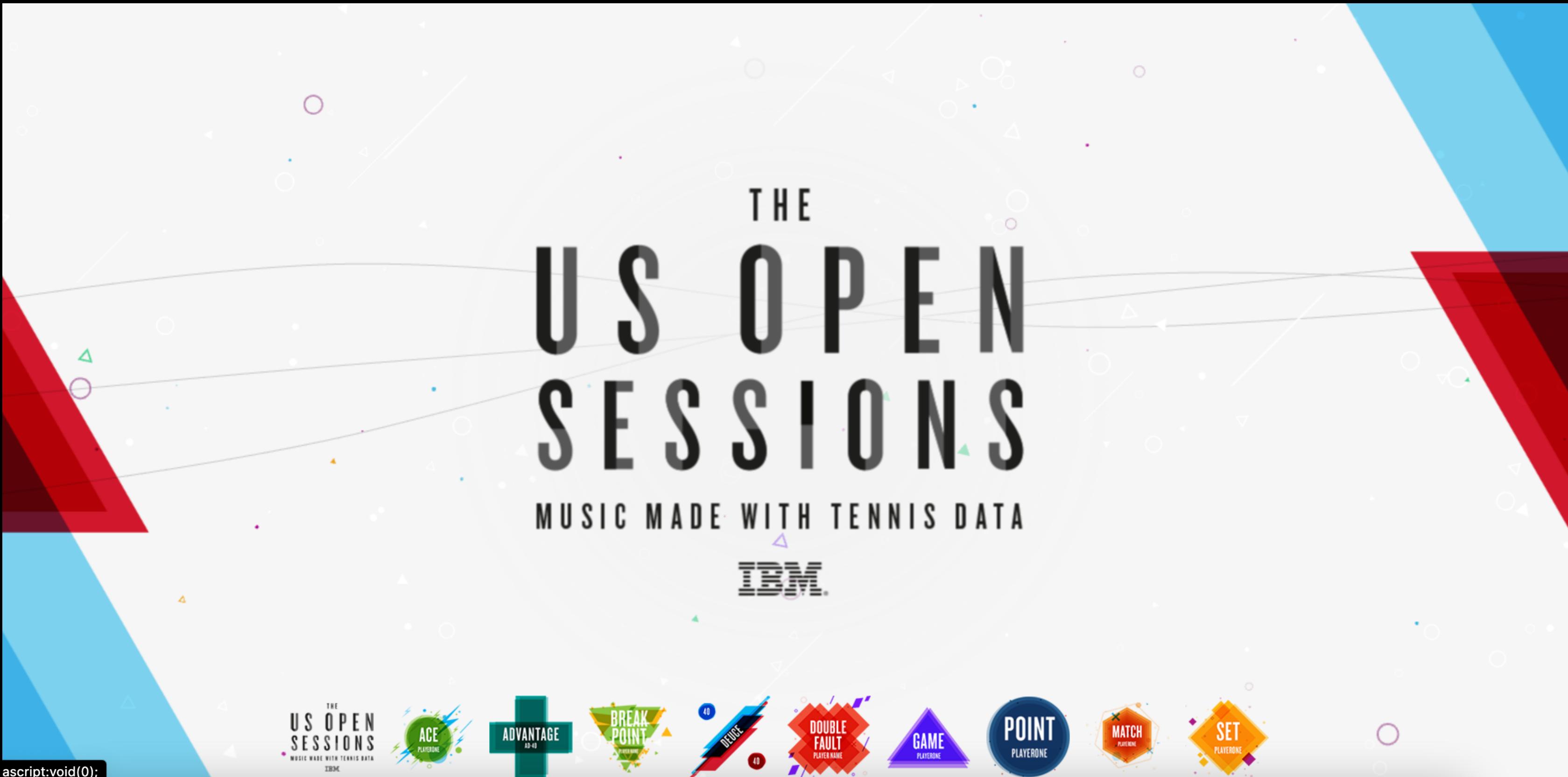
This experience is best  
with sound enabled.



Enter

<https://www.impact.compassion.com.au/>

# JS Animations



<https://www.shanemielke.com/archives/usopen-sessions/>

# JS Animations



## GSAP 3 Cheat Sheet [PDF](#)

Most code is linked to the appropriate page in the [Docs](#)

Links: [Get started](#) | [Install](#) | [Forums](#) | [Tips](#) | [Learning](#) | [CodePen](#) | [Club](#)

### Basics

```
// "to" tween (animate to provided values)
gsap.to(".selector", { // selector text, Array, or object
  x: 100, // any properties (not limited to CSS)
  backgroundColor: "red", // camelCase
  duration: 1, // seconds
  delay: 0.5,
  ease: "power2.inOut",
  stagger: 0.1, // stagger start times
  paused: true, // default is false
  overwrite: "auto", // default is false
  repeat: 2, // number of repeats (-1 for infinite)
  repeatDelay: 1, // seconds between repeats
  repeatRefresh: true, // invalidates on each repeat
  yoyo: true, // if true > A-B-B-A, if false > A-B-A-B
  yoyoEase: true, // or ease like "power2"
  immediateRender: false,
  onComplete: myFunc,
  // other callbacks:
  // onStart, onUpdate, onRepeat, onReverseComplete
  // Each callback has a params property as well
  // i.e. onUpdateParams (Array)
});

// "from" tween (animate from provided values)
```

[https://greensock.com/docs/v3/GSAP/gsap.to\(\)\\_Vars](https://greensock.com/docs/v3/GSAP/gsap.to()_Vars):

### Timelines

```
// Create a timeline
let tl = gsap.timeline({
  delay: 0.5,
  paused: true, // default is false
  repeat: 2, // number of repeats (-1 for infinite)
  repeatDelay: 1, // seconds between repeats
  repeatRefresh: true, // invalidates on each repeat
  yoyo: true, // if true > A-B-B-A, if false > A-B-A-B
  defaults: { // children inherit these defaults
    duration: 1,
    ease: "none"
  },
  smoothChildTiming: true,
  autoRemoveChildren: true,
  onComplete: myFunc,
  // other callbacks:
  // onStart, onUpdate, onRepeat, onReverseComplete
  // Each callback has a params property as well
  // i.e. onUpdateParams (Array)
});
```

```
// Sequence multiple tweens
tl.to(".selector", {duration: 1, x: 50, y: 0})
  .to("#id", {autoAlpha: 0})
  .to(elem, {duration: 1, backgroundColor: "red"})
```

<https://greensock.com/cheatsheet/>

# JS Animations

Let's fire up a codepen or two

<https://codepen.io/vijnv/pen/VwgrBee>

<https://codepen.io/vijnv/pen/abKwwgW>

# JS Animations

## Assignment:

Now it's your turn! Upgrade your SVG assignment from week 1 into an animation powered by GSAP.

**Uncaught SyntaxError  
Unexpected end of input**