Before we begin...

- Open up these slides:
  - https://goo.gl/NEPWDS

JavaScript & HTML

Learning Objectives

- **Identify** the differences between the Document Object Model and HTML
- **Explain** the methods and use the DOM in JavaScript
- **Manipulate** the DOM by using selectors and functions
- **Register** and trigger event handlers for JavaScript events

Agenda

- Adding JavaScript to the page
- How a browser renders a page
- **D**ocument **O**bject **M**odel
  - Selectors
  - Accessing Information
  - Creating Nodes
  - *Events*
- DevTools
- *Animations*

A quick review

- Functions
  - Parameters/arguments
  - Return values
- Scope
- Hoisting
- *Closures*
- *Higher Order Functions*

JS & Webpages

Approaches

How do we actually add JavaScript to an HTML document?

1. Attributes
2. Inline Scripts
3. **External Files**

Inline Attributes

```
<body onload="console.log('Welcome to my app!');">
```

The least desirable approach

Inline Scripts

```
<script>
    console.log("Welcome to my app");
</script>
```

Primarily used by back-end languages to pass data from the back-end to JavaScript

External Files with Scripts

```
<script src="main.js"></script>
```

The best approach!

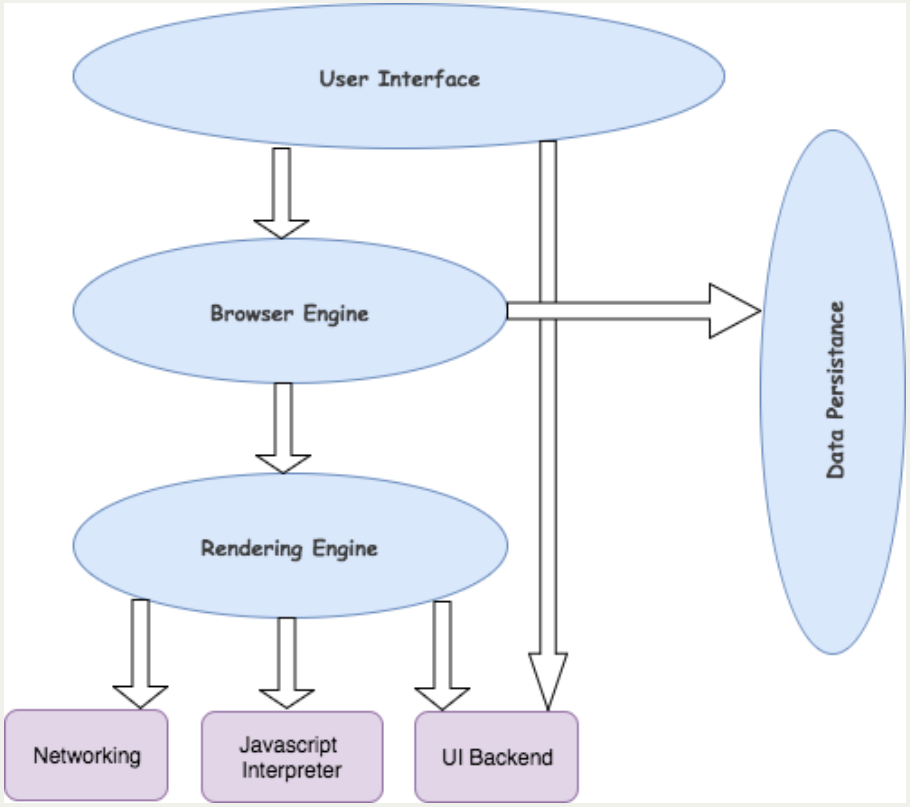
Make sure you wait until your HTML & CSS has loaded before you start manipulating them

Browsers?

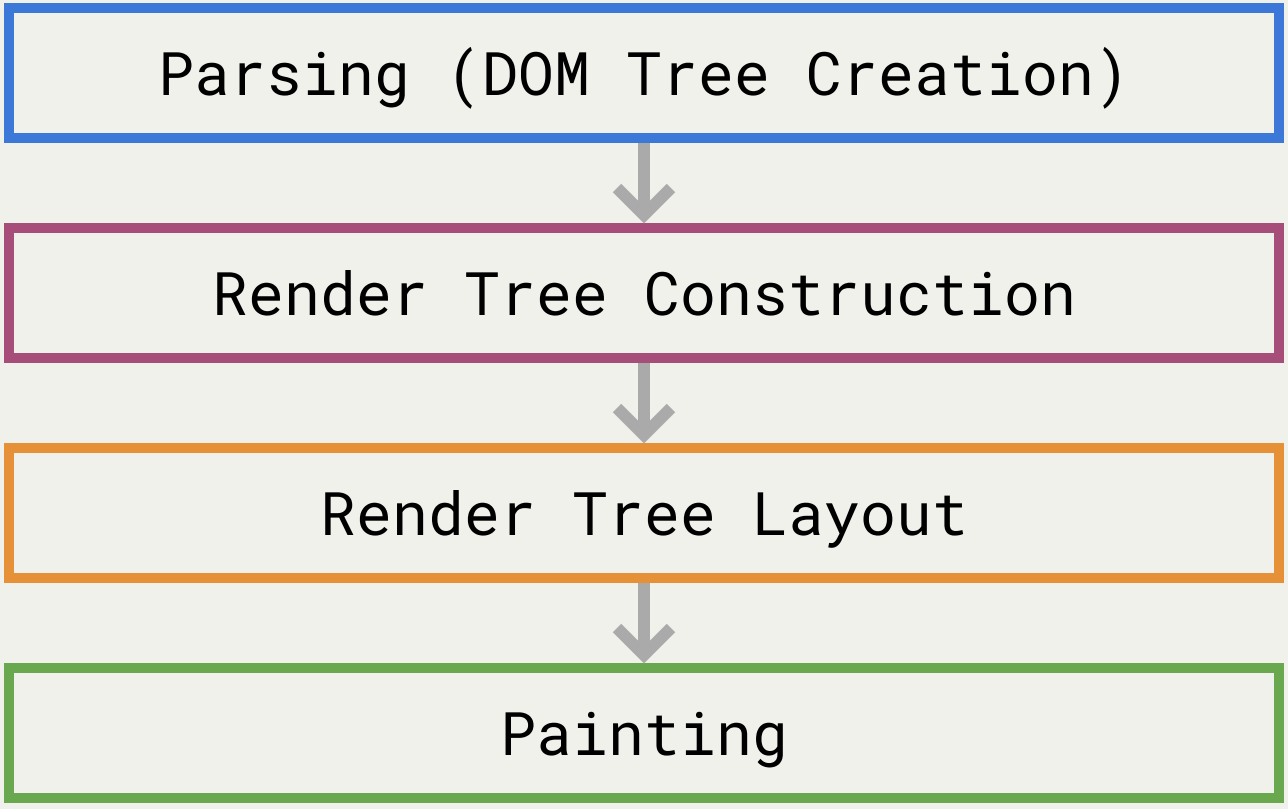
Browser Parts

- User Interface (search bar, menu etc.)
- Browser Engine (manipulates rendering engine)
- Rendering Engine (renders the page)
- Networking (retrieves URLs)
- UI Backend (draws basic widgets - not just for the browser)
- JavaScript Interpreter (interprets and executes JS)
- Data Storage (persistence layer)

Browser Parts



Rendering Engine



Resources

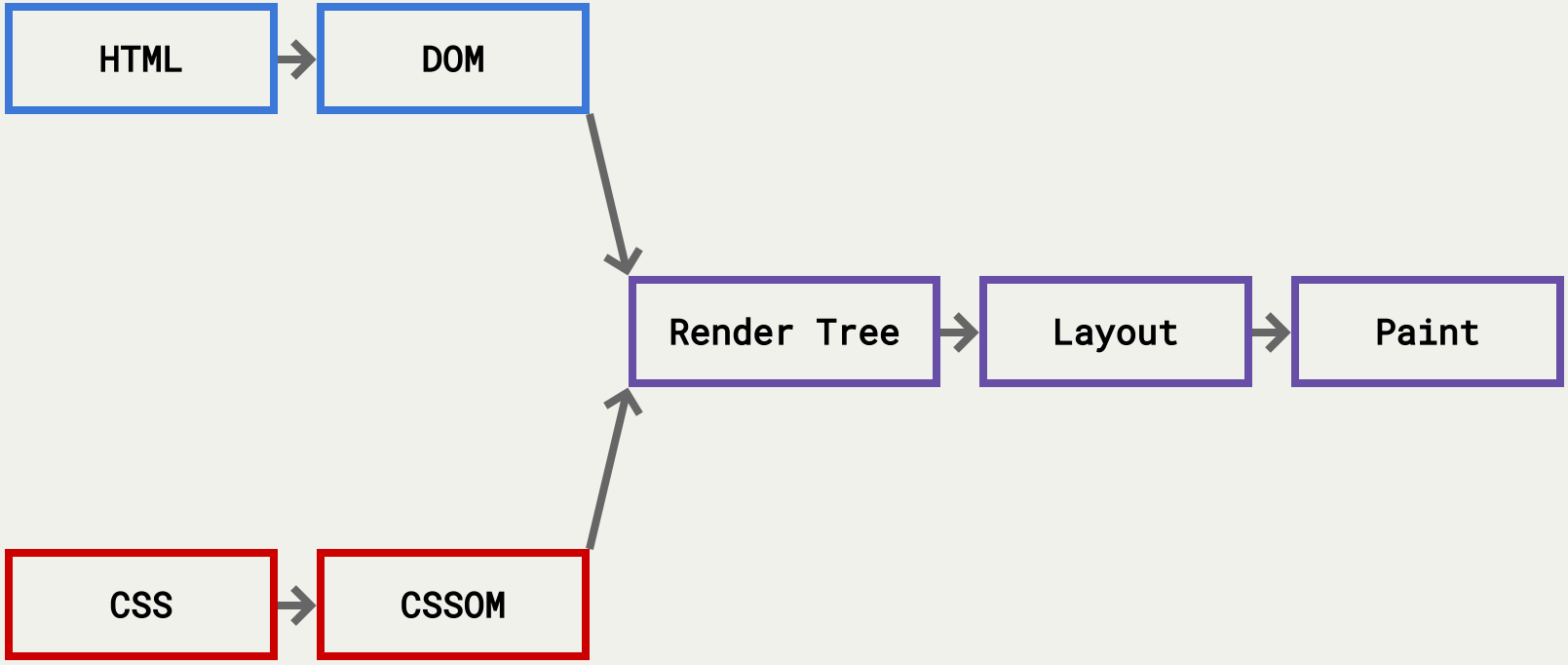
- Lin Clark: How do browsers work
  - Podcast by CodeNewbie
- HTML5 Rocks: How Browsers Work
- Moz://a Hacks: Building the DOM Faster
- Umar Hansa: An Introduction to Browser Rendering

Document Object Model

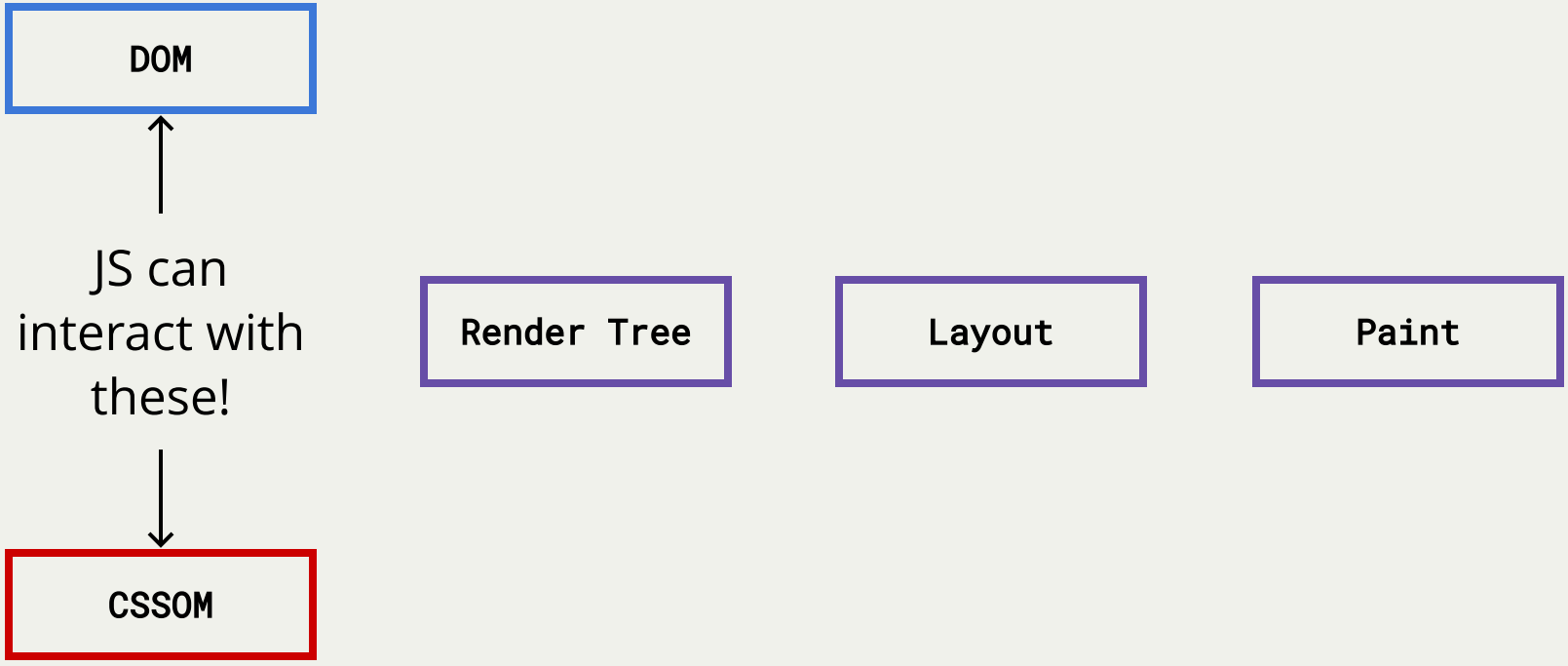
What is the DOM?

- It stands for the Document Object Model
- More or less, a (potentially) large object represented by the ***document*** variable
  - It has properties and methods
- It's a programming interface for HTML. The way that JavaScript can affect the page

Where does it come from?



Where does it come from?

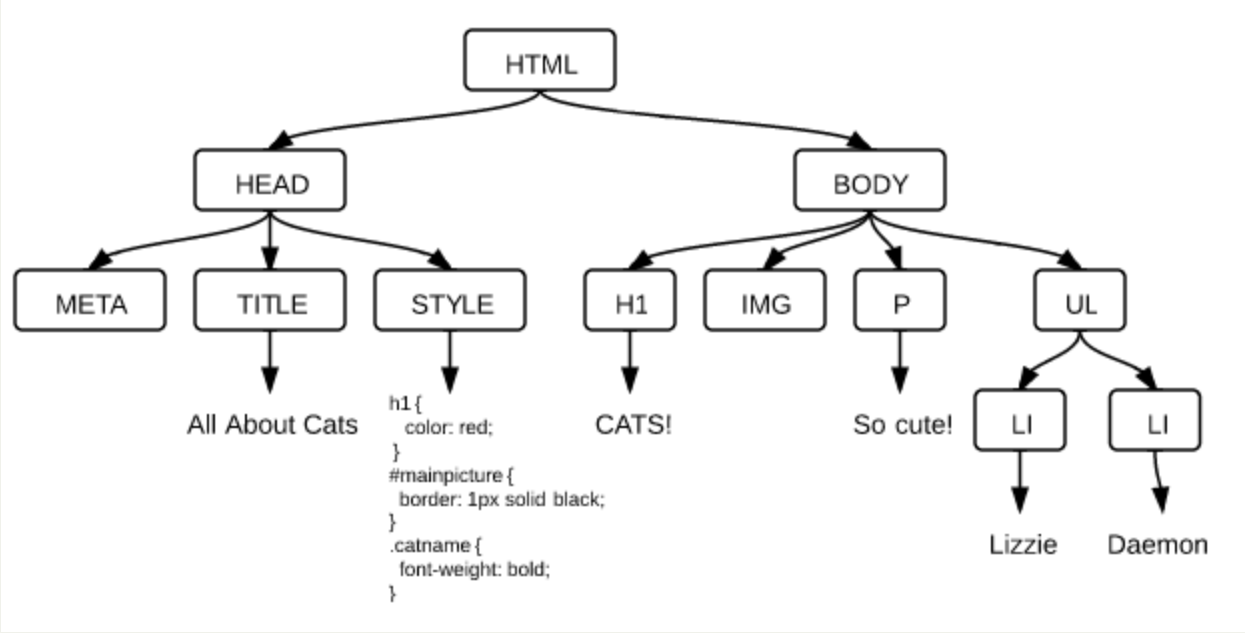


When the DOM changes...

When the DOM changes, the page gets updated!

1. You make a change to the DOM with JS
2. The browser creates a render tree
3. The browser figures out the layout tree
4. The browser re-paints the page

What does it look like?



Key Terms

- Each point of data is called a ***node***
- Each ***node*** can have ***parents***, ***children*** and ***siblings***
- The DOM is accessed through a global variable called:
  - `document`
- We can call methods and access properties - just like an object

Identify Away

```
<!DOCTYPE html>
<html>
<head>
    <title>Some website</title>
</head>
<body>
    <div class="container">
        <h1>Some heading</h1>
        <p>Some text</p>
        <a href="http://www.google.com">Some <span>link</span></a>
    </div>

    <ul>
        <li>A link</li>
        <li>Another link</li>
        <li>Another link</li>
    </ul>
</body>
</html>
```

DOM Access

The document object gives us ways of accessing the DOM, finding elements, changing styles, etc.

The general strategy for DOM manipulation:

- Find the DOM node by using an access method and store it in a variable
- Manipulate the DOM node by changing its attributes, style, inner HTML, or by appending nodes to it

document.querySelector

```
document.querySelector( CSS SELECTOR );
```

```
<h1>Our App</h1>

<p>Welcome</p>

<ul>
    <li>Item</li>
</ul>
```

```
var heading = document.querySelector("h1");

var para = document.querySelector("p");


var item = document.querySelector("ul li");
```

Returns the ***first*** DOM node that matches a given CSS selector

document.querySelectorAll

```
document.querySelectorAll( CSS SELECTOR );
```

```
<p>First para</p>
<p>Second para</p>

<ul>
    <li>Item 1</li>
    <li>Item 2</li>
</ul>
```

```
var paras = document.querySelectorAll("p");


var items = document.querySelectorAll("li");
```

Returns ***all*** DOM nodes that match a given CSS selector
(as an array-like thing called a NodeList)

Exercise

The DOM Detective

DOM Traversal

```
<div>
  <h1>Hi</h1>
  <p>P tag</p>
  <h3>Heading</h3>
</div>
```

```
var pTag = document.querySelector("p");

pTag.children;

pTag.childNodes;

pTag.parentNode;

// Let's create a sibling function!
```

el.getAttribute

```
<img src="http://fillmurray.com/200/200" alt="Bill!">

<a href="https://ga.co" id="generalAssembly">
    A link to GA
</a>
```

```
var image = document.querySelector("img");

var srcText = image.getAttribute("src");
var altText = image.getAttribute("alt");

var aTag = document.querySelector("a");

var href = aTag.getAttribute("href");
var id = aTag.getAttribute("id");
```

el.setAttribute

```
<img src="http://fillmurray.com/200/200" alt="Bill!">

<a href="https://ga.co" id="generalAssembly">
    A link to GA
</a>
```

```
var image = document.querySelector("img");

var srcText = image.setAttribute("src", "http://placecage.com/200/200");
var altText = image.setAttribute("alt", "Another image");

var aTag = document.querySelector("a");

var href = aTag.setAttribute("href", "/home");
var id = aTag.setAttribute("id", "home");
```

el.setAttribute

```
<img src="http://fillmurray.com/200/200" alt="Bill!">

<a href="https://ga.co" id="generalAssembly">
    A link to GA
</a>
```

```
var image = document.querySelector("img");

var srcText = image.setAttribute("src", "http://placecage.com/200/200");
var altText = image.setAttribute("alt", "Another image");

var aTag = document.querySelector("a");

var href = aTag.setAttribute("href", "/home");
var id = aTag.setAttribute("id", "home");
```

HTML

```
<h1>Hello World</h1>
```

```
var heading = document.querySelector("h1");

var currentText = heading.innerText;
var currentHTML = heading.innerHTML;

heading.innerText = "This is the text";
heading.innerHTML = "<span>Hi there</span>";

heading.innerHTML += "!!!";
```

Getting Values

```
<input type="text" value="User types here">
```

```
var input = document.querySelector("input");

var currentValue = input.value;

input.value = "Something else";

var newValue = input.value;
```

Styles

```
<h1>Hello World</h1>
```

```
var heading = document.querySelector("h1");

// Getting Styles
var currentStyles = getComputedStyle(heading);
var fontSize = currentStyles.fontSize;

// Setting Styles
heading.style.width = "400px";
heading.style.fontSize = "24px";
```

Styles

- CSS properties that normally have a hyphen in it, you must camelCase it
- Number properties must have a unit - they won't default to pixels

Exercise

The Logo Hijacker

Creating DOM Nodes

We can make our own HTML elements as well!

```
// Create Element in Memory
var newPara = document.createElement( "p" );

// Set the text
newPara.innerText = "Created with JS";

// Set the styles
newPara.style.fontSize = "24px";
newPara.style.color = "hotpink";

// Put it on the page
document.body.appendChild( newPara ); // Or...
document.body.insertBefore(newPara, document.body.firstChild); // Or...
document.body.innerHTML += newPara;
```

Exercise

DOM Manipulation

Events

Some Terminology

- **Event**: something that happens
- **Callback**: a function that executes after the event has happened
- **Event listener**: a method that binds an event to a callback

Events with JavaScript

- Three important things:
  - **The element** that is going to be interacted with (body, h1, p etc.)
  - **The event type** (click, hover, scroll etc.)
  - **The response** (often called *the callback* - a function!)

Events Pseudocode

```
WHEN the element with ID of toggle is CLICKED
    SELECT the body tag and save as body
    CHANGE the body CSS to have a hotpink background

WHEN the element with ID of toggle is CLICKED
    SELECT the body tag and save as body
    STORE the currentBackground of body
    IF currentBackground === "hotpink"
        CHANGE the body CSS to have a ghostwhite background
    ELSE
        CHANGE the body CSS to have a hotpink background
```

Events Pseudocode

```
WHEN the page is scrolled
    CREATE an image of bill, save it as bill
    CHANGE the src of bill to be http://fillmurray.com/500/500
    APPEND it to the page
```

el.addEventListener

```
var myButton = document.querySelector("button");

myButton.addEventListener("click", function() {
  console.log("button clicked!");
});
```

The basic process:

- Find the element
- Add the event listener and pass in a function to call

Anonymous Functions

```
var myButton = document.querySelector("button");

myButton.addEventListener("click", function() {
  console.log("button clicked!");
});
```

You can't ever remove that event handler!

Referenced Events

```
var myButton = document.querySelector("button");

function myCallback() {
  console.log("button clicked!");
}

myButton.addEventListener("click", myCallback);

myButton.removeEventListener("click", myCallback);
```

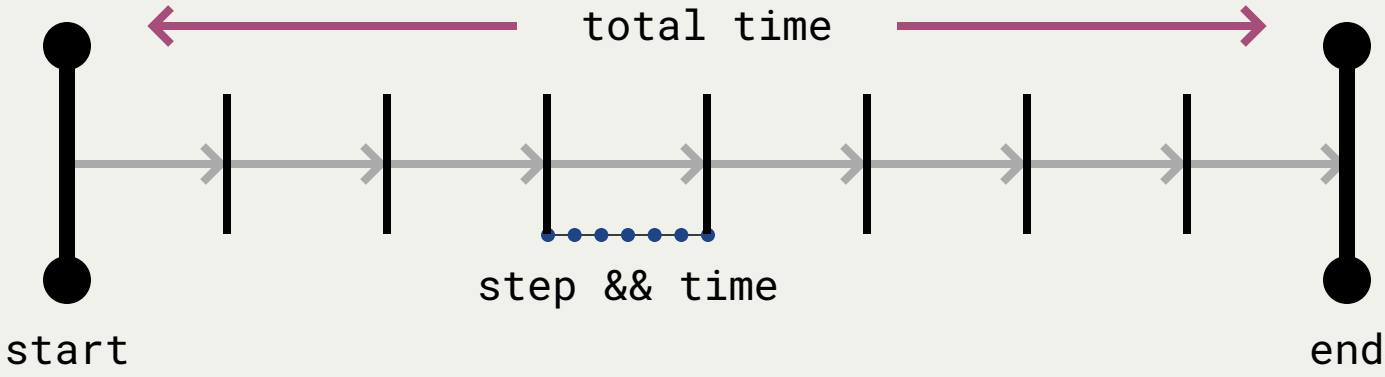
Much better!

What events are there?

- We always create them in the same way, but these are some of the available events:
  - Mouse Events
  - Keyboard Events
  - Browser Events
  - Form Events

Animations

Animations



Animations

Things you need to define:

1. **Starting Point**
2. **Step**
3. **Time between steps**
4. **Total time**
5. **Ending Point**

Timers in JavaScript

```
function delayedFunction() {}

window.setTimeout( delayedFunction, 1000 );


function regularlyScheduledProgram() {}

window.setInterval(regularlyScheduledProgram, 1000);
```

Fade Away: Pseudocode

```
SELECT and STORE the image as bill

CREATE a function called fadeBillAway
  GET the current opacity and store as currentOpacityAsString
  GET the current opacity as a number and store as currentOpacity
  CREATE newOpacity by subtracting 0.01 from currentOpacity
  UPDATE bill opacity to be newOpacity
  IF the currentOpacity is >= 0
    CALL fadeBillAway in 10ms

CALL fadeBillAway to start the animation
```

Fade Away

```
var bill = document.querySelector("img");

function fadeBillAway() {
  var currentOpacityAsString = getComputedStyle(bill).opacity;
  var currentOpacity = parseFloat(currentOpacityAsString, 10);
  var newOpacity = currentOpacity -= 0.01;
  bill.style.opacity = newOpacity;
  if (currentOpacity >= 0) {
    window.setTimeout(fadeBillAway, 10);
  }
}

window.setTimeout(fadeBillAway, 1000);
```

Homework

- Finish all exercises from class
  - DOM Detective, The Logo Hijack, and DOM Manipulation
  - Make previous exercises dynamic!
    - 99 Bottles && Working with Users
      - Bonus: Make Users work with Local storage
    - Train Stations
    - Create your own Endless Horse
    - Plus, anything else!

Homework (Extra)

- Watch Umar Hansa's Browser Rendering Talk
- Watch Jake Archibald's In The Loop
- Go through The Modern JavaScript Tutorial
- Read Eloquent JavaScript
- Read Speaking JavaScript

What's next?

- JavaScript & The Browser!
  - **D**ocument **O**bject **M**odel
  - Events
  - Animations
  - Rendering

Questions?

Feedback

https://ga.co/js05syd

Our first extra session!

Thanks!