Before we begin...

- Open up these slides:
  - https://goo.gl/8auWLi

Advanced JavaScript & HTML

Learning Objectives

- **Identify** common JavaScript patterns for manipulating the Document Object Model
- **Register** and trigger event handlers for JavaScript events
- **Effectively** work with the event parameter
- **Create** some basic JavaScript animations

Agenda

- JavaScript DOM manipulation patterns
- Events
  - Creating event handlers
  - Using the event parameter
- Timers
- Animations
- *Templating*

A quick review

- 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*

Patterns

Select, Manipulate, Admire

- **Step 1**: Select element and store a reference
  - `var p = document.querySelector("p");`
- **Step 2**: Manipulate the element (optional)
  - `p.innerText = "Something new";`
  - `p.style.color = "hotpink";`
- **Step 3**: Admire

Create, Manipulate, Inject

- **Step 1**: Select element and store a reference
  - `var p = document.createElement("p");`
- **Step 2**: Use a method to manipulate (optional)
  - `p.innerText = "Something new";`
  - `p.style.color = "hotpink";`
- **Step 3**: Add it to the page
  - `document.body.appendChild(p);`

Exercise

Creating Elements

Have a function that creates an img element using Unsplash and adds it to the end of the body tag

- *Bonus: Get the width and height to be random*
- *Bonus: Make it happen every second*
- *Bonus: Make it happen whenever there is a click*

Callbacks

What are callbacks?

- Just a fancy name for JavaScript functions
- Only difference is that you don't decide when these functions run
- They are functions that act as a response
- When X happens, call this callback
- They are a part of *Higher-Order Functions*
    - Functions that receive functions as input, or return functions as output

It's all about callbacks

Hey Girl,
I wish I were an Event Listener...
...so you would give me a callback

Events

GENERAL ASSEMBLY

What are events?

- Every browser has events built-in
- Events are important moments that take place on a webpage
- We can attach functions (or callbacks) to these moments, and the browser will call them for us
- There are lots of events
    - Mouse events, window events, keyboard events, form events etc.

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!

So many ways!

- The more variables you have, the easier it will be to debug - I would start off defining everything
- Once you get more comfortable, you can start storing less
    - But I much prefer using named functions rather than anonymous functions (for debugging purposes, and because you can remove the event listener)

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

Mouse Events

- click
- dblclick
- mousemove
- mousedown
- mouseup
- contextmenu
- ...

Key Events

- keydown
- keyup
- keypress
- …

Window/View Events

- resize
- scroll
- …

Form Events

- submit
- ...

They always look the same!

```
TARGET.addEventListener(
    EVENT_TYPE,
    CALLBACK_FUNCTION
);
```

The *event* parameter

The *event* parameter

- When JavaScript runs an event handler, it automatically provides us with a little bit of information as a parameter
  - How long we have been on the page
  - Where the mouse was
  - What key was pressed
  - The target of the event
  - etc.
- We can call it whatever we would like, but the names *e* and *event* are very common

The *event* parameter

```
var button = document.querySelector("button");
var eventType = "click";
function onButtonClick(event) {
    console.log(event);
}

button.addEventListener(eventType, onButtonClick);

window.addEventListener("mousemove", function (event) {
    console.log(event);
});
```

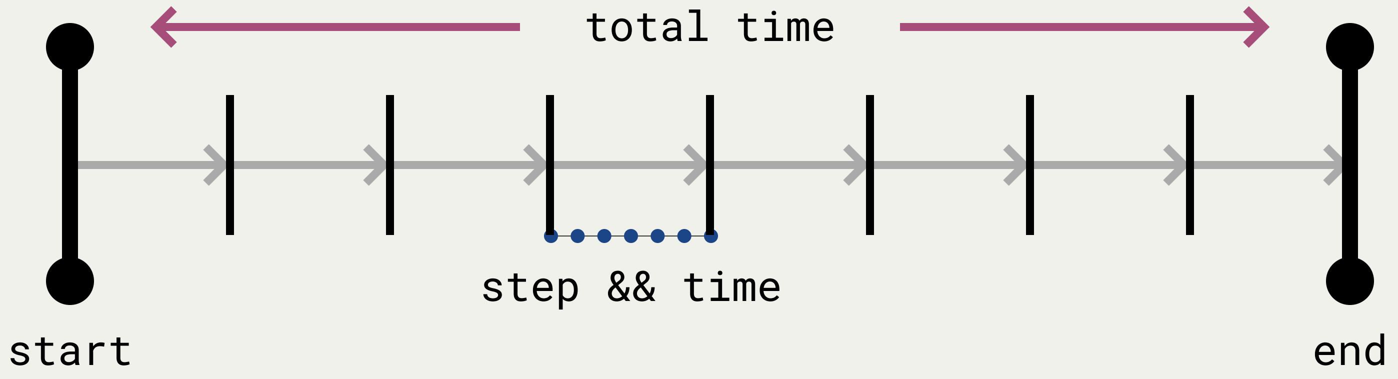
Exercise

Add some events!

- Add a click listener to an image
  - *Bonus: Log the width of that image*
  - *Bonus: Make the same function work for all images*
  - *Bonus: Add a border to the clicked images*
- Add a keypress listener on an input
  - *Bonus: Log the key that was pressed*

Animations

Animations



@threequal

Animations

Things you need to define:

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

What are functions?

- There are two main ways to work with time in JavaScript
- You can set a **delay** with *setTimeout*
- You can set an **interval** with *setInterval*

Timers in JavaScript

```
// window.setTimeout( CALLBACK, TIME_IN_MS );

function delayedFunction() {}

window.setTimeout( delayedFunction, 1000 );


// window.setInterval( CALLBACK, TIME_IN_MS );

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);
  }
}

fadeBillAway();
```

@threequal

Homework

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

The Real Homework

- Dancing Cats!
  - Here is some inspiration

Hopefully we will see some demos of this!

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?

- More JavaScript & The Browser!
  - Templating
  - Building larger apps
  - In-class Project/Exercise

Questions?

Feedback

https://ga.co/js05syd

Our first extra session!

Thanks!