Welcome

DOM and jQuery

Objectives

- Identify differences between the DOM and HTML.
- Explain the methods and use the DOM in javascript.
- Manipulate the DOM by using jQuery selectors and functions.
- Register and trigger event handlers for jQuery events.

But First!

- I’m going to put 15 minutes on the clock
- I want you to recreate fizzbuzz with a *new* javascript file that can be ran using **node**

> Write a program that prints the numbers from 1 to 100. But for multiples of three print “Fizz” instead of the number and for the multiples of five print “Buzz”. For numbers which are multiples of both three and five print “FizzBuzz

I like this page

- This page on FizzBuzz is a nice intro and a good way to understand it further

- http://wiki.c2.com/?FizzBuzzTest

The DOM

- Stands for Document Object Model
- We’re going to watch a quick video on the DOM and how it’s represented within a browser.
- https://www.youtube.com/watch?v=n1cKlKM3jYI

The “O” of DOM

- Last class we spent time learning about Javascript Objects and Methods.

- Javascript exposes the DOM of browser pages as an Object that we can access called “document”.

- This allows us access elements on the page such as links, images, tables, etc.

- We can change text, colors, etc, from there.

What does this mean?

- This means that every HTML element on the page has a corresponding Javascript object

- Every HTML object has properties and methods that be used to manipulate it, for example: Size or text content.

MDN Overview

- Let's take a look at the mozilla developer documentation about the DOM.

- https://developer.mozilla.org/en-US/docs/Web/API/Document

MDN Overview

- Let’s take a look at the mozilla developer documentation about the DOM.
- https://developer.mozilla.org/en-US/docs/Web/API/Document
    - Specifically: getElementById, querySelector, querySelectorAll

Working with the DOM

- There a few ways to manipulate the DOM with Javascript.

Inline Javascript

- You can use Javascript within HTML tags.
- While not recommended, you can do this to run Javascript within your DOM:
- `<body onload="window.alert('welcome to my app!');">`

Script Tags

- We can use the <script></script> tags to include Javascript within our page as well. Such as:

```
<html>
  <head>
    <script>
       alert('Welcome to my app!');
    </script>
  </head>
  <body>
  </body>
</html>
```

Including a Javascript file

- The script tags (mentioned last slide) also allow you to include javascript from separate files as well.

- Given we have a directory with 2 files in it:
  - index.html
  - app.js
- We could have a **<script>** tag the looks like this in our **index.html** file:
  - `<script src=“app.js"></script>`

Code Along

- Creating an HTML document
- Creating a Javascript file
- Including that Javascript file in the HTML

Patience is Key

- Before manipulating elements on a document, it's best to wait for the whole page to load.

- When HTML is loaded, it's loaded top to bottom. Because of this, you might have a script that loads before the HTML element has rendered

- This means you might not be able to even find the element on the page.

window.onload

- A lot of the time our javascript will be manipulating the DOM of the webpage.

- We can only manipulate the DOM once it's actually loaded.

- To avoid trying to change parts of the DOM before its loaded, there's a Javascript way of waiting until it is ready to go.

- You can accomplish this with:

```
window.onload = function() {
  alert("Page is loaded");
}
```

Breakdown

```
window.onload = function() {
  alert(“Page is loaded”);
}
```

- What is happening here is that we’re assigning property on the window object to a function
- Within that function, we’re saying to alert the user
- This function is only ran when the page has been considered “loaded”

Code Along

- Show all ways of defining javascript in HTML
- Show `onload` once more for each version of including JS

Break

Creating Elements

- You can create HTML elements with Javascript too!
- You can do things such as:
  - `document.createElement(“h1");`
  - `document.createTextNode("Hello dynamic world!”);`
- Notice how “**`h1`**” doesn’t have the **<** and **>** around it, you just pass the name of the element.
- If you just want to add text to a page, you need to create what’s called a text node to add to the DOM of the page.

Why would you want to create elements?

- Anytime you want to add content to a page, you're adding elements.

- For example, when using Facebook Messenger and a friend sends you a message.

- Facebook's Javascript is creating an element with the text of the message and inserting it into the page's DOM.

Elements are created with

- `document.createElement(“tagtype”)`
- `document.createTextNode(“this is my text content”)`
- Both of these are **methods** on the document **object**.

Adding these elements to the page

```
// run this function when the document is loaded
window.onload = function() {

  // create a couple of elements in an empty HTML page
  var main_heading = document.createElement("h1");
  var heading_text = document.createTextNode("Hello dynamic world!");
  main_heading.appendChild(heading_text);
  document.body.appendChild(main_heading);
}
```

- We first create the new H1 element through the **document.createElement** method.
- We create the text through the **createTextNode** method.
- The text is added to the newly created H1 element.
- The H1 element is added to the body. Both steps 3 and 4 use the **appendChild** method to the respective element. Think of H1 has a bunch of elements inside of it, they would be called children. **appendChild** adds another to the set of children.

Retrieving Elements

- Along with creating elements and adding them to the page, we can also retrieve elements that may already exist.
- One example of a method on the “document” object that allows you to do this is called “getElementById”.
  - `document.getElementById("hello");`

Retrieving Elements

- `document.getElementById(“hello");`
- This allows you retrieve an element from the DOM that looks something like this:
  - `<div id="hello">Hello world</p>`
- Take note of the blue text in the code, as this is what Javascript uses to retrieve that element for you.

Retrieving elements in another element

- Since the DOM is a tree of elements, its common you'll want to retrieve a subset of elements within a node.

- For example, lets say I want list items, but only within a specific list, not *every* list item on the page.

- What to do… what to do…

Example

- Given this HTML:

```html
<body>
  <div id="hello">Hello world</p>
  <ul id="gaCampuses">
    <li>DC</li>
    <li>NY</li>
    <li>SF</li>
    <li>LA</li>
    <li>HK</li>
  </ul>
</body>
```

We can retrieve only list items in “**gaCampuses**”

```js
var campusesContainer = document.getElementById("gaCampuses");
// The getElementsByTagName() method returns a live HTMLCollection of elements with the given tag name.
var gaCampuses = campusesContainer.getElementsByTagName("li");
```

Example

- Given this HTML:

```
<body>
  <div id="hello">Hello world</p>
  <ul id="gaCampuses">
    <li>DC</li>
    <li>NY</li>
    <li>SF</li>
    <li>LA</li>
    <li>HK</li>
  </ul>
</body>
```

We can retrieve only list items in "**gaCampuses**"

```
var campusesContainer = document.getElementById("gaCampuses");
// The getElementsByTagName() method returns a live HTMLCollection of elements with the given tag name.
var gaCampuses = campusesContainer.getElementsByTagName("li");
```

Code Along

Break

Events

- Javascript also allows you to react when things happen in the page (onload is just one example of many).

- This can be anything from clicking something, to moving your mouse around.

Events (cont)

- Events might be caused by a user, or by a page event.

- For example the “**onload**” on the document is fired when the page is loaded.

- Another event, “**onclick**” is fired when a user clicks on something that you’re listening for.

Steps to add an event

- To add an event listener (as they are called) to an element on the page, you must do the following:
  - Find the element (**document.getElementById**)
  - Attach a function by using a property such as “onclick” on the element returned
  - For the function, it is up to you to decide what the function does when the event occurs.

Lets setup some simple HTML

```
<form>
  <input id="my-input" />
  <input id="my-input-button" type="submit" value="Run button code"/>
</form>
```

And the Javascript

wutasdf.js

```
window.onload = function() {
    button = document.getElementById('my-input-button');
    // Event paremeter is the default object event that would have happened on user click
    button.onclick = function(event) {
      // The preventDefault() method lets us disable the default action, allowing us to override with our on functionality.
      event.preventDefault();
      MyApp.doSomething("world");
    };
};

// We can define things outside of the window.onload which are evaluated
// only when called.
MyApp = {};

MyApp.doSomething = function(name) {
    console.log("Hello " + name);
}
```

Code Along

- Attaching an event to a button when clicked
- Attaching an event to a form when submitted
- Adding an element to the page when a button is clicked

`

http://bit.ly/bt-js-07-1