Unit 4 – Web development

DOM Manipulation and jQuery

1. The browser gives us access to something called a “web API” which allows us to manipulate objects on the page.
2. Having a foundation of DOM manipulation will allow us to understand how libraries like React is built upon it.
3. **The Browser**
   1. 3 languages
   2. HTML
      1. Hypertext markup language
      2. The “what” in the document
      3. HTML is static.
   3. CSS
      1. Cascading style sheets
      2. The “how things look” in the document
      3. Fonts, font colors, border styles, shadowing around boxes, etc etc etc
   4. Javascript
      1. Javascript
      2. The “what will things do” in the document
      3. This is what provides us with the functionality.
      4. HTML and CSS are **not programming languages**!
      5. It is a “runtime language” – it changes and interacts with the user.
   5. The DOM
      1. The HTML DOM Standard defines a programming interface for HTML elements.
      2. The web page is a series of objects in a Document Object Model.
      3. Javascript is used to interact with the browser’s **web API** to manipulate the DOM.
         1. And **API** is something like the function calls you are allowed to make when you’re interacting with a library…
         2. It represents the data you can request and what you have access to.
         3. It is basically the middleman that sits between 2 programs, allowing them to communicate with one another.
         4. If I want to, for example, gather a list of tweets from Twitter.
            1. Twitter’s API can talk to twitter for us and get the data for us.
            2. It would be quite problematic if a dev could directly access the twitter databases.
      4. W3C (World wide web consortium / the “WEC”) puts out these rules.
         1. JS code can interact with HTML.
      5. The DOM standard gives us a set of rules for writing code when interacting with HTML elements.
   6. So.. What is the DOM exactly?
      1. It is an object!!
      2. It is about calling properties on the object, assigning values on the object, and performing methods on the object.
      3. It is a tree-structure composed of objects that represent a web page.
   7. In other words, a document may have a property of… divs. Or paragraphs. Or classes.
   8. *Example layout*
      1. Window -> document -> html
         1. Html
            1. Head

Meta

Meta

Meta

* + - * 1. Body

Div

Div

P

A

Input

Script

* + - * 1. Footer
        2. Etc etc
  1. “Nodes”
     1. To summarize the last few days – **a node is just an object.**
     2. **To reiterate. A node is just an object. Don’t overthink it.**
  2. Each node in a DOM tree has one parent
     1. And each node can have multiple children.
     2. Children sharing a parent node are sibling nodes.
  3. W3C standard is separated into 3 parts
     1. Core DOM – standard model for all document types
     2. XML DOM – standard model for XML Document types
     3. HTML DOM – standard model for HTML Document types.
  4. The “window” is the browser “window” that loads the “document”
     1. And we really only look at one document at a time, I think.
  5. HTML elements are represented as objects, and each object has methods and properties.
     1. And as with any other object, you should be able to add your own methods as necessary.
  6. Order of operations:
     1. HTML loads
     2. Browser creates DOM
     3. **Then** our browser should load our Javascript.
     4. We can do this in a lot of ways
        1. jQuery
        2. Native JS event handlers
        3. The “defer” keyword in JS at the end of the file
        4. Or the JS script added to end of body…
     5. DOM is exposed to JS via “document”

1. Manipulating the DOM
   1. Select the object
      1. We say we “traverse” the DOM
         1. document.getElementById
            1. by TagName, ClassName,
            2. querySelector

returns just the first object.

* + - * 1. querySelectorAll -> pass in class or id

returns NodeList

a collection of Nodes.

* 1. Do something
     1. Add
     2. Remove
     3. Change

1. Changing the elements in the DOM
   1. Changing text, attributes, styles of an HTML element.
   2. element.innerHTML = “my new content”
   3. element.attribute = newValue
   4. element.setAttribute(attribute, newValue)
   5. element.style.CSS-property = newValue
   6. we tried changing all the images on the codesmith website
      1. we changed the src, but not all the images changed… could we do that somehow?
2. Creating elements in the DOM
   1. document.createElement(element)
   2. *element.*appendChild(element)
   3. *element.*replaceChild(element)
   4. *element.*removeChild(newChild, oldChild)
3. We can also set event handlers
   1. On click
   2. On hover
   3. Etc etc
   4. addEventListener
      1. there are clicks
      2. And you can register multiple of the same event on the same thing.
      3. So I can actually have multiple click events (like if you click 5 times or something).
      4. Does not overwrite existing event handlers
      5. But can add event handlers
      6. And can remove with removeListener method.
4. jQuery
   1. abstracts and simplifies a lot of DOM manipulation.
   2. Basic syntax: ${*selector*}
   3. .on
      1. This is a method that allows us to flexibly select the kind of events that jQuery should listen for.
      2. Provides many shortcuts for common webAPI tasks.
      3. Uhhhh why not use it?
      4. jQuery requires an entire library to be loaded, and thus slows down a lot of websites that require it.
      5. And so, DOM manipulation, although it requires more written code, should be preferred over jQuery.
5. Editing in HTML vs Javascript
   1. When do we actually use HTML and when do we manipulate something with JS?
   2. If you have something static, then **change the HTML.**
   3. If you need something to be dynamic and changing, then **change the JS.**