A. DOM
    a. Document Object Model
    b. DOM is an object-oriented representation of HTML documents and their component parts as objects and nodes which can be modified with a scripting language
    c. Content is stored in the DOM and may be accessed and manipulated via JS
    d. HTML document = the topmost node
    e. Each element on the page = separate object that with own relationship to the document
    f. Allows interaction w/individual elements and page as a whole
        i. Modify elements
        ii. retrieve and set element properties
        iii. add and remove elements or objects
        iv. capture and respond to user or browser actions/events
    g. Viewing DOM
        i. HTML in devtools = visual representation of the DOM
        ii. created directly from your HTML, but it's often not the same, possible reasons:
            1. mistakes/looseness in HTML → browser has fixed them for you
                a. Eg table
                b. browser will insert missing <tbody>
                c. Viewable in DOM
            2. Manipulated DOM via JS
                a. Eg empty container injected with content via js (react app node)
    h. Structure
        i. Live DOM viewer: http://software.hixie.ch/utilities/js/live-dom-viewer/
        ii. browsers read an HTML page and turn data into objects logically arranged as a data structure tree (DOM tree)
            1. every element of the document = object or a node organized in a hierarchical fashion
            2. Each node can have a parent node (the node right above it)
            3. each node can also have any number of child nodes (the nodes below it), either other elements or text nodes
            4. siblings nodes (other nodes belonging to the same parent)
        iii. Types of nodes
            1. 12 node types
            2. https://dom.spec.whatwg.org/#node
            3. Element Nodes
                a. individual tags or tag pairs in HTML
            4. Text Nodes
                a. Text content of HTML tags
                b. usually have a parent node and sometimes sibling nodes
                c. cannot have their own child nodes
            5. Comment Node
                a. Doesn’t affect presentation
                b. Included in DOM because everything in an HTML document becomes DOM, even comments
    i. Accessing and using DOM

i. As soon as JS is loaded, you can immediately use API for the document or window elements to manipulate the document itself and access the children of document (elements in the web page)
ii. We use the document and window objects used most often
    1. window object represents the browser
        a. global scope
        b. all functions and methods built into JS are built off the window object
        c. JS checks window for any variables we haven't defined in the JS script (global variables)
    2. document object represents the root of the document (HTML tag node)
        a. Its DOM representation is a property of window (window.document)

B. JS crash course
    a. https://javascript.info
    b. seainfo6150-webapp
        i. npm install
        ii. npm start
        iii. http://localhost:3000/?demo=concepts
    c. Javascript is the interactivity and behavior layer of a web site
        i. Javascript is a very powerful client-side scripting language
        ii. It's used mainly for
            1. facilitating a user's interaction with the webpage or application
            2. adding new HTML to the page, change the existing content, modifying styles
            3. responding to user actions: mouse clicks, pointer movements, key presses, etc.
            4. sending requests over the network to remote servers, downloading and upload files
            5. getting and setting data on the client-side (cookies or local storage) to remember for later use
        iii. Javascript browser abilities are limited for the sake of the user's safety to prevent the website from accessing private information or harming the user's data
            1. Examples
                a. Javascript can't read/write arbitrary files on the hard disk, copy them or execute programs. It has no direct access to OS system functions.
                b. Javascript can work with files, but the access is limited and only provided if the user does certain actions, like "dropping" a file into a browser window or selecting it via an <input> tag
                c. Javascript requires the user's explicit permission to interact with camera/microphone and other devices.
                d. Javascript from one page may not access another other if they come from different sites (from a different domain, protocol or port)
                    i. called the "Same Origin Policy"
                    ii. both pages must agree for data exchange and contain a special Javascript code that handles it
                    iii. Similarly, Javascript cannot receive data from other sites/domains without an explicit agreement (expressed in HTTP headers) from the remote side
        iv. Despite the similarities in name, Javascript and Java are pretty much completely unrelated
            1. Java is a complex programming language that is compiled

2. Javascript is a scripting language whose syntax is mostly influenced by the programming language C
3. Why Javascript?
    a. When Javascript was created, it initially had another name: “LiveScript”
    b. But Java was becoming very popular at that time, so it was decided that positioning a new language as a “younger brother” of Java would help increase Javascript’s popularity as well.
    c. As it evolved, Javascript became a fully independent language with its own specification called ECMAScript, and now it has no relation to Java at all

v. Being a scripting language, Javascript cannot run on its own
1. Javascript engine is responsible for running JS code. It can execute not only in the browser, but also on the server, or on any device that the Javascript engine
2. When a user requests an HTML page with Javascript in it, the script is sent to the JS engine in the browser and that executes it
3. Scripts are provided and executed as plain text; they don’t need special preparation or compilation to run

vi. The main advantage of using Javascript as a scripting language for the web is that
1. It has full integration with HTML/CSS and can modify them extensively on-demand
2. It is supported by all major browsers and enabled by default.

d. Variables
i. A box to store any JS value in until you need them
1. Numbers: 1, 10000
2. Strings (alphanumeric characters: “April”, “123 Main Street”, “10000”)
    a. Single quotes: used for plain strings
    b. Double quotes: used for strings that might have a single quote in theme
    c. Backticks: used for interpolating variable values into strings
3. Objects
4. Arrays
5. Functions

ii. Use an equal sign to put a value into the box
1. Example `let foo = 5;`
2. foo box now has the number 5 in it.

iii. Use let to declare variables you think may change
1. Example `let foo = 5;`
2. `foo = foo + 2; // no errors, foo box now has the number 7 in it`

iv. Use const to declare variables that cannot change
1. Example `const foo = 5;`
2. `foo = foo + 1; // will throw an error`

e. Function
i. A reusable JS structure that performs some operation
ii. Can accept parameters and return some result

iii. Could also just perform an operation with no return
iv. Example addTwoNumbers

f. Objects
   i. A collection of key-value pairs, with no specific order
   ii. The keys have to be strings, but the values don't
   iii. To access the values of an object, use a dot followed by the key name
      1. Example user.name
   iv. To set values on an object, use a dot on the variable and use the equal sign
      1. Example anObject

g. Arrays
   i. An array is an ordered list of elements
   ii. To create an array, separate the elements with commas, and surround the whole thing with square brackets
      1. Example `let myArray = ["foo", "bar", "baz"];`
   iii. To add an element to an array, use the built-in `push` function
      1. Example `myArray.push("bleem");`
   iv. To get the element of an array at a given index, use square brackets
      1. `myArray[0]`
      2. Will be "foo" -- arrays start counting from 0, not 1
   v. To get the length of an array, use the `.length` property
      1. Example `let myArray = ["foo", "bar", "baz"];`
      2. `let length = myArray.length // equals 3`

h. Control flow
   i. if statement
   ii. while statement

C. React
   a. http://localhost:3000/?demo=hello
   b. http://localhost:3000/?demo=hello-user-class
   c. http://localhost:3000/?demo=adding
   d. React is a JS library for building user interfaces for single page applications
   e. Why use React?
      i. React makes it very easy to create a visual representation of all the logic and parameters that are in an application (the application state)
         1. Examples of things we might store in state
            a. Which navigation item is active when the page loads or when you click on a link
            b. Whether a button is disabled or not based on external logic
            c. The default value of a text input
         2. For a given application state, your view should always look the same way
            a. For example, for some parameter having a value, a particular warning message may need to be showing
         3. React makes it easy to set up components and UI pieces that can incorporate logic about how your view will behave in a very predictable way
            a. Makes it easy to debug
            b. Also makes it easy to set up reusable pieces of UI (buttons, forms, etc.)

f. Those reusable pieces of UI are called components!
    i. A React app is made up of components
    ii. A component is a reusable piece of code which defines how certain features should look and behave on the UI
    iii. Does that sound like some other JS structure we were just talking about?
        1. Functions!
        2. React components are JS functions that accept accept arbitrary input parameters (called “props”) and return React elements describing what should appear on the screen
            a. Props can be strings, numbers, other React components, functions
            b. Component can also be passed special props called children
                i. Like HTML tags, component tags can surround content
                ii. <Foo>this is text, an image, another component</Foo>
                iii. The text, img, other component would be accessed from Foo.js via props.children
    iv. Working with components
        1. Components are imported from their component files into the app or component where you want to use them
            a. Eg `import Foo from 'Foo.js'`
            b. Referenced with <Foo />
            c. If component can take children, then <Foo>{children}</Foo>
    v. React has two types of components
        1. Functional
            a. Also known as stateless components
            b. A functional component in React consumes props that you pass to it and returns the UI that React should render
            c. Example
        2. Class
            a. Also known as stateful components
            b. Class components have additional property state, which you can use to hold data that’s private to the component
            c. Class components can also access what’s called the React lifecycle methods
                i. These are special functions that React calls when certain events happen
                    1. Whenever the component is rendered
                    2. Whenever the component is added to and removed from the DOM
                    3. Whenever the component is updated with new state or props
                ii.
            d. The render() function must be present in a class because React looks for this method in order to know what UI it should render on screen
            e. Props can be accessed class component using this.props object
            f. Example
g. Event handlers
    i. DOM nodes have special functions that are triggered when they are interacted with

ii. They broadcast the interaction and if we are listening for that event, we can capture it and do something in response, either with the information that is sent with the event or not
    1. For example
        a. buttons can broadcast a “click” event when they are clicked on
        b. Form inputs can broadcast events like “click”, “change”, “focus”, “blur”

iii. React gives elements special props where we can attach functions to listen for the specific event and then use it to trigger some change to state, props, etc.

iv. Example adding machine