|  |  |  |  |
| --- | --- | --- | --- |
|  | **index.html** | **style.css** | **scripts.js** |
| **Portfolio** | Reference CSS and JS script files.  Link google fonts. | Observed how to create a const element as "div" directly from JS as opposed to HTML.  Add css characteristics (i'm assuming) to those div elements, then append them to "root" (which is html). | Create id in HTML then perform action with them in scripts by "document.getElementByID('id');"  Access hours, minutes, and seconds from Date API.  Change inner text of an id (or span element) using ".innerText" or ".textContent".  *setInterval* to continuously activate a function at set time interval.  ".innerText" returns the visible text contained in a node (if its style="display:none;", that text will not be displayed); but this option yields the benefit of defining html objects, showing <br> properly when requested.  ".textContent" returns the full text regardless of the display status; but this option will ignore html  elements like <br>, yields a result that concatenate all texts with no line break. |
| **To-do** | Learned to link Font Awesome 4’s icons using <link> in header and <i> during in line html. | text-decoration:line-through;  A list item can have effects on the element itself “li { }”, its effects “li:hover { }”, or its subsidiary elements “li > button { }”.  If there’s a hover modification effect, we can adjust its transition speed with “transition: all 0.75s ease;”, then repeat the following settings to ensure different browser are consistent “-webkit/moz/ms/o-transition: all 0.75s ease;”. Write each browser variation on different lines pls. | Append/remove elements such as a button, or a list item, using a DOM method (e.g, .appendChild()).  Nested functions are used to keep things private. Example, if our fxn is creating a list item, we create the nest right away to ensure effects are only applied to current list item. Taking the nest out may result in unwanted items being affected.  *addEventListener* to watch for events such as "mousemove" or “click” by triggering a transforming function to a div/cursor element.  DOM (Document Object Model) manipulation means adding or changing the HTML elements & attributes, CSS styles, creating trigger events. (e.g., there’s a method and a property in *document.getElementById(id).innerHTML*)  Found out about the overlay effect similar to VBA’s UserForm’s frame. Clicking on a list item’s built-in x button triggers the list’s listener event... except it’s indirect and doesn’t activate… unless... idfk. |
| **8 Ball** | Add images with *<img id="" src="" alt="">* |  | ***Math.random() \* n*** to get a random number between 0 and n. Follow up with a ***Math.floor()*** or ***Math.round()*** to avoid decimals.  Dictionary is created by ***var varname = { };***, array is created with ***[ ]***.  IDs can have innate JS functions and are denoted with ***$(“#id”).effect(“shake”); $(“#id”).attr(“src”, img\_src); $(“#id”).fadeIn(4000); $(“#id”).text(string);***  Creating a function by giving it a name.   * var showCurrentTime = function() { // code here } * function addListAfterKeypress(event) { // code here } * Where event.keyCode === 13 denotes the Enter key. * Function declared by var doesn’t exist until that line is reach. Self-declared function exists immediately at runtime. Determine what’s best for you.   Call a function by itself.   * showCurrentTime();   Pass function as a parameter.   * setInterval(showCurrentTime, 1000); * setTimeout(function() { // code here }, 1000); |
| **Joke Generator** |  |  | Fetch API from a website and set headers parameter to accept JSON responses. Pass the API ‘response’ to a function parameter with **.then ( )** and convert Stringified JSON response to a Javascript Object with **response.json();**. Pass the JS Object to another innate function parameter *then* extract a particular key from the JSON.  Manipulate twitter link and button (did not explore). |
| **Inventory Management** |  | *.editButton { // code*  *&:hover { // code } }*  To immediately modify a class or id’s hover effect. | React.js being used in action. |
| **Movie List** | Add “forms” that allow user input, integrate with **div**, **label**, and **input** to create labels followed by input boxes.  Create a button that links to a JS function directly *onClick* using <input>. | Exhibit error in an input box by changing its **border-color** to red.  Variable declared by **let** cannot be redeclared and must be declared before use. **Block scoped**.  Variable declared by **var** are hoisted, meaning it exists but remains *undefined* unless the declaring line is reached. **Function scoped.**  Variable declared by **const** cannot be changed thru reassignment and cannot be redeclared – unless inside a block. If the constant is an *object* or *array,* however, its properties or items can be updated or removed. **Block scoped.** | Generate a table structure with **document.createElement(“table/thead/tr/th”),** then change heading (th)’s innerHTML text.  Add bootstrap styling to table such as **movieTable.setAttribute("class", "table table-striped");**  Append TableHeaders to TableRow. Append TableRow to TableHead. Append TableHead to Table.  Conclude by attach this JS table object to a div set in HTML. **document.getElementById("divID").appendChild(Table);** |