FS II Notes - Day 5 (Oct 10, 2019)

Client-Side Javascript

- Inline scripts

```
<script>some javascript</script>
```

- External references

```
<script ref="http://javascriptserver.com/js/scriptName.js">
```

or

```
<script ref="js/scriptname.js">
```

- Documenting & Commenting

```
// Text following two forward slashes is a comment
```

and/or

```
/*
Text here is commented.
This makes it easier to add more words
*/
```

External Information Resource:

https://www.w3schools.com/js/js_whereto.asp

https://www.w3schools.com/js/js_comments.asp

TASK:

Demonstrate basic HTML file setup and understanding of file handling.

- Create a new file named index.html with basic HTML structure and external references to CSS and JS files using placeholder names.
  - This file should be saved where it can be retrieved for the next task

Development Environment

- Project set up
  - Local Testing
  - Deployment Prep

TASK:

Demonstrate management of local files for use in development environment and understanding of file and folder relationships.

- Create a development folder that will be dedicated to client-side code exercises named “js_client_side” - This is our “Development Exercise Root Repository” (DERR) for client-side javascript
- Create a folder within the DERR that will be used as a template named “0_template”
- Create the following files using the template folder as the root:

index.html

/css/mycss.css

/js/myScript.js

Code Blocks

- Semicolons

```
// This is commented code to describe the upcoming lines of code that execute
// If you forget to end your lines of code with a semicolon you'll have problems.
// The next line is properly ended with a semicolon.
var welcomeMsg = "Hello to a declared variable";
// comments don't need semicolons
```

- Code blocks

```
// code inside { } is executed separately
function sayHelloFunction() {
  var welcomeMsg = "Hello to a declared variable";
  console.log(welcomeMsg);
}
// the closing } does not require a semicolon
```

External Information Resource:

https://www.w3schools.com/js/js_statements.asp

https://www.w3schools.com/js/js_conventions.asp

TASK:

Demonstrate understanding of external references to code and syntax.

- Make a copy of the template folder "0_template" & rename to "1_hello_world" in the DERR. The folder should be at the same directory level as the template folder
- Display "Hello World" in the HTML of index.html using Javascript within the myScript.js file

Outputting

- Try/Catch error handling

```
try {
  alert(“Hello User");
}
catch(err) {
  document.getElementById("msgBox").innerHTML = err.message;
}
finally {
  // If needed, you can do something here regardless of what happens in try/catch
}
```

- HTML output
  - HTML element with innerHTML
  - HTML output with document.write()
- Alert Box output
  - Alert box with window.alert()
- Console output
  - Browser console with console.log(), console.error() & console.table()

External Information Resource:

https://www.w3schools.com/js/js_errors.asp

https://www.w3schools.com/js/js_output.asp

TASK:

Demonstrate understanding of javascript output to the browser and awareness of error handling.

- Make a copy of the template folder “0_template” & rename to “2_outputting” in the DERR. The folder should be at the same directory level as the template folder

- Using the index.html file display in the browser title, top level document a human-friendly welcome message that is assigned to a variable using the externally referenced myScript.js file. All JS files should have error handling implemented
- Display an ‘ok’ message to the console when successful

Lists

- Lists
  - Ordered, Unordered & List Items

```
// Unordered list - list with bullets
// Disc (default), Circle, Square, None
<ul>
  <li>This</li>
  <li>That</li>
  <li>Other</li>
</ul>

// Ordered list - list with incrementing indexes
// 1, A, a, I, i
<ol>
  <li>This</li>
  <li>That</li>
  <li>Other</li>
</ol>
```

TASK:

Demonstrate use of lists in HTML pages.

- Make a copy of the template folder “0_template” & rename to “3_lists” in the DERR. The folder should be at the same directory level as the template folder
- Using the files in the newly created folder, display three different lists of different words in the browser window using list item functionality. Each list must be a different type of list and be easy to distinguish from the other by the user
- After the lists have been displayed the user should be presented with a human readable alert informing them about the words in the lists
- The console should display a success message if there are no problems or the error that occurred if there is a problem.

Variables

- Variable types & Arrays
  - Strings - Single and double quotes
  - Values & Objects
- Operators
- Variable declarations - Identifiers (names) - Lower “camelCase"
- Statements - Assigning values

Javascript is Case Sensitive

Strings - Text enclosed by single or double quotes
Numbers - Double-precision 64-bit binary format IEEE 754
Booleans - True or False
Objects - Any Javascript object
Arrays - A list of variables

```
textValue = “Hello”;
numberValue = 100;
BooleanValue = true;
```

External Information Resource:

https://www.w3schools.com/js/js_variables.asp

https://www.w3schools.com/js/js_arrays.asp

https://www.w3schools.com/js/js_string_methods.asp

https://www.w3schools.com/js/js_datatypes.asp

TASK:

Demonstrate using arrays to store text and numbers. Demonstrate displaying values from variables in HTML. Demonstrate using different operators on different types of variables.

- Make a copy of the template folder “0_template” & rename to “4_variables” in the DERR. The folder should be at the same directory level as the template folder
- Using the files in the newly created folder, create an array using a list of words and a list of numbers. Display the list of words with an accompanying number by combining strings from the values of each list. Each of the words displayed should be human-readable
- All JS files should have error handling implemented

Conditionals

- if/else

```
if (thisIsTrue) {
  //  do something if the above is true
} else if (thisIsTrue) {
  //  do this if the above is true
} else {
  //  otherwise do this
}
```

- Comparisons

= equal sign assigns a value

== two equal signs is comparing the value, === also compares the type

!= means not equal in value, !== means not equal in value and type

&& is a logical **and** comparison

|| is a logical **or** comparison

! is a logical **not** comparison

- Switch

```
switch(someThing) {
  case valueA:
    // do this if the value of someThing is the same as valueA
    break;  // stop checking values and continue after the switch
    // without the break it will keep checking
  case valueB:
    // do this if the value of someThing is the same as valueB
    break;
  default:
    // do this if none of the cases above match the value of someThing
}
```

External Information Resource:

https://www.w3schools.com/js/js_if_else.asp

https://www.w3schools.com/js/js_switch.asp

https://www.w3schools.com/js/js_comparisons.asp

TASK:

Demonstrate using multiple variables for storing information. Demonstrate using conditions for determining output to be displayed in the browser and for debugging.

- Make a copy of the template folder “0_template” & rename to “5_conditionals” in the DERR. The folder should be at the same directory level as the template folder
- Using the files in the newly created folder, create an array using a list of three words and an array using a list of three numbers
- Also assign a number to different variable and a letter to a different variable
- In the browser window display two lists, one list of words from the array that begins with the letter in the single letter variable and the second with numbers that are either greater or less than the single number variable.
- Each list should have accompanying descriptions of what they are displaying
- All JS files should have error handling implemented

Loops

- For loops

```
var aNumber; // a number we are using
var nCount = 0;  // a number we are using with our loop
var aDifferentNumber = 10; // could be assigned from elsewhere
for (aNumber = 0; aNumber < aDifferentNumber; aNumber++) {
  nCount++;
}
```

- While loops

```
while (thisIsTrue) {
  // do something
}
```

so…

```
var aNumber = 0; // a number we are using
var nCount = 0;  // a number we are using with our loop
var aDifferentNumber = 10; // could be assigned from elsewhere
while (aNumber < aDifferentNumber) {
  nCount++;
}
```

External Information Resource:

https://www.w3schools.com/js/js_loop_for.asp

https://www.w3schools.com/js/js_break.asp

TASK:

Demonstrate using loops for cycling through data to perform tasks.

- Make a copy of the template folder “0_template” & rename to “6_loops” in the DERR. The folder should be at the same directory level as the template folder
- Accomplish the same functionality as the previous task using loops

Functions

- Using functions
- Creating functions
- Passing values
- Returning values

```
function somethingToDi(someData) {
// declaring a variable that we can use for when we are done here
var resultValue;
  // this is where we can do stuff
  // the someData variable is optional but if it's here we can use it
  // we also have the option of sending data back when done
return resultValue;
}
```

External Information Resource:

https://www.w3schools.com/js/js_functions.asp

TASK:

Demonstrate understanding of how functions work by using a single set of functions from multiple sources.

- Make a copy of the template folder "0_template" & rename to "7_functions" in the DERR. The folder should be at the same directory level as the template folder
- Create two additional HTML files based on the index.html file named page1.html and page2.html
- Using the javascript file create three functions each with different names
- Within the HTML of each of the HTML pages display the output from one different function with no page showing the results for the same function
- At minimum the output from the function should be text that is human readable and not the same as the other functions
- All JS files should have error handling implemented

HTML Forms & Fields

- Forms

```
<form action=“/path/dosomething“ method="post">
  <input type="text" name="firstName" required>
  <input type="submit" value="Continue">
</form>
```

- Buttons
  - Button events
- Fields
  - Field attributes

```
<input id="aNumber" type="number" min="0" max="1000000" required>
<button onclick="doSomething()">OK</button>
<p id="resultMessage"></p>

function doSomething() {
  var theNumber = document.getElementById("aNumber");
  if (! theNumber.checkValidity()) {
    document.getElementById("resultMessage").innerHTML =
theNumber.validationMessage;
  }
}
```

External Information Resource:
https://www.w3schools.com/js/js_validation.asp

TASK:

Demonstrate understanding the use of buttons for navigation. Demonstrate the use of form elements within a HTML page.

- Make a copy of the template folder “0_template” & rename to “8_forms” in the DERR. The folder should be at the same directory level as the template folder
- Present a page that provides two or more buttons for the user to choose from where the resulting interaction loads a different page for the selected button

- One of the pages after the first page should display a human-readable message welcoming the user. Additional pages should have a different message and appropriate page distinction
- A different page from the welcoming page should present the user with the ability to enter information into a form (must have at least two text fields) with buttons that provide options for returning to the previous page or to proceed
- The page with fields should also display to the user in human-readable form what information is being requested
- All JS files should have error handling implemented

User Interactivity

- Event handling
- Affecting HTML

```
somethingHappened(function (someData) {
  if (someData) {  //  this checks to see if there is data, if there is then it does stuff
    //  here we are displaying the incoming data to the console for debugging
    console.log(someData);
    //  the console logging could also be used only for debugging
    document.getElementById('anElementId').innerHTML = someData;
    document.getElementById('aButtonId').onclick = function () {
      // do something when that button is clicked

      };
  } else {
  // if there is no data then do something else
  window.location.href = 'login.html';
  // in this case we are redirecting to a different HTML page
  }
});
```

External Information Resource:

https://www.w3schools.com/js/js_events.asp

TASK:

Demonstrate use of event handling to determine results displayed to a user.
Demonstrate re-use of code.

- Make a copy of the template folder “0_template” & rename to “9_events” in the DERR. The folder should be at the same directory level as the template folder
- With the template files and methods used previously, create the necessary pages for a user to be presented with options that will navigate them to additional pages. There should be three or more pages. The first page will provide options for the user. The subsequent pages will be the pages based on the options chosen by the user. Pages should display different information based on input from the user.

Deployment Testing

- Deployment
- Server testing

Be aware of path handling and the location of external resources and dependencies.

TASK:

Demonstrate understanding of the purpose for server-side testing. Demonstrate configuration of a server that can be used for application development.

- Using Firebase create a new project with hosting for this task
- Create a new folder named “10_testing”, this is our “testing folder” in the DERR
- Use Firebase Tools to initialize the testing folder
- Deploy the content of your “9_events” folder to the Firebase servers so it functions the same as local testing with your “9_events” folder

FINAL TASK:

- Using your Git account push your DERR as a repo named “FS2_JS_Client”