**Preparation**

Copy the files and folders from **your Web app** in week 2 to your week 3 lab folder.  
**Don't include anything from the example Flask Web app.**Create a folder named **js** in your week 3 folder that will contain your JavaScript code.

My class GitHub repo is: [**https://github.com/oit-gaden/Web-Development-2020-Winter (Links to an external site.)**](https://github.com/oit-gaden/Web-Development-2020-Winter)  
  
**Exercise 1**

1. Create a JavaScript file named **students.js** in the js folder.
2. You will replace the hard-coded data you put in the students table in week 1 with data from a JSON string.  
     
   Add a string variable in students.js that contains a JSON string similar to:

**var products = '{"products": [{"name": "television", "price": 29.00}, {"name": "recliner", "price": 199.00}]}'**  
  
Except, your data will reflect the contents of the students table.  
  
Refer to:  
[**https://www.w3schools.com/js/js\_json\_intro.asp (Links to an external site.)**](https://www.w3schools.com/js/js_json_intro.asp) to become familiar with JSON syntax.

1. Remove all of the data rows from the student table in students.html (remove the tr and td elements).
2. Add a function to students.js that returns an array of JavaScript objects from the JSON using a line of code similar to:

**return JSON.parse(testProducts);**The parameter is the variable you created in step 1.  
  
Refer to:

[**https://www.w3schools.com/js/js\_arrays.asp (Links to an external site.)**](https://www.w3schools.com/js/js_arrays.asp)[**https://www.w3schools.com/js/js\_functions.asp (Links to an external site.)**](https://www.w3schools.com/js/js_functions.asp)[**https://www.w3schools.com/js/js\_json.asp (Links to an external site.)**](https://www.w3schools.com/js/js_json.asp)

1. Add a function to students.js that you will call from students.html to populate the student table.  This function will first get the array of student objects using the function from step 4.  It will then call a another function from step 6 to display the students in the student table.  Here is a sample of what the code would look like:

**var data = getTestProducts();**

**displayProducts(data.products);**

1. Add a function to students.js that displays the array of students in the student table.  Here is a sample of what the code would look like:

function displayProducts(products) {

    var tableList = "";

    for(i = 0; i < products.length; i++) {

        var name = products[i].name;

        var price = products[i].price;

        tableList += "<tr><td>" + name + "</td><td>" + price + "</td></tr>";

    }

    document.getElementById("product-list").innerHTML = tableList;

}

You need to add an id to the tbody element (table element if you don't have a tbody element in your table) so you can reference it in the code above.  In the example code above, the id that is being referenced is "product-list".  
  
Refer to:

[**https://www.w3schools.com/js/js\_operators.asp (Links to an external site.)**](https://www.w3schools.com/js/js_operators.asp)

[**https://www.w3schools.com/js/js\_loop\_for.asp (Links to an external site.)**](https://www.w3schools.com/js/js_loop_for.asp)

[**https://www.w3schools.com/jsref/met\_document\_getelementbyid.asp (Links to an external site.)**](https://www.w3schools.com/jsref/met_document_getelementbyid.asp)

1. Edit your students.html file and add a reference to schools.js file inside the head element.

**<script src="js/students.js"></script>**

1. Add a call to the function you created in step 6 inside students.html just prior to the body end tag (</body>) like this:

**<script>initializeProducts()</script>**

1. Start your Web application from VSCode using LiveServer (make sure you have index.html as the current file in VSCode when you start LiveServer).  Click on the students link. You should now see the list of students populated from the JSON string.  It is common to get JSON strings from REST services and displaying all or some of the content received.

**Exercise 2**

1. Repeat Exercise 1 for the instructors page.

**Exercise 3**

1. Create a JavaScript file named **login.js** in the **js** folder.
2. Add a function to **login.js** that will validate that the user has entered both a username and password in your login page.  Use the validate function from my GitHub class repository Week3/app1 as an example of a way to validate form input and display errors.  You'll need to add error messages, element ids, JavaScript file reference and a button click event handler to your login.html.

**Exercise 4**

1. Build the Docker image.
2. Run the Docker image to test.
3. Push your Docker image to Docker Hub
4. Include a URL to your Docker Hub registry in your lab submission.