**Exercise 2.2 – Hello JavaScript**

**Setup**

Each week you will be asked to create a new folder under web-231 following a naming convention of “week-<number>.” If we are on week two, the folder name should be “week-2.” All files associated with the weekly assignment will be added to the appropriate weekly folder. All programs must be linked in the index.html landing page under the “Weekly Assignments” section. Projects will be linked under the “Projects” section of the index.html landing page. To be clear, **all** of the JavaScript, HTML, images, and CSS files associated with a weekly assignment must be placed under the appropriate weekly folder. The page title for all HTML files in this course must say “WEB 231 – Enterprise JavaScript I.” And, all HTML and CSS files must be valid HTML/CSS, tested through the WC3 validator. The links were provided during WEB 200 and were added to the index.html landing page. Also, the blue border around the provided images is to show they are images and should not be included in your submission. In other words, do not add a blue border around your work, unless the instructions explicitly ask for it.

Name this week’s exercise file: **<yourLastName>-exercise2.html**

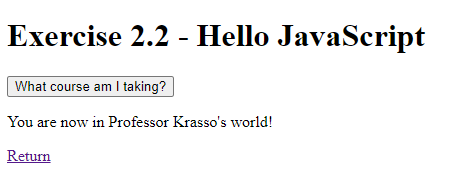
**~~Grading Reminders~~**

1. ~~(50%-points) All code sources (.html, .css, .js) must be cited in the opening programmers’ comments, following the format specified in the code attribution document.~~
2. ~~(25%-points) All code sources (.html, .css, .js) must show evidence of code comments. This means each section of the program (.html, .css, .js) must include code comments that explains what the block of codes purpose is, what the required parameters are (data type, if any), and what the expected output is.~~
3. ~~(rubric) All code sources (.html, .css, .js) are measured against~~ 
   1. ~~Code functionality: Does it work? Does it meet requirements?~~
   2. ~~Adherence to standards and conventions. Are you using the appropriate data types, including proper indention, are variables named appropriate (variable x is an example of poor naming conventions), is there an appropriate use of whitespace, is the code organized, and are semicolons being used to terminate code sentences?~~
   3. ~~Efficiency: Use of language features. Are you practicing DRY (Don’t-Repeat-Yourself?), are you leveraging built-in language features where appropriate, and are you using classes/functions to reduce code clutter?~~
   4. ~~Documentation: Self-documenting, naming conventions, code is maintainable by others. Is the code your write easy to read and maintainable by others?~~
   5. ~~Error trapping/handling. Are there errors in the program? Is there evidence of coding best practices to reduce user errors?~~
   6. ~~Assignment Specific Compliance. Does the delivered solution follow the instructions, as they are written? Does the output match what was provided in the screenshots (including spaces, styling, etc.)?~~

**Required Modifications**

Cite any sources in your opening programmer’s comment

**User Interface (final solution)**



1. **HTML** page load
2. Bind the text value “You are now in <youLastName’s> world!” to the innerHTML of the txtMyWorld paragraph tag.

**Exhibit B. txtMyWorld paragraph tag**



1. **What course am I taking?** Button
2. Give the button an id of btnMyCourse.
3. onclick alert “WEB 231 – Enterprise JavaScript I.”

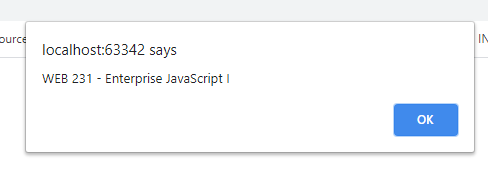
**Exhibit C. What course am I taking? Button**

**A computer code with white text

Description automatically generated**



**Exhibit D. onclick alert message**



1. **Return** anchor tag
2. Give it a class of “return-home” and text value of Return.
3. Link back to the index.html landing page.

**Exhibit E. anchor tag**



1. **Styling** requirements
   1. Add a CSS class return-home to the global.css file with a color of 4F3674, a text decoration of none, an on hover background color of D6A800, an on hover text color of 4F3674, and a pseudo visited text color of 3C275A