**Assignment 6 – In-N-Out-Books**

**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.

**User interface styling and formatting requirements are located in the HTML, CSS, and JavaScript Requirements document.**

HTML: **<yourLastName>-in-n-out-books.html**

CSS: **<yourLastName>-in-n-out-books.css**

**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
* Link the appropriate CSS files and Google fonts
* onload…

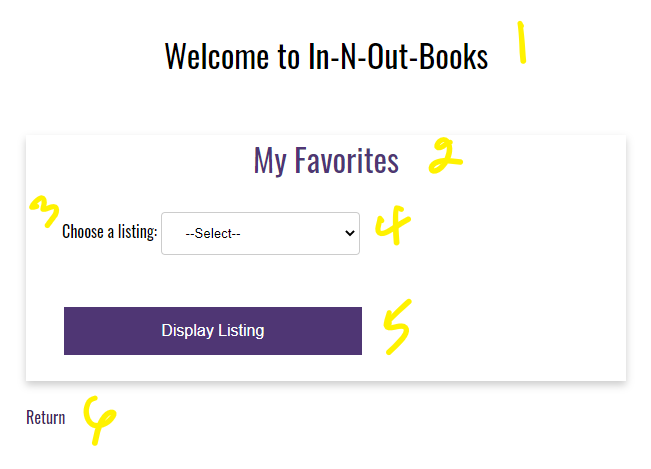
**Additional JavaScript Requirements**

1. Create 3 object literals for your top 3 favorite books and assign them to three separate variables. The object literal properties are: **isbn**, **title**, and **pages**.
2. Create 3 object literals for your top 3 favorite authors and assign them to three separate variables. The object literals properties are: **name** and **genre**.
3. Using document.getElementById, assign the “favorite-books” div (Exhibit B, #2-4) to a variable named **books**.
4. Using document.getElementById assign the “favorite-authors” div (Exhibit C, #2-4) to a variable named **authors**.
5. Set the **books** style.display to none.
6. Set the **authors** style.display to none.

**Additional Programming Assistance**

1. To assign a div to a variable in JavaScript you use document.getEelementById(“id”). For example, **let books = document.getElementById(“favorite-books”);**
2. Once you have the div assigned to a variable, you can change its styling through dot notation. For example, **books.style.display = “none”;**

**Exhibit A. User Interface (Initial view)**

****

1. h1 with a CSS class of app-header and a text value of “Welcome to In-N-Out-Books”
2. card-title with a text value of “My Favorites”
3. form-field label with a text value of “Choose a listing”
4. HTML select list with a CSS class of drop-down-menu and an id of favoriteList

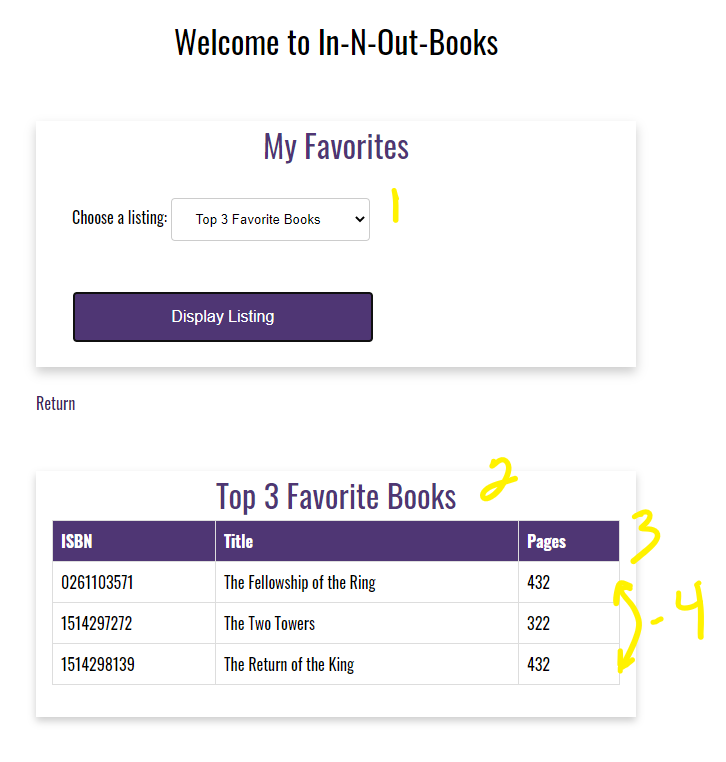
**Additional HTML Requirements**

1. The select form-field will need three HTML options
2. value=select; text=--Select--
3. value=books; text=Top 3 Favorite Books
4. value=authors; text=Top 3 Favorite Authors
5. HTML button with an id of btnDisplayListing and a text value of “Display Listing”

**Additional JavaScript Requirements**

1. Register an onclick event for btnDisplayListing
2. Capture the value from the select list and assign it to a variable named **choice**
3. Bind your top 3 favorite book object literals to an HTML table.
4. Bind your top 3 favorite author object literals to an HTML table.
5. Using if…else if…else, compare choice against the string values: books and authors.
6. If books are selected (Exhibit B, #1), set the authors display to none and set the books display to block (Exhibit B, #2-4).

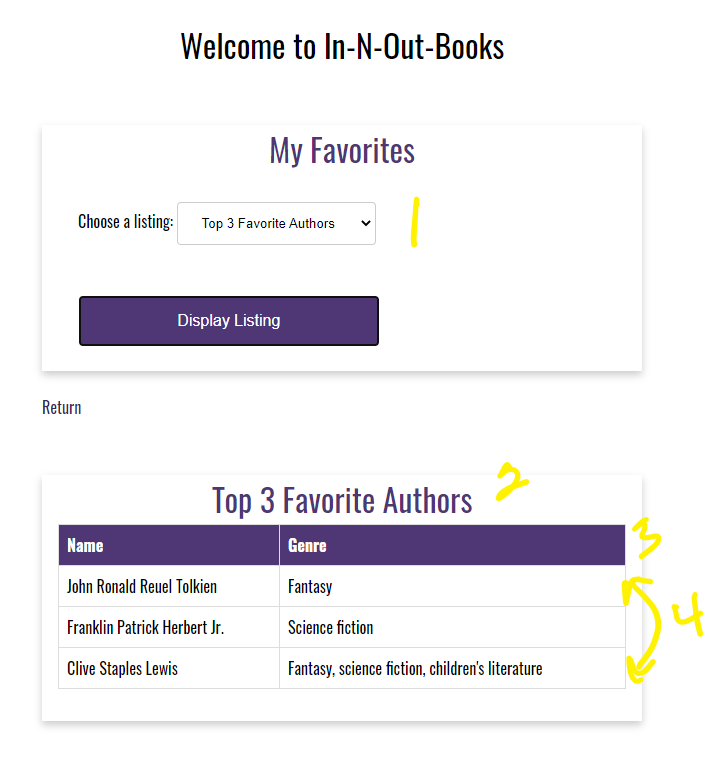
**Exhibit B. Top 3 Favorite Books**

****

**Additional Assistance**

1. It is important to note that the values for the HTML table are being mapped during page load. We are simply showing the appropriate HTML table based on the user’s selection.
2. Exhibit B, #2-4 are inside a card. Exhibit B, #2 is the card-title, Exhibit B, #3 are the table headers, and Exhibit B, #4 are the table rows.
3. If authors are selected, set the books display to none and set the authors display to block.

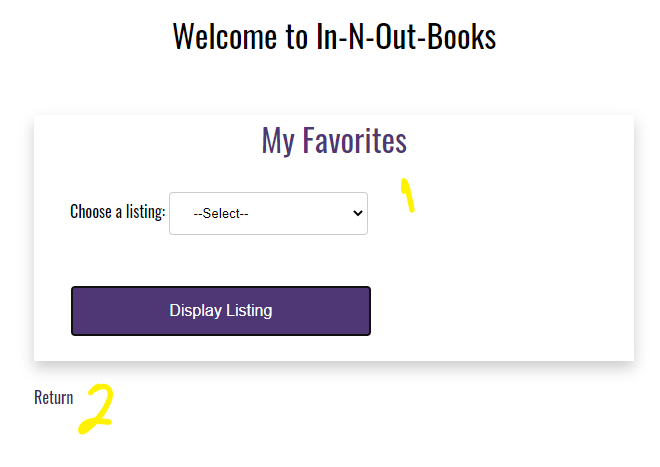
**Exhibit C. Top 3 Favorite Authors**

****

**Additional Guidance**

1. It is important to note that the values for the HTML table are being mapped during page load. We are simply showing the appropriate HTML table based on the user’s selection.
2. Exhibit C, #2-4 are inside a card. Exhibit C, #2 is the card-title, Exhibit C, #3 are the table headers, and Exhibit B, #4 are the table rows.
3. If “select” is selected, set the books display to none and set the authors display to none.

**Exhibit D. --Select--**

****

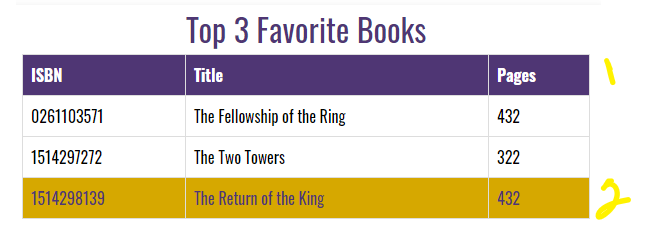
**Additional Assistance**

1. As shown in Exhibit D, #2 the anchor link is above Exhibit B and C, #2-4. This means the anchor link should be placed in the “My Favorites” card.

**Additional Styling Requirements**

1. Give the table a border-collapse, a width of 100%, and a margin bottom of 30 pixels
2. Give the table header a padding top of 12 pixels, a padding bottom of 12 pixels, align the text to the left, a background color of 4F3674, and a text color of white.
3. Give the table data and table header a border of 1 pixel with a solid line and a color of ddd.
4. On table row hover, set the background color to D6A800, the text color to 4F3674, and set the cursor to a pointer

**Exhibit E. Table Styling**

****

**Additional Assistance**

1. Exhibit E, #1 shows the table header styling.
2. Exhibit E, #2 shows the table row on hover styling.
3. This styling applies to the “Top 3 Favorite Books” and “Top 3 Favorite Authors” tables.
4. In order to bind the object literal properties to each of the appropriate tables, you will need to give the table data elements unique ids. For example, **<td id=”txtReturnOfTheKingISBN”></td>**
5. I typically recommend having the id values match the object literal properties. For example, txtReturnOfTheKingISBN, txtReturnOfTheKingTitle, txtReturnOfTheKingPages
6. anchor link with the CSS class return-home and a link back to the index.html landing page.