**Assignment 4 – Number Guessing Game**

**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>-number-game.html**

CSS: **<yourLastName>-number-game.css**

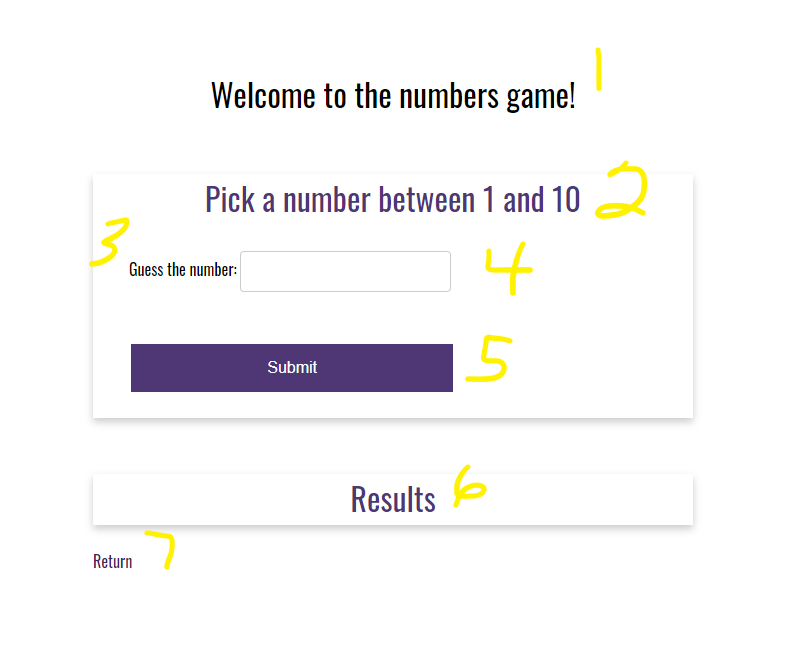
**Grading Reminders**

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

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

****

* On page load…

**Additional JavaScript Requirements**

1. Generate a random number between 1 and 10 and assign it to a variable named **rnd**

**Exhibit B. Random number generator**

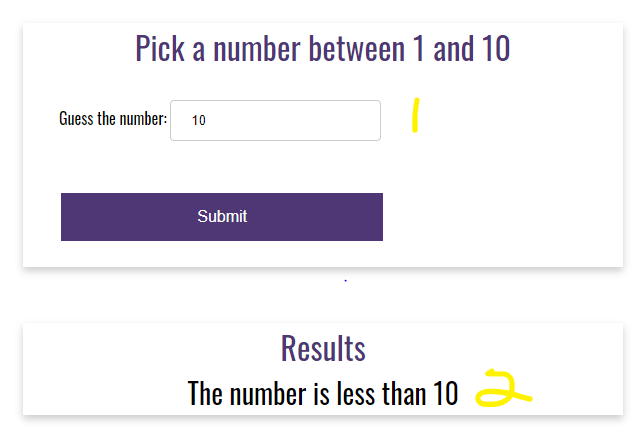


1. h1 with a CSS class of app-header and a text value of “Welcome to the numbers game!”
2. card-title with a text value of “Pick a number between 1 and 10”
3. form-field label with a text value of “Guess the number:”
4. input form-field with an id of txtMyNumber.
5. HTML button with an id of btnSubmit and a text value of “Submit”

**Additional JavaScript Requirements**

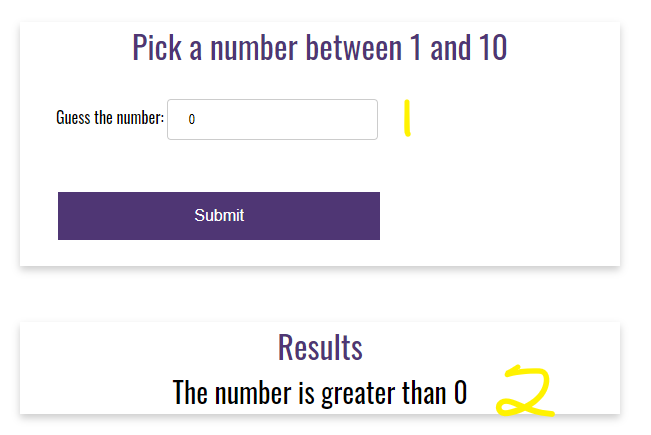
1. Register an onclick event for **btnSubmit**.
2. Capture the inputted value from **txtMyNumber** and assign it to a variable named **myNumber**
3. Using if…else if…else, compare the **myNumber** variable from **6.b.** against the **rnd** variable from **1.a.**
4. if **myNumber** is greater than **rnd**, build a string with the message “The number is less than <myNumber>” and bind it to the results card content (Exhibit C, #2). You will need to replace <myNumber> with the actual number the user inputs into **txtMyNumber** (Exhibit C, #1).

**Exhibit C. The number is less than <myNumber>**

****

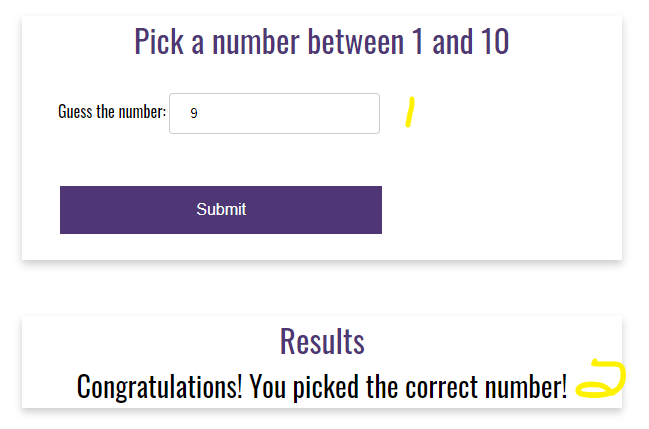
1. if **myNumber** is less than **rnd**, build a string with the message “The number is greater than <myNumber>” and bind it to the results card content (Exhibit D, #2). You will need to replace <myNumber> with the actual number the user inputs into **txtMyNumber** (Exhibit D, #1).

**Exhibit D. The number is greater than <myNumber>**

****

1. if myNumber matches rnd, build a string with the message “Congratulations! You picked the correct number!” and bind it to the results card-content (Exhibit E, #2).

**Exhibit E. Congratulations! You picked the correct number!**

****

**Additional Styling Requirements**

1. As shown in Exhibit C, D, and E, the text is centered with a font size of 28 pixels.
2. card-title with a text value of “Results”
3. anchor link with the CSS class return-home and a link back to the index.html landing page.