Intro to Web Dev
Prework Project Instructions

Introduction

For your final assessment, please make the following enhancements to the Higher-Lower game from the JavaScript exercises. You may start with the Higher-Lower solution or utilize your own solution to the practice exercise.

1. Prompt for Max Number

Instead of locking the game into a number between 1 and 20, use the `prompt()` method to ask the user what the maximum number should be. The prompt should be in a loop with validation as demonstrated previously in the course making sure that the inputted value is a positive number. If the user inputs a decimal, simply round it.

When a valid number is inputted, change the content of the instructions to specify guesses between 1 and N (where N is the user-provided maximum number).

Grading Criteria

Requirements:

- The application prompts the user for a maximum number.
- The application validates the user input and does not allow invalid entries (negative numbers, 0, or non-numbers), re-prompting the user if an invalid entry is provided.
- If the user provides a decimal number, the application rounds it.
- The application selects a random number between 1 and N (where N is the user-provided maximum number).

Points:

- 2 points will be given if all the requirements are implemented.
- 1 point if one requirement was missed.
- 0 points if two or more requirements were missed.

2. Validate the Guess

When the user presses the guess button, validate the input:

- If the guess is not a number, display a message: "That is not a number!"
- If the guess is out of range (1 to N), display a message: "That number is not in range, try again."

Grading Criteria

Requirements:

- The application prevents the user from guessing a non-number and displays an appropriate error message.
- The application prevents the user from guessing a number less than 1 and displays an appropriate error message.
- The application prevents the user from guessing a number greater than N and displays an appropriate error message.

Points:

- 2 points will be given if all the requirements are implemented.
- 1 point if one requirement was missed.
- 0 points if two or more requirements were missed.

3. Track the Guesses

Using an array, keep track of each guess by the user. When the user wins the game by guessing correctly, add the number of guesses and the list of guesses to the victory message. For example:

"You got it! It took you 5 tries and your guesses were 3, 14, 7, 9, 5"

Do not count invalid guesses (not numbers or out of range).

Grading Criteria

Requirements:

- The application correctly initializes an array and uses the `push()` function to add the guesses.
- The application correctly formats the win message to include the comma-delimited guesses as part of the output.
- The application uses the `length` property and does not use an extra variable to count the number of guesses.

Points:

- 2 points will be given if all the requirements are implemented.
- 1 point if one requirement was missed.
- 0 points if two or more requirements were missed.

4. Prevent Duplicate Guesses

Since you are tracking the guesses, add validation to check if a number has already been guessed. If it has, display a message and do not count it as a guess.

Grading Criteria

Requirements:

- The application correctly checks the array for a guess first using any means (loop, `find()`, etc.) before adding a guess to the array.
- The application correctly displays a message that a number has already been guessed.

Points:

- 2 points will be given if all the requirements are implemented.
- 1 point if one requirement was missed.
- 0 points if two or more requirements were missed.