SENG 4640 - Lab 04 - jQuery

In this lab, you will develop a calculator app using JavaScript and jQuery. In completing this lab, you will:

- Use the jQuery library to select HTML elements and modify their contents
- Define callback functions that are invoked as the result of user actions in the HTML page
- Assemble a JavaScript application consisting of multiple functions

Getting Started

Start by downloading the original HTML and JS files, "calc.html", "calc.js", and "jquery.js" from Moodle and then saving them on your computer.

The calc.html file is an HTML page that you can open in your browser and consists of the basic layout of the HTML elements that comprise the calculator. In particular, this page provides the HTML for:

- An input field at the top with the ID "display," which is where the calculator's output will be shown. This field is marked as "disabled" so that the user may not directly enter values into it.
- Buttons for the digits 0-9, with IDs "button0", "button1", etc.
- Buttons for the add, subtract, multiply, and divide operations, clear, etc. with IDs "addButton", "subtractButton", etc.
- A span with the ID "output" that you can use for debugging and displaying other output, but is not part of the calculator function itself

At the bottom of calc.html is a script tag that includes "calc.js," which is where you will implement all of the JavaScript/jQuery code. At the top of calc.html is a script tag that includes "jquery.js" which links the page with the jQuery library (for online/offline use). You also may change the CSS styling if you prefer, but it will not be considered for grading.

Important Note: You are not allowed to use *eval* function due to its security vulnerabilities.

Activity

In calc.js, write the JavaScript code using jQuery to implement the calculator functionality. Your calculator should work as a "normal" calculator would be expected to operate, but here are the different use cases that your app needs to consider:

1. **Case 1. Performing an operation on two numbers (10 marks)**
    - If the user chooses the divide operation and the result is not an integer, it should be displayed using floating point notation, e.g. "10" divided by "4" should produce "2.5".
    - If the user chooses the subtract operation and the result is negative, it should be displayed as a negative number, e.g. "5" minus "8" should produce "-3". This includes subtracting from zero, too, of course.

- If the user attempts to divide by 0, the result should be shown as "Infinity".

2. **Case 2. Continuing an operation (15 marks)**

   - If user chooses another operation, and then entered another number and clicked on "=" button, the app performs the arithmetic operation on the result of the previous operation and the one that was most recently entered.

3. **Case 3. Starting a new operation (15 marks)**

   - The user enter another operation on two numbers and clicks the equals button. In this case, the app performs the arithmetic operation on the two numbers that were input, and ignores the result of the previous operation.

4. **Case 4. Performing an operation on multiple numbers (20 marks)**

5. **Case 5: Using the Equals and Clear button (20 marks)**

   - During or after any of the cases above, if the user clicks the clear button, then the app should reset itself back to the state in which the page was just loaded. It should not reload the page, of course, but rather should clear the display and "forget" the results of prior inputs or operations.
   - If the app is in the "reset" state – because the page has just been loaded, or because an operation was just completed, or because the user clicked the clear button – and the user enters one or more numbers and then clicks the equals button without first selecting an operator and entering another operand, the display should be the same and the equals button should be ignored. For instance, if the app is reset and the user clicks "2" and then "3" and then "=", the display should still read "23" and that value should be used as normal for the next button click.
   - Likewise, if the app is in the reset state and the user enters some numbers, and then an operator, and then clicks the equals button without entering another operand, the display should be the same and the equals button should be ignored. For instance, if the app is reset and the user clicks "1" and then "5" and then "+" and then "=", the display should still read "15" and that value and the "+" operator should be used as normal for the next button click.
   - However, if the user has just completed an operation using the equals button and then clicks the equals button again, the previous operation should be repeated using the result of the operation and the most recently entered operand. For instance, if the app is reset and the user enters "8" and then "+" and then "6" and then "=", the display should show "14" as normal. If the user enters "=" again, the "+6" operation should be repeated and the display should now show "20". If the user then enters "=" again, the "+6" operation should again be repeated and the display should read "26" and so on.

6. **Case 6: Selecting multiple operators (20)**

   - If the app is in the "reset" state and the user enters some numbers, and then an operator, and then a different operator, the first operator should be ignored and the second operator should be used in the operation. For example, if the app is reset and the user enters "6" and then "+" and then "*" and then "2" and then "=", the "+" operator should be ignored and the "*" operator should be used, so the display should read "12".

7. **What about...?**

   - You may encounter other cases that are not addressed in this document, e.g. what to do when the result of an operation exceeds the largest number that JavaScript can represent, or other sequences of clicking buttons that do not follow the ones described above. You may handle those cases in any manner you choose or ignore them entirely!

Notes

- You are not allowed to use *eval* function due to its security vulnerabilities.
- Each item is worth 20% of your grade.
- Keep in mind that JavaScript will perform concatenation on string variables, even if the values are numeric. So don't be surprised if you run into a situation in which '5' + 1 = '51'.
- You can use the JavaScript *Number()* function to convert a string variable to a numeric variable.
- Please properly cite any sources that you used. This includes any code snippets or examples that you found online. But you don't need to cite the tutorials or resources that you used for guidance.

Good Luck!