CET2045 - Web Programming - Practicum 03

Topics Covered: Javascript

Learning Objectives:

- Familiarize with Javascript interactive front-end coding. This includes
  - Event based coding to respond when buttons are pressed
  - Target elements using classes and IDs
  - Perform mathematical operations
- Apply Javascript coding best practices such as
  - proper and consistent naming conventions
  - proper Javascript structure throughout
  - resolving Javascript errors

Deliverables:

- Submit a single css file called `CET2045_P03_<Your_Name>.js` (e.g. `CET2045_P03_John_Doe.js`).

Background

A designer created a simple online calculator using html and CSS. Since he is not a programmer, he has enlisted your help to build the calculator's functionalities using Javascript.

Task

You are given a zip file consisting of a html file, a CSS file and a js file. A blank js file has been provided to you. You should build up the required javascript into this js file based on the instructions in order to meet the requirements.

Requirements

- Do not attempt to change anything from the html or CSS files, you will only be graded based on your js code.
- This is a simple calculator that only deals with integers
- When AC is pressed, any ongoing operations will be stopped and the calculator value returns to 0.
- Pressing the "=" button when no operation key has been pressed results in nothing, the value that is displayed remains the same
- When entering a value. any zeroes in front are ignored, eg: 054 will be simply 54

if i type 9 x 9 then equal then x 9 then i keep pressing equal, will it work?

allow storing of negative numbers into m and allow for appending of numbers behind it.

- Whenever a number reaches thousands or more (more than 3 digits), a comma (thousands separator) is added for display purpose, despite the added commas, subsequent operations should still be possible and valid. Eg 3,453, a comma is added after every 3 digits from the right.
- If the result of a completed operation is not an integer (eg. after division), display the rounded value with a "r" at the end. Eg. `5 ÷ 2 = 3r`. Note that the "r" is only for display purpose to indicate that there is a remainder. For subsequent operations, only the whole number is used - the "r" is ignored.
- Negative numbers are allowed only as result of calculation, there is no need to create any means for the user to enter a negative number
- There is a limit of 9 digits, pressing any additional buttons does nothing and should be ignored

  if you type 999999999 then type 9 again, it won't give you E, just stays at 999999999

  As it is a simple calculator, if the number exceeds range of 999999999 or -999999999 due to any operations, an error message "E" is displayed. Nothing else can work until AC is pressed to reset the calculator

take care of division by zero.

after you press M1 or M2, and you press a value, it will append.

- The calculator works on the principle of `first value => Operator => second value => Calculate(=)`. If either operator or second value is missing, the calculator will not proceed with any calculation
- Right after obtaining a result, the user can further calculate using `result => Operator => next value => Calculate(=)` and continue in this format unless an error or out of range occurs.

if i got result and i press operator and next value and press calc, i get a result, if I press calc again, it should not do anything.

- If the page has just been loaded, or after pressing the "AC" button, pressing any operators before a first value is entered will not do anything
- Right after obtaining a result, the user can append to the result by entering more numbers. Eg. `2+5=7`, if user presses "6" after the result has been displayed, "6" is appended to "7" and the value becomes 76.
- Pressing different operators one after another simply overrides the previous operation. Eg, `9 + - 5 = 4`. The addition operation (+) would be replaced with the subtraction operation. The final operator before the next value is entered will be hte operator that takes effect.
- M1 and M2 provides storage, pressing M1+ or M2+ will store the current value into the memory.
- If an error has occurred and "E" (Error) is being displayed, that will not be saved into the memory if M1+ or M2+ are pressed.
- M1 or M2 are retrieved by pressing the M1 or M2 button. If no values are stored before hand, the default value of 0 will be retrieved when the buttons are pressed
- Values stored in M1 and M2 will stay in memory even when AC is pressed. M1 and M2 will always store their current values except for 2 situations.
  - If the calculator page is refreshed, M1 and M2 will reset to values of 0.
  - If new values are saved using the M1+ or M2+ buttons
- Errors should not show up on the browser's developer console, whatever the sequence of buttons pressed. All potential errors by users need to be resolved nicely without throwing errors (errors show up in red in the developer console)
- Do **not** use any plugins or libraries (eg. jQuery), everything should be coded by yourself using vanilla Javscript (https://thisinterestsme.com/vanilla-javascript/)
- As Chrome is the most popular browser, compatibility is based on the Chrome browser.

For the maximum allocation of marks, refer to the table below.

Description	Marks (%)
Successful implementation of basic Calculator functions	45
Successful implementation of storage functions	15
Proper and sufficient comments to explain code	10
Proper and consistent naming conventions.	5
Program able to handle edge cases	25

Once you have completed, save your file in the following format `CET2045_P03_<Your_Name>.js` e.g. `CET2045_P03_John_Doe.js`.