Homework 2: JavaScript Calculator

Due date: January 30, 2017 at 11:59pm

This homework is the second in a series of three homeworks in which you will build calculator applications. In this homework, you will implement the calculator functions using JavaScript running client-side in the web browser. (For the third homework, you will implement the calculator functions by sending web request to a Django application running server-side.)

For this assignment, you will use the front end you created in the last assignment and add functionality by writing the calculator behavior with JavaScript.

The learning goals for this assignment are to:

- Gain familiarity with client-side programming by using JavaScript to add functionality to a static front end through DOM manipulation and event dispatch.

- Ensure robustness of a small-scale application with some simple validation.

Specification

This section describes the behavior of the calculator that you will implement. The calculator itself behaves as a simple four-function calculator.

- The calculator will use the front end that you wrote for the previous homework.
- The number displayed at all times is a single integer value. This value is initially zero (0).
    - … unless the user attempts to divide by zero, in which case you should reset the state and display an error message until the next button is clicked.
- The calculator does integer math (e.g. 9 ÷ 4 = 2).
- No operator precedence. Do the operations in the order they are encountered.

Below are a few examples of the expected behavior of the calculator (you may use these as test cases for your own calculator).

Button Pressed	2	4	+	1	9	×	2	0	-	1	0	=
Display	2	24	24	1	19	43	2	20	860	1	10	850

Button Pressed	6	8	-	9	7	=	1	5	÷	1	3	=
Display	6	68	68	9	97	-29	1	15	15	1	13	1

Requirements

Your submission must also follow these requirements:

- Your submission must follow all of the requirements specified in the last homework assignment.
- You may not use any external libraries (e.g. jQuery, Bootstrap, etc.).
- You may not use the JavaScript `eval()` function in the implementation of your calculator.
- Your calculator must be able to support numbers up to ±1,000,000. We will not test numbers beyond that range, so you don't have to worry about overflow.
- Your JavaScript may not crash at any user input—there must not be any errors reported by the browser JavaScript console.
    - If two (or more) operation buttons are clicked without a digit in between, you must not crash. The computation for this is up to you.
- All of your JavaScript functions must be defined in a file called `calculator.js`. Minimize the amount of JavaScript written directly in your HTML file.

Implementation hints

- You may want to maintain an internal "new value", "previous value", and "previous operation". Initially, "new value" and "previous value" would be zero, and "previous operation" would be "+".

- When a digit is pressed, update the "new value" accordingly (multiply by 10 and add digit) and then display "new value".

- When an operation is clicked (+, -, ×, ÷, or =), use the "previous value", "previous operation" and "new value" to do the computation and get a "result" to display. Update as follows:
    - "Previous value" ← "result"
    - "Previous operation" ← operation clicked
    - "New value" ← 0

- Note that after an operation button is clicked, the display will be replaced by the next digit that is clicked. (E.g. if the last button clicked is an operator and the current display is 43, then clicking 1 will change the current value to 1, not 431).

- Your calculator will not be able to support entering in negative numbers, but it should be possible to use the negative numbers resulting from previous operations.

Grading criteria

This section contains some specific criteria we will follow while grading your homework. However, keep in mind that *for substantial credit your solution must clearly demonstrate the learning goals for this assignment, which are described above in the introduction.*

Committing your work [10pt]

As with the previous homework, we will be evaluating your version control usage. Keep in mind that good version control usage typically means (1) incremental, modular commits with (2) descriptive and useful commit messages.

Specification fulfillment [30pt]

Validation [10pt]

As mentioned in the learning goals and requirements, your JavaScript should not crash or fail at any input. We will be evaluating how robust your application is through various test-cases including the cases mentioned above.

Coverage of technologies [50pt]

Turning in your work

Your submission should be turned in via Git in a directory called **`hw2`** and should consist of an HTML file called `calculator.html`, a JavaScript file called `calculator.js`, and associated assets (CSS and images). You may want to organize your files logically. Here is an example directory structure (some directories/files omitted) of a submission.

```
[YOUR-ANDREW-ID]/hw2/
  |-- calculator.html
  |-- js/
    |-- calculator.js
  |-- css/
    |-- calculator.css
  |-- README.md
```