Project 3-JavaScript Report

HTML and Bootstrap

The first step to completing this assignment was to find a Bootstrap theme to use to create the application. In the previous project, I used a relatively complicated theme which did not require me to create some of the elements I used from scratch. For that reason, I decided to use the Scrolling Nav template from Start Bootstrap, which was very simple and only contained a scrolling navigation bar and a few sections containing paragraphs in them. It gave me the perfect simple base structure to build the application. It was simple enough that it required me to create many of the elements that I needed, but also contained enough to have a strong base structure to use. Once I had the template loaded into Brackets, I modified the HTML to create three input boxes for the numbers and a button to calculate the output. I also included an empty div that the JavaScript would output the results into. I also modified the navigation bar and added some information about the application.

JavaScript

The second step was to write the JavaScript for the application. I have never used JavaScript before, but I have a background in C++, so the learning curve for the JavaScript was not very difficult. I needed to use references such as Zybooks and W3Schools to create the event handlers and get elements from the DOM but writing the functions to handle the calculations was relatively straight forwards. Additionally, I needed to use refences to understand how to add CSS classes to HTML elements and how to modify the HTML inside certain HTML elements.

Possible Improvements

My base application included three textboxes and a calculate button. Each textbox received an input of one integer, and when the calculate button was pressed, the required calculations were displayed below. One possible improvement is the creation of a single textbox so that a larger set of numbers can be inputted. Additionally, the base application only accepted integers and produced logic errors if floats were used, so another improvement is to allow floating point numbers to be inputted. The application also did not include any error checking. If incorrect inputs were provided, the output would either be incorrect due to a logic error or would display NaN. An improvement would be to add error checking to prevent the calculation from occurring if the input is incorrect. A final possible improvement is to add other calculators that would preform more advanced operations on the set of data.

Implementation of Improvements

I implemented all of the improvements listed above in my final app. I modified the JavaScript so that it would accept a single textbox input that contained floats or integers separated by commas. The JavaScript separated the numbers into an array and parsed the array into floats. The floats were then checked for correct input, and if they were the calculation would be completed. If the input was incorrect, the textbox would turn red and an error message would be displayed. Due to the way I implemented the input parsing, if the user inputs an incorrect symbol after the number but before the comma (e.g., 1 a, 3, 4, …), the symbol would be automatically disregarded, and no error would appear. In this way, the program functions as an input interpreter if the input is incorrectly inputted in a certain way. The final improvement that I added was two additional calculators that had all the features that my initial calculator had, but preformed different operations. My second calculator preformed variance and standard deviation calculations and my third calculator performed a sum and product of the data set. I called the calculators advanced and other, respectively.

Final Thoughts

One remaining improvement that I would like to add if I had the opportunity is to make the application handle floating point error better. If certain float inputs are used, the output is off by a very insignificant amount (e.g., an error of 0.000000000001) due to floating point error. I could deal with this by rounding the output to the decimal place of the most precise input. This would remove the insignificant error from appearing because it would round down. I have written an algorithm before that preformed a function similar to this, however I do not know how I would go about it in JavaScript because I wrote it initially in C.