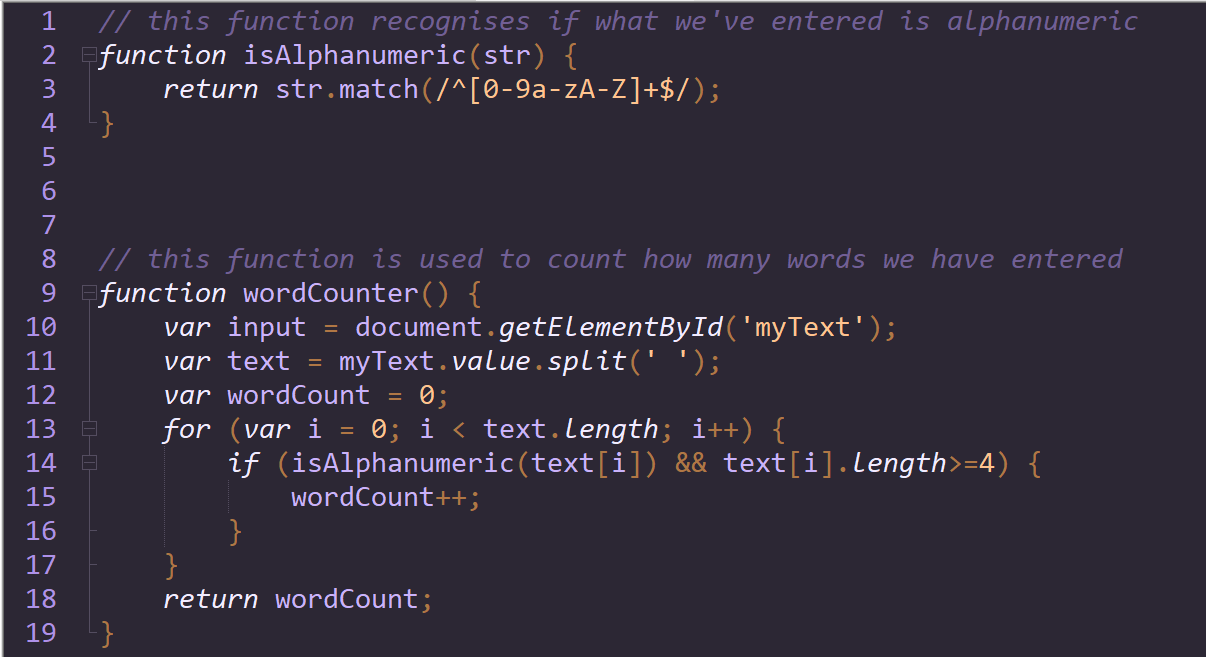
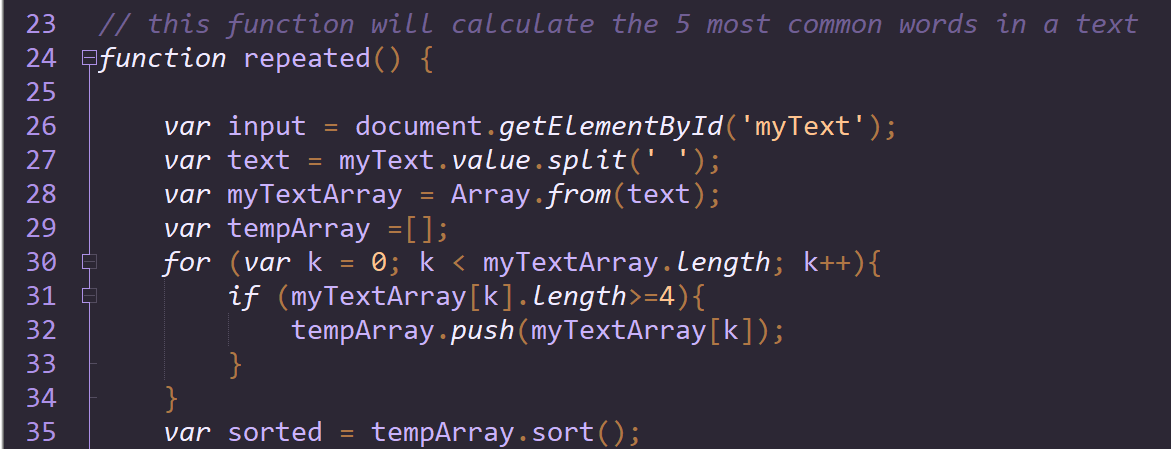
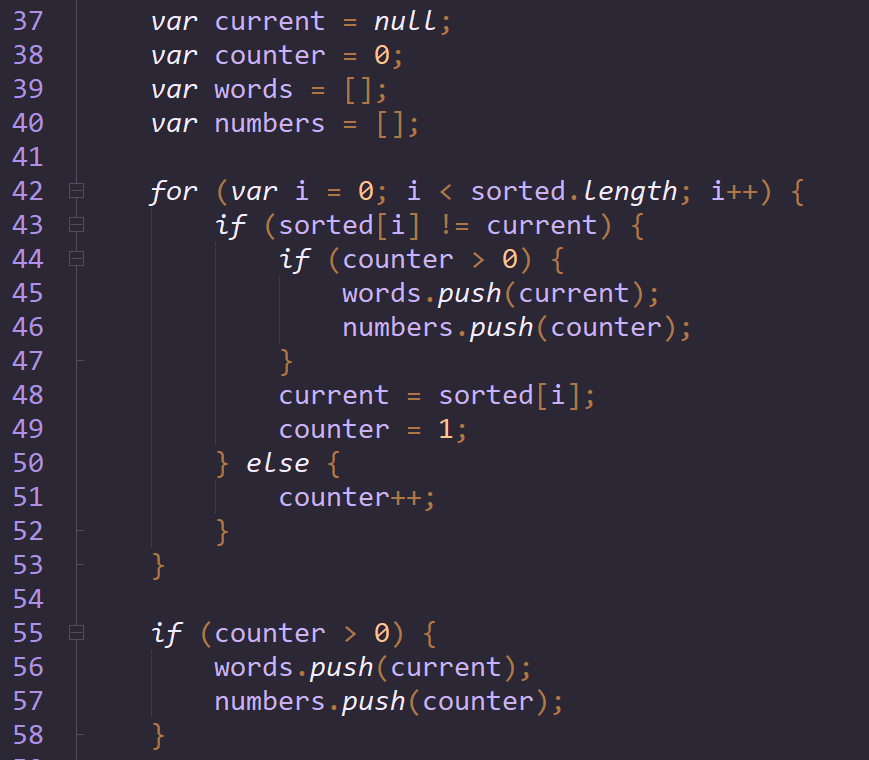
**Please read!!! It won’t take more than 5 minutes!!!**

The HTML code is pretty simple, so there is no need to explain it.

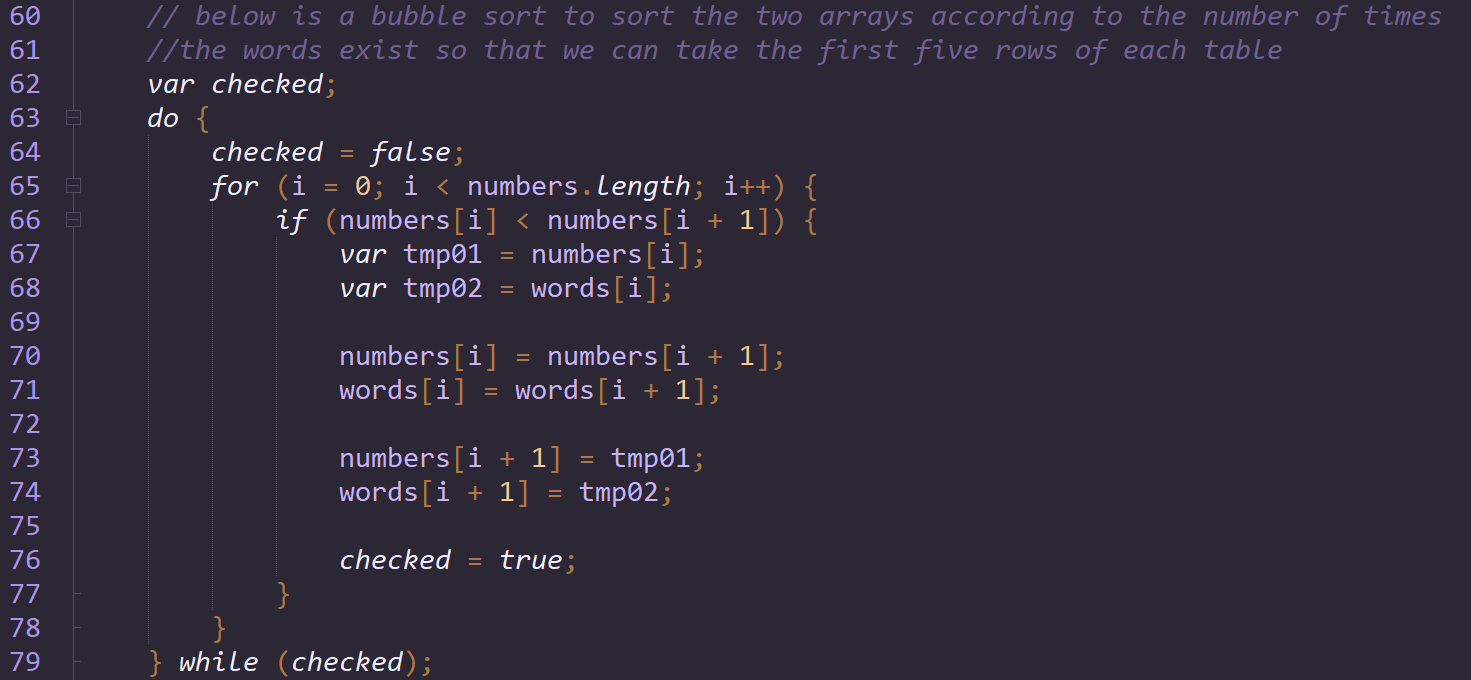
I had an assignment that required the user to write a text in a box created with HTML and then use JavaScript to alert the user which were the 5 words that appeared the most. I will now explain the code in detail. If you use this code, please try to understand the logic first and then try to do it yourself. Copy the code only if you have no time to work on it. It took me two days to work my way through the logical and syntax errors, but it was a beneficial experience.

These two functions are really simple, but essential. The **isAlphanumeric()** is needed so that we can get the text from the text box. The **wordCounter()** will count how many words we have entered. I have added an *if* statement to “ignore” any words that had less that 4 letters for the sake of the assignment, so you can skip this step and just have the command *wordCount++* in the *for* loop. I initially made this function so that I could experiment with JavaScript and an easier version of this exercise, but later it proved to be useful (I will explain it when the time comes).

With the following lines of code we get the text input the user gives and use the split(‘ ’) method to split it according to the spaces between the words and put each word in an array named **myTextArray**. Next, an array named **tempArray** is created and the reason why is because the exercise wants us to only take into consideration words that have 4 letters or more. This sounds like what I did in the **wordCounter()** function, but both of them are needed. The program runs fine without the code from the **wordCounter()** function, until the time to show the alert comes. If I didn’t use it, then if I didn’t type any words, the program took it as a *space* and just created two arrays with a single element, displaying the alert *“ comes 1 times.”*, which is wrong. This is why I check for the length of the words twice in the code. Finally, we sort the array in alphabetical order, because it will help us later.

Now we want to go through the array with the sorted words and count them. The **current** variable will be the word we are currently counting, the **counter** will be the number of times this word exists in the text, the **words** array and the **numbers** array will be arrays of equal length which will store the words of the text and the number of times they appear. *It’s important to understand that the difference between the* ***sorted*** *array and the* ***words*** *array is that the* ***sorted*** *array has duplicates of the words and the* ***words*** *array doesn’t’* (if it’s confusing now, it will become clear later).

For the first time the program runs **current** is null, so it will become the first item in the sorted array and **counter** will become 1, meaning we have our first word and we've counted it one time. *Since the array is sorted, then if a word is shown multiple times, it will occupy multiple consecutive places in the array*, so the next time the loop runs the code under *else* will run, increasing the count of that word. When we reach a different word then we push the current word in the array **words** and the number it exists in the array **numbers** and then we make the new word the **current** word and set the **counter** back to 1. The last *if* seen in the screenshot only exists to push the last word into the array **words** and the number it appears on the **numbers** array.

The comment on this section pretty much says it all. At this point, we have two arrays that are sorted according to the alphabetical order of the **words** array. Since we want to show the user the most common words, we need to sort the tables according to the **numbers** array, in a descending order (meaning going from the word with the highest count to the word with the lowest). Bubble sort has high complexity, so you could probably use other sorting algorithms.

Now here’s the final part, the part where we show an alert to the user, depending on what they wrote. We have three *if* cases: the user has typed …

* Nothing
  + In this case we will use the **wordCounter()** function with the help of a variable. If the user didn’t type anything the message *“You did not type any words.”* will appear in the alert.
* Less than 5 different words
  + In this case we can only show in an alert the number of different words the user gave and the times they appeared, so that’s why the loop runs for the length of the ‘numbers’ array.
* 5 or more different words
  + In this case we can use 5 for the number of times the loop will run, since we definitely have 5 or more rows on the arrays.

