****

**Assignment Cover Sheet**

**BSc Creative Computing**

To be completed **electronically** by the student and submitted with each piece of work. Please upload this completed cover sheet via TurnitinTM .

**Assignment Title:**Tomorrow’s Web (Proof of Concept - Assessment 2)

**Tutor:** Billy McQueer

**Student Name:** Kyle Daniel D. Co

**Student Number:** 415220

**Date of Submission:** 1/5/2023

**Details of your submission**

Online submission: Enter the URL of where you project files can be accessed (e.g., Google Drive)

|  |
| --- |
| **GitHub Repository Link:** <https://github.com/KyleDaniel-BSU-04-009/TEXT-BASE>  **YouTube Video Link:** <https://youtu.be/dIN596N56NM>  **Google Drive Link:** <https://drive.google.com/drive/folders/13xOnoG0R5zzJFH6f659NQbRfRpaFpdHv?usp=share_link> |

In submitting this assignment, I am confirming that I have read and understood the regulations for assessment, and I am aware of the seriousness with which the University regards unfair practice.

Signed: Kyle Daniel D. Co Date: 1/5/2023

|  |  |  |
| --- | --- | --- |
| TEXT BASE | December 18  2022 | |
| By Kyle Daniel Co | | CC Y3 | #415220 |

Contents

[Abstract 3](#_Toc123851246)

[1.0 Features 4](#_Toc123851247)

[2.0 API Research 4](#_Toc123851248)

[3.0 Main Development 5](#_Toc123851249)

[3.1 Front-end 5](#_Toc123851250)

[3.2 Back-end 7](#_Toc123851251)

[4.0 Technical Description 9](#_Toc123851252)

[4.1 Unload Warning 9](#_Toc123851253)

[4.2 Unobtrusive Code 10](#_Toc123851254)

[4.3 Code Readability 10](#_Toc123851255)

[4.4 Selection Styling 11](#_Toc123851256)

[4.5 Editor Copying 11](#_Toc123851257)

[5.0 Reflection 12](#_Toc123851258)

[5.1 Successes 12](#_Toc123851259)

[5.2 Code optimization 13](#_Toc123851260)

[5.3 Limitations 15](#_Toc123851261)

Abstract

TEXT BASE is an applied proof of concept of a text editor that features a word counter, character counter, and Grammarly's API. The purpose of Grammarly's API is to enhance the experience of the user when using TEXT BASE. With the API applied, TEXT BASE is also able to provide service for different dialects and writing styles.

1.0 Features

TEXT BASE is a text editor that features Grammarly's API. Its features include the bullets listed below:

* **Word Counter** - This feature provides convenient data for the user; specifically the number of words that have been typed inside the editor. It is located at the right side of the screen.
* **Character Counter** - This feature will show the user the amount of characters that they have typed inside the editor. It is located at the right side of the screen below the word count.
* **Dialect Selection** - This is a feature that allows the user to select which dialect they prefer to use when writing with TEXT BASE. The functionality is provided by Grammarly's API and lets the user select between 4 individual dialects being American, Australian, British, and Canadian.
* **Style Selection** - This is also a feature that is provided by Grammarly's API and it allows the user to select which style they prefer to write in with TEXT BASE. It lets the user decide between 6 options being academic, business, general, mail, casual, and creative.

2.0 API Research

I conducted my research based on the following conditions and the API that fit these conditions will be applied:

* Clear API documentation
* Multiplatform compatibility
* Service Availability

These are the requirements that I have decided to seek when searching for an API to apply for this project; clearly, Grammarly met my requirements. Upon further research, Grammarly requires a client ID in order to allow developers to use their API. All in all, Registration was easy and did not hinder further development.

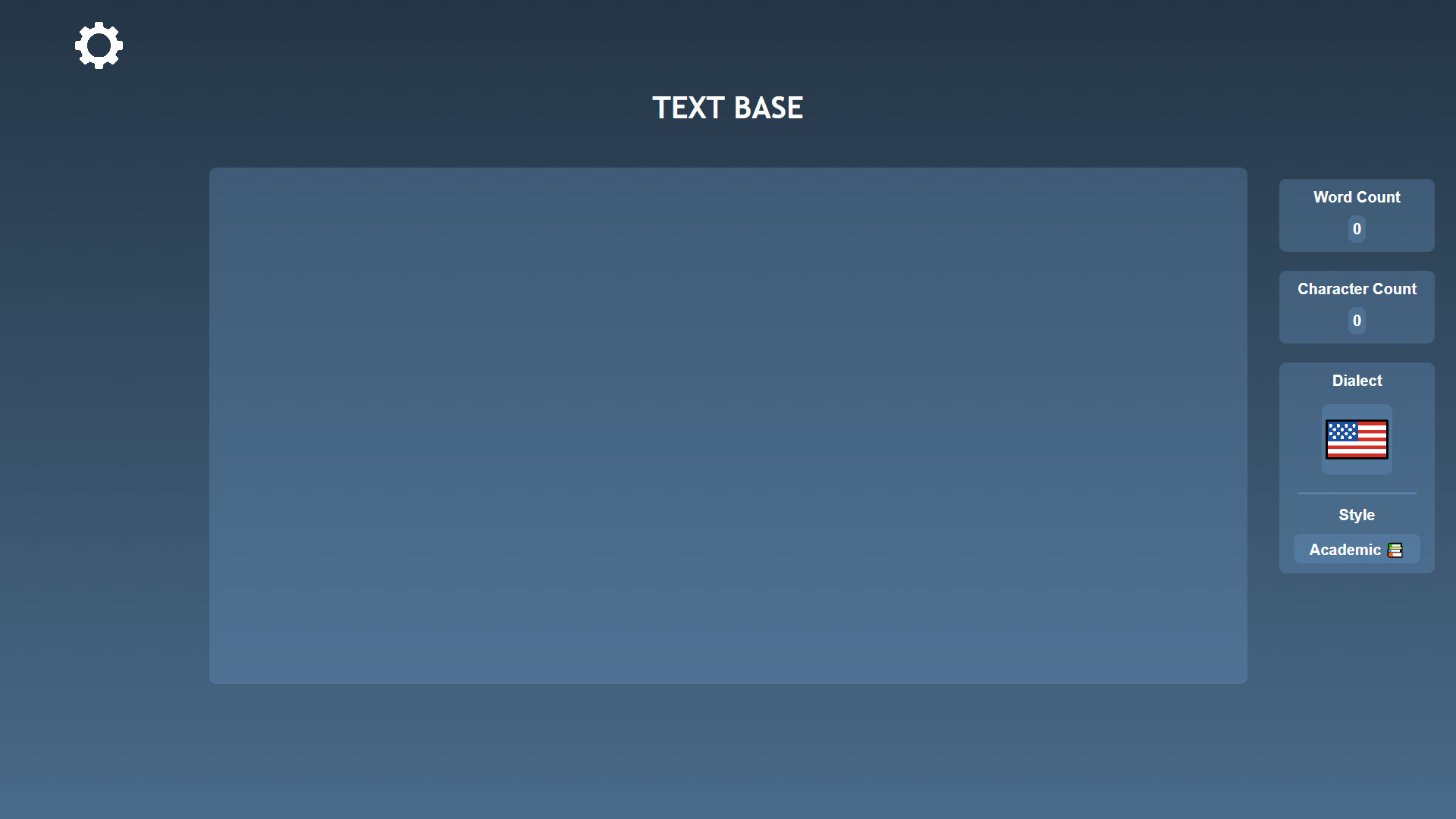
Furthermore, the service is only available online; this means that TEXT BASE needs to be hosted in a web server in order for Grammarly's API to function. During the development of TEXT BASE, the site was hosted using a web server plug-in from Visual Studio Code. The testing phase went smoothly and ran without error.

The reasons behind the conditions listed above are all considerations of clarity and efficiency. The project needs to meet its goal with the aid of an appropriate API.

3.0 Main Development

3.1 Front-end

* **Style** - The style that TEXT BASE revolves around modernity, as it is inspired by other modern user interfaces. I chose a dark-to-light gradient for the background and used a lighter transparent tone for everything else.



***Figure 1 shows TEXT BASE’s main page***

* **Animation** - Majority of the elements of TEXT BASE use animations. The main input and results section blurs out of the background whenever the user decides to hover over the settings icon; this will slide a menu from the left and introduce the user to the settings menu where the dialect and style selection is located.



***Figure 2 shows the activated settings menu and what it does to the background***

* **Font** - There are 3 fonts used in TEXT BASE. 'Arial' is used for the main text editor, 'Trebuchet MS' is used for headers, and 'Helvetica' is used for everything else such as selection options and results.



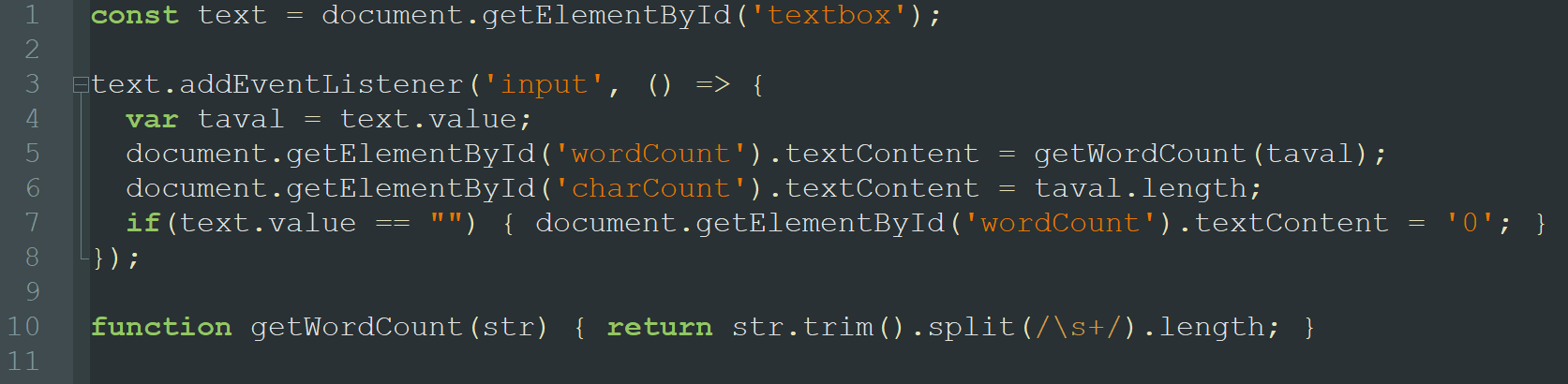
***Figures 3, 4, and 5 shows all the fonts that TEXT BASE uses***

3.2 Back-end

* **Word and Character Counter** - The word counter gets updated whenever the user types something in the editor. Each time a 'word' gets typed in the editor, a function immediately returns the length of the editor's value by trimming the string and splitting it each after spaces; this value will change the DOM property of the word counter. The character counter gets updated the same way but changes the DOM property of the character counter based on the editor's value.

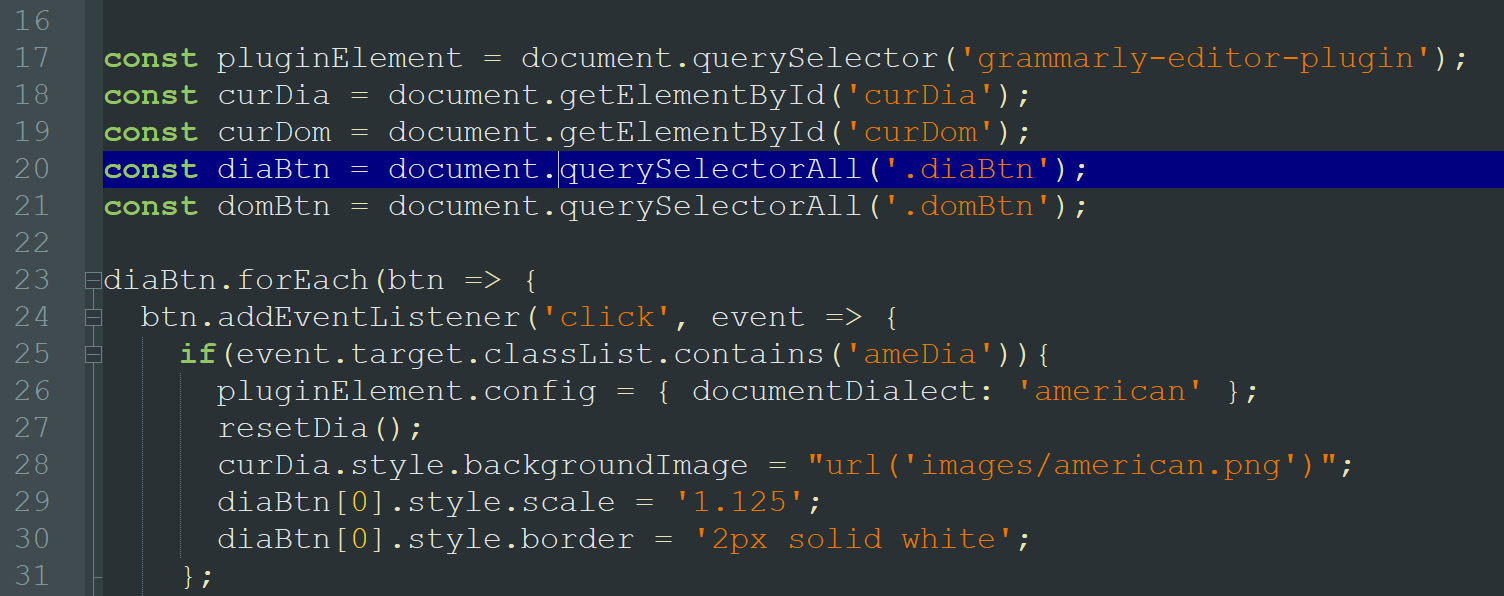


***Figure 6 shows the results section where the current data is presented***

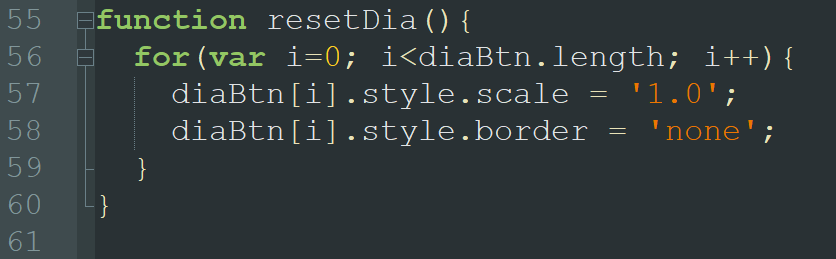


***Figure 7 shows the code for the word counter and character counter***

* **Dialect and Style Selection** - The dialect and style selection process both have the same concept. When the user selects an option from each value, it calls a function to Grammarly's API to change the dialect or selection. Afterwards, calls a function with a for-loop that resets the current selection from the options menu. Lastly, it will change the DOM elements from the results bar to the new selection of the user.

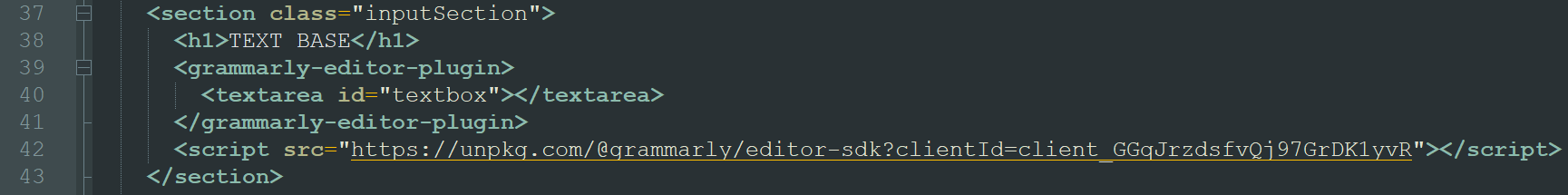


***Figure 8 shows the code for dialect selection***



***Figure 9 shows the code of a function called in the process of selection where it resets the styling***

* **Grammarly's Menu Implementation** - This feature is only available as long as TEXT BASE has an internet connection; it needs to connect to Grammarly's services. Whenever the user types something in the main input, a small loading icon will appear in the bottom right corner of the screen. If you made grammatical errors, a number will appear and the problematic word or section of your input will have a thick underline (completely different from default). Clicking these underlines will open Grammarly's menu and allow you to either ignore it or make changes to it.

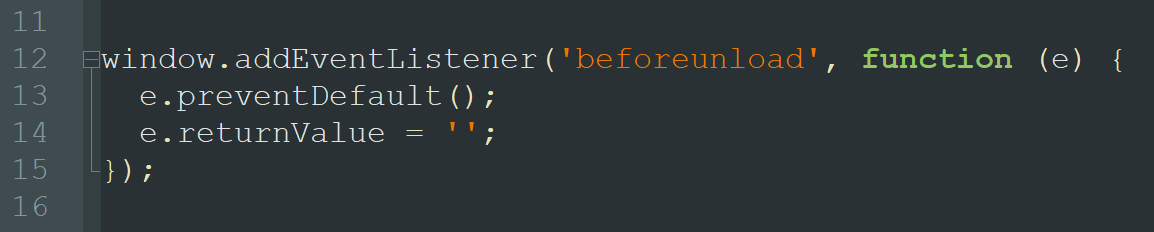


***Figure 10 shows the code where it implements Grammarly’s API and allows TEXT BASE to use it***

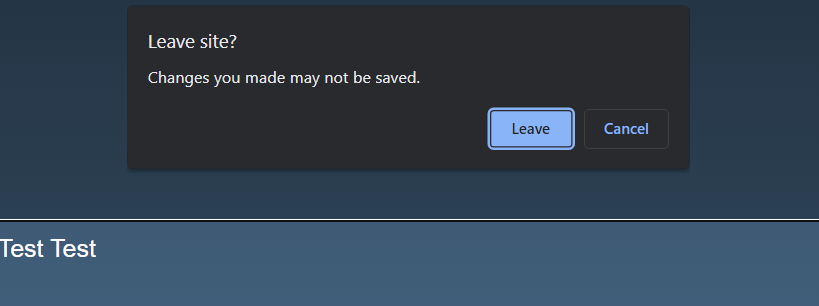
4.0 Technical Description

4.1 Unload Warning

Regarding the technical aspects, TEXTBASE mainly uses javascript to detect changes made in the editor and Grammarly's services to spot mistakes and errors made by the user. TEXT BASE runs a function whenever the user switches to a previous or the following page, restarting the page, and closing the page. This function will warn the user that the page will close and any changes made to the site will be gone.



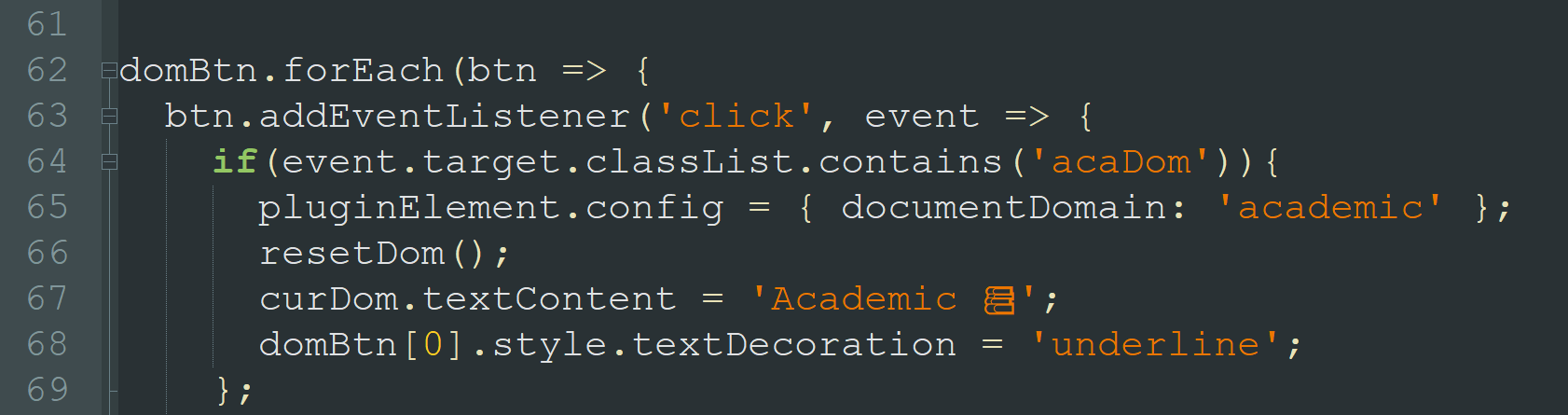
***Figure 11 shows the code that warns the user when unloading TEXT BASE***



***Figure 12 shows what happens when the user attempts to unload TEXT BASE***

4.2 Unobtrusive Code

The javascript is written unobtrusively; there is no inline javascript found in the main HTML file. All events and functions are handled in the main javascript file and collectively add event listeners for each button. Each button gets checked if they have a certain class before they make any changes in the DOM.



***Figure 13 shows the code for the main button functionality when selecting a writing style***

4.3 Code Readability

The readability of the code is aided with comments; a short explanation is provided with each integral part of the javascript to increase its readability. Nesting statements, spaghetti code, and convoluted functions have been significantly reduced as an improvement from the first version.



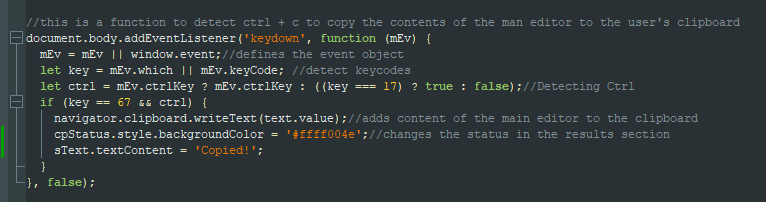
***Figure 14 shows the dialect selection code and its comments***

4.4 Selection Styling

After every selection made by the user, every DOM element styling related to that option will be reset thanks to a function getting called. It cycles through the respective selection's index and reset its styling to a default unselected version before the final changes are made. Please refer to figure 8 for the previous visual aid.

4.5 Editor Copying

When the ‘CTRL’ key is detected as well as the ‘C’ key, a function will be called to take the value of the main text editor. The function will also change the styling of its status.



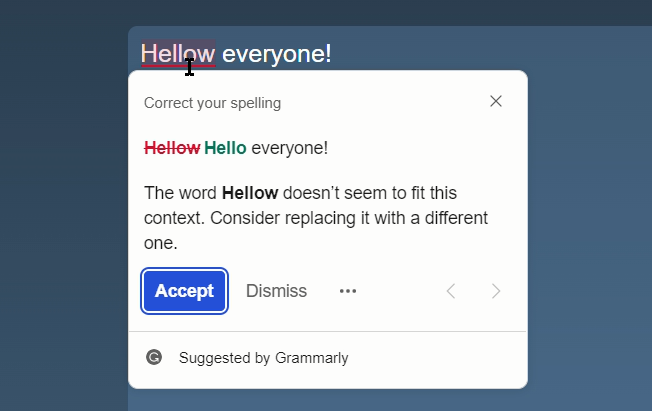
***Figure 15 shows the code for copying the text editor’s content using a function***

5.0 Reflection

5.1 Successes

**Successful API implementation**

Grammarly's API is very easy to implement. Instructions are provided in their developers page to help with implementation. They prepared an HTML ready with a script tag that contains your client ID.



***Figure 16 shows the successful implementation of Grammarly’s API***

**CSS Success**

Styling TEXT BASE successfully was intermediately difficult. The difficulty was more focused on the settings menu sliding from the left when hovered over by the user. It was successful nonetheless. Please refer to figure 2 for the visual aid.

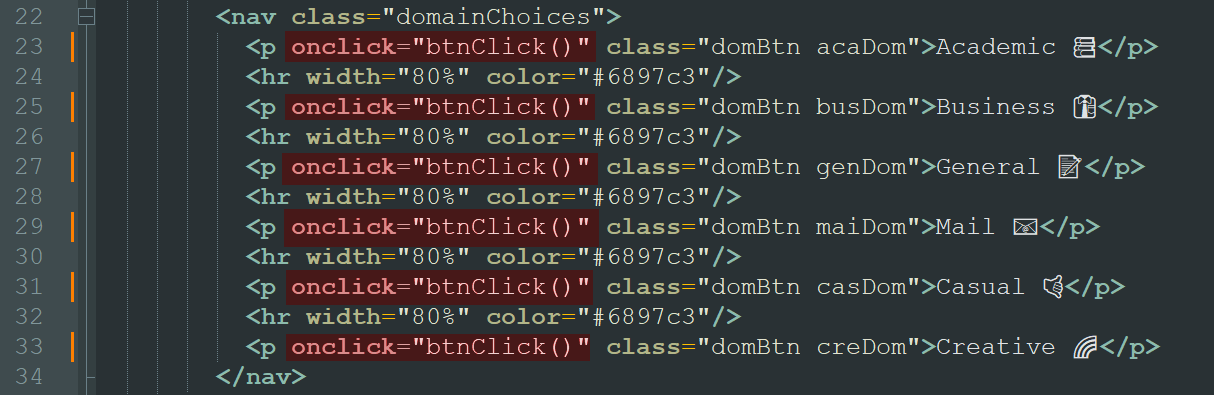
**Javascript Execution**

TEXT BASE's functionality greatly depends on javascript; every button has a function that will be executed when called. TEXT BASE's javascript does not depend on any additional libraries.

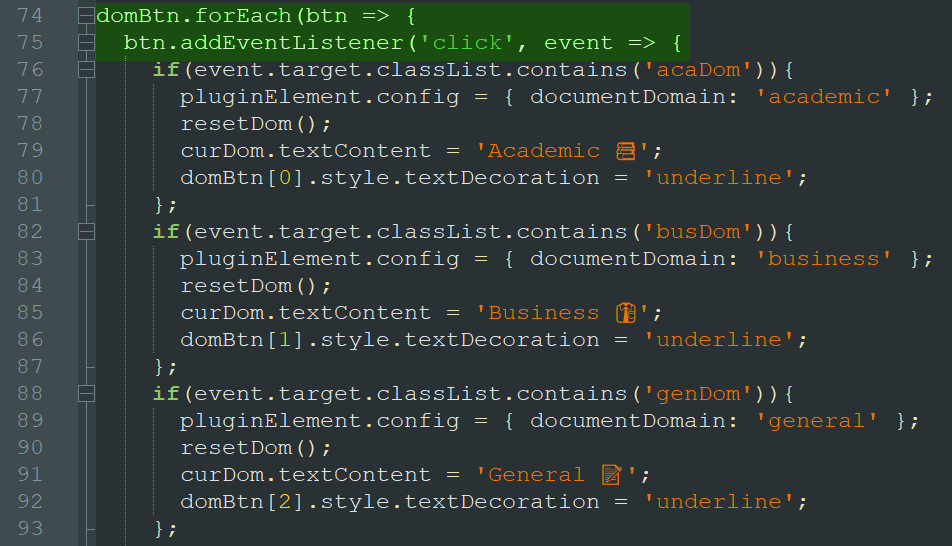
5.2 Code optimization

**Obtrusion**

Being used to basic javascript level coding, inline javascript was used in the initial phase of development. After a reading javscript codes made by other developers from the industry, I decided to optimize it and remove inline javascript from the main HTML file.



***Figure 17 shows obtrusive code; inline javascript***

****

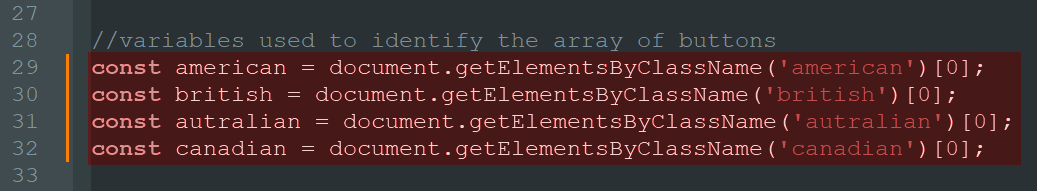
***Figure 18 shows an external javascript code that prevents obtrusion***

**Readability**

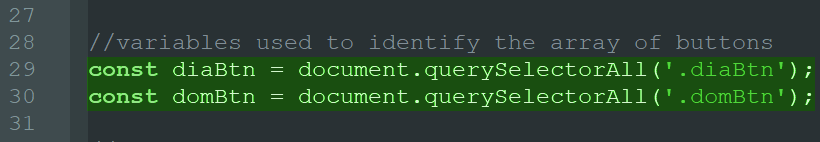
Further analysis of work made by other developers in the industry, I decided to add short comments and describe what each action of segments, functions, and events for future reference (including me). Please refer to figure 14 for code commenting.

**Flexibility**

The javascript I developed for TEXTBASE before is hardcoded; this means that it doesn't leave any room for future adjustments. Further improvements involved dynamic values that adjust in case TEXT BASE gets updated.

******

***Figure 19 shows spaghetti code; hardcoded values***

******

***Figure 20 shows simplified variables; an array of values that allows flexibility***

5.3 Limitations

**Pay wall**

Grammarly's API holds advanced functions and features that are bricked behind a pay wall and so, TEXT BASE is using a free and costless plan. The additional functionality that the premium membership provides does not qualify for the basic direction and intent that TEXT BASE leads.

**Network**

Grammarly's API is limited as it only functions when it's connected to the internet. TEXT BASE's basic functionality such as a word and character counter however is not hindered by the absence of an internet connection. Though TEXT BASE does use Grammarly's service to fulfill its primary function, it certainly cannot fully operate without the internet.

**Web** **Server** **Breakdown**

The web server that TEXT BASE was supposed to use kept breaking down, this has caused massive delays during the development and texting phase and schedules had to be extended by the developer. Fortunately, after seeking aid from fellow developers and colleagues, TEXT BASE was able to run on a local web server and was successfully tested.