11/2/2022

Web Development 2.1

**Web Development 2.1**

TABLE OF CONTENTS

[Introduction 3](#_Toc118297053)

[HTML Content 3](#_Toc118297054)

[CSS Content 4](#_Toc118297055)

[JavaScript Content 7](#_Toc118297056)

[Conclusion 12](#_Toc118297057)

# Introduction

My goal in this project is to create a Calculator using HTML, CSS and JavaScript Programming Language. The calculator will be running on a web browser. User will be able to add, subtract, multiply, divide and get the power of desired numbers. User will be able to make multiple operations at a single time - once the “equals” button is pressed, the user will get the result.

The logic behind this project is to expand my knowledge on how to create applications and run the final product on the web browser, gain an understanding of functions in JavaScript programming language, as well as become proficient with CSS styling.

# HTML Content

As my starting point for the page, I have created new .html file called Index which includes doctype and some standard html tags like <html>, <head>, <body>. I have also linked w3 styling as well as my custom CSS and JavaScript file to the <head> tag.

Text

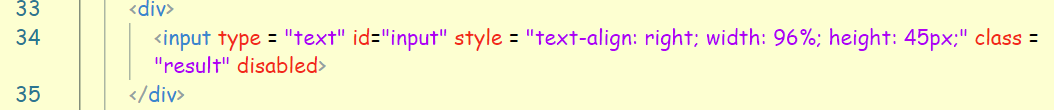
Description automatically generated

Once I had the standards done, I started creating my Calculator in the <body> tag. Firstly, I created the layout by using <table>. Shortly after completing the calculator buttons, I decided to change my tactic and use <button> as well as <div> tags instead. Each row is in a new div. Depending on whether the button is a number or an operation, it has a different CSS class assigned to it. As there is one less column in the first row of buttons, I have assigned width to the AC button (All Clear) which fits the gap. Each button has an appointed function that performs the desired operation.

Text

Description automatically generated

As a display screen, I used a text box and disabled it for the user.



The final input to the HTML page was <footer> tag which includes my student details. I have included W3C CSS and W3C XHTML validation images after validating my page and checking for errors.

Text

Description automatically generated with medium confidence

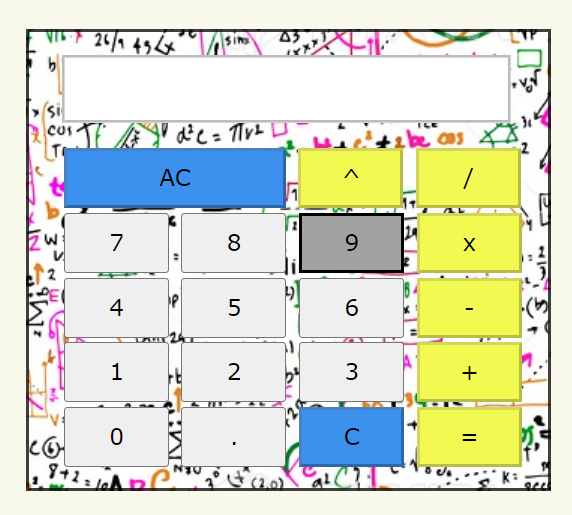
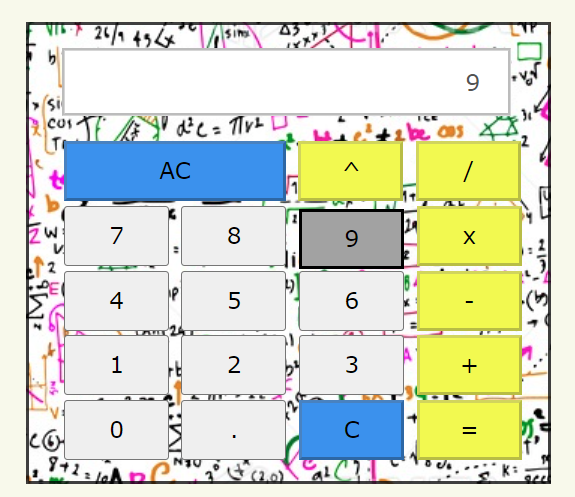
# 

# CSS Content

To style the Calculator I created a few classes: numbers, operator, and clear. Each clickable button has a hoover style and pressed action added to it. The buttons that the user hovers over are highlighted in different colours and once it’s pressed, the “click” effect will happen, which will move the button down by 2px.

Timeline

Description automatically generated



.calculator class styles the overall layout. It has a solid border added to it as well as a background image to make it look more interesting. Content is also centered. To center the actual calculator in the middle of the web page, I implemented another class called .center.

Text

Description automatically generated

For the output text box, I implemented a class called .result which includes groove border style – a border is displayed with a carved appearance.

Text

Description automatically generated

I have also added a background color to my page and styled the header to make the page look more appealing. Once I had that done, the page looked a bit boring, so I decided to include custom font to the header. I achieved that by adding style tag to my html file which includes the path to the desired font. Once I got the font added, I assigned it to h2 tag.

Timeline

Description automatically generated

The final application:

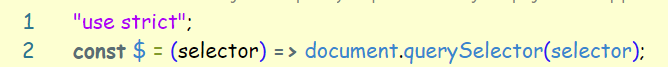
Graphical user interface

Description automatically generated

# 

# JavaScript Content

Firstly, in my JavaScript file, I have created a dollar method to return element within the document that matches the specified selector.



Then I created a function to clear the text-box output. That action will happen once AC (All Clear) button is clicked.

Text

Description automatically generated with low confidence

Graphical user interface, text

Description automatically generated with medium confidence

Function append reads the value of pressed button and adds it to the text-box output. It allows user to see what numbers or operations have been selected. In addition, I have created a limit of possible entries to 25 characters (including numbers and operations). If the user will try to enter more, an alert will appear to notify of the input limit.

Text

Description automatically generated

Text

Description automatically generated

In addition to the above condition, I implemented button “C” (clear) that will delete last entered input. This extra function lets the user delete last entry, rather than clearing the whole input. I achieved the above by converting the text-box value into String and then using substr method to delete last element from the string. New output is visible in the text box.

Text

Description automatically generated

After cosmetic functions were finished, I had to implement main method that calculates the result. I achieved that by using eval() function. The eval() function is mostly used in applications that need to evaluate mathematical expressions. It takes the string expression and returns the integer as a result. Eval() is not recommended as invoking it can possibly crash the computer and in general, is classified as dangerous and not secure.

Text

Description automatically generated with medium confidence

While testing my calculator and its functionality I came across a few errors that needed to be looked after. When it comes to user input – we never know what user will enter. I decided to add error handling if last input is an operator and prompt user to finish the expression with a number. I achieved that by using JavaScript slice method which takes the last element of a String.

Text

Description automatically generated with medium confidence

Text

Description automatically generated

The second condition handles first input. I used JavaScript substring method that takes the first element of a String and then checks the condition – once it is operator divide/multiply or power of, user will be alerted to start the expression again and the textbox will be cleared with clearResult() function that I already implemented. I left out plus/minus since these two operations are not affecting the calculations. What’s more, user needs to be able to start calculations with negative numbers.

Text

Description automatically generated with medium confidence

Text

Description automatically generated

Since there are many more possible errors that could occur, I decided to use try-catch statement. If the error occurs in the try block, it will be caught by catch statement and will alert user of the error. This method allows for errors to happen without crashing the application.

Graphical user interface, text

Description automatically generated

Text

Description automatically generated

# Conclusion

In conclusion, I created a fully functioning calculator using JavaScript language. I styled it by CSS file, and it can be run on the web browser by opening index file. All possible errors are handled appropriately, and user is always notified about them. I enjoyed creating my own calculator, and I will end up using it for my Maths assignment. The calculator could have more options like percentage or square of – but that is something I might consider adding in the future.