**Web Development 2.1 – Project**

**Report**

**Name: Wye Zhen Ng**

**Student Number: A00290501**

**Year: 2**

**Course Title: Bachelor of Science (Honours)**

**Software Design in Artificial Intelligence for Cloud Computing**

**Lecturer: Enda Farrell**

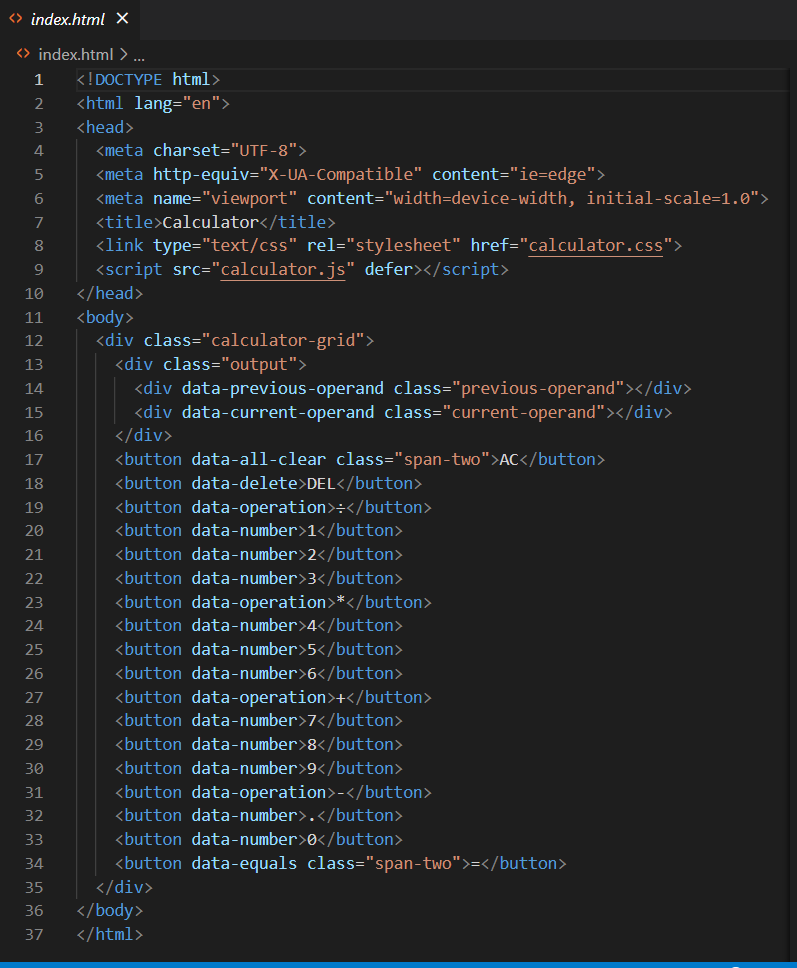
**Project Title: Designing a Calculator**

**Introduction**

The goal of the project is to test students on their ability to create a basic html page, styling the html page using CSS and to create interaction within the page using JavaScript. The task was to design a calculator. When designing a calculator, the first things to consider is that the page must have the basic buttons of a calculator. Second is to style the calculator so that it should look like a calculator. Third is to ensure all button are working as intended.

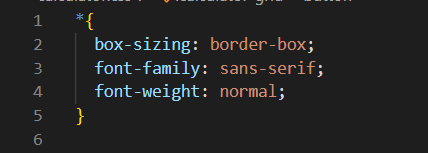
**HTML**

Things to note in the html is that it’s link to the calculator.css file for styling and the calculator.js file for coding. Inside the body element, the div element with class “calculator-grid” is to indicate the area in which will be use for the calculator. Inside of this div element, the div element with the class “output” is the area in which the output of the calculator will be displayed. Within this div element contain another 2 div element which are have the class “previous-operand” and “current-operand”. The “previous-operand” is to display inputs that are previous inputted by the user and will be evaluated based on the operation and the value of the “current-operand”. The “current-operand” displays the inputs that are currently being entered by the user as well as the result of the evaluation. There are a total of 18 buttons which are “AC”, “DEL”, “÷”, “1”, “2”, “3”, “\*”, “4”, “5”, “6”, “+”, “7”, “8”, “9”, “-”, “.”, “0”, “=”. The buttons are assigned to its custom data types based on its type. The “AC” button is assigned to the data-all-clear data and the class “span-two”. The “DEL” button is assigned to the data-delete data. The “÷”, “\*”, “+”, “-” buttons are assigned to the data-operation data. The “1”, “2”, “3”, “4”, “5”, “6”, “7”, “8”, “9”, “0” buttons are assigned to the data-number data. The “=” button is assigned to the data-equals data and the class “span-two”.

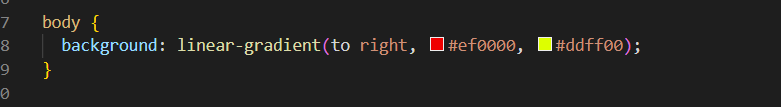


**CSS**

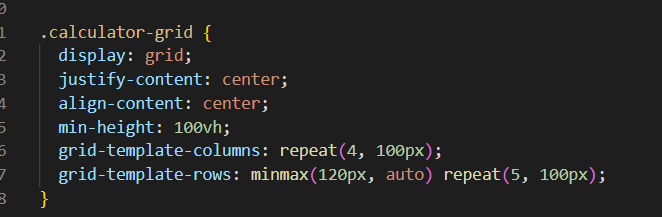
All elements have been padded using box-sizing property to border-box. The font of all element has been changed to sans-serif font. The font boldness has been adjusted to normal.



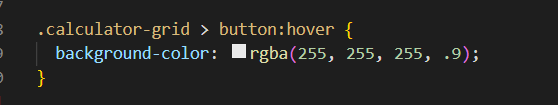
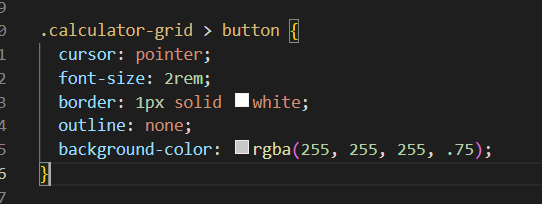
The body element has been styled where its background colour is red from the left and is linearly changing to yellow as it travels to the right.



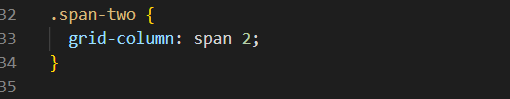
The calculator-grid class has been styled using the grid layout format and all its content is justified to the centre. The alignment of the content is also centred with a minimum height of 100vh. In the grid, it has been set to 4 columns with 100px on repeat of buttons and the row for the output is adjusted to a minimum of 120px and maximum of “auto” and 5 rows with 100px on repeat of buttons.



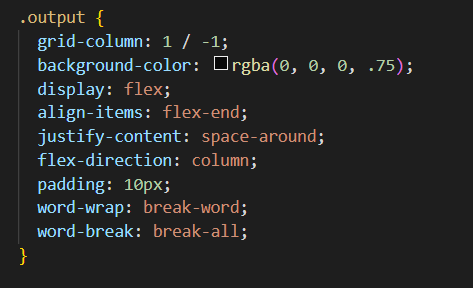
The button element within the calculator-grid class has been styled where the cursor will become a pointer as it hovers on the button. The font size is also adjusted to 2rem. Each button has a border of 1px of solid white. The outline property was also set to none to not display an outline. The background colour of the buttons was set to white with an opacity of 75%. As the button is hovered, the button background opacity will be changed to 90%.



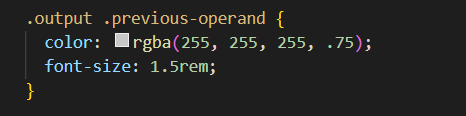
The span-two class only spans the grid column by 2.

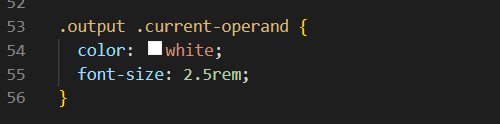


The output class spans from the first column to the last column. It has a background colour of black with an opacity of 75%. The output class was also adjusted so that the display becomes flexible. It has an alignment to the right-hand side of the column. The content of the class is also justified so that it will space around the column. Its direction of content is from up to down. It has a padding of 10px and words that extends outside the box will be wrap to the next line.



The output for the previous operand is set to the colour of white with an opacity of 75% and a font size of 1.5rem while the output for the current operand is set to the colour of white and a font size of 2.5rem.





**JavaScript**

In the JavaScript file, “use strict” was coded. Its purpose is to indicate that the code should be executed in "strict mode". With strict mode, you cannot, for example, use undeclared variables. After that, the class Calculator was created so that the program is structured, and it can reduce the amount of code that is present in the program.

In the class, the first function is the clear() function. This function will clear all outputs as well as the operation that is stored in the data. The delete() function will slice the current operand string by 1 character. The append(number) function adds the number parameter onto the current operand. If the current operand already has a “.” and the user clicks it, it will return nothing so that the “.” no longer appends after the first one. The chooseOperation(operation) function stores the clicked operation and stores it into the class data and transfer the current operand to the previous operand. If the current operand is empty, it will return nothing so that nothing will happen. If the previous operand already has an operation, it will run the eval() function. The eval() function will make the previous and current operand into a float and store it to the prev and current variables and will evaluate the equation based on what the class operation is. In a scenario where the prev and current variables are not a number, it will return nothing so that nothing happens. After evaluation, it will store the result into the current operand, set the previous operand string to empty and set the class operation to undefined so nothing is in the class operation data. The getDisplayNumber(number) function will store the number parameter in the variable stringNumber as a string. Then the stringNumber is stored into integerDigits as a float but only the number before the “.” appears. The stringNumber is then stored to decimalDigits, but only the digit that comes after the “.”. If the integerDigits is not a number, it will not display anything. Otherwise, it will display the digit while adding a comma to every 3 digits. If the decimalDigits is not null, it will display the number from before and display the digit in decimalDigits. The whole purpose of this function is to style the output so that the numbers look more neat and easier to read without adding comma after the “.”. The updateDisplay() function gets the current operand, runs the getDisplayNumber function with it as its variable and sets it to the class current operand. If the class operation data is not null, then it will set the previous operand while running the function getDisplayNumber and putting the operation in the data after the previous operand so that the user knows what operation is currently in the data.

The $ sign is a replacement of the whole line of code document.querySelector(). All the data that was created in the html file is then stored in the following variable, numberButtons, operationButtons, equalsButtons, deleteButton, allClearButton, previousOperandTextElement, and currentOperandTextElement. Then the Calculator class is pass on to the calculator object. For each button that is clicked in the numberButtons variable, it will run the appendNumber and updateDisplay function. For each operation clicked in the operationButtons variable, it will run the chooseOperation and updateDisplay function. When the button in equalsButton variable is clicked, it will run the eval and updateDisplay function. When the button in allClearButton variable is clicked, it will run the clear and updateDisplay function. When the button in deleteButton variable is clicked, it will run the delete and updateDisplay function.

