IT5039 Client-side Development
Practical Task 2: Interactive Web App (LO2, 4, 6)
Jeshua Hertzke - 20210843

For this task I decided to create a calculator app like the application in the Course Demo but more complex and in the format of a more standard calculator. The app files for the calculator are in the same folder this document is in.

Design and Accessibility:

The calculator features a display screen where the numbers and symbols are displayed along with the display of the previous input but in a smaller font and above the active display area. The calculator also has buttons for all the inputs including the numbers, symbols, a backspace button and a clear display button. Under the calculator is a button to close the calculator which is replaced with a thank you message. You can return to the calculator by clicking the start button in the thank you message display.

The design is based off a typical calculator making the app easily navigable and reducing cognitive load on the user. A simple design is used with effective use of white space, minimal colors and a color range that accommodates those who are visually impaired to enhance user accessibility.

The user can also input the numbers and symbols by using the keys on the keyboard, along with the enter key for the equals button, the delete key for backspace, and the c key for clearing the display. A modulus was included in the calculator which was an additional required functionality. Each button also has an aria-label to assist voice over. These features increase accessibility as the application can be used without a mouse and assists users who are visually impaired.

The app is easy to navigate as all the buttons are well identified, are a good size and the symbols are clear helping assist user predictability. All lines, font sizes and border radiuses are the same providing visual consistency. Input by the user is reversible, if the user inputs a wrong number or symbol, they can remove it clicking the backspace button on the calculator or pressing the delete key. HTML buttons are also unused instead of divs to allow the symbols to be disabled to reduce user errors. The calculator also momentarily displays a ERROR message if an invalid equation is input by the user and then clears the screen for the user to continue. All this helps the application be more user friendly.

Interactive Features:

Number event listener:

Each number button is declared as a constant using querySelectorAll() and selecting all numbers by their class. A “for of” loop is then used to cycle through the collection with a ‘click’ Event Listener with a function called addNum. The addNum function checks to see if there is a symbol in the display. If it does, it pushes the symbol in to the “calculation” array, adds the symbol to the history display using the showPreviousEq() function then clears the main displays before adding the number the user inputs.

Symbol event listener:

This event listeners runs similar to the numbers event listener but includes a for of loop in the function called by the event listener which cycles through the symbols and disables them after a symbol is input by the user. The

minus button is then enabled but again disabled if two consecutive minus buttons a used allowing the user to minus a negative number.

Decimal event listener:

The decimal button is declared as a constant and selected using querySelector() and its id. A click event listener is used to call the addDecimal() function which uses an if statement to see if the number in the display does not include a decimal. If it doesn't, a decimal is added, if it does, the button is disabled until a symbol is input allowing the user to add a decimal in the next number. Doing this adds a user constraint to reduce equation errors.

Backspace and clear event listeners:

The backspace and clear button are declared as constants using querySelector() and selecting them by their id. Click event listeners are used linking them to their related function which either deletes one single character at a time or clears and resets the entire screen and buttons.

Equals event listener and decimal rounding feature:

The equal button is declared as a constant and selected using querySelector() and its id. A click event listener is used to call the calcDisplay() arrow function. The function uses a try catch to catch any errors that may result from user input. The function tries to add the number in the display to the array, creates a variable that joins the array together and evaluates it using the eval() function. A function called roundToFive is then used on the variable to round the result to max five decimal points if needed. The result is then displayed on the screen.

If an error occurs, the catch sets the display to "ERROR" and then runs a setTimeout callback method which sets a time of 2 seconds then runs the clearDisplay() function resetting the screen.

Mouse down/up event listeners:

Each button is declared as a constant using querySelectorAll() and selects the buttons by their class. A for of loop is used to cycle through the collection and adds a mousedown and mouseup event listener with the function mouseStyle. The function toggles a class onto the button element which decreases the size of the button when clicked and then returns to the original size after clicked giving the impression that the button is being pushed.

Mouse hover event handlers:

One of the requirements was to change the styling when the button is hovered on through event handling. This could be done through CSS using the hover pseudo class, however, for the exercise of the assignment this was done through an event handler. A for of loop is used to cycle through the collection adds a onmouseover and onmouseoff event handler.The onmouseover event handler adds a class to the button hovered on which enlarges the button and the onmouseoff event handler removes the class from the button. This allows the users to see where their mouse cursor is more easily on the calculator as it hovers over the buttons.

Exit button/start button event listeners:

The two buttons are declared as constants using querySelector() and selecting them by their id. A click event listener is used with the same function called changeDisplay() which toggles the class of different blocks of code. The class sets the display to none hiding the block of code. This allows the display of the calculator to swap with the thank you message.

Keyboard events listener:

A key down event listener is used through the window which calls the function inside it that contains the event object inside the params. The function holds an if statement which adds the value of the key attribute of the object

to the display if the value is a number, otherwise the function uses a switch if a symbol, delete, or clear key is input by the user. This allows the user to control the calculator using the keyboard.

Testing:

I developed the app through VS Code and tested the app throughout the design process with the debugging tool and also used Chromes inspector to help test and debug the code. When an incorrect equation was input into the calculator the app would crash. To resolve this, try/catch was used to display an error message then clear the screen preventing the app from crashing and allowing the user to carry on. Each function and Event handling was vigorously tested and fine-tuned to work with minimal code. For example, arrow functions were used when possible.

When testing the following issues were found and resolved:

Key zero would not work. This was because the if statement in the event listener was a boolean and would recognise number zero as false. To resolve this, I added "|| e.key ==="0" to the statement to accept zero.

If you added a number that had zeros at the front, when clicking equals it would cause an error. To fix this I created a function that contains a while loop which removes the first index if it is a zero. I then added this function inside other functions that sent the number to the array.

Browser compatibility:

A reset sheet was used to ensure that the design was consistent on different browsers. I also added CSS code to support all browsers. I tested the application on the following browsers to check that they all ran well.

Chrome Browser - Version 99.0.4844.51 (Official Build) (x86_64) - Latest version:

As I had been using Chrome's Inspector as a reference in the design of the site, everything looked good and fitted well as expected.

Safari - Version 15.3 (17612.4.9.1.8) – Latest version:

Font size was larger making the styling not as nice.

Firefox - Version 98.0 - Latest version:

Same issues as Safari.

With some minor adjustments to fonts sizes and padding, the calculator worked well on all browsers with minimal differences in stylings.

Responsiveness:

I also tested the app through different screen widths. As responsive units were used in CSS everything worked well.

Conclusion:

Given the constraint of time I am pleased with the design and functionality of the application. I also enjoyed this task.

Jeshua Hertzke - 20210843