Name: Mingwei You (2837632), Rafael Carrillo (2746190)

Date: 12/4/2025

Developer's Guide

CalcWeb

Mingwei You (2837632), Rafael Carrillo (2746190)

1.0 Introduction

In this document, we will introduce CalcWeb, a calculator on a website, to guide developers what are the elements of this website and what can be changed.

2.0 Interface Function

In this section, we will introduce the elements that are related to the interface.

2.1 Color Theme

The "COLOR THEME" button is located on the top left corner, which can change the colors of the UI.

- Color are defined in stylesheet.css:

```css
:root {
    --header-color: #2FEED4;
    --background-color: #e0ffff;
    --dropbtn-color: #47AFE6;
    --dropbtn-hover-color: #9DDBFA;
    --sidebar-content-color: #66D2FF;
    --calc-btn-hover-color: #94E3E3;
    --text-color: #000000;
    --close-btn-color: #C41700;
    --close-btn-active-color: #731500;
    --transition-speed: 0.6s;
}
```

- It is triggered by the eventListener in interface.js, hooked with the sidebar trigger event.

2.2 History

The "History" button is located inside the "MORE" (next to the "COLOR THEME" button), allowing users to check and use their calculation history.

- History are defined in interface.js:
    - updateDisplay: After a calculation is performed, it will be recorded in history (error will be ignored).
    - deleteHistoryRecord: Use to delete a single calculation history

- clearHistory: Use to delete all calculation history.
- useHistoryRecord: Allow users to use the selected calculation record.

2.3 Tutorial & About

Both are located inside “MORE” too, allowing users to view the tutorial and about.

- Tutorial and About are both defined in interfaces, but the contents inside are located in CalcWeb.html (for better editing view):
  - Tutorial: Introduce users arithmetics and functions
  - About: About the authors and the website.

3.0 Calculator

The calculator display is where all entered values, operations, and results appear.

3.1 Calculator Display

The calculator display is where all entered values, operations, and results appear.

- It is implemented using a combination of:
  - provides the display container (…) and enables text content updates.
  - CSS – styles the display area, including sizing, alignment, colors, and responsiveness.
  - JavaScript – updates the displayed expression, processes user input, and evaluates calculations.

3.2 Basic Arithmetic

Basic arithmetic operations are performed using the main calculator keypad. When a button is pressed, it triggers JavaScript functions inside **Computation.js** that handle input, validation, and evaluation. The calculator uses the HTML element with the ID **display** with the **const display = document.getElementByID(“display”)**. **display.textContent** is a value that stores *the current expression as text* (numbers and operators). Every time the user presses a key, the script updates **display.textContent** to build the full expression via **appendToDisplay()**.

Before evaluating the expression, the calculator checks whether the input is valid. For example, the helper function: **const isOP = ch => [‘+’, ‘-’, ‘*’, ‘/’, ‘%’].includes(ch);. isOP** is used to determine whether a character is an operator. This ensures operators are only appended in places where they create valid expressions. If an invalid sequence is detected (such as two operators in a row or an incomplete expression), the calculator will display "Error" during

evaluation. The complete expression is then processed and evaluated through the calculator's evaluation pipeline, which handles normal arithmetic, exponents, functions, and fractions when needed.

- These buttons allow the user to enter and evaluate simple expressions.
  Supported basic functions:
    - Addition (+)
    - Subtraction (-)
    - Multiplication (*)
    - Division (/)
    - Open and close parenthesis (())
    - Decimal (.)
    - Percentage (%)
    - Equal (=)
        - When the user presses "=", the full expression in display.textContent is passed through the evaluation pipeline via **function calculate()**. If the expression is incomplete or invalid, the system displays "Error."

3.3 Advanced Arithmetic

Advanced arithmetic features are located inside the collapsible Advanced Function menu. In **Computation.js** , these functions required their own set of codes for it to work with numbers and basic operators from the calculator buttons. Sine (sin), Cosine (cos), Tangent (tan), Log (log), ln (ln), and Squareroot (√) are called by the method **appendToFunction(),** which makes it possible for these functions to display as *sin(30)* for instance. These functions particularly are passed down to **preprocessExpression()**, which convert the functions names into JavaScript Math calls. Thus, this will transform *2+sin(π/2)* to *2+Math.sin(Math.PI/2)* for instance. Pi (π), exponent (^), and Euler's number (e) are regulated to **appendToDisplay()** instead.

- These buttons allow evaluation of more complex expressions that go beyond standard arithmetic. Supported advanced functions:
    - Fraction (a/b)
        - When a/b is clicked, the html button calls **showFractionEditor(),** the same method used in **Computation.js.** It will create a small fraction editor overlay instead of typing directing into the main display. Within this box,

it will process the digits being typed in its numInput.value and denInput.value (used for numerator and denominator respectively). **fractionActive** is set to true so **appendToDisplay()** knows it should send digits into the fraction inputs instead of the main display. When the fraction is entered, it will appear in the display between parenthesis, which is handled by the method **tryFractionExpression()**, which is used to determine if the fraction itself would lead to a fraction or decimal.

- Sine (sin)
- Cosine (cos)
- Tangent (tan)
- Log (log)
- ln (ln)
- Squareroot (√)
- Factorial (!)
  - The factorial button will call **appendFactorial()**, which determines if a factorial sign is used correctly or not. It will then be passed to **preprocessExpression()**.
- Pi (π)
- exponent (^)
- Euler's number (e)

4.0 Unit Conversion Calculator

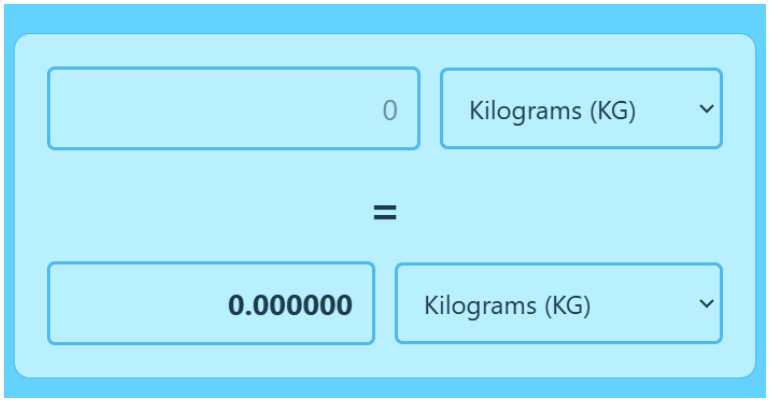
In this section, we introduce the elements that are related to the unit conversion calculator.

4.1 Open

The button of "UNIT CONVERSION" is located at the top left corner in the header, next to the "COLOR THEME" button. By clicking it, a sidebar will slide out from the left and show the calculator. This button is separated in its own class and id in calcweb.html: *unit-conversion-sidebar*.

4.2 Conversion

In the first container, show as below:



The first box allows users to input any numbers they want, and choose the unit on the right side (for both input and output units). Without having to click anything, it will automatically convert and show the results.

- User input, result output, auto convert, and units are all located in calcweb.html and unitConversion.js.
    - User Input: located in both.
        - In calcweb.html, inside the class *conversion-from*, it is an input box, type is *text*, class *conversion-input*, id *conversion-value*.
        - In unitConversion.js, inside the *initialize()*, the *valueInput* is reading from the id *conversion-value* in calcweb.html, to perform the conversion.
    - Result Output: located in both.
        - In calcweb.html, inside the *class conversion-to*, it is an output box, class and id both are *conversion-result*.
        - In unitConversion.js, inside the *initialize()*, the *resultDisplay* is reading from the id *conversion-result* in calcweb.html, to display the result in the box.
    - Auto Convert: in *bindEvent()*, [*this.valueInput.addEventListener('input', () => this.performConversion())*] will detect the input in *valueInput*, if valid, it will auto convert to the result.
    - Unit: located in both.
        - In calcweb.html, both classes are *conversion-select* for the input and output, but with different id: *conversion-from-unit* (for input) and

conversion-to-unit (for output), both are in the type of *option* (only chose one at a time).

- In unitConversion.js, inside the initialize(), the input unit is *fromUnitSelect* read from id *conversion-from-unit*, and the output unit is *toUnitSelect* read from id *conversion-to-unit*.

4.3 Units

In unitConversion.js, each different unit has its own type and factors:

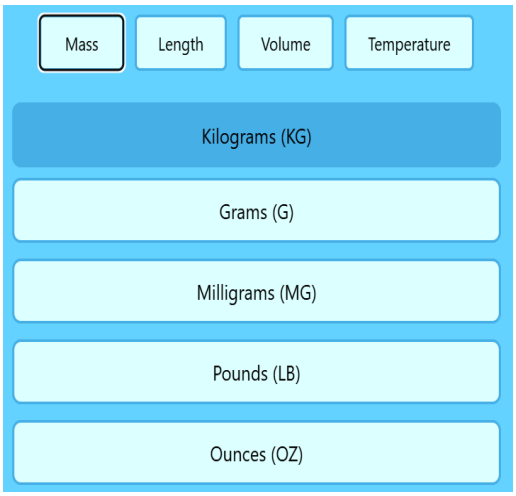
```
//Mass
'kg': {type: 'mass', factor: 1, display: 'KG'},
'g': {type: 'mass', factor: 0.001, display: 'G'},
'mg': {type: 'mass', factor: 0.000001, display: 'MG'},
'lb': {type: 'mass', factor: 0.453592, display: 'LB'},
'oz': {type: 'mass', factor: 0.0283495, display: 'OZ'},
```

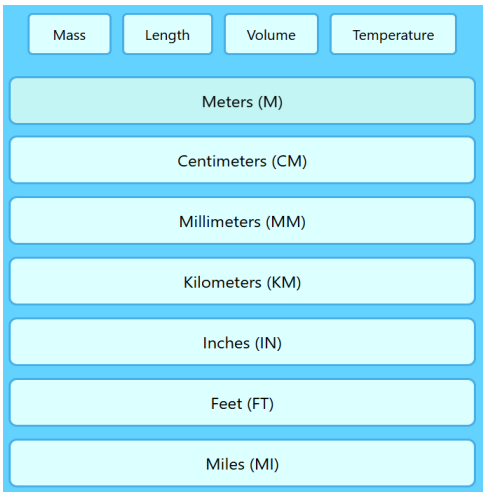
```
//Length
'm': {type: 'length', factor: 1, display: 'M'},
'cm': {type: 'length', factor: 0.01, display: 'CM'},
'mm': {type: 'length', factor: 0.001, display: 'MM'},
'km': {type: 'length', factor: 1000, display: 'KM'},
'in': {type: 'length', factor: 0.0254, display: 'IN'},
'ft': {type: 'length', factor: 0.3048, display: 'FT'},
'mi': {type: 'length', factor: 1609.34, display: 'MI'},
```

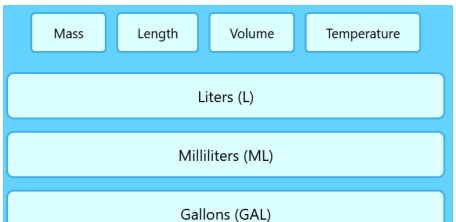
```
//Volume
'L': {type: 'volume', factor: 1, display: 'L'},
'mL': {type: 'volume', factor: 0.001, display: 'ML'},
'gal': {type: 'volume', factor: 3.78541, display: 'GAL'},
```

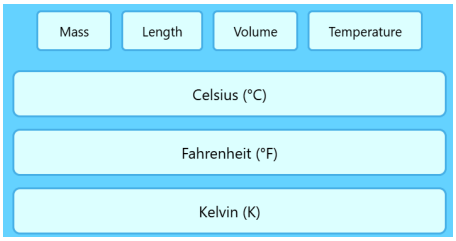
```
//Temperature
'C': {type: 'temperature', display: '°C'},
'F': {type: 'temperature', display: '°F'},
'K': {type: 'temperature', display: 'K'}
```

- When the users change the type of units, if it does not match the input/output unit, *updateCurrentCategory()* will detect and change the input/output unit to match the selected unit.










```
this.unitCategories = {
    'mass': ['kg', 'g', 'mg', 'lb', 'oz'],
    'length': ['m', 'cm', 'mm', 'km', 'in', 'ft', 'mi'],
    'volume': ['L', 'mL', 'gal'],
    'temperature': ['C', 'F', 'K']
};
this.currentCategory = 'mass';
```

- In the middle section, users are provided with four different type units: mass, length, volume, and temperature. Each type unit is stored in *unitCategories*, and the initial unit type is mass.

- On the bottom, this allows users to copy the converted result.
    - Located in *formatResultDisplay()*, of *formatDisplay*.