# Project Calculator Documentation

## 1. Description

The ‘Project Calculator’ is a simulation of a traditional calculator. It is developed as a part of The Odin Project's curriculum, focusing on the foundational concepts of web development. The application provides a simple yet interesting interface for performing basic arithmetic operations.

Constructed with HTML, CSS, and JavaScript, this project solidifies my understanding of web technologies. HTML portrays the calculator's buttons and display, CSS styles the application for an engaging user experience, and JavaScript implements the calculator's logic and response to user inputs.

The interface is intentionally minimalist, simulating the look and of a physical calculator, with clear labels for each function, including addition, subtraction, multiplication, division, and modulus operation, as well as special functions like clear and sign toggle.

## 2. Objectives

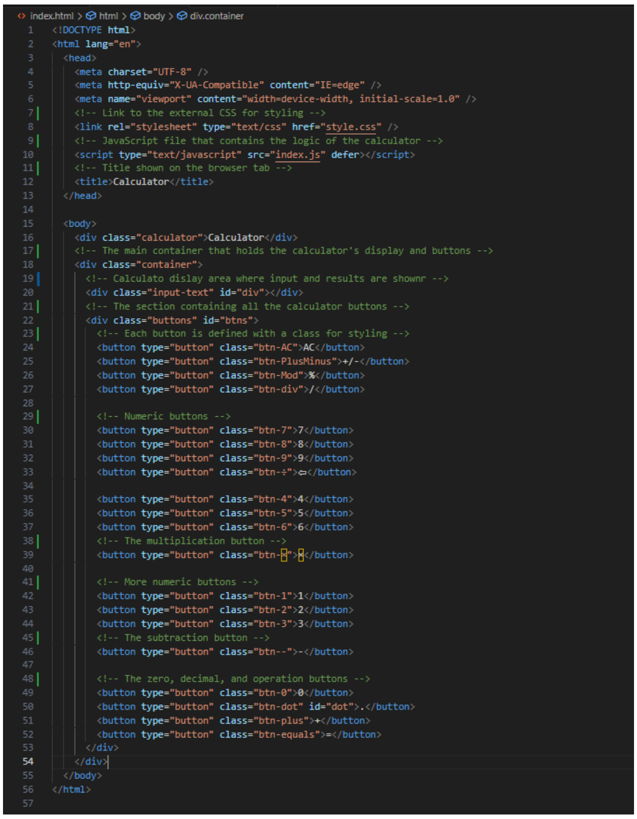
Key objectives include:

- Illustrating the ability to structure web pages with HTML, ensuring that all necessary calculator functions are accessible.
- Applying CSS skills to design a user interface that is visually appealing and also provides a seamless and intuitive experience.
- Demonstrating knowledge in JavaScript by developing a fully functional calculator that can handle a variety of arithmetic operations.
- Showcasing the ability to handle user inputs, perform calculations, and update the display without page reloads, offering immediate feedback to the user.

## 3. Design and Code

The project is structured into three primary files, each serving a distinct purpose in the application's functionality and design: ‘index.html’ , ‘index.js’ and ‘style.css’.

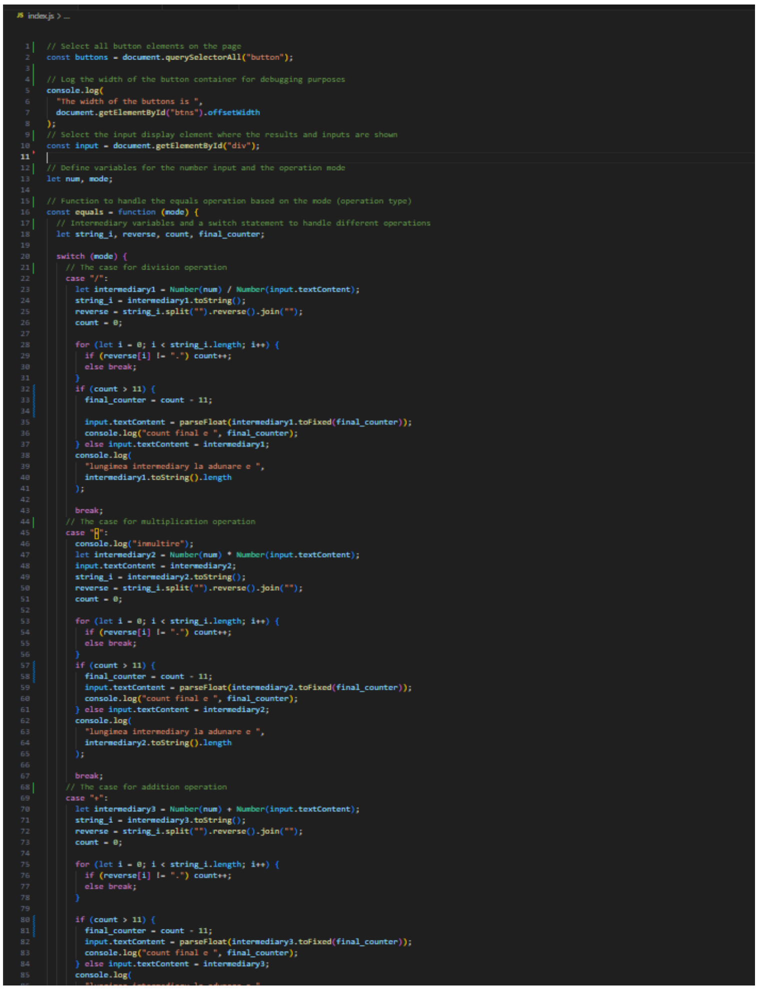
The ‘index.html’ file sets up the structure of the web calculator. It includes all the necessary HTML elements that form the interface of the calculator. This is how it looks like:



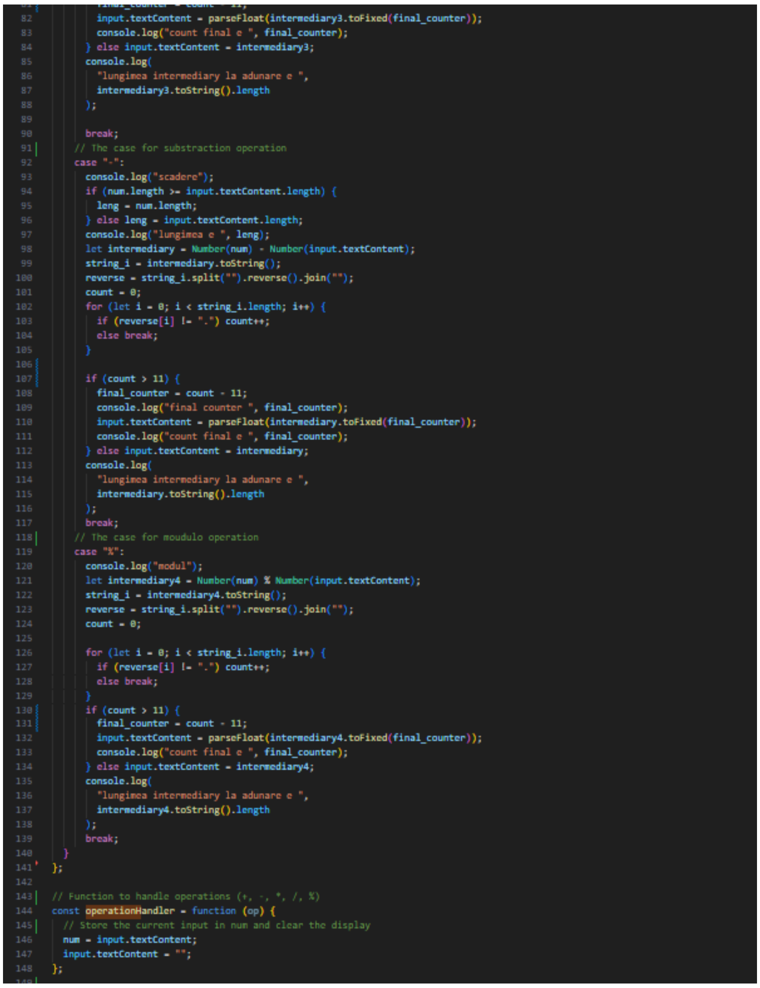
```
index.html > html > body > div.container
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <!-- Link to the external CSS for styling -->
    <link rel="stylesheet" type="text/css" href="style.css" />
    <!-- JavaScript file that contains the logic of the calculator -->
    <script type="text/javascript" src="index.js" defer></script>
    <!-- Title shown on the browser tab -->
    <title>Calculator</title>
  </head>

  <body>
    <div class="calculator">Calculator</div>
    <!-- The main container that holds the calculator's display and buttons -->
    <div class="container">
      <!-- Calculato dislay area where input and results are shownr -->
      <div class="input-text" id="div"></div>
      <!-- The section containing all the calculator buttons -->
      <div class="buttons" id="btns">
        <!-- Each button is defined with a class for styling -->
        <button type="button" class="btn-AC">AC</button>
        <button type="button" class="btn-PlusMinus">+/-</button>
        <button type="button" class="btn-Mod">%</button>
        <button type="button" class="btn-div">/</button>

        <!-- Numeric buttons -->
        <button type="button" class="btn-7">7</button>
        <button type="button" class="btn-8">8</button>
        <button type="button" class="btn-9">9</button>
        <button type="button" class="btn-÷">⇦</button>

        <button type="button" class="btn-4">4</button>
        <button type="button" class="btn-5">5</button>
        <button type="button" class="btn-6">6</button>
        <!-- The multiplication button -->
        <button type="button" class="btn-×">×</button>

        <!-- More numeric buttons -->
        <button type="button" class="btn-1">1</button>
        <button type="button" class="btn-2">2</button>
        <button type="button" class="btn-3">3</button>
        <!-- The subtraction button -->
        <button type="button" class="btn--">-</button>

        <!-- The zero, decimal, and operation buttons -->
        <button type="button" class="btn-0">0</button>
        <button type="button" class="btn-dot" id="dot">.</button>
        <button type="button" class="btn-plus">+</button>
        <button type="button" class="btn-equals">=</button>
      </div>
    </div>
  </body>
</html>
```

It sets up a calculator on the web page with buttons for all digits (0-9), operations (addition, subtraction, multiplication, division, and modulus), and other functions like clear (AC), toggle sign (+/-), and equals (=). The defer attribute in the script tag ensures that the JavaScript file is executed after the HTML document has been parsed. This ensures that all HTML elements are loaded before any JavaScript attempts to interact with them.

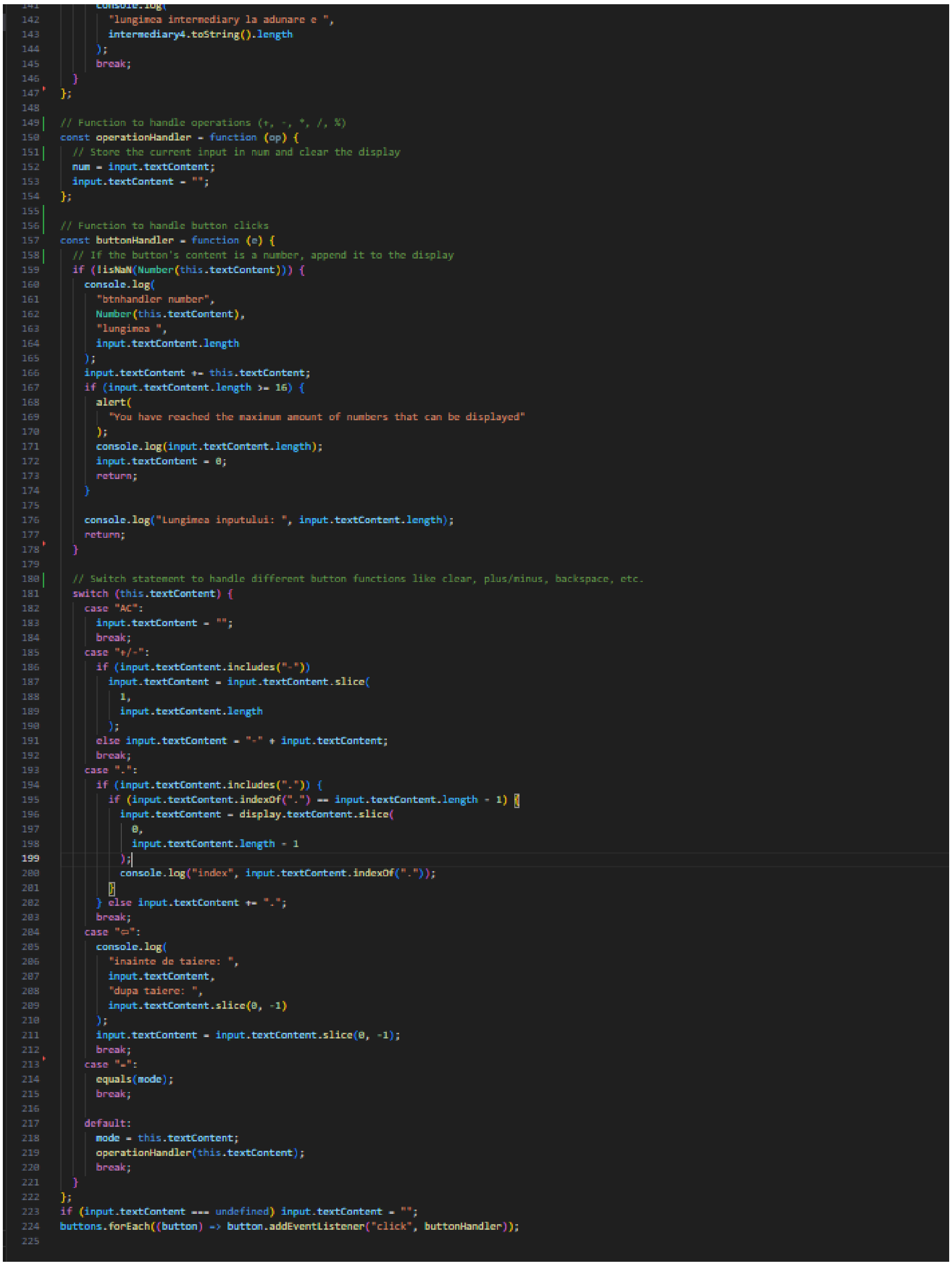
Next, we will take a look at the the 'index.js' file to understand how it attaches functionality to these elements. This is how the code looks like:



```
index.js > ...

// Select all button elements on the page
const buttons = document.querySelectorAll("button");

// Log the width of the button container for debugging purposes
console.log(
  "The width of the buttons is ",
  document.getElementById("btns").offsetWidth
);
// Select the input display element where the results and inputs are shown
const input = document.getElementById("div");

// Define variables for the number input and the operation mode
let num, mode;

// Function to handle the equals operation based on the mode (operation type)
const equals = function (mode) {
  // Intermediary variables and a switch statement to handle different operations
  let string_i, reverse, count, final_counter;

  switch (mode) {
    // The case for division operation
    case "/":
      let intermediary1 = Number(num) / Number(input.textContent);
      string_i = intermediary1.toString();
      reverse = string_i.split("").reverse().join("");
      count = 0;

      for (let i = 0; i < string_i.length; i++) {
        if (reverse[i] != ".") count++;
        else break;
      }
      if (count > 11) {
        final_counter = count - 11;

        input.textContent = parseFloat(intermediary1.toFixed(final_counter));
        console.log("count final e ", final_counter);
      } else input.textContent = intermediary1;
      console.log(
        "lungimea intermediary la adunare e ",
        intermediary1.toString().length
      );

      break;
    // The case for multiplication operation
    case "×":
      console.log("inmultire");
      let intermediary2 = Number(num) * Number(input.textContent);
      input.textContent = intermediary2;
      string_i = intermediary2.toString();
      reverse = string_i.split("").reverse().join("");
      count = 0;

      for (let i = 0; i < string_i.length; i++) {
        if (reverse[i] != ".") count++;
        else break;
      }
      if (count > 11) {
        final_counter = count - 11;
        input.textContent = parseFloat(intermediary2.toFixed(final_counter));
        console.log("count final e ", final_counter);
      } else input.textContent = intermediary2;
      console.log(
        "lungimea intermediary la adunare e ",
        intermediary2.toString().length
      );

      break;
    // The case for addition operation
    case "+":
      let intermediary3 = Number(num) + Number(input.textContent);
      string_i = intermediary3.toString();
      reverse = string_i.split("").reverse().join("");
      count = 0;

      for (let i = 0; i < string_i.length; i++) {
        if (reverse[i] != ".") count++;
        else break;
      }

      if (count > 11) {
        final_counter = count - 11;
        input.textContent = parseFloat(intermediary3.toFixed(final_counter));
        console.log("count final e ", final_counter);
      } else input.textContent = intermediary3;
      console.log(
```



```
        input.textContent = parseFloat(intermediary3.toFixed(final_counter));
        console.log("count final e ", final_counter);
      } else input.textContent = intermediary3;
      console.log(
        "lungimea intermediary la adunare e ",
        intermediary3.toString().length
      );

      break;
    // The case for substraction operation
    case "-":
      console.log("scadere");
      if (num.length >= input.textContent.length) {
        leng = num.length;
      } else leng = input.textContent.length;
      console.log("lungimea e ", leng);
      let intermediary = Number(num) - Number(input.textContent);
      string_i = intermediary.toString();
      reverse = string_i.split("").reverse().join("");
      count = 0;
      for (let i = 0; i < string_i.length; i++) {
        if (reverse[i] != ".") count++;
        else break;
      }

      if (count > 11) {
        final_counter = count - 11;
        console.log("final counter ", final_counter);
        input.textContent = parseFloat(intermediary.toFixed(final_counter));
        console.log("count final e ", final_counter);
      } else input.textContent = intermediary;
      console.log(
        "lungimea intermediary la adunare e ",
        intermediary.toString().length
      );
      break;
    // The case for moudulo operation
    case "%":
      console.log("modul");
      let intermediary4 = Number(num) % Number(input.textContent);
      string_i = intermediary4.toString();
      reverse = string_i.split("").reverse().join("");
      count = 0;

      for (let i = 0; i < string_i.length; i++) {
        if (reverse[i] != ".") count++;
        else break;
      }
      if (count > 11) {
        final_counter = count - 11;
        input.textContent = parseFloat(intermediary4.toFixed(final_counter));
        console.log("count final e ", final_counter);
      } else input.textContent = intermediary4;
      console.log(
        "lungimea intermediary la adunare e ",
        intermediary4.toString().length
      );
      break;
  }
};

// Function to handle operations (+, -, *, /, %)
const operationHandler = function (op) {
  // Store the current input in num and clear the display
  num = input.textContent;
  input.textContent = "";
};
```



```
      console.log(
        "lungimea intermediary la adunare e ",
        intermediary4.toString().length
      );
      break;
  }
};

// Function to handle operations (+, -, *, /, %)
const operationHandler = function (op) {
  // Store the current input in num and clear the display
  num = input.textContent;
  input.textContent = "";
};

// Function to handle button clicks
const buttonHandler = function (e) {
  // If the button's content is a number, append it to the display
  if (!isNaN(Number(this.textContent))) {
    console.log(
      "btnhandler number",
      Number(this.textContent),
      "lungimea ",
      input.textContent.length
    );
    input.textContent += this.textContent;
    if (input.textContent.length >= 16) {
      alert(
        "You have reached the maximum amount of numbers that can be displayed"
      );
      console.log(input.textContent.length);
      input.textContent = 0;
      return;
    }

    console.log("Lungimea inputului: ", input.textContent.length);
    return;
  }

  // Switch statement to handle different button functions like clear, plus/minus, backspace, etc.
  switch (this.textContent) {
    case "AC":
      input.textContent = "";
      break;
    case "+/-":
      if (input.textContent.includes("-"))
        input.textContent = input.textContent.slice(
          1,
          input.textContent.length
        );
      else input.textContent = "-" + input.textContent;
      break;
    case ".":
      if (input.textContent.includes(".")) {
        if (input.textContent.indexOf(".") == input.textContent.length - 1) {
          input.textContent = display.textContent.slice(
            0,
            input.textContent.length - 1
          );
          console.log("index", input.textContent.indexOf("."));
        }
      } else input.textContent += ".";
      break;
    case "⇐":
      console.log(
        "inainte de taiere: ",
        input.textContent,
        "dupa taiere: ",
        input.textContent.slice(0, -1)
      );
      input.textContent = input.textContent.slice(0, -1);
      break;
    case "=":
      equals(mode);
      break;

    default:
      mode = this.textContent;
      operationHandler(this.textContent);
      break;
  }
};
if (input.textContent === undefined) input.textContent = "";
buttons.forEach((button) => button.addEventListener("click", buttonHandler));
```

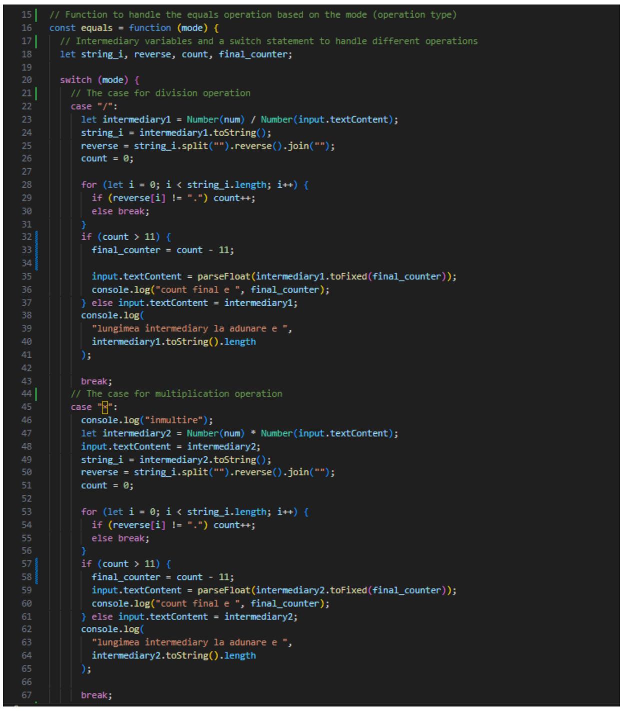
Let's take a look at each function within the ‚index.js' file.

*equals(mode)*

This is an important function that handles the calculation when the equals button is pressed. It takes the operation mode (like "+", "-", "*", "/", "%") as an argument and performs the corresponding mathematical operation using the previously stored number (num) and the current number displayed (input.textContent).

It begins by converting the result of the operation to a string to analyze its length and decide how to format it. It reverses the string to count how many digits follow the decimal point, using a ‚for' loop.

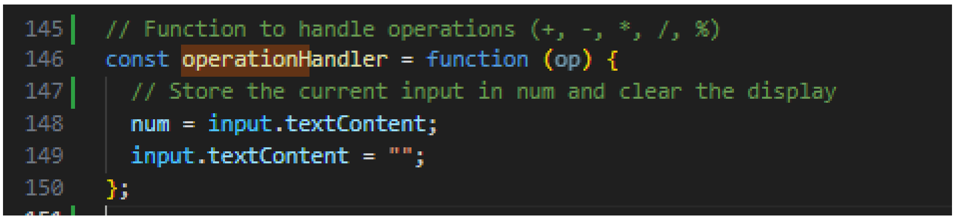
Depending on the length of the result and the number of decimal places, it either displays the result directly or formats it to a fixed number of decimal places using ‚toFixed'.



```javascript
// Function to handle the equals operation based on the mode (operation type)
const equals = function (mode) {
  // Intermediary variables and a switch statement to handle different operations
  let string_i, reverse, count, final_counter;

  switch (mode) {
    // The case for division operation
    case "/":
      let intermediary1 = Number(num) / Number(input.textContent);
      string_i = intermediary1.toString();
      reverse = string_i.split("").reverse().join("");
      count = 0;

      for (let i = 0; i < string_i.length; i++) {
        if (reverse[i] != ".") count++;
        else break;
      }
      if (count > 11) {
        final_counter = count - 11;

        input.textContent = parseFloat(intermediary1.toFixed(final_counter));
        console.log("count final e ", final_counter);
      } else input.textContent = intermediary1;
      console.log(
        "lungimea intermediary la adunare e ",
        intermediary1.toString().length
      );

      break;
    // The case for multiplication operation
    case "×":
      console.log("inmultire");
      let intermediary2 = Number(num) * Number(input.textContent);
      input.textContent = intermediary2;
      string_i = intermediary2.toString();
      reverse = string_i.split("").reverse().join("");
      count = 0;

      for (let i = 0; i < string_i.length; i++) {
        if (reverse[i] != ".") count++;
        else break;
      }
      if (count > 11) {
        final_counter = count - 11;
        input.textContent = parseFloat(intermediary2.toFixed(final_counter));
        console.log("count final e ", final_counter);
      } else input.textContent = intermediary2;
      console.log(
        "lungimea intermediary la adunare e ",
        intermediary2.toString().length
      );

      break;
```

*operationHandler(op)*

This function captures the current value displayed on the calculator when an operation button is clicked. It stores this value in the ‚num’ variable for later use, then clears the display in preparation for the next number to be entered by the user.

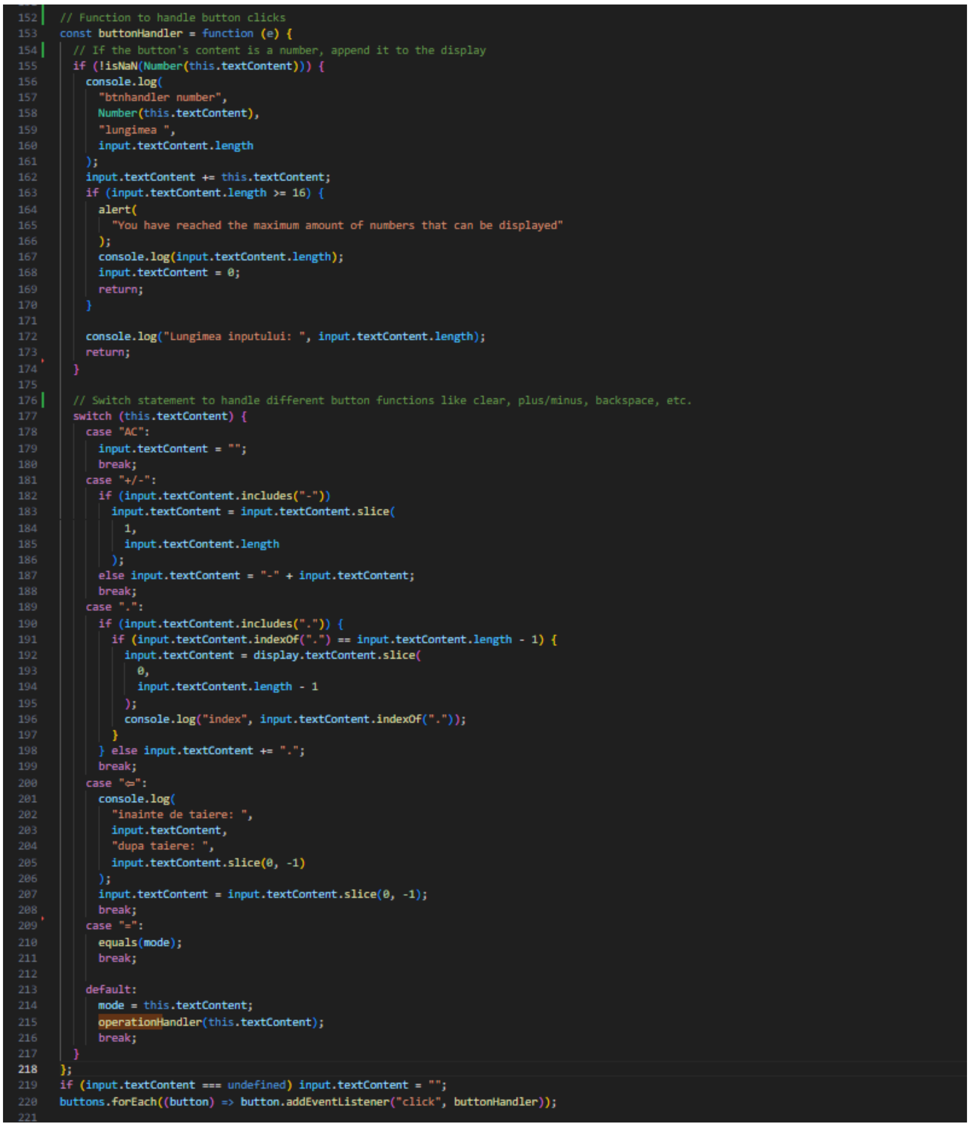


```javascript
// Function to handle operations (+, -, *, /, %)
const operationHandler = function (op) {
  // Store the current input in num and clear the display
  num = input.textContent;
  input.textContent = "";
};
```

*buttonHandler(e)*

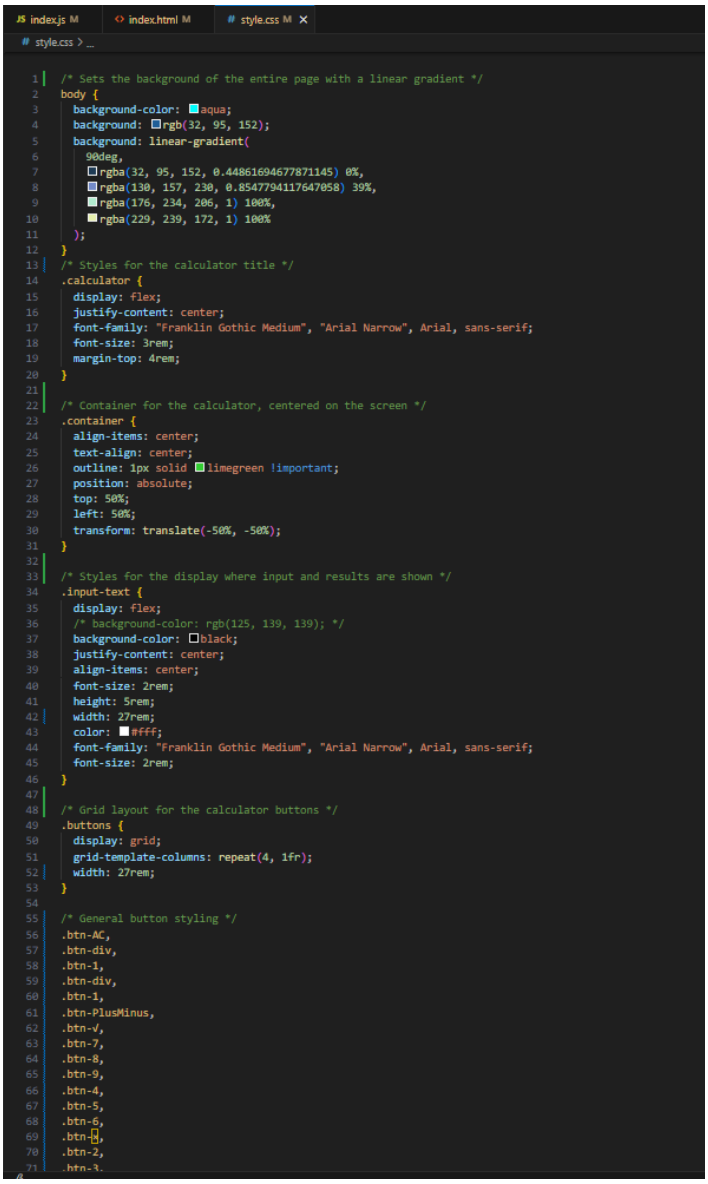
This function is attached to each button's click event and handles the logic for when a button is pressed. It checks the text content of the button to determine what action to take:

- If the button content is a number, it appends this number to the display.
- For function buttons like "AC" (All Clear), it sets ‚input.textContent’ to an empty string, clearing the display.
- For ‚+/-, , it toggles the sign of the displayed number by checking if it includes a minus sign and adding or removing it accordingly.
- For ‚.’ , it ensures only one decimal point is present in the number.
- For ‚⇦’ , it removes the last character from the display, which is a basic backspace function.
- For ‚=’, it calls the ‚equals’ function to perform the calculation based on the stored mode.
- For any other content, it assumes an operation is being set (like "+", "-", "*", "/"), stores the current display value in ‚num’ , clears the display, and sets the mode to the operation symbol.

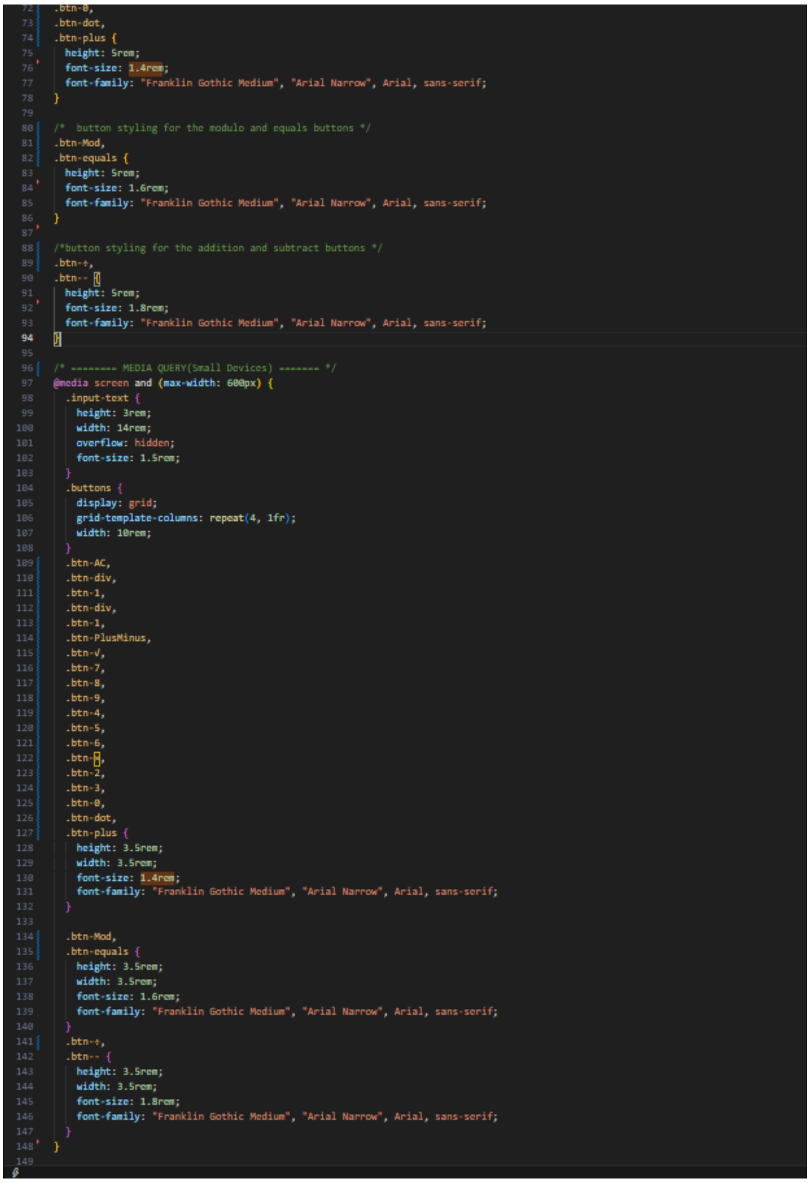


```javascript
// Function to handle button clicks
const buttonHandler = function (e) {
  // If the button's content is a number, append it to the display
  if (!isNaN(Number(this.textContent))) {
    console.log(
      "btnhandler number",
      Number(this.textContent),
      "lungimea ",
      input.textContent.length
    );
    input.textContent += this.textContent;
    if (input.textContent.length >= 16) {
      alert(
        "You have reached the maximum amount of numbers that can be displayed"
      );
      console.log(input.textContent.length);
      input.textContent = 0;
      return;
    }

    console.log("Lungimea inputului: ", input.textContent.length);
    return;
  }

  // Switch statement to handle different button functions like clear, plus/minus, backspace, etc.
  switch (this.textContent) {
    case "AC":
      input.textContent = "";
      break;
    case "+/-":
      if (input.textContent.includes("-"))
        input.textContent = input.textContent.slice(
          1,
          input.textContent.length
        );
      else input.textContent = "-" + input.textContent;
      break;
    case ".":
      if (input.textContent.includes(".")) {
        if (input.textContent.indexOf(".") == input.textContent.length - 1) {
          input.textContent = display.textContent.slice(
            0,
            input.textContent.length - 1
          );
          console.log("index", input.textContent.indexOf("."));
        }
      } else input.textContent += ".";
      break;
    case "⇐":
      console.log(
        "inainte de taiere: ",
        input.textContent,
        "dupa taiere: ",
        input.textContent.slice(0, -1)
      );
      input.textContent = input.textContent.slice(0, -1);
      break;
    case "=":
      equals(mode);
      break;

    default:
      mode = this.textContent;
      operationHandler(this.textContent);
      break;
  }
};
if (input.textContent === undefined) input.textContent = "";
buttons.forEach((button) => button.addEventListener("click", buttonHandler));
```

Lastly, we have the ‚style.css’ which is responsible for styling the visual presentation of the calculator. This is how the code looks like:



```css
/* Sets the background of the entire page with a linear gradient */
body {
  background-color: aqua;
  background: rgb(32, 95, 152);
  background: linear-gradient(
    90deg,
    rgba(32, 95, 152, 0.44861694677871145) 0%,
    rgba(130, 157, 230, 0.8547794117647058) 39%,
    rgba(176, 234, 206, 1) 100%,
    rgba(229, 239, 172, 1) 100%
  );
}
/* Styles for the calculator title */
.calculator {
  display: flex;
  justify-content: center;
  font-family: "Franklin Gothic Medium", "Arial Narrow", Arial, sans-serif;
  font-size: 3rem;
  margin-top: 4rem;
}

/* Container for the calculator, centered on the screen */
.container {
  align-items: center;
  text-align: center;
  outline: 1px solid limegreen !important;
  position: absolute;
  top: 50%;
  left: 50%;
  transform: translate(-50%, -50%);
}

/* Styles for the display where input and results are shown */
.input-text {
  display: flex;
  /* background-color: rgb(125, 139, 139); */
  background-color: black;
  justify-content: center;
  align-items: center;
  font-size: 2rem;
  height: 5rem;
  width: 27rem;
  color: #fff;
  font-family: "Franklin Gothic Medium", "Arial Narrow", Arial, sans-serif;
  font-size: 2rem;
}

/* Grid layout for the calculator buttons */
.buttons {
  display: grid;
  grid-template-columns: repeat(4, 1fr);
  width: 27rem;
}

/* General button styling */
.btn-AC,
.btn-div,
.btn-1,
.btn-div,
.btn-1,
.btn-PlusMinus,
.btn-√,
.btn-7,
.btn-8,
.btn-9,
.btn-4,
.btn-5,
.btn-6,
.btn-×,
.btn-2,
.btn-3,
```

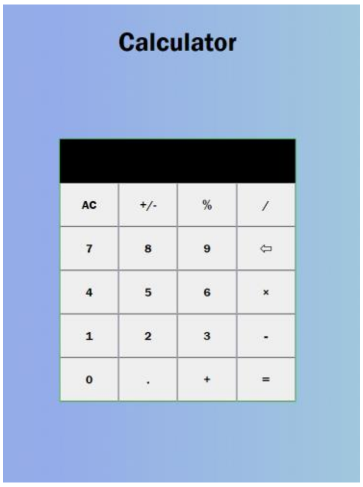


```css
.btn-0,
.btn-dot,
.btn-plus {
  height: 5rem;
  font-size: 1.4rem;
  font-family: "Franklin Gothic Medium", "Arial Narrow", Arial, sans-serif;
}

/*  button styling for the modulo and equals buttons */
.btn-Mod,
.btn-equals {
  height: 5rem;
  font-size: 1.6rem;
  font-family: "Franklin Gothic Medium", "Arial Narrow", Arial, sans-serif;
}

/*button styling for the addition and subtract buttons */
.btn-+,
.btn-- {
  height: 5rem;
  font-size: 1.8rem;
  font-family: "Franklin Gothic Medium", "Arial Narrow", Arial, sans-serif;
}

/* ======== MEDIA QUERY(Small Devices) ======= */
@media screen and (max-width: 600px) {
  .input-text {
    height: 3rem;
    width: 14rem;
    overflow: hidden;
    font-size: 1.5rem;
  }
  .buttons {
    display: grid;
    grid-template-columns: repeat(4, 1fr);
    width: 10rem;
  }
  .btn-AC,
  .btn-div,
  .btn-1,
  .btn-div,
  .btn-1,
  .btn-PlusMinus,
  .btn-√,
  .btn-7,
  .btn-8,
  .btn-9,
  .btn-4,
  .btn-5,
  .btn-6,
  .btn-×,
  .btn-2,
  .btn-3,
  .btn-0,
  .btn-dot,
  .btn-plus {
    height: 3.5rem;
    width: 3.5rem;
    font-size: 1.4rem;
    font-family: "Franklin Gothic Medium", "Arial Narrow", Arial, sans-serif;
  }

  .btn-Mod,
  .btn-equals {
    height: 3.5rem;
    width: 3.5rem;
    font-size: 1.6rem;
    font-family: "Franklin Gothic Medium", "Arial Narrow", Arial, sans-serif;
  }
  .btn-+,
  .btn-- {
    height: 3.5rem;
    width: 3.5rem;
    font-size: 1.8rem;
    font-family: "Franklin Gothic Medium", "Arial Narrow", Arial, sans-serif;
  }
}
```

The CSS file includes:

- Background styling for the entire page.
- Styling for the calculator title to center it horizontally.
- The container styling to center the calculator in the middle of the page.
- Styling for the display of the calculator where inputs and results are shown.
- Grid layout and button styling to give a uniform look to the calculator buttons.
- Special styling for certain operation buttons to increase their font size.
- A media query for small screen devices to adjust the button sizes and display width for better responsiveness on devices like phones.

The final product will look like below:


Calculator
AC
+/-
%
/
7
8
9
⇦
4
5
6
×
1
2
3
-
0
.
+
=