```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Interactive Calculator</title>
    <link rel="stylesheet" href="style.css">
    <link
href="https://fonts.googleapis.com/css2?family=Poppins:wght@400;600&family=Roboto
:wght@400;500&display=swap" rel="stylesheet">
</head>
<body>
    <div class="grid-container">
        <header class="main-header">
            <h1>Day 8 - Interactive Calculator</h1>
        </header>

        <div class="main-content">
            <div class="container">
                <h2>Calculator</h2>

                <!-- Output box -->
                <div class="output-box">
                    <div class="history-box" id="historyNum"></div>
                    <div id="result" class="result-box">0</div>
                </div>

                <!-- Calculator and buttons -->
                <div class="calculator">
                    <div class="button-container">
                        <!-- Calculator Buttons -->
                        <button class="operator" data-op="%">%</button>
                        <button class="operator" data-op="^">^</button>
                        <button class="operator" data-op="sqrt">√</button>
                        <button class="operator" data-op="/">/</button>

                        <button class="number" data-num="7">7</button>
                        <button class="number" data-num="8">8</button>
                        <button class="number" data-num="9">9</button>
                        <button class="operator" data-op="*">×</button>

                        <button class="number" data-num="4">4</button>
                        <button class="number" data-num="5">5</button>
                        <button class="number" data-num="6">6</button>
                        <button class="operator" data-op="-">-</button>
                        <button class="number" data-num="1">1</button>
                        <button class="number" data-num="2">2</button>
                        <button class="number" data-num="3">3</button>
                        <button class="operator" data-op="+">+</button>

                        <button class="clear" data-action="back">⍰</button>
                        <button class="number" data-num="0">0</button>
                        <button class="number" data-num=".">.</button>
                        <button class="equals" data-action="equal">=</button>
                    </div>
                </div>
            </div>
        </div>

        <footer class="main-footer">
            <p>&copy; 2025 John Rich Nicolas. All rights reserved.</p>
        </footer>
    </div>

    <script src="script.js"></script>
</body>
</html>
```

CSS

```css
body {
    font-family: 'Roboto', sans-serif;
    background-color: #1E1E1E;
    color: #F0F0F0;
    padding: 20px;
    margin: 0;
}

.grid-container {
    display: grid;
    grid-template-areas:
        "header"
        "main"
        "footer";
    grid-template-columns: 1fr;
    gap: 20px;
    max-width: 1200px;
    margin: auto;
    padding: 20px;
    box-sizing: border-box;
}

.main-header {
    grid-area: header;
    display: flex;
    justify-content: center;
    align-items: center;
    background-color: #2A2A2A;
    color: #FFFFFF;
    padding: 30px;
    border-radius: 1.5rem;
    border-bottom: 4px solid #444444;
    box-shadow: 0 6px 16px rgba(0, 0, 0, 0.6);
}

.main-header h1 {
    font-family: 'Poppins', sans-serif;
    font-size: 2.5rem;
    letter-spacing: 1px;
    text-transform: uppercase;
    margin: 0;
}

.main-content {
    grid-area: main;
    display: flex;
    flex-direction: column;
    gap: 20px;
    align-items: center;
}

.container {
    width: 100%;
    max-width: 500px;
    background-color: #1E1E1E;
    border: 2px solid #444444;
    border-radius: 1rem;
    padding: 20px;
    box-shadow: 0 4px 12px rgba(0, 0, 0, 0.4);
    box-sizing: border-box;
}

.output-box {
```

```css
    background-color: #2A2A2A;
    color: #FFFFFF;
    border: 1px solid #444444;
    border-radius: 10px;
    padding: 10px;
    margin-bottom: 20px;
    box-sizing: border-box;
    min-height: 100px;
    display: flex;
    flex-direction: column;
    justify-content: space-between;
}

.history-box {
    font-size: 0.9rem;
    color: #CCCCCC;
    text-align: right;
    overflow: hidden;
    white-space: nowrap;
    text-overflow: ellipsis;
}

.result-box {
    font-size: 1.8rem;
    font-weight: bold;
    text-align: right;
    color: #FFFFFF;
}

.calculator {
    display: flex;
    flex-direction: column;
    width: 100%;
    max-width: 500px;
    border: 2px solid #444444;
    border-radius: 10px;
    box-sizing: border-box;
    overflow: hidden;
}

.button-container {
    display: grid;
    grid-template-columns: repeat(4, 1fr);
    gap: 10px;
    padding: 15px;
    background-color: #1E1E1E;
    box-sizing: border-box;
}

button {
    padding: 15px;
    font-size: 1rem;
    font-weight: 600;
    border-radius: 10px;
    border: 1px solid #444444;
    cursor: pointer;
    transition: filter 0.2s;
    position: relative;
    z-index: 1;
}

button:hover {
    filter: brightness(1.05);
}

/* Number buttons */
button.number {
    background-color: #2D2D2D;
    color: #E0E0E0;
}

/* Operator buttons */
button.operator {
    background-color: #37474F;
    color: #4FC3F7;
}

/* Equals button */
button.equals {
    background-color: #81C784;
    color: #121212;
}

/* Clear / Reset button */
button.clear {
    background-color: #E57373;
    color: #121212;
}

.main-footer {
    grid-area: footer;
    text-align: center;
    color: #888;
    font-size: 0.9rem;
    padding: 10px;
}
```

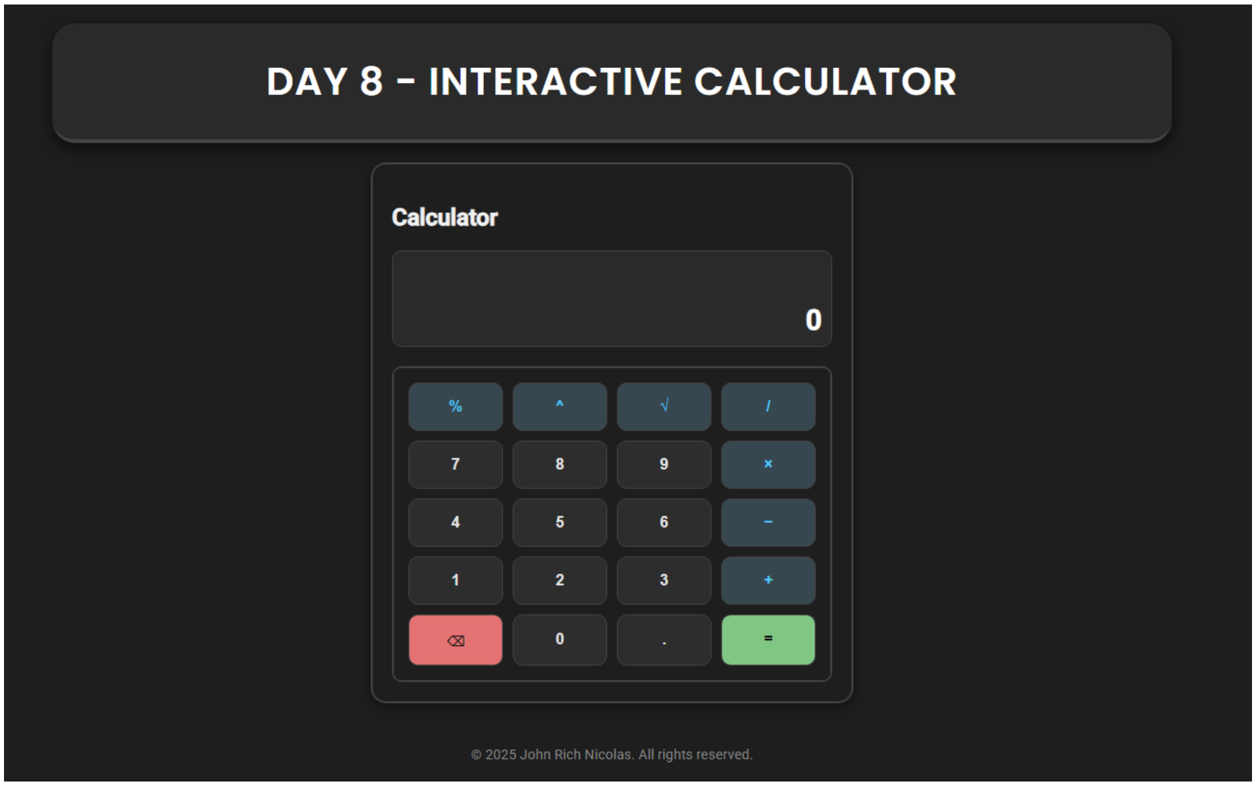
SCRIPT

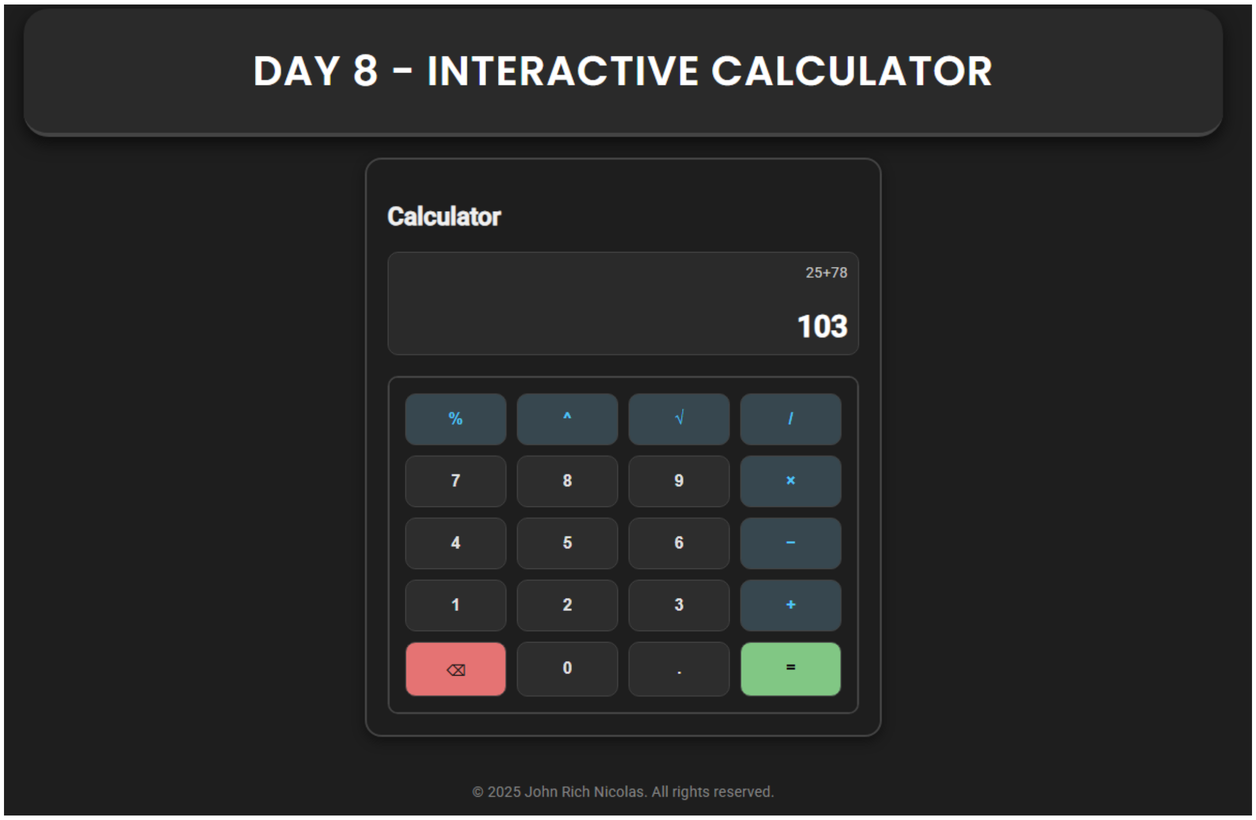
```
const resultDisplay = document.getElementById("result");
const historyDisplay = document.getElementById("historyNum");

let expression = "";
let hasError = false;

function updateDisplay() {
  resultDisplay.textContent = expression || "0";
}

function showError(message) {
  resultDisplay.textContent = message;
  historyDisplay.textContent = expression || "";
  expression = "";
  hasError = true;
}

function clearDisplay() {
  expression = "";
  resultDisplay.textContent = "0";
  historyDisplay.textContent = "";
  hasError = false;
}

function handleInput(input) {
  if (hasError) clearDisplay();

  try {
    const lastChar = expression.slice(-1);

    let displayInput = input;
    if (input === "*") displayInput = "×";
    if (input === "/") displayInput = "÷";
    if (input === "-") displayInput = "−";
    if (input === "sqrt") displayInput = "√";
    if ("+-×÷%^".includes(lastChar) && "+×÷%^".includes(displayInput)) {
      throw new Error("Cannot have two operators in a row");
    }

    if (expression === "" && "+×÷%^".includes(displayInput)) {
      throw new Error("Expression cannot start with an operator");
    }

    if (displayInput === ".") {
      const currentNumberMatch = expression.match(/[\d.]*$/);
      if (currentNumberMatch && currentNumberMatch[0].includes(".")) {
        throw new Error("Number already contains a decimal point");
      }
      if (expression === "" || "+-×÷%^(".includes(lastChar)) {
        expression += "0.";
        updateDisplay();
        return;
      }
    }

    if (displayInput === "√") {
      if (expression !== "" && (/[\d)]/.test(lastChar))) {
        expression += "×√(";
      } else {
        expression += "√(";
      }
    } else {
      if (displayInput === "(" && expression !== "" && (/[\d)]/.test(lastChar)))
{
        expression += "×(";
      } else {
        expression += displayInput;
      }
    }

    updateDisplay();
  } catch (error) {
    showError(error.message);
  }
}

function calculateResult() {
  try {
    if (!expression) return;
    // Save the original expression before any mutation
    const originalExpression = expression;

    let evalExpr = expression
      .replace(/×/g, "*")
      .replace(/÷/g, "/")
      .replace(/−/g, "-")
      .replace(/√\(/g, "Math.sqrt(")
      .replace(/\^/g, "**");

    // Auto-close unmatched parentheses
    const openParens = (evalExpr.match(/\(/g) || []).length;
    const closeParens = (evalExpr.match(/\)/g) || []).length;
    evalExpr += ")".repeat(openParens - closeParens);

    // Handle percentage
    evalExpr = evalExpr.replace(/(\d+(\.\d+)?)%(?![\d(])/g, "($1/100)");

    // Prevent divide by zero
    if (/\/\s*0(?![\d.])/.test(evalExpr)) {
      throw new Error("Cannot divide by zero");
    }

    // Validate square roots of negative numbers
    const sqrtMatches = evalExpr.match(/Math\.sqrt\(([^()]+)\)/g);
    if (sqrtMatches) {
      for (const match of sqrtMatches) {
        const innerExpr = match.match(/Math\.sqrt\(([^()]+)\)/)[1];
        try {
          const innerValue = Function('"use strict";return (' + innerExpr +
')')();
          if (typeof innerValue !== "number" || isNaN(innerValue)) {
            throw new Error("Invalid value inside square root");
          }
          if (innerValue < 0) {
            throw new Error("Cannot calculate square root of a negative number");
          }
        } catch (e) {
          throw new Error("Cannot calculate square root of a negative number");
        }
      }
    }

    const result = eval(evalExpr);
    if (!isFinite(result)) {
      throw new Error("Result is too large or too small to display");
    }

    if (isNaN(result)) {
      throw new Error("Invalid mathematical operation");
    }

    // Display history correctly using saved original expression
    historyDisplay.textContent = originalExpression;
    expression = result.toString();
    hasError = false;
    updateDisplay();
  } catch (error) {
    if (error.message.includes("Cannot calculate square root of a negative 
number")) {
      showError("Cannot calculate square root of a negative number");
    } else if (error.message.includes("Cannot divide by zero")) {
      showError("Cannot divide by zero");
    } else if (error.message.includes("Result is too large")) {
      showError("Result is too large or too small to display");
    } else if (error.message.includes("Invalid mathematical operation")) {
      showError("Invalid mathematical operation");
    } else if (error.message.includes("Unexpected token")) {
      showError("Invalid expression format");
    } else if (error.message.includes("SyntaxError")) {
      showError("Invalid expression syntax");
    } else {
      showError("Calculation error - please check your input");
    }
  }
}

// Button Event Listeners
document.querySelectorAll(".number").forEach(btn => {
  btn.addEventListener("click", () => handleInput(btn.dataset.num));
});

document.querySelectorAll(".operator").forEach(btn => {
  btn.addEventListener("click", () => handleInput(btn.dataset.op));
});

document.querySelector(".clear").addEventListener("click", () => {
  if (hasError || expression === "") {
    clearDisplay();
  } else {
    if (expression.endsWith("√(")) {
      expression = expression.slice(0, -2);
    } else {
      expression = expression.slice(0, -1);
    }
    updateDisplay();
  }
});

document.querySelector(".equals").addEventListener("click", calculateResult);

// Keyboard Support
document.addEventListener("keydown", (e) => {
  const key = e.key;

  if (hasError && !["Escape", "Backspace", "Delete"].includes(key)) {
    clearDisplay();
  }

  if (/\d/.test(key) || key === ".") {
    handleInput(key);
  } else if (["+"].includes(key)) {
    handleInput(key);
  } else if (key === "-") {
    handleInput("−");
  } else if (key === "*") {
    handleInput("×");
  } else if (key === "/") {
    e.preventDefault();
    handleInput("÷");
  } else if (key === "%") {
    handleInput("%");
  } else if (key === "^") {
    handleInput("^");
  } else if (key === "(" || key === ")") {
    handleInput(key);
  } else if (key === "Enter") {
    e.preventDefault();
    calculateResult();
  } else if (key === "Escape" || key === "Delete") {
    clearDisplay();
  } else if (key === "Backspace") {
    if (expression.endsWith("√(")) {
      expression = expression.slice(0, -2);
    } else {
      expression = expression.slice(0, -1);
    }
    updateDisplay();
  } else if (key.toLowerCase() === "r") {
    handleInput("sqrt");
  }
});
```

John Rich Nicolas
DAY 8

FRONTEND
CALCULATOR

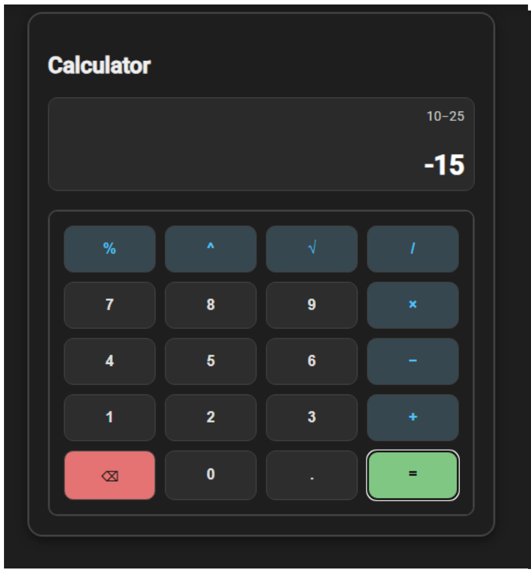
OUTPUT


DAY 8 – INTERACTIVE CALCULATOR
Calculator
0
%
^
√
/
7
8
9
×
4
5
6
−
1
2
3
+
⌫
0
.
=
© 2025 John Rich Nicolas. All rights reserved.

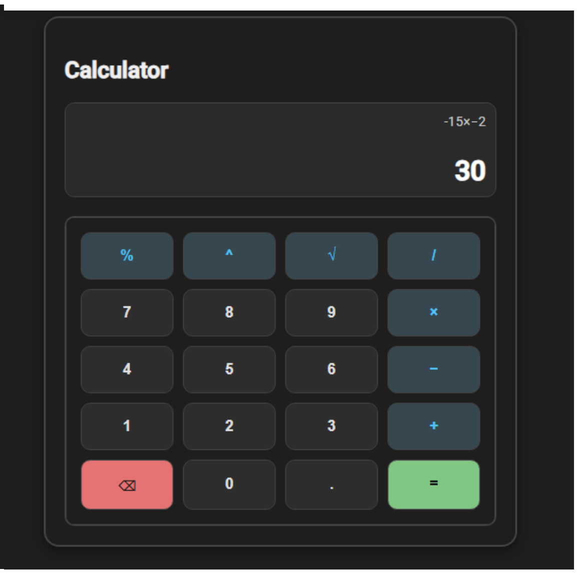

DAY 8 – INTERACTIVE CALCULATOR
Calculator
25+78
103
%
^
√
/
7
8
9
×
4
5
6
−
1
2
3
+
⌫
0
.
=
© 2025 John Rich Nicolas. All rights reserved.

John Rich Nicolas
DAY 8

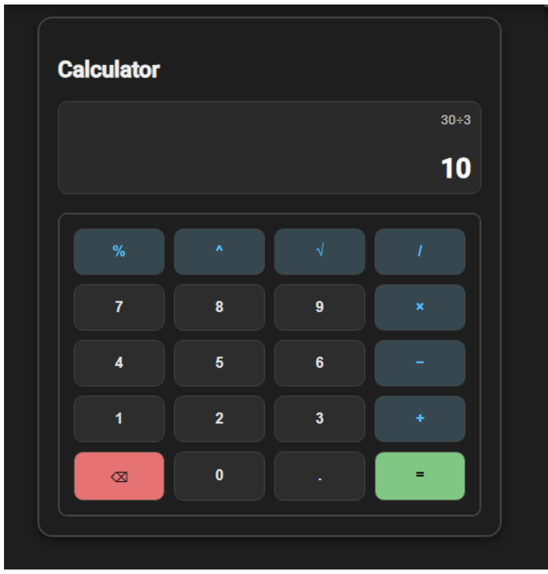
FRONTEND
CALCULATOR


Calculator

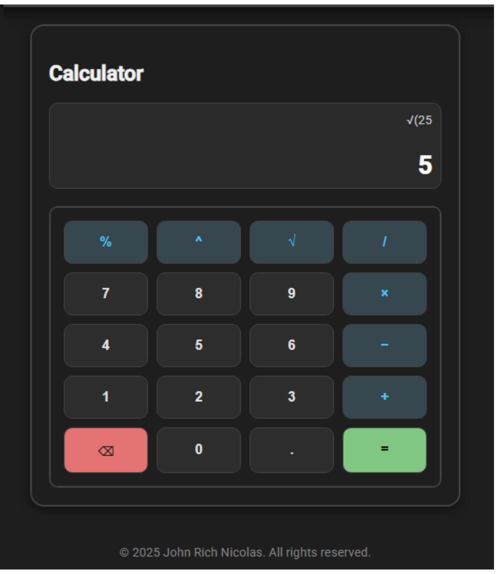
10−25
-15


Calculator

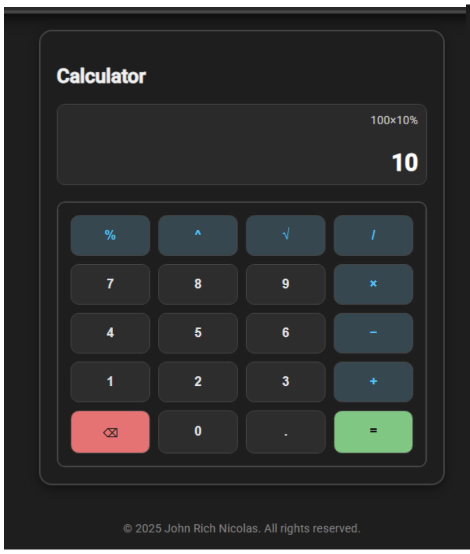
-15×−2
30


Calculator

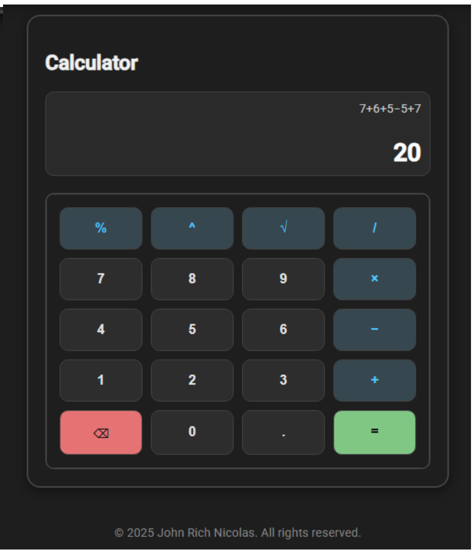
30÷3
10


Calculator

√(25
5


Calculator

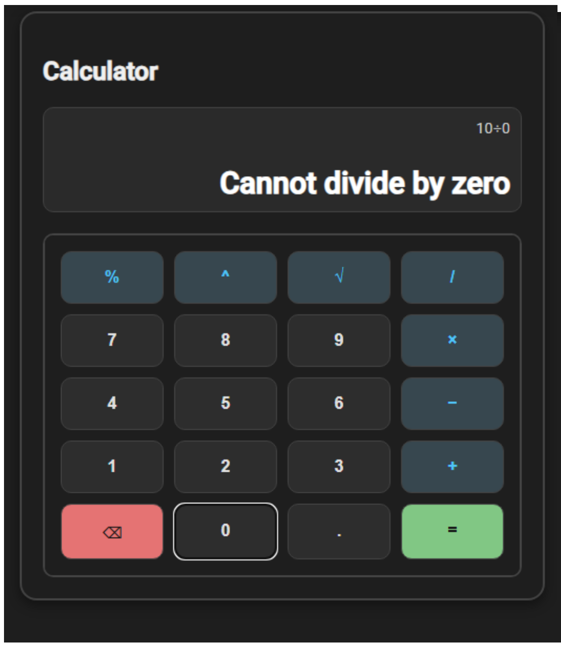
100×10%
10


Calculator

7+6+5−5+7
20

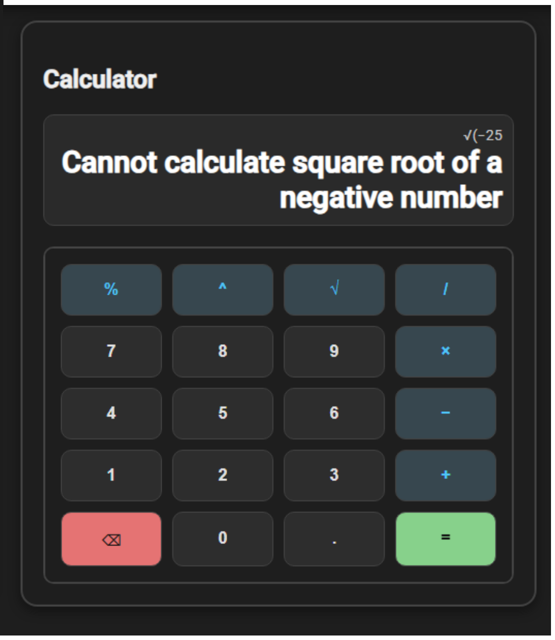
John Rich Nicolas
DAY 8

FRONTEND
CALCULATOR


Calculator

10÷0

Cannot divide by zero


Calculator

√(−25

Cannot calculate square root of a negative number

John Rich Nicolas FRONTEND
DAY 8 CALCULATOR

REFLECTION

In my calculator project, I used different functions to organize the code and separate the logic. For example, I created one function to handle user input, another to calculate results, one for updating the display, and one for handling errors. This helped me keep the code clean and easier to manage. When a user enters something, the input function adds it to the current expression, and the display updates right away. The calculateResult() function is only called when the user presses the equals button or hits Enter, and that's when the result is shown. If something goes wrong, like an invalid input, the showError() function displays a message to let the user know what happened.

I also added input validation so the calculator wouldn't accept invalid combinations like two operators in a row or multiple decimals in one number. I made sure it checks for divide-by-zero errors, and I added special handling for square roots of negative numbers. Before evaluating those parts, the code checks if the value inside the square root is less than zero. If it is, the calculator shows an error instead of crashing or showing NaN. I also used try...catch in the calculateResult() function to catch any unexpected errors and show a friendly message to the user.

To make the calculator feel responsive, I used functions that update the screen in real time as the user types or clicks buttons. For example, when a button is clicked or a key is pressed, the calculator updates the result area immediately. I used DOM manipulation to get the elements from the HTML and update their textContent. I added event listeners to all the number and operator buttons, and also added support for keyboard input using the keydown event. This way, users can interact with the calculator using either their mouse or keyboard, which makes it feel more complete.

The calculator supports basic operations like addition, subtraction, multiplication, and division, but I also included more advanced functions like modulus (%), exponents (^), and square roots (√). These required some extra logic because I had to replace them with valid JavaScript syntax before evaluating the expression. For example, I converted ^ to ** and √(to Math.sqrt(. That part was tricky but made the calculator more powerful.

Honestly, I struggled a lot while working on this project. It was my first time learning how to use event listeners properly, and it was confusing at first especially when I tried to add keyboard functionality. The logic was also overwhelming because there were so many things to consider just to make the calculator behave properly. I spent a lot of time researching and testing different solutions just to fix small issues. It was definitely time-consuming, but in the end, I learned a lot about combining JavaScript logic with HTML and how the DOM works to make interactive web apps. Even though it was hard, I'm proud of how much I improved by doing this project.