Simple JavaScript Calculator

Yubraj Niraula
05/29/2024

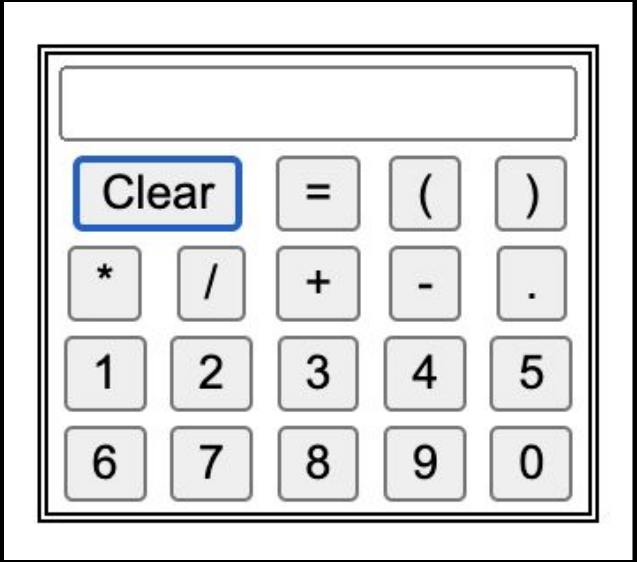

Table of Contents

Introduction

- Project Goal: Create a simple, functional calculator using HTML, CSS, and JavaScript.
- Features: Basic arithmetic operations, error handling, keyboard support.
- Audience: Beginners in web development, educators, hobbyists.

Design

- Problems: Need for a basic calculator, user-friendly interface, error handling.
- Possible Solutions:
  - Use of eval() for calculation.
  - Error handling mechanisms for invalid inputs.
- Comparison Criteria: Ease of implementation, simplicity, reliability.
- Chosen Solution: Use **eval()** with error handling due to its straightforward implementation.

Implementation

- HTML Structure:
  - Table layout for buttons and screen.
  - Readonly input field for display.
- CSS Styling:
  - Simple, clean design for clarity and ease of use.
  - Responsive layout for various screen sizes.
- JavaScript Functionality:
  - Button event listeners for input.
  - **eval()** for expression evaluation.
  - Error handling with **try-catch**.
  - Keyboard support for enhanced user experience.

HTML Structure

Element	**Description**
input	Display screen
button	Calculator buttons

Example:

```
<input type="text" id="screen" readonly>

<button>1</button>
```

CSS Styling

Example:

```
table {
  border-collapse: collapse;
  margin: 30px auto;
  border: 3px double black;
}
td {
  padding: 10px;
  border: 1px solid black;
}
```

Javascript Functionality

Key Points

- Event listeners for buttons
- Use of eval() for calculation
- Error handling
- Keyboard support

Example:

```
    buttons.forEach(button => {

  button.addEventListener('click',
function() {

    handleInput(this.textContent);

  });

});
```

Test

- Testing Process
    - Functional Testing:
        - Valid operations (e.g., 1+1)
        - Invalid operations (e.g., 1++)
        - Edge cases (e.g., division by zero)
- User Testing:
    - User feedback on interface and usability
- Tools Used:
    - Browser developer tools for debugging

Enhancement Ideas

- UI Improvements:
  - Better styling and layout.
  - Responsive design for mobile devices.
- Advanced Features:
  - Scientific functions (e.g., square root, power).
  - History of calculations.
- Accessibility:
  - Support for screen readers.
  - Keyboard navigation improvements.

Conclusion

- Achievements:
  - Functional basic calculator.
  - Error handling and keyboard support.
- Learning Outcomes:
  - Enhanced understanding of HTML, CSS, and JavaScript.
  - Practical experience with event handling and error management.
- Next Steps:
  - Implement enhancement ideas.
  - Gather more user feedback.

Bibliography/References

- MDN Web Docs: JavaScript Reference
- W3Schools: HTML, CSS, and JavaScript Tutorials
- Stack Overflow: Community solutions and discussions

- Google slides link:
  - https://docs.google.com/presentation/d/1cOqLtc2XlBybLJqFViCR3alNdWndNDdE00ZD_IxLd-o/edit?usp=sharing
- Github project link:
  - https://github.com/yubrajniraula/Cloud-Computing