Web Programming Final Project

Developed by

Tuncay EŞSİZ - 191005053

Contents

[Project Overview 1](#_Toc158163777)

[Implementation Details 1](#_Toc158163778)

[HTML Structure 1](#_Toc158163779)

[CSS Styling 1](#_Toc158163780)

[ Responsive Design 1](#_Toc158163781)

[ Button Interaction 1](#_Toc158163782)

[ Fade-in Effect for Results 1](#_Toc158163783)

[JavaScript Logic 1](#_Toc158163784)

[ Event Listeners 1](#_Toc158163785)

[ Calculation and Error Handling 1](#_Toc158163786)

[ History Feature 1](#_Toc158163787)

[Challenges and Solutions 2](#_Toc158163788)

[Responsive Design 2](#_Toc158163789)

[Calculation Logic 2](#_Toc158163790)

[History Management 2](#_Toc158163791)

[References and Learning Resources 2](#_Toc158163792)

[Conclusion 2](#_Toc158163793)

# Project Overview

My goal goal was to build a calculator that not only performs calculations but also enhances user experience with a responsive design and additional features like calculation history. I structured the application into three main components: the HTML for the layout, CSS for styling, and JavaScript for functionality.

# Implementation Details

## HTML Structure

I started by defining the HTML structure, ensuring that the calculator's interface was intuitive. The calculator class wraps the entire calculator, including the display area (result-container) and the button grid (calculator-keys). I included a separate history div to display past calculations, initially hidden.

## CSS Styling

For the styling, I aimed for a clean and modern look. I used Flexbox to center the calculator on the page and Grid layout for the button arrangements to ensure responsiveness. Styling highlights include:

* Responsive Design: The calculator's width adjusts based on the screen size, with a maximum width set to ensure it remains user-friendly on large screens.
* Button Interaction: I applied hover and active states to the buttons to provide visual feedback to the user. For example, the :hover and :active pseudo-classes were used to change the background color and scale of the buttons, enhancing the interactive experience.
* Fade-in Effect for Results: I introduced a fade-in effect for the result display using CSS transitions, which added a polished feel to the calculator's operations.

## JavaScript Logic

The core of the calculator's functionality lies in the JavaScript code. Key aspects include:

* Event Listeners: I attached an event listener to the calculator's keys to handle clicks, capturing the button values to perform operations or update the display.
* Calculation and Error Handling: I used the eval function for calculation, wrapped in a try...catch block to handle any errors, such as invalid expressions, displaying an "Error" message when necessary.
* History Feature: Implementing the history feature involved storing calculations in an array and updating the history div content. A toggle function was also added to show or hide the history panel.

# Challenges and Solutions

## Responsive Design

One challenge was ensuring the calculator was fully responsive and user-friendly across devices. I addressed this by using CSS Flexbox and Grid, which allowed for flexible layout adjustments. Media queries were not necessary due to the inherent responsiveness of these CSS features.

## Calculation Logic

Handling calculations securely and efficiently was another challenge, given the potential security risks of using eval for evaluating expressions. I mitigated this by limiting the calculator's functionality to basic arithmetic operations and implementing error handling to catch any malicious or incorrect input.

## History Management

Managing the calculation history without cluttering the UI was tricky. I implemented a simple yet effective solution by displaying only the most recent entries and providing a toggle button to show or hide the history panel, keeping the interface clean.

# References and Learning Resources

Throughout the development process, I relied on several key resources:

* **MDN Web Docs** for understanding HTML, CSS, and JavaScript best practices.
* **CSS Tricks** for guides on Flexbox and Grid.
* **Stack Overflow** for community advice and example code pices on specific issues, particularly around JavaScript.
* **Grammarly** for writing & translating.

# Conclusion

Developing the Responsive Calculator was a rewarding experience that honed my skills in web development, particularly in creating intuitive and responsive interfaces. The challenges encountered along the way were valuable learning opportunities, leading to a deeper understanding of effective web design and development practices for me.