**CHAPTER 1**

**INTRODUCTION**

In modern web development, three core technologies serve as the foundation for creating interactive, well-designed, and functional websites: HTML, CSS, and JavaScript. Each of these plays a distinct but complementary role in the development process, and understanding their individual functions is crucial for building any web-based application.

**HTML (HyperText Markup Language)**

**HTML** is the standard language used to structure content on the web. It defines the layout and structure of web pages by using a system of elements and tags that describe various types of content—such as text, images, links, and multimedia—on a webpage.

**Key Concepts of HTML:**

* **Elements**: HTML documents are made up of elements, typically represented by tags. Each tag has an opening and closing counterpart, with the content placed in between. For example: <p>This is a paragraph.</p>.
* **Attributes**: Elements can have attributes that provide additional information or modify their behavior. For instance, an <img> element can include an src attribute to specify the image source.
* **Document Structure**: A basic HTML document consists of the following key components:
  + <html>: The root element that encompasses the entire document.
  + <head>: Contains metadata, links to external resources (like CSS files), and the page’s title.
  + <body>: Contains the visible content of the web page.

**CSS (Cascading Style Sheets)**

**CSS** is used to control the **presentation** of the web page's content, enabling developers to implement sophisticated layouts, styling, and responsiveness. While HTML handles the structure, CSS is responsible for the visual appeal of a webpage.

**Key Concepts of CSS:**

* **Selectors**: CSS rules are applied to HTML elements using selectors. A selector targets HTML elements, and properties define the styles to be applied to those elements.
* **Box Model**: Every element on a webpage is treated as a box, which consists of content, padding, border, and margin. This concept is essential for understanding spacing and layout.
* **Positioning**: CSS provides various ways to position elements on the page:
* **Flexbox and Grid**: These are powerful CSS layout systems that allow for more flexible and complex designs without the need for floats or positioning.

**JavaScript**

**JavaScript** is a dynamic programming language used to **add interactivity** and functionality to websites. While HTML structures the content and CSS styles it, JavaScript enables dynamic changes and real-time interaction with the user.

**Key Concepts of JavaScript:**

* **Variables and Data Types**: Variables store data that can be used and manipulated throughout the code. JavaScript supports various data types, such as strings, numbers, booleans, and arrays.
* **Control Flow**: JavaScript includes control flow structures like **if/else**, **switch**, and loops (**for**, **while**) to execute code conditionally or repetitively.
* **DOM Manipulation**: JavaScript allows for direct interaction with the Document Object Model (DOM), which represents the structure of an HTML document. This enables the dynamic updating of the content, structure, and style of a page.

**Asynchronous JavaScript:**

JavaScript also supports asynchronous programming through mechanisms such as **callbacks**, **Promises**, and **async/await**, enabling efficient handling of tasks like data fetching from APIs or server communication without blocking the main thread.

**CHAPTER 2**

**EXECUTIVE SUMMARY**

The **CodePen** enables users to write and view HTML, CSS, and JavaScript code in real-time with an interactive live preview of the output. The interface is divided into an **editor section** for code input and an **output section** for displaying the result. Key features include **theme toggling**, **output expansion**, and **real-time code execution**.

**Key Features:**

1. **Code Editors**:
   * The editors for HTML, CSS, and JavaScript are powered by **CodeMirror**, a versatile JavaScript library for code editing. Each editor provides syntax highlighting, line numbers, and supports the respective coding languages (HTML, CSS, JavaScript).
2. **Real-Time Output**:
   * The output area is dynamically updated as the user types. Code from all three editors is compiled into a single HTML document and rendered in an embedded **iframe** to provide immediate visual feedback.
3. **Dark Mode & Theme Toggle**:
   * Users can toggle between **dark** and **light themes** via a button in the header. When dark mode is enabled, the background, text, and editor areas are styled accordingly for better visibility in low-light conditions.
4. **Responsive Design**:
   * The layout uses **flexbox** to ensure that the interface is flexible and adjusts properly across different screen sizes. The editor sections are displayed side-by-side on larger screens and stack vertically on smaller devices.
5. **Output Section with Expand/Collapse**:
   * The output section contains a button that allows users to **expand** or **shrink** the output area. When expanded, the output occupies the full screen, providing an immersive view of the rendered result.

**Technical Details:**

* **HTML Structure**: The page consists of a **header** with the title and theme toggle button, a main editor area containing three separate editors for HTML, CSS, and JavaScript, and an **iframe** for rendering the live output.
* **CSS Styling**: The application is styled with custom CSS for layout, spacing, and theming. The dark mode is implemented with a specific set of styles that alter background colors, text colors, and borders.
* **JavaScript Functionality**:
  + **CodeMirror** editors are initialized for each of the three languages, providing a rich code editing experience.
  + **Real-time Preview**: The updateOutput() function listens for changes in any of the editors and updates the iframe with the combined HTML, CSS, and JavaScript content.
  + **Theme Toggle**: The toggleTheme() function switches between dark and light modes by toggling a dark class on the body element.
  + **Output Resizing**: The expandOutput() and shrinkOutput() functions toggle the fullscreen state of the output section, allowing users to view the result in a larger format.

This application mimics the functionality of a simple web-based code editor like **CodePen**, providing users with a streamlined interface to quickly prototype and test HTML, CSS, and JavaScript code. The interactive nature, real-time feedback, and aesthetic modes make it suitable for developers, learners, and anyone needing a lightweight code testing environment.

**CHAPTER 3**

**CODEPEN**

**Step 1:** Set Up the HTML Structure

Start by setting up the basic structure of the HTML document, which include the <head> and <body> sections.

* **HTML Declaration**: Use <!DOCTYPE html> to specify the document type.
* **Meta Tags**: Add <meta> tags for charset and viewport settings.
* **Title**: Set the title of the document to "CodePen Clone".

**Key Sections:**

1. **Header**:
   * A header section with a title "CodePen" and a button to toggle the theme.
2. **Editor Section**:
   * A container with three sub-sections for the HTML, CSS, and JavaScript editors.
   * Each editor should be wrapped in a <div> with a label for the title (HTML, CSS, or JavaScript) and a <textarea> to input the code.
3. **Output Section**:
   * An output container with a header and an embedded <iframe> to display the result.

**Code:**

<header>

CodePen

<button class="theme-toggle" onclick="toggleTheme()">🌙</button>

</header>

<div class="editor-container">

<div class="editor">

<label>HTML</label>

<textarea id="html-editor"></textarea>

</div>

<div class="editor">

<label>CSS</label>

<textarea id="css-editor"></textarea>

</div>

<div class="editor">

<label>JavaScript</label>

<textarea id="js-editor"></textarea>

</div>

</div>

<div class="output-container" id="output-container">

<div class="output-header">

<span>Output</span>

<button class="expand-btn" onclick="expandOutput()">Expand</button>

<button class="shrink-btn" onclick="shrinkOutput()">Shrink</button>

</div>

<iframe id="output"></iframe>

</div>

**Step 2:** Integrate CodeMirror for Enhanced Code Editing

You will now integrate **CodeMirror**, a powerful JavaScript library, to enable advanced code editing with syntax highlighting, line numbers, and more.

1. **Include the CodeMirror CSS and JavaScript libraries** in the <head> section of the document.
2. **Initialize CodeMirror** for each editor (HTML, CSS, JavaScript) by creating instances using CodeMirror.fromTextArea().

**Code:**

const htmlEditor = CodeMirror.fromTextArea(document.getElementById('html-editor'), {

mode: 'htmlmixed',

theme: 'neo',

lineNumbers: true,

});

const cssEditor = CodeMirror.fromTextArea(document.getElementById('css-editor'), {

mode: 'css',

theme: 'neo',

lineNumbers: true,

});

const jsEditor = CodeMirror.fromTextArea(document.getElementById('js-editor'), {

mode: 'javascript',

theme: 'neo',

lineNumbers: true,

});

**Step 3:** Add Real-Time Output Preview

To update the output in the iframe as users type code, write a function that combines the HTML, CSS, and JavaScript from each editor and injects it into the iframe.

1. **Combine Code**: The HTML code from the HTML editor, the CSS code wrapped in <style>, and the JavaScript code wrapped in <script> are concatenated together.
2. **Write to the iframe**: Use outputDoc.open(), outputDoc.write(), and outputDoc.close() to inject the combined content into the iframe.

**Code**:

function updateOutput() {

const htmlCode = htmlEditor.getValue();

const cssCode = `<style>${cssEditor.getValue()}</style>`;

const jsCode = `<script>${jsEditor.getValue()}<\/script>`;

const outputDoc = outputFrame.contentDocument || outputFrame.contentWindow.document;

outputDoc.open();

outputDoc.write(htmlCode + cssCode + jsCode);

outputDoc.close();

}

**Step 4:** Implement Dark and Light Mode Toggle

Add a theme toggle button that allows users to switch between dark and light modes. This is achieved by toggling a CSS class (dark) on the body element.

1. **Create the Toggle Button**: The button with the theme-toggle class toggles the theme between dark and light.
2. **CSS Classes for Dark Mode**: Define the styles for the dark theme and apply them when the dark class is added to the body.

**Code**:

function toggleTheme() {

isDark = !isDark;

document.body.classList.toggle('dark');

themeToggleBtn.textContent = isDark ? '☀️' : '🌙';

}

CSS Involved:-

body.dark {

background-color: #2b2b2b;

color: #e0e0e0;

}

body.dark textarea,

body.dark input {

background-color: #000;

color: #fff;

border: 1px solid #555;

}

body.dark .editor label,

body.dark .output-header {

background-color: #555;

}

**Step 5:** Expand and Shrink Output Section

Allow users to **expand** and **shrink** the output container for better viewing.

1. **Expand Button**: When clicked, the output container is made fullscreen.
2. **Shrink Button**: When clicked, the output returns to its original size.

**Code**:

function expandOutput() {

outputContainer.classList.add('fullscreen');

document.querySelector('.expand-btn').style.display = 'none';

document.querySelector('.shrink-btn').style.display = 'inline-block';

}

function shrinkOutput() {

outputContainer.classList.remove('fullscreen');

document.querySelector('.expand-btn').style.display = 'inline-block';

document.querySelector('.shrink-btn').style.display = 'none';

}

**Step 6: CSS Styling and Layout Adjustments**

Style the layout using **Flexbox** to ensure that the editor sections are responsive and the output area is properly aligned.

1. **Editor Layout**: Use Flexbox to position the three editor areas (HTML, CSS, JS) side by side on large screens and stack vertically on smaller screens.
2. **Output Section Styling**: The output area should be styled with a background color, a header, and an iframe. When expanded, the output should occupy the entire screen.

**Code:**

.editor-container {

display: flex;

gap: 10px;

flex-wrap: wrap;

}

.editor {

flex: 1;

min-width: 300px;

}

.CodeMirror {

height: 350px;

}

.output-container {

flex: 1;

position: relative;

padding: 10px;

}

.output-container.fullscreen {

position: fixed;

top: 0;

left: 0;

width: 100%;

height: 100%;

z-index: 1000;

}

iframe {

width: 100%;

height: 150px;

border: none;

}

**CHAPTER 4**

**SOURCE CODE & OUTPUT**

**Program:**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>CodePen Clone</title>

<!-- CodeMirror Library -->

<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/codemirror/5.65.12/codemirror.min.css">

<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/codemirror/5.65.12/theme/material.min.css">

<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/codemirror/5.65.12/theme/neo.min.css">

<script src="https://cdnjs.cloudflare.com/ajax/libs/codemirror/5.65.12/codemirror.min.js"></script>

<script src="https://cdnjs.cloudflare.com/ajax/libs/codemirror/5.65.12/mode/htmlmixed/htmlmixed.min.js"></script>

<script src="https://cdnjs.cloudflare.com/ajax/libs/codemirror/5.65.12/mode/css/css.min.js"></script>

<script src="https://cdnjs.cloudflare.com/ajax/libs/codemirror/5.65.12/mode/javascript/javascript.min.js"></script>

<style>

/\* General styles \*/

\* {

margin: 0;

padding: 0;

box-sizing: border-box;

}

body {

font-family: Arial, sans-serif;

display: flex;

flex-direction: column;

height: 100vh;

background-color: #f5f5f5;

color: #333;

transition: background-color 0.3s, color 0.3s;

}

/\* Header \*/

header {

background-color: #333;

color: #fff;

padding: 15px;

font-size: 1.8rem;

text-align: center;

position: relative;

}

.theme-toggle {

position: absolute;

right: 20px;

font-size: 1.5rem;

background: none;

border: none;

color: #fff;

cursor: pointer;

}

/\* Editors Section \*/

.editor-container {

display: flex;

flex: 3;

gap: 10px;

padding: 10px;

flex-wrap: wrap;

margin-bottom: 5px;

}

.editor {

flex: 1;

display: flex;

flex-direction: column;

min-width: 300px;

margin-bottom: 20px; /\* Added gap \*/

}

.editor label {

background-color: #444;

color: #fff;

padding: 8px;

text-align: center;

}

.CodeMirror {

height: 350px; /\* Increased height \*/

}

/\* Output Section \*/

.output-container {

flex: 1;

position: relative;

padding: 10px;

margin-top: 5px; /\* Gap above output box \*/

display: flex;

flex-direction: column;

align-items: center;

}

.output-header {

display: flex;

justify-content: space-between;

align-items: center;

background-color: #444;

color: #fff;

padding: 8px;

width: 100%;

text-align: center;

border-radius: 5px 5px 0 0;

}

.output-header span {

flex-grow: 1;

text-align: center;

}

iframe {

width: 100%;

border: none;

height: 150px; /\* Default height \*/

transition: all 0.3s ease;

}

/\* Fullscreen Output \*/

.output-container.fullscreen {

position: fixed;

top: 0;

left: 0;

width: 100%;

height: 100%;

z-index: 1000;

background-color: #fff;

}

.output-container.fullscreen iframe {

height: calc(100% - 50px);/\* Adjust iframe height within fullscreen \*/

}

.output-container.fullscreen .output-header {

border-radius: 0; /\* Remove rounded corners in fullscreen \*/

}

.expand-btn, .shrink-btn {

background-color: #555;

color: #fff;

border: none;

padding: 5px 10px;

border-radius: 5px;

cursor: pointer;

}

.shrink-btn {

display: none;

}

/\* Dark Mode \*/

body.dark {

background-color: #2b2b2b;

color: #e0e0e0;

}

header {

background-color: #000; /\* Black background \*/

color: #fff; /\* White text \*/

}

/\* Input boxes and textareas in dark mode \*/

body.dark textarea,

body.dark input {

background-color: #000; /\* Black background inside the input box \*/

color: #fff; /\* White text \*/

border: 1px solid #555; /\* Add a subtle border for visibility \*/

}

/\* Normal mode input box styling \*/

textarea,

input {

background-color: #fff; /\* Default white background \*/

color: #000; /\* Default black text \*/

border: 1px solid #ccc; /\* Default border \*/

}

/\* Editor labels and output headers \*/

.editor label,

.output-header {

background-color: #444; /\* Dark gray background for editors and output in normal mode \*/

color: #fff; /\* White text \*/

}

body.dark .editor label, body.dark .output-header {

background-color: #555;

color: #fff;

}

</style>

</head>

<body>

<!-- Header -->

<header>

CodePen

<button class="theme-toggle" onclick="toggleTheme()">🌙</button>

</header>

<!-- Editors Section -->

<div class="editor-container">

<div class="editor">

<label>HTML</label>

<textarea id="html-editor"></textarea>

</div>

<div class="editor">

<label>CSS</label>

<textarea id="css-editor"></textarea>

</div>

<div class="editor">

<label>JavaScript</label>

<textarea id="js-editor"></textarea>

</div>

</div>

<!-- Output Section -->

<div class="output-container" id="output-container">

<div class="output-header">

<span>Output</span>

<button class="expand-btn" onclick="expandOutput()">Expand</button>

<button class="shrink-btn" onclick="shrinkOutput()">Shrink</button>

</div>

<iframe id="output"></iframe>

</div>

<!-- JavaScript -->

<script>

// CodeMirror initialization

const htmlEditor = CodeMirror.fromTextArea(document.getElementById('html-editor'), {

mode: 'htmlmixed',

theme: 'neo',

lineNumbers: true,

});

const cssEditor = CodeMirror.fromTextArea(document.getElementById('css-editor'), {

mode: 'css',

theme: 'neo',

lineNumbers: true,

});

const jsEditor = CodeMirror.fromTextArea(document.getElementById('js-editor'), {

mode: 'javascript',

theme: 'neo',

lineNumbers: true,

});

const outputFrame = document.getElementById('output');

const outputContainer = document.getElementById('output-container');

const themeToggleBtn = document.querySelector('.theme-toggle');

let isDark = false;

// Update the Output

function updateOutput() {

const htmlCode = htmlEditor.getValue();

const cssCode = `<style>${cssEditor.getValue()}</style>`;

const jsCode = `<script>${jsEditor.getValue()}<\/script>`;

const outputDoc = outputFrame.contentDocument || outputFrame.contentWindow.document;

outputDoc.open();

outputDoc.write(htmlCode + cssCode + jsCode);

outputDoc.close();

}

htmlEditor.on('change', updateOutput);

cssEditor.on('change', updateOutput);

jsEditor.on('change', updateOutput);

// Dark Mode

function toggleTheme() {

isDark = !isDark;

document.body.classList.toggle('dark');

themeToggleBtn.textContent = isDark ? '☀️' : '🌙';

}

// Expand and Shrink Output

function expandOutput() {

outputContainer.classList.add('fullscreen');

document.querySelector('.expand-btn').style.display = 'none';

document.querySelector('.shrink-btn').style.display = 'inline-block';

}

function shrinkOutput() {

outputContainer.classList.remove('fullscreen');

document.querySelector('.expand-btn').style.display = 'inline-block';

document.querySelector('.shrink-btn').style.display = 'none';

}

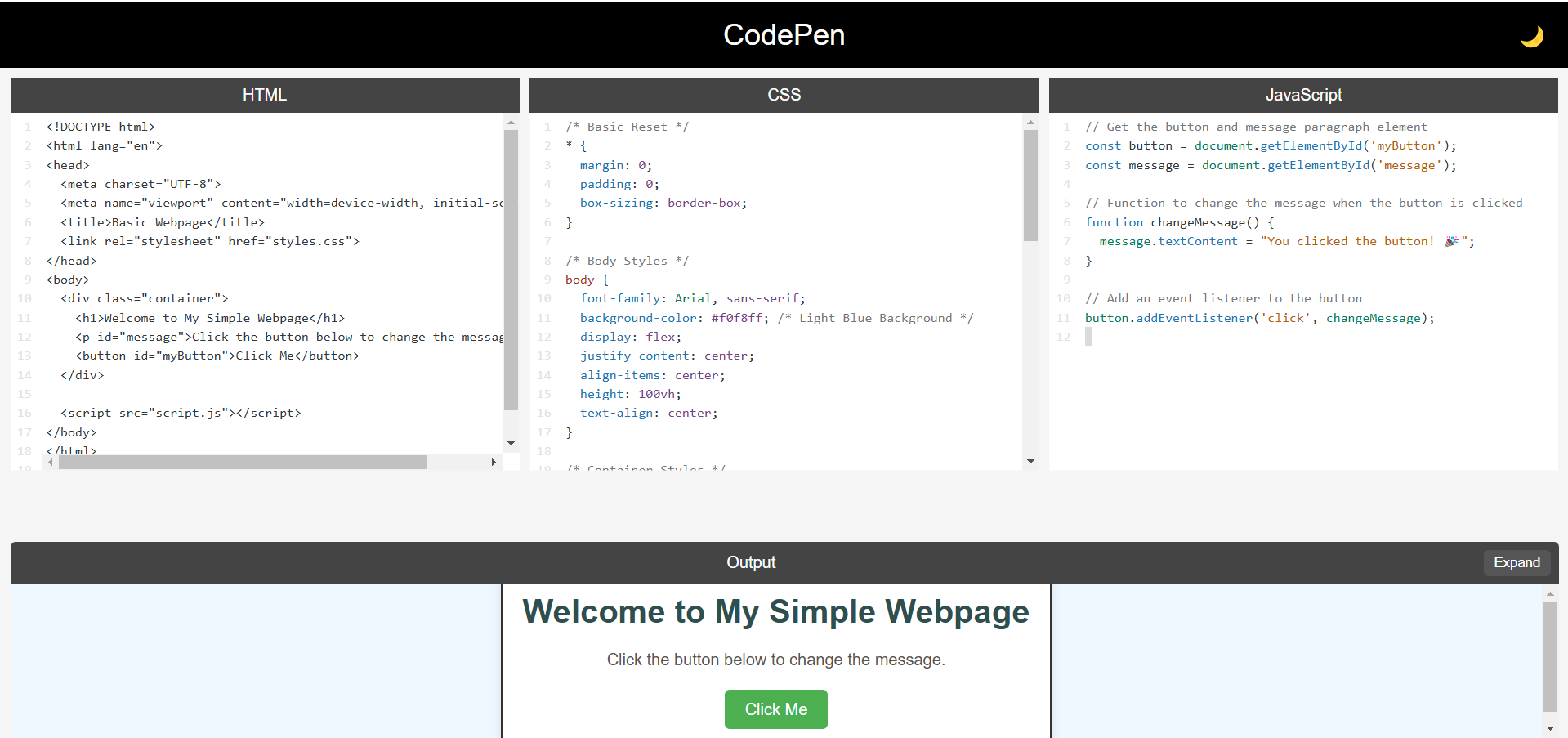
updateOutput(); // Initialize output

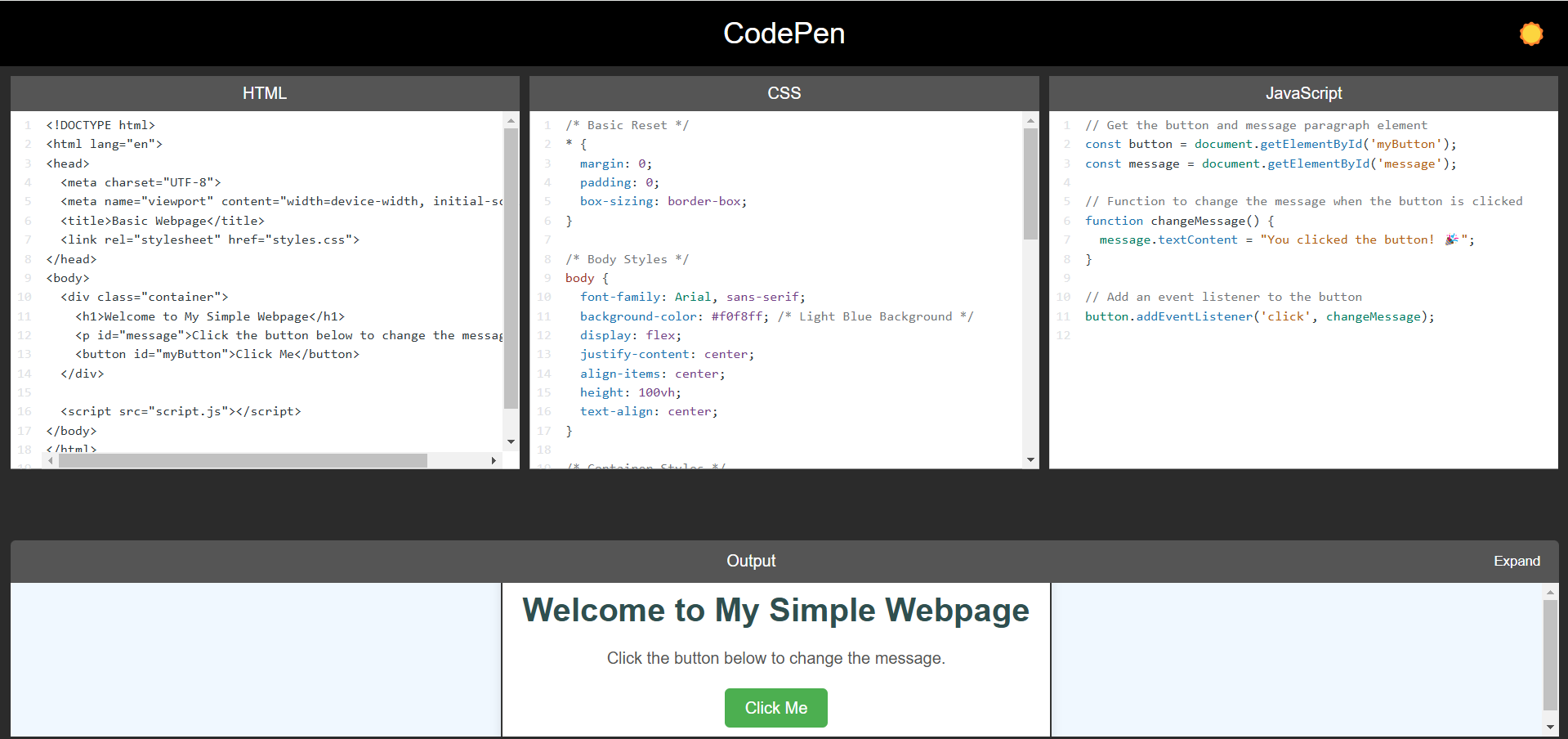
</script>

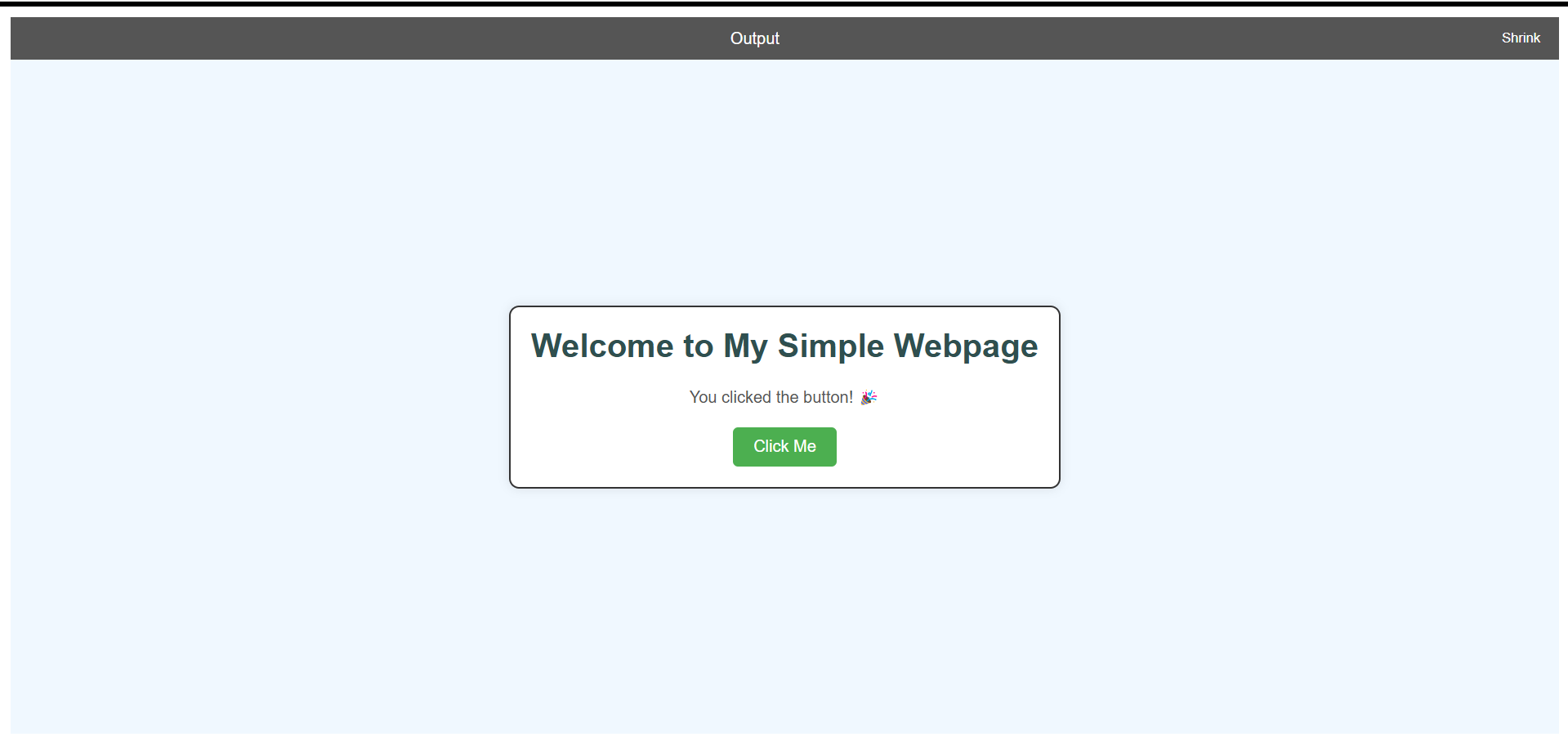
</body>

</html>

**Output:**

****

****

****

**CHAPTER 5**

**CONCLUSION**

Codepen integrates HTML, CSS, and JavaScript to create a real-time interactive code editor, allowing users to edit and view their code instantly. It uses **CodeMirror** for syntax-highlighted editing, which enhances the overall coding experience by making it more intuitive and user-friendly. As users type, the output is rendered live in an iframe, providing immediate feedback and ensuring a seamless workflow. Additionally, a dark/light mode toggle has been implemented to allow users to personalize their experience. Designed using only native HTML, CSS, and JavaScript, the tool is lightweight and accessible, making it ideal for beginners looking to learn web development basics.

**Challenges Faced:**

* **Handling real-time output rendering**: Ensuring that the output updates dynamically in response to code changes without delay or performance issues.
* **Integrating CodeMirror**: Setting up the CodeMirror library properly to work with different modes (HTML, CSS, JavaScript) while ensuring the user experience remains smooth.
* **Responsive Layout**: Ensuring the application was responsive across different screen sizes, especially when resizing the output or switching between dark and light themes.
* **Iframe Rendering**: Dynamically injecting HTML, CSS, and JavaScript into the iframe and ensuring it renders correctly with no security issues (like script injection).
* **State management**: Ensuring the theme toggle and output resize functionality worked smoothly across user interactions without causing UI glitches.

**How It Is Different from Others:**

* **No external dependencies**: Unlike many other CodePen clones, this program doesn't rely on external libraries or frameworks (like React or Vue.js). It uses only **native HTML, CSS, and JavaScript**.
* **Simplicity**: Focuses on **core web technologies** without the complexity of advanced features, making it highly suitable for beginners to understand the basic structure of a web application.
* **Minimalistic design**: The user interface is intentionally kept **simple and clean**, without excess features, making it less overwhelming compared to other feature-rich CodePen clones.
* **Dark/Light Mode**: The inclusion of a **dark/light mode toggle** makes it customizable, offering a more personalized experience compared to the default design of many similar platforms.
* **Real-time Output**: It provides a real-time, interactive output rendering mechanism using iframes, allowing users to instantly see the effects of their code without needing to refresh or click buttons.
* **No reliance on advanced JS frameworks**: The program is built with **vanilla JavaScript**, avoiding the complexity and overhead of frameworks, which makes it easier to maintain and more educational for learners.