**Cycle 1:**

**Problem Statement 1: Responsive Personal Portfolio Website (2 hours)**

Create a responsive personal portfolio website using HTML5 and CSS3. The website should include sections for your profile, skills, experience, projects, and contact information. Use Tailwind CSS to style the website and ensure it is responsive across different screen sizes.

<!doctype html>

<html>

<head>

  <meta charset="UTF-8">

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

  <link href="output.css" rel="stylesheet">

</head>

<body>

  <header class="bg-gray-800 text-white p-4">

    <h1 class="text-center text-4xl font-bold">My Portfolio</h1>

</header><br>

  <h1 class="text-3xl font-bold underline m-5">

        PORTFOLIO

  </h1>

  <div class="bg-stone-400 p-8 m-5 rounded-lg shadow-md font-roboto">

    <p>Hello Guys! My Name is jshree,  I'm interested in developing websited and making them more in teractive and easy to the user.</p>

  </div>

  <h1 class="text-3xl font-bold underline m-5">

    SKILLS

</h1>

<div class="grid grid-cols-3 gap-4 bg-slate-300 p-8 m-5">

  <div class="bg-gray-50 text-black rounded-lg p-4 m-2">COMMUNICATION</div>

  <div class="bg-gray-50 text-black rounded-lg p-4 m-2">TEAMWORK</div>

  <div class="bg-gray-50 text-black rounded-lg p-4 m-2">LISTENING</div>

  <div class="bg-gray-50 text-black rounded-lg p-4 m-2">QUICK LEARNER</div>

  <div class="bg-gray-50 text-black rounded-lg p-4 m-2">HTML5 & CSS</div>

  <div class="bg-gray-50 text-black rounded-lg p-4 m-2">REACT JS</div>

</div>

  <h1 class="text-3xl font-bold underline m-5">

    EXPERIENCE

</h1>

<div class="bg-stone-400 p-8 m-5 rounded-lg shadow-md font-roboto">

  <ul class="list-disc list-inside mt-2">

    <li>Developed and maintained web applications using React and Node.js.</li>

    <li>Collaborated with designers to create user-friendly interfaces.</li>

    <li>Implemented responsive design to ensure optimal performance across devices.</li>

</ul>

</div>

<h1 class="text-3xl font-bold underline m-5">

  PROJECTS

</h1>

<div class="grid grid-cols-3 gap-4 bg-slate-300 p-8 m-5">

<div class="bg-gray-50 text-black rounded-lg p-4 m-2">PROJECT 1</div>

<div class="bg-gray-50 text-black rounded-lg p-4 m-2">PROJECT 2</div>

<div class="bg-gray-50 text-black rounded-lg p-4 m-2">PROJECT 3</div>

<div class="bg-gray-50 text-black rounded-lg p-4 m-2">PROJECT 4</div>

<div class="bg-gray-50 text-black rounded-lg p-4 m-2">PROJECT 5</div>

<div class="bg-gray-50 text-black rounded-lg p-4 m-2">PROJECT 6</div>

</div>

<h2 class="text-3xl font-bold mb-4">Contact</h2>

<div class="bg-white p-6 rounded-lg shadow-md">

    <form>

        <div class="mb-4">

            <label for="name" class="block text-gray-700">Name</label>

            <input type="text" id="name" class="w-full p-2 border border-gray-300 rounded-lg">

        </div>

        <div class="mb-4">

            <label for="email" class="block text-gray-700">Email</label>

            <input type="email" id="email" class="w-full p-2 border border-gray-300 rounded-lg">

        </div>

        <div class="mb-4">

            <label for="message" class="block text-gray-700">Message</label>

            <textarea id="message" class="w-full p-2 border border-gray-300 rounded-lg"></textarea>

        </div>

        <button type="submit" class="bg-blue-500 text-white px-4 py-2 rounded-lg">Submit</button>

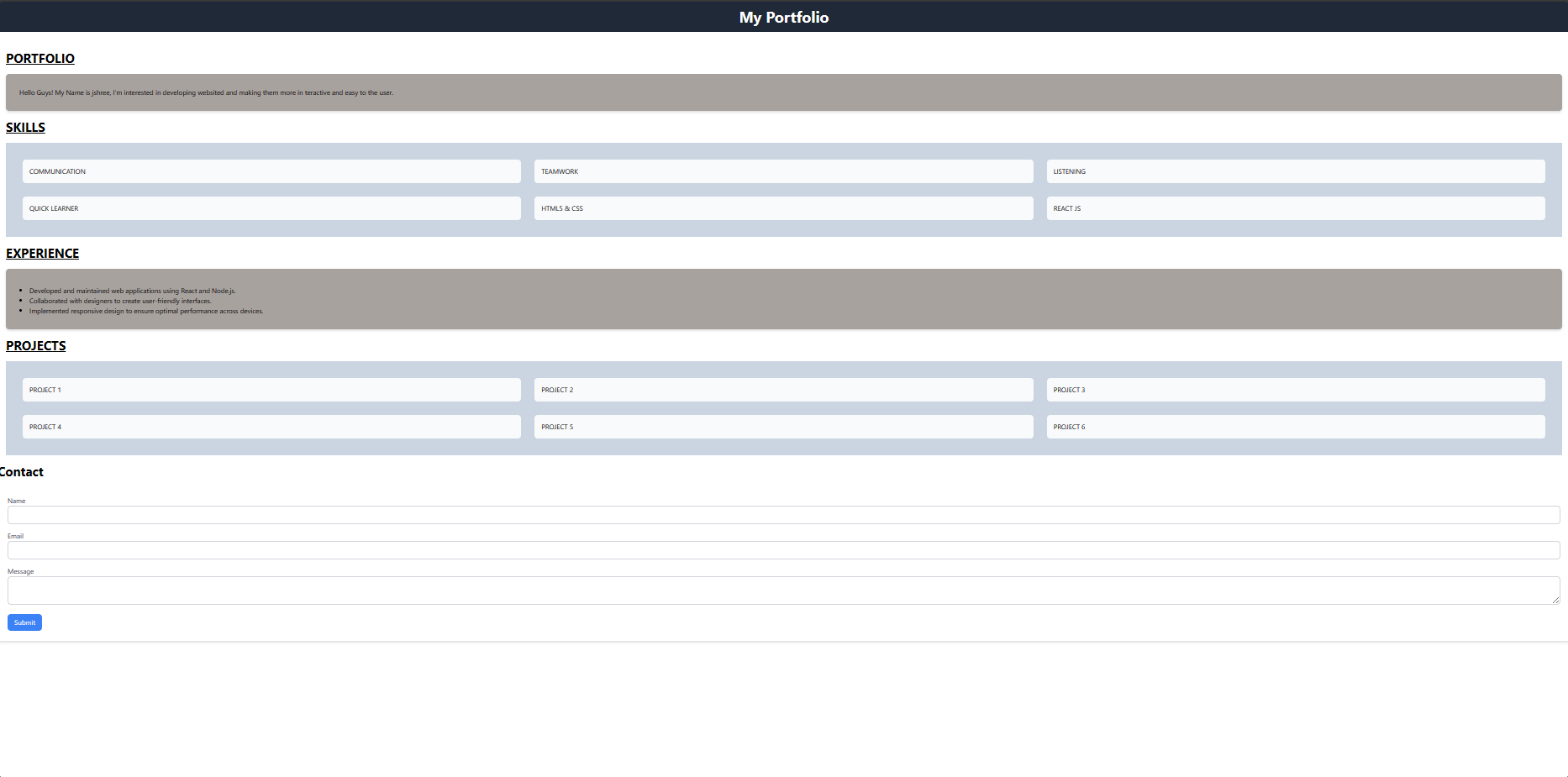
    </form>

</div>

</body>

</html>

**OUTPUT:**



**Problem Statement 2: E-commerce Product Page**

Design and develop an e-commerce product page using HTML5, CSS3, and Tailwind CSS. The page should display product information, including images, descriptions, prices, and options for adding to cart. Use Tailwind CSS to create a visually appealing and consistent layout.

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

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

    <link href="output.css" rel="stylesheet">

    <title>E-Shop</title>

    <style>

        body {

            font-family: Arial, sans-serif;

            background-color: #f0f4f3;

            margin: 0;

            padding: 0;

        }

        header {

            background-color: #f8b7d4;

            padding: 20px;

            text-align: center;

            color: white;

            font-size: 24px;

        }

        .container {

            display: flex;

            justify-content: space-around;

            margin: 40px;

        }

        .product-image {

            width: 50%;

        }

        .product-image img {

            max-width: 100%;

            height: auto;

            border-radius: 8px;

        }

        .product-details {

            width: 40%;

            padding-left: 20px;

        }

        .product-title {

            font-size: 24px;

            font-weight: bold;

        }

        .product-description {

            margin-top: 10px;

            font-size: 14px;

            color: #777;

        }

        .product-price {

            margin-top: 20px;

            font-size: 22px;

            color: #333;

        }

        .quantity {

            margin-top: 15px;

        }

        .quantity input {

            width: 50px;

            padding: 5px;

            margin-right: 10px;

        }

        .add-to-cart {

            margin-top: 20px;

        }

        .add-to-cart button {

            background-color: #008cba;

            color: white;

            padding: 10px 20px;

            border: none;

            cursor: pointer;

            font-size: 16px;

        }

        .additional-images {

            display: flex;

            margin-top: 20px;

        }

        .additional-images img {

            width: 100px;

            margin-right: 10px;

            border-radius: 8px;

        }

        footer {

            text-align: center;

            background-color: #333;

            color: white;

            padding: 20px;

            margin-top: 50px;

        }

    </style>

</head>

<body>

<header>

    E-Shop

</header>

<div class="container">

    <div class="product-image">

        <img src="<https://encrypted-tbn0.gstatic.com/images?q=tbn:ANd9GcTLgC2NeYGimWduD-7i9XiezTvyAclcEHiwQQ&s>" alt="Product Image">

    </div>

    <div class="product-details">

        <div class="product-title">Product Title</div>

        <div class="product-description">

            This is a brief description of the product, highlighting its key features and benefits. It gives the customer an overview of what to expect.

        </div>

        <div class="product-price">Rs. 99.99</div>

        <div class="quantity">

            <label for="quantity">Quantity:</label>

            <input type="number" id="quantity" name="quantity" value="1" min="1">

        </div>

        <div class="add-to-cart">

            <button>Add to Cart</button>

        </div>

        <div class="additional-images">

            <img src="<https://i.etsystatic.com/26740374/r/il/ea0faf/4037447446/il_570xN.4037447446_jjrv.jpg>" alt="Additional Image 1">

            <img src="<https://img.freepik.com/free-photo/vintage-photo-camera-composition_23-2148913937.jpg>" alt="Additional Image 2">

            <img src="<https://petapixel.com/assets/uploads/2014/03/argus_c31-620x465.jpg>" alt="Additional Image 3">

        </div>

    </div>

</div>

<footer>

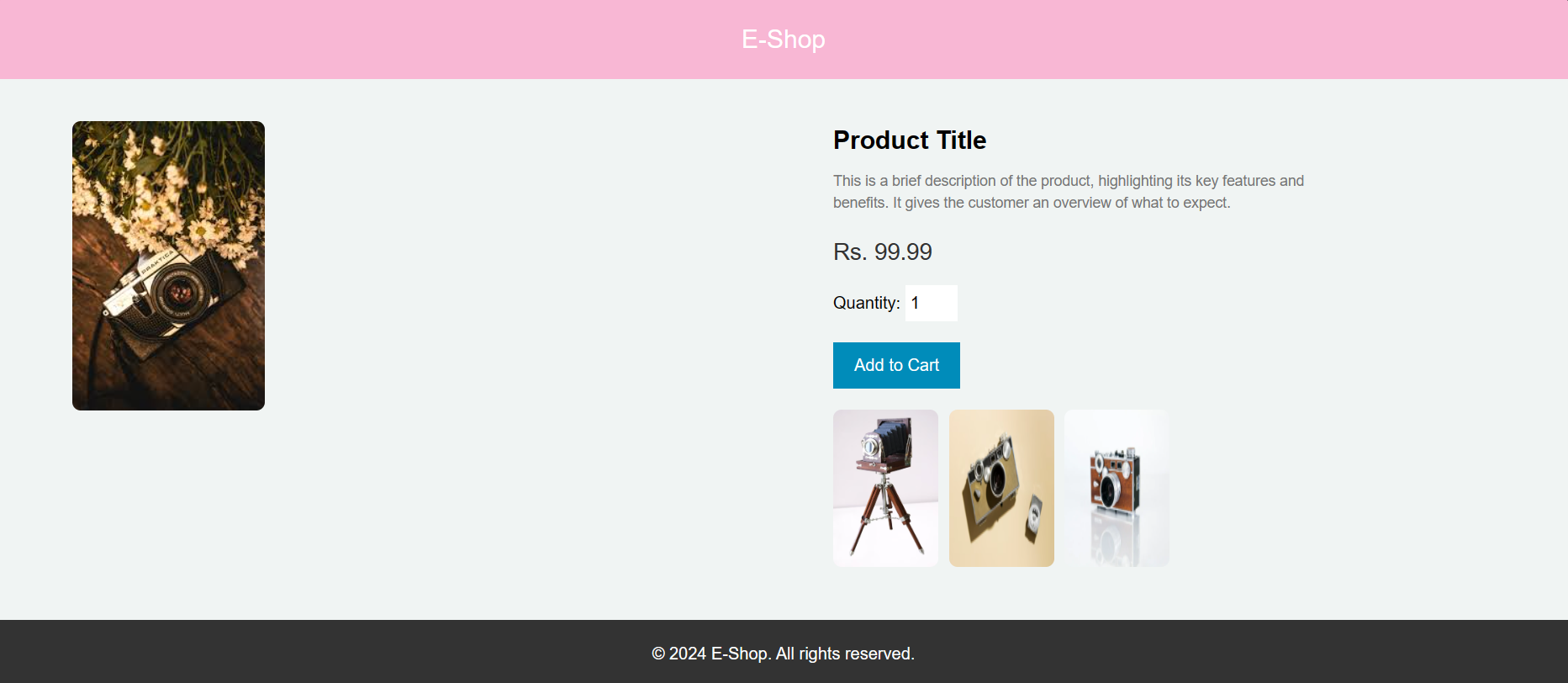
    © 2024 E-Shop. All rights reserved.

</footer>

</body>

</html>

**OUTPUT:**

****

**Problem Statement 3: Interactive Blog Post with Comments (3 hours)**

Create an interactive blog post with comments using HTML5, CSS3, and JavaScript. The blog post should include a title, author, content, and a comment section. Use JavaScript to enable users to submit comments and display them on the page.

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Interactive Blog Post</title>

<style>

/\* Basic reset \*/

\* {

margin: 0;

padding: 0;

box-sizing: border-box;

}

/\* Styling for the header \*/

header {

background-color: lightblue;

padding: 20px;

text-align: center;

}

/\* Styling for the main content area \*/

main {

padding: 20px;

margin: 20px;

background-color: lightgray;

}

/\* Styling for the comments section \*/

#comments {

margin-top: 30px;

}

#commentList {

list-style-type: none;

margin-bottom: 20px;

}

#commentList li {

background-color: white;

padding: 10px;

margin-bottom: 10px;

border: 1px solid #ddd;

border-radius: 5px;

}

form {

margin-top: 20px;

}

label {

font-weight: bold;

}

input, textarea {

width: 100%;

padding: 10px;

margin-top: 5px;

margin-bottom: 20px;

border: 1px solid #ddd;

border-radius: 5px;

}

button {

padding: 10px 20px;

background-color: lightblue;

border: none;

border-radius: 5px;

cursor: pointer;

}

button:hover {

background-color: deepskyblue;

}

/\* Styling for the footer \*/

footer {

text-align: center;

padding: 10px;

background-color: #333;

color: white;

}

</style>

</head>

<body>

<header>

<h1>My Blog</h1>

</header>

<main>

<article>

<h2>Web Technologies</h2>

<p><strong>Author:</strong> ABCDE</p>

<p>Welcome to the world of Web Technologies, where we design and create beautiful and interactive websites.</p>

</article>

<section id="comments">

<h3>Comments</h3>

<ul id="commentList">

<!-- Comments will be appended here by JavaScript -->

</ul>

<form id="commentForm" onsubmit="return addComment()">

<label for="name">Name:</label><br>

<input type="text" id="name" name="name" required><br><br>

<label for="comment">Comment:</label><br>

<textarea id="comment" name="comment" rows="4" required></textarea><br><br>

<button type="submit">Submit Comment</button>

</form>

</section>

</main>

<footer>

<p>&copy; 2024 My Blog</p>

</footer>

<script>

function addComment() {

// Get the values from the form

var name = document.getElementById('name').value;

var comment = document.getElementById('comment').value;

// Validate non-empty fields

if (name !== "" && comment !== "") {

// Create a new list item for the comment

var li = document.createElement('li');

li.innerHTML = '<strong>' + name + '</strong>: ' + comment;

// Add the new comment to the list

var commentList = document.getElementById('commentList');

commentList.appendChild(li);

// Clear the form fields

document.getElementById('name').value = '';

document.getElementById('comment').value = '';

}

// Prevent form submission

return false;

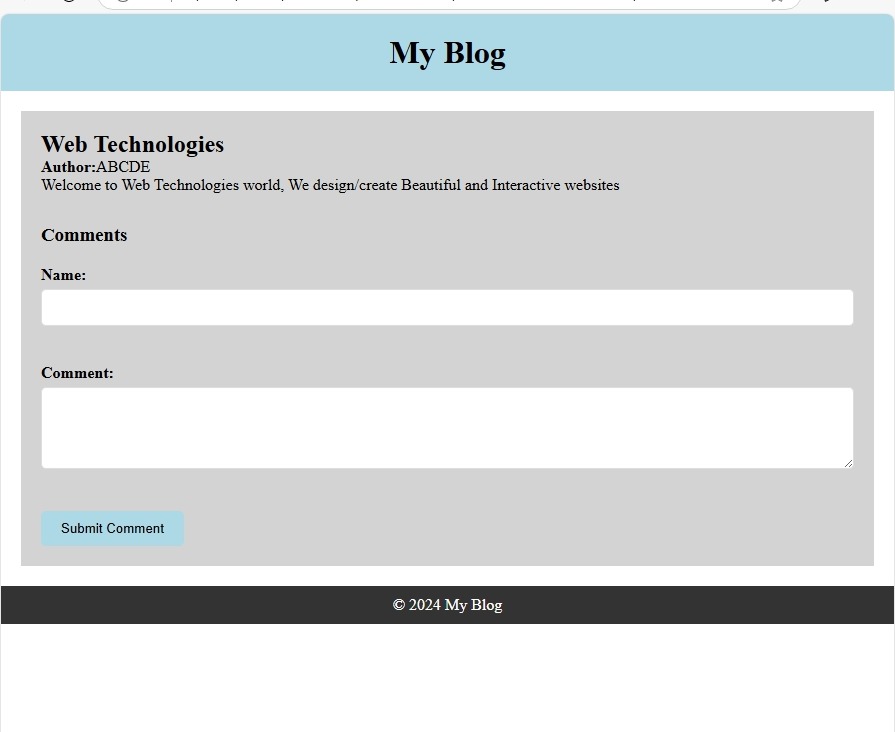
}

</script>

</body>

</html>

**OUTPUT:**

****

**Problem Statement 4: Adaptive Landing Page for Different Devices (2 hours)**

Develop an adaptive landing page that adjusts its layout and content based on the user's device. Use HTML5, CSS3, and JavaScript to detect the device type and display the appropriate content. Employ media queries and responsive design techniques to ensure the page looks great on all devices.

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Adaptive Landing Page</title>

<style>

/\* Basic reset \*/

\* {

margin: 0;

padding: 0;

box-sizing: border-box;

}

/\* Styling for the header \*/

header {

background-color: lightcoral;

color: white;

padding: 20px;

text-align: center;

}

/\* Styling for the main content \*/

main {

padding: 20px;

background-color: #f9f9f9;

}

/\* Styling for the hero section \*/

#hero {

text-align: center;

margin-bottom: 20px;

}

#heroTitle {

font-size: 2em;

margin-bottom: 10px;

}

#heroDescription {

font-size: 1.2em;

}

/\* Styling for the features section \*/

#features {

margin-bottom: 20px;

}

#featureList {

display: flex;

flex-wrap: wrap;

gap: 10px;

}

.feature {

background-color: white;

padding: 20px;

border: 1px solid #ddd;

border-radius: 5px;

flex: 1;

}

/\* Styling for the contact form \*/

#contactForm {

display: flex;

flex-direction: column;

}

#contactForm label {

margin-top: 10px;

}

#contactForm input,

#contactForm textarea {

margin-top: 5px;

padding: 10px;

border: 1px solid #ddd;

border-radius: 5px;

}

#contactForm button {

margin-top: 10px;

padding: 10px 20px;

background-color: lightcoral;

border: none;

border-radius: 5px;

color: white;

cursor: pointer;

}

#contactForm button:hover {

background-color: darkred;

}

/\* Styling for the footer \*/

footer {

text-align: center;

padding: 10px;

background-color: #333;

color: white;

}

/\* Media Queries for responsiveness \*/

/\* For tablets and larger devices \*/

@media (min-width: 768px) {

#featureList {

flex-direction: row;

}

}

/\* For desktops and larger devices \*/

@media (min-width: 1024px) {

#hero {

text-align: left;

}

#heroTitle {

font-size: 2.5em;

}

#heroDescription {

font-size: 1.5em;

}

}

</style>

</head>

<body>

<header>

<h1>Welcome to Our Service</h1>

<p id="deviceType"></p>

</header>

<main>

<section id="hero">

<h2 id="heroTitle">Discover Our Solutions</h2>

<p id="heroDescription">We provide top-notch solutions tailored to your needs.</p>

</section>

<section id="features">

<h2>Features</h2>

<div id="featureList">

<!-- Features will be dynamically inserted here -->

</div>

</section>

<section id="contact">

<h2>Contact Us</h2>

<p>Get in touch with us for more information.</p>

<form id="contactForm">

<label for="name">Name:</label>

<input type="text" id="name" name="name" required>

<label for="email">Email:</label>

<input type="email" id="email" name="email" required>

<label for="message">Message:</label>

<textarea id="message" name="message" rows="4" required></textarea>

<button type="submit">Send Message</button>

</form>

</section>

</main>

<footer>

<p>&copy; 2024 Our Service</p>

</footer>

<script>

// Function to detect device type

function detectDevice() {

var deviceTypeElement = document.getElementById('deviceType');

var featureList = document.getElementById('featureList');

if (window.innerWidth < 768) {

deviceTypeElement.textContent = "You are using a mobile device.";

// Example mobile features

featureList.innerHTML = `

<div class="feature">Mobile Feature 1</div>

<div class="feature">Mobile Feature 2</div>

`;

} else if (window.innerWidth < 1024) {

deviceTypeElement.textContent = "You are using a tablet.";

// Example tablet features

featureList.innerHTML = `

<div class="feature">Tablet Feature 1</div>

<div class="feature">Tablet Feature 2</div>

<div class="feature">Tablet Feature 3</div>

`;

} else {

deviceTypeElement.textContent = "You are using a desktop.";

// Example desktop features

featureList.innerHTML = `

<div class="feature">Desktop Feature 1</div>

<div class="feature">Desktop Feature 2</div>

<div class="feature">Desktop Feature 3</div>

<div class="feature">Desktop Feature 4</div>

`;

}

}

// Detect device type on load and resize

window.onload = detectDevice;

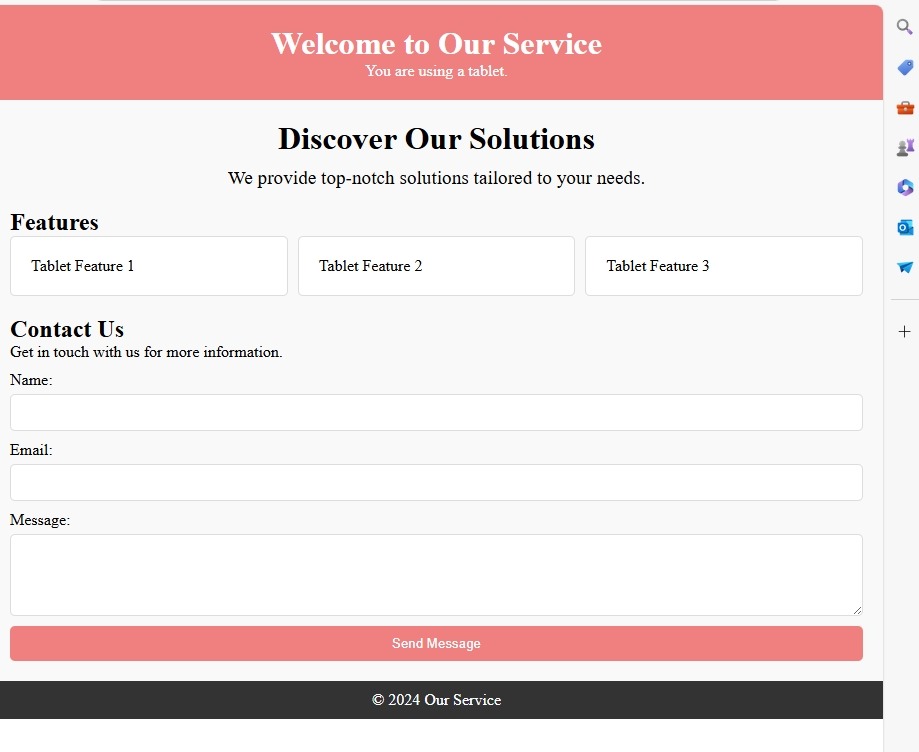
window.onresize = detectDevice;

</script>

</body>

</html>

**OUTPUT:**



**Cycle 2:**

**Problem Statement 1: Dynamically Generated Content with JavaScript**

Create a web page that dynamically generates content using JavaScript. The page should include a button that, when clicked, generates a new random number and displays it on the page. Use JavaScript to manipulate the Document Object Model (DOM) to add and remove elements.

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Random Number Generator</title>

<style>

/\* Basic reset \*/

\* {

margin: 0;

padding: 0;

box-sizing: border-box;

}

/\* Styling for the header \*/

header {

background-color: lightcoral;

color: white;

padding: 20px;

text-align: center;

}

/\* Styling for the main content \*/

main {

padding: 20px;

text-align: center;

background-color: #f9f9f9;

}

/\* Styling for the button \*/

button {

padding: 10px 20px;

background-color: lightcoral;

border: none;

border-radius: 5px;

color: white;

font-size: 16px;

cursor: pointer;

margin: 10px;

}

button:hover {

background-color: darkred;

}

/\* Styling for the number display \*/

#numberDisplay {

margin-top: 20px;

font-size: 24px;

font-weight: bold;

}

/\* Styling for the list \*/

ul {

list-style: none;

padding: 0;

}

ul li {

font-size: 18px;

margin: 5px 0;

}

/\* Styling for the footer \*/

footer {

text-align: center;

padding: 10px;

background-color: #333;

color: white;

}

</style>

</head>

<body>

<header>

<h1>Random Number Generator</h1>

</header>

<main>

<button id="generateButton" onclick="generateRandomNumber()">Generate Random Number</button>

<div id="numberDisplay">

<!-- Random number will be displayed here -->

</div>

<button id="addItemButton" onclick="addItem()">Add Item</button>

<button id="removeItemButton" onclick="removeItem()">Remove Last Item</button>

<ul id="itemList">

<!-- List items will be added and removed here -->

</ul>

</main>

<footer>

<p>&copy; 2024 Random Number Generator</p>

</footer>

<script>

// Function to generate and display a random number

function generateRandomNumber() {

// Generate a random number between 1 and 100

var randomNumber = Math.floor(Math.random() \* 100) + 1;

// Get the numberDisplay div

var numberDisplay = document.getElementById('numberDisplay');

// Clear any existing content

numberDisplay.innerHTML = '';

// Create a new paragraph element

var paragraph = document.createElement('p');

paragraph.innerHTML = 'Random Number: ' + randomNumber;

// Append the paragraph to the numberDisplay div

numberDisplay.appendChild(paragraph);

}

// Function to add a new item to the list

function addItem() {

// Get the itemList element

var itemList = document.getElementById('itemList');

// Create a new list item element

var newItem = document.createElement('li');

newItem.innerHTML = 'Item ' + (itemList.children.length + 1);

// Append the new item to the itemList

itemList.appendChild(newItem);

}

// Function to remove the last item from the list

function removeItem() {

// Get the itemList element

var itemList = document.getElementById('itemList');

// Remove the last item if the list is not empty

if (itemList.children.length > 0) {

itemList.removeChild(itemList.lastElementChild);

}

}

</script>

</body>

</html>

**OUTPUT:**



**Problem Statement 2: Interactive Shopping Cart with JavaScript**

Develop an interactive shopping cart using JavaScript. The cart should allow users to add and remove items, update quantities, and calculate the total price. Utilize JavaScript arrays and objects to store product information and manage cart operations.

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Shopping Cart</title>

<style>

/\* Basic reset \*/

\* {

margin: 0;

padding: 0;

box-sizing: border-box;

}

/\* Styling for the header \*/

header {

background-color: lightcoral;

color: white;

padding: 20px;

text-align: center;

}

/\* Styling for the main content \*/

main {

padding: 20px;

display: flex;

justify-content: space-around;

background-color: #f9f9f9;

}

/\* Styling for product and cart sections \*/

#product-list, #cart {

width: 45%;

}

.product, #cartItems li {

background-color: white;

padding: 10px;

margin-bottom: 10px;

border: 1px solid #ddd;

border-radius: 5px;

}

button {

padding: 10px;

background-color: lightcoral;

border: none;

border-radius: 5px;

color: white;

cursor: pointer;

}

button:hover {

background-color: darkred;

}

/\* Styling for the total price \*/

#totalPrice {

font-weight: bold;

margin-top: 20px;

}

/\* Styling for the footer \*/

footer {

text-align: center;

padding: 10px;

background-color: #333;

color: white;

}

</style>

</head>

<body>

<header>

<h1>Sample Shopping Cart</h1>

</header>

<main>

<section id="product-list">

<h2>Products</h2>

<div class="product">

<span>Product 1</span>

<button onclick="addToCart('Product 1', 10)">Add to Cart (Rs.10)</button>

</div>

<div class="product">

<span>Product 2</span>

<button onclick="addToCart('Product 2', 15)">Add to Cart (Rs.15)</button>

</div>

<div class="product">

<span>Product 3</span>

<button onclick="addToCart('Product 3', 20)">Add to Cart (Rs.20)</button>

</div>

</section>

<section id="cart">

<h2>Your Cart</h2>

<ul id="cartItems">

<!-- Cart items will be added here -->

</ul>

<p id="totalPrice">Total Price: Rs.0</p>

</section>

</main>

<footer>

<p>&copy; 2024 Shopping Cart Example</p>

</footer>

<script>

// Array to hold cart items

let cart = [];

// Function to add item to cart

function addToCart(name, price) {

// Check if item already exists in cart

let item = cart.find(item => item.name === name);

if (item) {

// If item exists, increase quantity

item.quantity += 1;

} else {

// If item does not exist, add new item

cart.push({ name, price, quantity: 1 });

}

updateCart();

}

// Function to remove item from cart

function removeFromCart(name) {

// Filter out the item to be removed

cart = cart.filter(item => item.name !== name);

updateCart();

}

// Function to update item quantity

function updateQuantity(name, quantity) {

let item = cart.find(item => item.name === name);

if (item) {

item.quantity = parseInt(quantity);

if (item.quantity <= 0) {

removeFromCart(name);

} else {

updateCart();

}

}

}

// Function to update the cart display

function updateCart() {

let cartItems = document.getElementById('cartItems');

cartItems.innerHTML = ''; // Clear existing items

// Add each item to the cart display

cart.forEach(item => {

let li = document.createElement('li');

li.innerHTML = `${item.name} - Rs.${item.price} x ${item.quantity}

<button onclick="removeFromCart('${item.name}')">Remove</button>

<input type="number" value="${item.quantity}" min="1"

onchange="updateQuantity('${item.name}', this.value)">`;

cartItems.appendChild(li);

});

// Update total price

let totalPrice = cart.reduce((total, item) => total + item.price \* item.quantity, 0);

document.getElementById('totalPrice').innerHTML = `Total Price: Rs.${totalPrice}`;

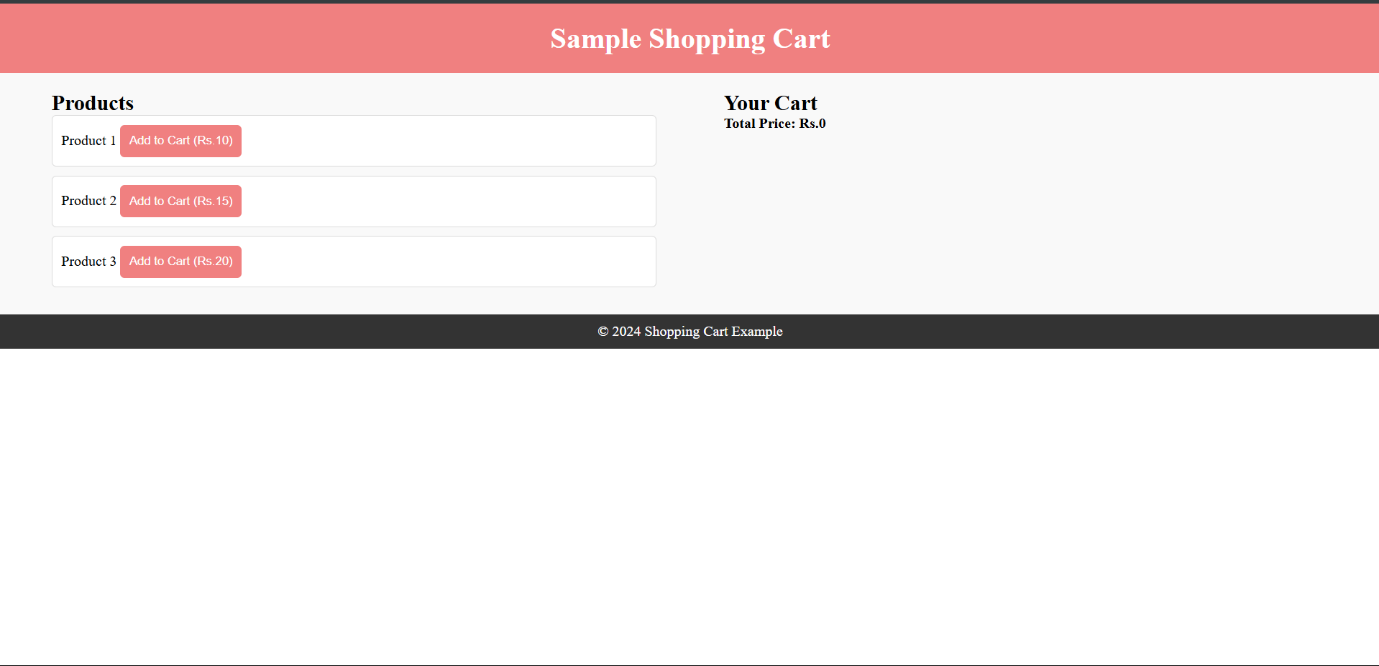
}

</script>

</body>

</html>

**OUTPUT:**



**Problem Statement 3: Regular Expression-Based Text Manipulation**

Build a web application that performs text manipulation using regular expressions. The application should allow users to enter a text string and provide options for search, replace, and formatting. Implement regular expression patterns to identify and modify specific text elements.

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Text Manipulation with Regex</title>

<style>

/\* Basic reset \*/

\* {

margin: 0;

padding: 0;

box-sizing: border-box;

}

/\* Styling for the header \*/

header {

background-color: lightcoral;

color: white;

padding: 20px;

text-align: center;

}

/\* Styling for the main content \*/

main {

padding: 20px;

text-align: center;

background-color: #f9f9f9;

}

/\* Styling for the textarea and input fields \*/

textarea, input {

margin-top: 10px;

padding: 10px;

border: 1px solid #ddd;

border-radius: 5px;

width: 100%;

}

button {

padding: 10px 20px;

background-color: lightcoral;

border: none;

border-radius: 5px;

color: white;

cursor: pointer;

margin: 10px;

}

button:hover {

background-color: darkred;

}

/\* Styling for the footer \*/

footer {

text-align: center;

padding: 10px;

background-color: #333;

color: white;

}

</style>

</head>

<body>

<header>

<h1>Text Manipulation with Regular Expressions</h1>

</header>

<main>

<div class="container">

<textarea id="inputText" rows="10" cols="50" placeholder="Enter text here..."></textarea>

<br>

<label for="searchPattern">Search Pattern:</label>

<input type="text" id="searchPattern">

<br>

<label for="replacePattern">Replace With:</label>

<input type="text" id="replacePattern">

<br>

<button onclick="performSearch()">Search</button>

<button onclick="performReplace()">Replace</button>

<button onclick="performFormatting()">Format</button>

<br><br>

<h2>Result</h2>

<textarea id="resultText" rows="10" cols="50" readonly></textarea>

</div>

</main>

<footer>

<p>&copy; 2024 Text Manipulation Example</p>

</footer>

<script>

// Function to perform search using regex

function performSearch() {

const inputText = document.getElementById('inputText').value;

const searchPattern = document.getElementById('searchPattern').value;

const resultText = document.getElementById('resultText');

try {

const regex = new RegExp(searchPattern, 'g');

const matches = inputText.match(regex);

resultText.value = matches ? matches.join('\n') : 'No matches found.';

} catch (e) {

resultText.value = 'Invalid regular expression.';

}

}

// Function to perform replace using regex

function performReplace() {

const inputText = document.getElementById('inputText').value;

const searchPattern = document.getElementById('searchPattern').value;

const replacePattern = document.getElementById('replacePattern').value;

const resultText = document.getElementById('resultText');

try {

const regex = new RegExp(searchPattern, 'g');

const replacedText = inputText.replace(regex, replacePattern);

resultText.value = replacedText;

} catch (e) {

resultText.value = 'Invalid regular expression.';

}

}

// Function to perform formatting (e.g., capitalizing first letter of each word)

function performFormatting() {

const inputText = document.getElementById('inputText').value;

const resultText = document.getElementById('resultText');

// Example formatting: Capitalize the first letter of each word

const formattedText = inputText.replace(/\b\w/g, char => char.toUpperCase());

resultText.value = formattedText;

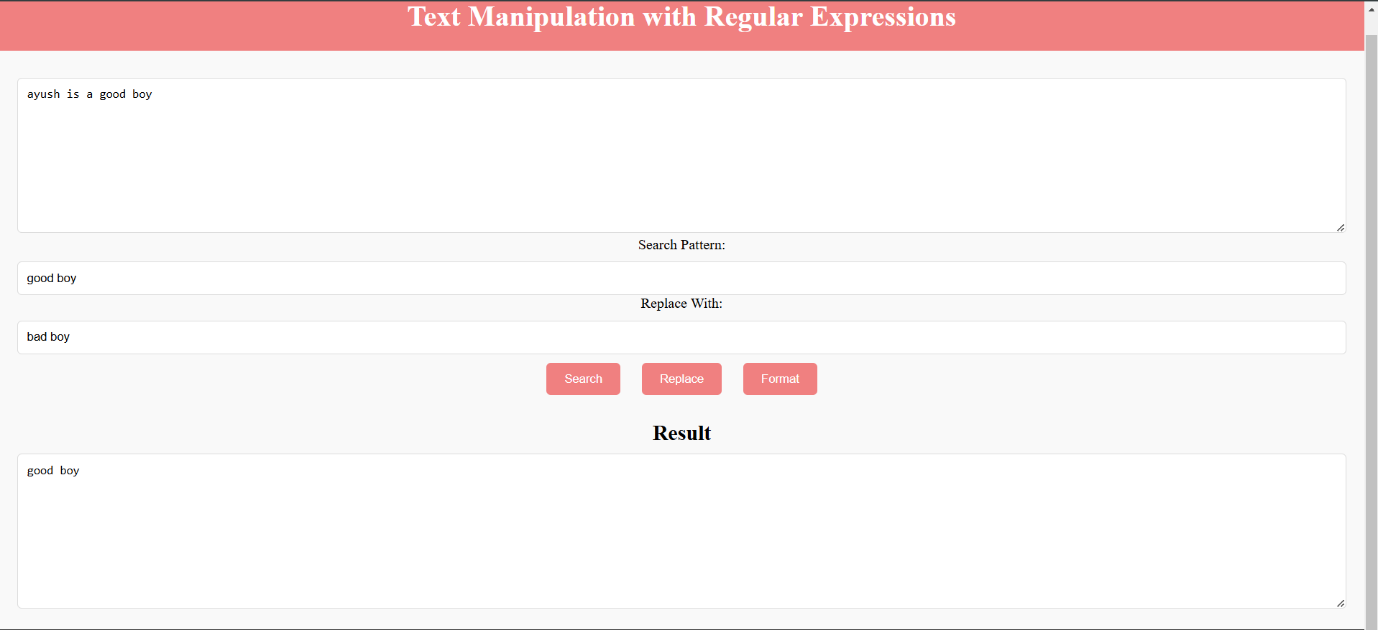
}

</script>

</body>

</html>

**OUTPUT:**



**Problem Statement 4: Asynchronous Data Fetching and Display with JavaScript Promises and Async/await**

Create a web page that fetches data from an API asynchronously using JavaScript promises and Async/await. The page should display a loading indicator while the data is being fetched and then render the data in a list or table. Demonstrate the use of promises to handle asynchronous operations and improve code readability.

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Async Data Fetching</title>

<style>

/\* Basic reset \*/

\* {

margin: 0;

padding: 0;

box-sizing: border-box;

}

/\* Styling for the header \*/

header {

background-color: lightcoral;

color: white;

padding: 20px;

text-align: center;

}

/\* Styling for the main content \*/

main {

padding: 20px;

text-align: center;

background-color: #f9f9f9;

}

/\* Styling for the loading indicator \*/

#loading {

font-size: 20px;

font-weight: bold;

margin-bottom: 20px;

}

/\* Styling for the table \*/

table {

width: 100%;

border-collapse: collapse;

}

th, td {

border: 1px solid #ddd;

padding: 10px;

text-align: left;

}

th {

background-color: lightcoral;

color: white;

}

/\* Styling for the footer \*/

footer {

text-align: center;

padding: 10px;

background-color: #333;

color: white;

}

</style>

</head>

<body>

<header>

<h1>Async Data Fetching Example</h1>

</header>

<main>

<div id="loading">Loading...</div>

<table id="dataTable">

<thead>

<tr>

<th>ID</th>

<th>Name</th>

<th>Email</th>

</tr>

</thead>

<tbody id="dataBody">

<!-- Data rows will be added here -->

</tbody>

</table>

</main>

<footer>

<p>&copy; 2024 Async Data Fetching Example</p>

</footer>

<script>

// Function to fetch data from the API

async function fetchData() {

try {

// Show loading indicator

document.getElementById('loading').style.display = 'block';

// Fetch data from the API

const response = await fetch('https://jsonplaceholder.typicode.com/users');

// Check if the response is ok

if (!response.ok) {

throw new Error('Network response was not ok');

}

// Parse the JSON data

const data = await response.json();

// Render the data in the table

renderData(data);

} catch (error) {

// Handle errors

console.error('There was a problem with the fetch operation:', error);

} finally {

// Hide loading indicator

document.getElementById('loading').style.display = 'none';

}

}

// Function to render data in the table

function renderData(data) {

const dataBody = document.getElementById('dataBody');

// Clear existing rows

dataBody.innerHTML = '';

// Add rows for each item

data.forEach(user => {

const row = document.createElement('tr');

row.innerHTML = `

<td>${user.id}</td>

<td>${user.name}</td>

<td>${user.email}</td>

`;

dataBody.appendChild(row);

});

}

// Call fetchData to get the data and render it

fetchData();

</script>

</body>

</html>

**OUTPUT:**

