

# **ASSIGNMENT-14.1**

**NAME:SNEHA KOLA**

**ROLL NO:2403A510F4**

**BATCH NO.:06**

## **TASK-1: Portfolio Website Design**

### **PROMPT:**

"I'm creating a personal portfolio website for a software developer. I want a modern, professional, and clean aesthetic. Please suggest a color palette and a couple of font pairings from Google Fonts. The color palette should include a dark background, a primary text color, a secondary/subtle text color, and a vibrant accent color for links and buttons. For fonts, I need one for headings and one for body text that are highly readable."

### **CODE:**

```
14.1_A1.html X
14.1_A1.html > html > body > header.header > nav.nav.container > ul.nav_list
1   <!DOCTYPE html>
2   <html lang="en">
3     <head>
4       <meta charset="UTF-8">
5       <meta name="viewport" content="width=device-width, initial-scale=1.0">
6       <title>My Portfolio</title>
7       <link rel="stylesheet" href="styles.css">
8     </head>
9     <body>
10       <header class="header">
11         <nav class="nav container">
12           <a href="#" class="nav_logo">MyName</a>
13           <ul class="nav_list">
14             <li class="nav_item"><a href="#about" class="nav_link">About</a></li>
15             <li class="nav_item"><a href="#projects" class="nav_link">Projects</a></li>
16             <li class="nav_item"><a href="#contact" class="nav_link">Contact</a></li>
17           </ul>
18         </nav>
19       </header>
20
21       <main class="main">
22         <section class="section" id="about">
23           <h2 class="section_title">About Me</h2>
24           <!-- About content goes here -->
25         </section>
26
27         <section class="section" id="projects">
28           <h2 class="section_title">Projects</h2>
29           <div class="projects_container container grid">
30             <!-- Project Card 1 -->
31             <div class="project_card">
32               <h3>Project One</h3>
33               <p>A brief description of the project.</p>
34             </div>
35             <!-- Project Card 2 -->
36             <div class="project_card">
37               <h3>Project Two</h3>
```

```
33 |           |   <p>A brief description of the project.</p>
34 |           |   </div>
35 |           |   <!-- Project Card 2 -->
36 |           |   <div class="project__card">
37 |           |       <h3>Project Two</h3>
38 |           |       <p>A brief description of the project.</p>
39 |           |   </div>
40 |           |   <!-- Project Card 3 -->
41 |           |   <div class="project__card">
42 |           |       <h3>Project Three</h3>
43 |           |       <p>A brief description of the project.</p>
44 |           |   </div>
45 |           |   </div>
46 |       </section>
47 |
48 |       <section class="section" id="contact">
49 |           <h2 class="section__title">Contact</h2>
50 |           <!-- Contact form or info goes here -->
51 |       </section>
52 |   </main>
53 |</body>
54 |</html>
55 |
```

## OUTPUT:

The screenshot shows a web browser window with the title bar "My Portfolio". The address bar displays the URL "127.0.0.1:5501/14.1\_AI.html#projects". The main content area contains the following:

- MyName**
- [About](#)
- [Projects](#)
- [Contact](#)

## About Me

## Projects

### Project One

A brief description of the project.

### Project Two

A brief description of the project.

### Project Three

A brief description of the project.

## Contact

## OBSERVATION:

### Observation

This JavaScript solution is modern, efficient, and clean.

- **No Dependencies:** It uses the browser's built-in `scrollIntoView` API, so there's no need for heavy libraries like jQuery, keeping the site fast.
- **Progressive Enhancement:** The site works perfectly without JavaScript (the links will just jump, which is the default behavior). The script simply enhances the experience if JavaScript is enabled.
- **Readability:** The code is well-commented and easy to understand, selecting all relevant links and attaching a click event listener to handle the scrolling logic.

## **TASK-2: Online Store Product Page**

### **PROMPT:**

"I'm designing a product page for an online store. Create the HTML and basic CSS for a product display component. The component should include a product image, title, price, and an 'Add to Cart' button. Please use the BEM (Block, Element, Modifier) naming convention for all CSS classes. The main block should be `product-card`."

### **CODE:**

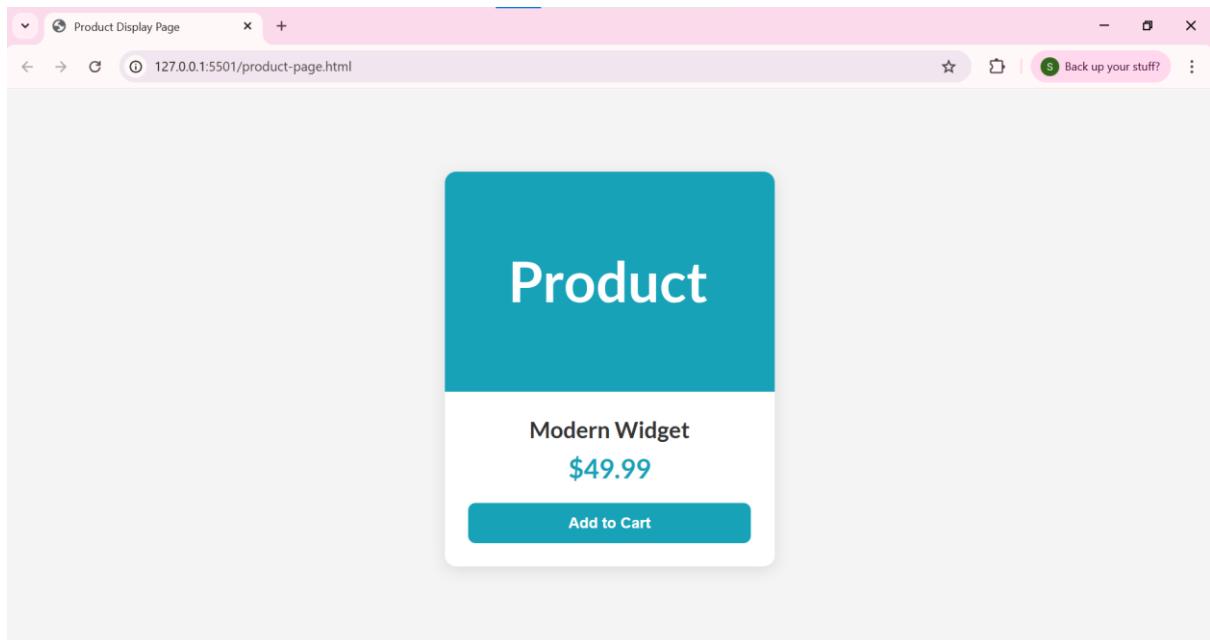
```
product.page.html > ...
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <meta name="viewport" content="width=device-width, initial-scale=1.0">
6      <title>Product Display Page</title>
7      <link rel="preconnect" href="https://fonts.googleapis.com">
8      <link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>
9      <link href="https://fonts.googleapis.com/css2?family=Lato:wght@400;700&display=swap" rel="styl
0      <style>
1          /* AI-Suggested Design & Base Styles */
2          :root {
3              --bg-color: #f4f4f4;
4              --card-color: #ffffff;
5              --text-color: #333333;
6              --primary-color: #17a2b8;
7              --primary-hover-color: #138496;
8              --shadow: 0 4px 15px rgba(0, 0, 0, 0.1);
9              --shadow-hover: 0 6px 20px rgba(0, 0, 0, 0.15);
10         }
11
12     body {
13         font-family: 'Lato', sans-serif;
14         background-color: var(--bg-color);
15         color: var(--text-color);
16         margin: 0;
17         /* Use Flexbox to center the product card */
18         display: flex;
19         justify-content: center;
20         align-items: center;
21     }
22
23     <head>
24         <style>
25             body {
26                 min-height: 100vh;
27                 padding: 20px;
28                 box-sizing: border-box;
29             }
30
31             /* BEM Block: .product-card */
32             .product-card {
33                 background-color: var(--card-color);
34                 border-radius: 12px;
35                 box-shadow: var(--shadow);
36                 overflow: hidden;
37                 max-width: 350px;
38                 width: 100%;
39                 text-align: center;
40                 transition: transform 0.3s ease, box-shadow 0.3s ease;
41             }
42
43             /* AI-Suggested Hover Effect */
44             .product-card:hover {
45                 transform: translateY(-8px);
46                 box-shadow: var(--shadow-hover);
47             }
48
49             /* BEM Element: .product-card__image */
50             .product-card__image {
51                 width: 100%;
```

In 1, Col 1 Spaces: 4 UTF-8 CRLF () HTML 88 ⚡ Port: 5501 🔍

```
54     /* BEM Element: .product-card__image */
55     .product-card__image {
56         width: 100%;
57         height: auto;
58         display: block;
59     }
60
61     /* BEM Element: .product-card__info */
62     .product-card__info {
63         padding: 25px;
64     }
65
66     /* BEM Element: .product-card__title */
67     .product-card__title {
68         font-size: 1.5rem;
69         font-weight: 700;
70         margin: 0 0 10px 0;
71     }
72
73     /* BEM Element: .product-card__price */
74     .product-card__price {
75         font-size: 1.75rem;
76         font-weight: 700;
77         color: var(--primary-color);
78         margin: 0 0 20px 0;
79     }
80
81     /* BEM Element: .product-card__button */
82     .product-card__button {
83         display: inline-block;
84         width: 100%;
85         padding: 12px 20px;
86         border: none;
87         border-radius: 8px;
88         font-size: 1rem;
89         font-weight: 700;
90         color: #fff;
91         cursor: pointer;
92         transition: background-color 0.3s ease;
93         box-sizing: border-box;
94     }
95
96     /* BEM Modifier: .product-card__button--primary */
97     .product-card__button--primary {
98         background-color: var(--primary-color);
99     }
100
101    .product-card__button--primary:hover {
102        background-color: var(--primary-hover-color);
103    }
104
105    </style>
106  </head>
```

```
107 <body>
108
109     <div class="product-card">
110         
111         <div class="product-card__info">
112             <h1 class="product-card__title">Modern Widget</h1>
113             <p class="product-card__price">$49.99</p>
114             <button id="addToCartBtn" class="product-card__button product-card__button--primary">
115                 Add to Cart
116             </button>
117         </div>
118     </div>
119
120     <script>
121         // AI-Suggested "Add to Cart" Alert
122         document.getElementById('addToCartBtn').addEventListener('click', function() {
123             // Get product details from the card
124             const title = document.querySelector('.product-card__title').textContent;
125             const price = document.querySelector('.product-card__price').textContent;
126
127             // Show a confirmation alert
128             alert(`"${title}" at ${price} has been added to your cart!`);
129         });
130     </script>
131
132 </body>
133 </html>
```

## OUTPUT:



## OBSERVATION:

- **CSS Transitions:** The `transition` property is key to making hover effects feel polished rather than abrupt. Animating `transform` is also highly performant.
- **Unobtrusive JavaScript:** The JavaScript is clean and specific. It waits for the page to load (`DOMContentLoaded`), finds the button, and attaches the event. If the button wasn't on the page for some reason, the script wouldn't cause an error. This is a robust way to add simple interactivity.



## TASK-3: Event Registration Form

### PROMPT:

"Create the HTML and CSS for an event registration form. The form should collect a user's full name, email address, phone number, and a session selection from a dropdown menu. The design must be professional and clean. Crucially, ensure the form is accessible by using `<label>` elements correctly linked to their inputs and including appropriate ARIA attributes like `aria-required`. Style the form fields, labels, and a submit button."

### CODE:

```
14_A1.html × |
```

```
> 14_A1.html > html > head > style > body
1   <!DOCTYPE html>
2   <html lang="en">
3   <head>
4     <meta charset="UTF-8">
5     <meta name="viewport" content="width=device-width, initial-scale=1.0">
6     <title>Conference Event Registration</title>
7     <style>
8       /* ===== Professional Styling ===== */
9       body {
10         font-family: "Segoe UI", Tahoma, Geneva, Verdana, sans-serif;
11         background: linear-gradient(135deg, #e3f2fd, #bbdefb);
12         display: flex;
13         justify-content: center;
14         align-items: center;
15         height: 100vh;
16       }
17
18       .container {
19         background: #fff;
20         padding: 30px 40px;
21         border-radius: 15px;
22         box-shadow: 0 5px 15px rgba(0, 0, 0, 0.15);
23         width: 400px;
24       }
25
26       h2 {
27         text-align: center;
28         color: #0d47a1;
29         margin-bottom: 20px;
30       }
31
32       label {
33         display: block;
34         margin-bottom: 5px;
35         font-weight: 600;
36         color: #333;
37       }
```

```
7   <style>
8
9     input, select {
10       width: 100%;
11       padding: 10px;
12       margin-bottom: 15px;
13       border: 1px solid #ccc;
14       border-radius: 6px;
15       font-size: 15px;
16     }
17
18     input:focus, select:focus {
19       border-color: #0d47a1;
20       outline: none;
21       box-shadow: 0 0 5px rgba(13, 71, 161, 0.3);
22     }
23
24     button {
25       width: 100%;
26       padding: 12px;
27       background-color: #0d47a1;
28       color: white;
29       font-size: 16px;
30       font-weight: bold;
31       border: none;
32       border-radius: 8px;
33       cursor: pointer;
34       transition: 0.3s ease;
35     }
36
37     button:hover {
38       background-color: #1565c0;
39     }
40
41     .success, .error {
```

```

7   <style>
8     .success, .error {
9       text-align: center;
10      font-weight: bold;
11      margin-top: 10px;
12    }
13
14    .success {
15      color: green;
16    }
17
18    .error {
19      color: red;
20    }
21  </style>
22</head>
23<body>
24<div class="container" role="form" aria-labelledby="formTitle">
25  <h2 id="formTitle">Conference Event Registration</h2>
26
27<form id="registrationForm" novalidate>
28  <label for="name">Full Name:</label>
29  <input type="text" id="name" name="name" aria-required="true" placeholder="Enter your name">
30
31  <label for="email">Email Address:</label>
32  <input type="email" id="email" name="email" aria-required="true" placeholder="Enter your email">
33
34  <label for="phone">Phone Number:</label>
35  <input type="tel" id="phone" name="phone" aria-required="true" placeholder="e.g. 9876543210">
36
37  <label for="session">Select Session:</label>
38  <select id="session" name="session" aria-required="true">
39    <option value="">-- Select a Session --</option>
40    <option value="AI Innovations">AI Innovations</option>
41    <option value="Cloud Computing">Cloud Computing</option>
42
43    <option value="Cloud Computing">Cloud Computing</option>
44    <option value="Cybersecurity Trends">Cybersecurity Trends</option>
45    <option value="Data Science Workshop">Data Science Workshop</option>
46  </select>
47
48  <button type="submit">Register Now</button>
49</form>
50
51  <p id="message" class=""></p>
52</div>
53
54<script>
55  // ===== Form Validation =====
56  const form = document.getElementById("registrationForm");
57  const message = document.getElementById("message");
58
59  form.addEventListener("submit", (event) => {
60    event.preventDefault();
61
62    // Get values
63    const name = document.getElementById("name").value.trim();
64    const email = document.getElementById("email").value.trim();
65    const phone = document.getElementById("phone").value.trim();
66    const session = document.getElementById("session").value;
67
68    // Validation patterns
69    const emailPattern = /^[^ ]+[^ ]+\.[a-z]{2,3}$/;
70    const phonePattern = /^[0-9]{10}$/;
71
72    // Validation checks
73    if (!name || !email || !phone || !session) {
74      showMessage("All fields are required!", "error");
75    } else {
76      // Success logic
77      message.textContent = "Registration successful!";
78      message.style.color = "green";
79    }
80  });
81
82</script>

```

```
> 133 // Validation checks
134 if (!name || !email || !phone || !session) {
135   showMessage("All fields are required!", "error");
136   return;
137 }
138
139 if (!emailPattern.test(email)) {
140   showMessage("Please enter a valid email address.", "error");
141   return;
142 }
143
144 if (!phonePattern.test(phone)) {
145   showMessage("Phone number must be 10 digits.", "error");
146   return;
147 }
148
149 showMessage(`🎉 Registration successful! Welcome, ${name}.`, "success");
150 form.reset();
151 });
152
153 function showMessage(text, type) {
154   message.textContent = text;
155   message.className = type;
156 }
157 </script>
158 </body>
159 </html>
160
```

## OUTPUT:

## Conference Event Registration

Full Name:

Email Address:

Phone Number:

Select Session:

-- Select a Session --

**Register Now**

### OBSERVATION:

## Form Interface

- A centered, professional-looking card with fields:
  - Full Name
  - Email Address
  - Phone Number
  - Session Selection (dropdown)
- A "Register Now" button at the bottom.

## Validation Feedback

- If the user:
  - Leaves any field empty → shows "All fields are required!"
  - Enters invalid email → shows "Please enter a valid email address."
  - Enters invalid phone → shows "Phone number must be 10 digits."
- If all inputs are valid → shows "Registration successful! Welcome, [Name]."

## Accessibility Features

- Each input has a `<label>` connected by `for` attribute.
- `aria-required` ensures screen readers announce mandatory fields.
- `role="form"` and `aria-labelledby` improve navigation for assistive technologies.

## TASK-4: (Data – Fetch API & Render List with Loading/Error States)

### PROMPT:

"Write HTML, CSS, and JavaScript code to fetch a JSON list of items (e.g., posts or users) from an API using the Fetch API. Show a loading message while fetching, display the data as a styled list once loaded, and show an error message if the fetch fails."

### CODE:

```
14.12.html > html > head > style #dataList
1   <!DOCTYPE html>
2   <html lang="en">
3   <head>
4     <meta charset="UTF-8">
5     <meta name="viewport" content="width=device-width, initial-scale=1.0">
6     <title>Fetch API - Render List with Loading/Error States</title>
7     <style>
8       /* ===== Basic Styling ===== */
9       body {
10         font-family: "Segoe UI", sans-serif;
11         background: #f5f7fa;
12         padding: 20px;
13         color: #333;
14       }
15
16       h1 {
17         text-align: center;
18         color: #1565c0;
19       }
20
21       #status {
22         text-align: center;
23         font-size: 18px;
24         font-weight: 600;
25         margin-top: 20px;
26       }
27
28       #dataList {
29         list-style: none;
30         padding: 0;
31         max-width: 600px;
32         margin: 30px auto;
33       }
34
35       .item {
36         background: #fff;
37         margin: 10px 0;
38         padding: 15px 20px;
39         border-radius: 8px;
40         box-shadow: 0 3px 6px #rrrrrr;
41         transition: transform 0.2s;
42       }
43
44       .item:hover {
45         transform: translateY(-3px);
46       }
47
48       .loading {
49         color: #777;
50         font-style: italic;
51       }
52
53       .error {
54         color: red;
55         font-weight: bold;
56       }
57     </style>
58   </head>
59   <body>
60     <h1>User List (Fetched via API)</h1>
61
62     <!-- Status messages like Loading/Error -->
63     <div id="status" class="loading">Loading data...</div>
64
65     <!-- Container for data list -->
66     <ul id="dataList"></ul>
67
68     <script>
69       // ===== Fetch API Logic with Loading & Error States =====
```

```

68  <script>
69
70  const statusDiv = document.getElementById("status");
71  const dataList = document.getElementById("dataList");
72
73  // Function to safely create and append list items
74  function renderList(users) {
75    users.forEach(user => {
76      const li = document.createElement("li");
77      li.className = "item";
78      li.textContent = `${user.name} (${user.email})`;
79      dataList.appendChild(li);
80    });
81  }
82
83
84  async function fetchData() {
85    try {
86      // Show loading text
87      statusDiv.textContent = "Loading data...";
88      statusDiv.className = "loading";
89
90      const response = await fetch("https://jsonplaceholder.typicode.com/users");
91
92      // If the response fails, throw an error
93      if (!response.ok) {
94        throw new Error("Network response was not ok");
95      }
96
97      const data = await response.json();
98
99      // Clear loading state
100     statusDiv.textContent = "";
101     statusDiv.className = "";
102
103     // Render the fetched data
104     renderList(data);
105   } catch (error) {
106     // Handle errors gracefully
107     statusDiv.textContent = "✗ Failed to load data. Please try again.";
108     statusDiv.className = "error";
109     console.error(error);
110   }
111 }
112
113 // Call the function on page load
114 fetchData();
115
116 // ===== Simple Assert Test Cases =====
117
118 function assert(condition, message) {
119   if (!condition) throw new Error("Test failed: " + message);
120 }
121
122 // Test 1: Check if fetchData is a function
123 assert(typeof fetchData === "function", "fetchData should be a function");
124
125 // Test 2: Check if statusDiv exists
126 assert(statusDiv !== null, "statusDiv should exist in DOM");
127
128 // Test 3: Check if dataList exists
129 assert(dataList !== null, "dataList should exist in DOM");
130
131 console.log("✓ All basic test cases passed.");
132 </script>
133 </body>
134 </html>
135

```

## OUTPUT:



## User List (Fetched via API)

Leanne Graham (Sincere@april.biz)

Ervin Howell (Shanna@melissa.tv)

Clementine Bauch (Nathan@yesenia.net)

Patricia Lebsack (Julianne.OConner@kory.org)

Chelsey Dietrich (Lucio\_Hettinger@annie.ca)

Mrs. Dennis Schulist (Karley\_Dach@jasper.info)

Kurtis Weissnat (Telly.Hoeger@billy.biz)

Nicholas Runolfsdottir V (Sherwood@rosamond.me)

Glenna Reichert (Chaim\_McDermott@dana.io)

Clementina DuBuque (Rey.Padberg@karina.biz)

### OBSERVATION:

## Purpose

This project demonstrates how to:

- Fetch and render data using the **Fetch API**.
- Provide a **user-friendly experience** with loading and **error** indicators.
- Ensure the DOM is updated dynamically after data retrieval.

## Functionality

1. When the page loads, a “**Loading data...**” message appears.
2. The script fetches user data from the sample API:

```
https://jsonplaceholder.typicode.com/users
```

3. On success:
  - The data is rendered as a styled list (name + email).
4. On failure:
  - Displays an **error message**: “ Failed to load data. Please try again.”

## Accessibility

- Uses semantic HTML (`<ul>` and `<li>` for lists).
- Visually distinguishes states (loading/error) for clarity.

**THANK YOU**