**Module 16) CSS in Full Stack Course**

**CSS Selectors & Styling :**

**Theory Assignment :**

**Question 1**: What is a CSS selector? Provide examples of element, class, and ID selectors.

A **CSS selector** is a pattern used to select and style specific HTML elements. It tells the browser which elements to apply the styles to.

**Examples:**

**Element Selector:** Selects all elements of a specific type.

p {

color: blue;

}

This targets all <p> (paragraph) elements.

**Class Selector:** Selects all elements with a specific class.

.highlight {

background-color: yellow;

}

This targets any element with class="highlight".

**ID Selector:** Selects a single element with a specific ID.

#header {

font-size: 24px;

}

This targets the element with id="header".

**Question 2: Explain the concept of CSS specificity. How do conflicts between multiple styles get resolved?**

**CSS specificity** is a system of rules that the browser uses to determine which CSS rule to apply when multiple rules target the same element.

**Specificity hierarchy (from lowest to highest):**

Element selectors (e.g., div, p) → **1 point**

Class selectors (.class), attribute selectors ([type="text"]), and pseudo-classes (:hover) → **10 points**

ID selectors (#id) → **100 points**

Inline styles (e.g., style="color:red;") → **1000 points**

!important overrides all (but should be used sparingly)

**Example:**

<p id="para1" class="text">Hello</p>

p { color: blue; } /\* 1 point \*/.text { color: green; } /\* 10 points \*/#para1 { color: red; } /\* 100 points \*/

The paragraph will be red because #para1 has the highest specificity.

**Question 3: What is the difference between internal, external, and inline CSS? Discuss the advantages and disadvantages of each approach.**

| **Type** | **Description** | **Example** | **Advantages** | **Disadvantages** |
| --- | --- | --- | --- | --- |
| **Inline CSS** | Styles applied directly to an HTML element using the style attribute. | <h1 style="color:red;">Hello</h1> | - Quick to apply  - Overrides other styles | - Not reusable  - Clutters HTML  - Hard to maintain |
| **Internal CSS** | Styles written within a <style> tag inside the <head> section of the HTML file. | <style> h1 { color: blue; } </style> | - Good for single-page styling  - No external files needed | - Not reusable  - Slows page load if large |
| **External CSS** | Styles are defined in a separate .css file and linked via <link> tag. | <link rel="stylesheet" href="style.css"> | - Reusable across pages  - Clean HTML  - Better caching | - Extra HTTP request  - Requires file management |

**Lab Assignment**

 **Task**: Style the contact form (created in the HTML Forms lab) using external CSS. The following should be implemented:

o Change the background color of the form.

o Add padding and margins to form fields.

o Style the submit button with a hover effect.

o Use class selectors for styling common elements and ID selectors for unique

elements.

1. **Index.html FILE :**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>Contact Form</title>

<link rel="stylesheet" href="style.css"> <!-- External CSS linked here -->

</head>

<body>

<form id="contactForm">

<h2>Contact Us</h2>

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

<input type="text" id="name" class="form-field" required>

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

<input type="email" id="email" class="form-field" required>

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

<textarea id="message" class="form-field" rows="4" required></textarea>

<button type="submit" id="submitBtn">Send</button>

</form>

</body>

</html>

1. **style.css FILE :**

/\* Change the background color of the form \*/

#contactForm {

background-color: #f4f8ff;

width: 400px;

margin: 50px auto;

padding: 20px;

border-radius: 10px;

box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);

font-family: Arial, sans-serif;

}

/\* Common form field styles using class selector \*/

.form-field {

display: block;

width: 100%;

padding: 10px;

margin: 12px 0;

border: 1px solid #ccc;

border-radius: 5px;

box-sizing: border-box;

}

/\* Style the submit button using ID selector \*/

#submitBtn {

background-color: #4CAF50;

color: white;

border: none;

padding: 12px 20px;

font-size: 16px;

border-radius: 5px;

cursor: pointer;

transition: background-color 0.3s ease;

}

/\* Add hover effect to submit button \*/

#submitBtn:hover {

background-color: #45a049;

}

1. **CSS Box Model :**

**Theory Assignment**

**Question 1: Explain the CSS box model and its components (content, padding, border, margin). How does each affect the size of an element?**

The **CSS Box Model** describes how the size of every HTML element is calculated. It consists of **four components** (from innermost to outermost):

### ****Content :****

The actual content of the element (text, images, etc.)

You can set the width and height of this area using width and height properties.

### ****Padding :****

The space between the content and the border.

It increases the space inside the box **without adding visible lines**.

Padding is **inside** the border.

### ****Border :****

The line surrounding the padding (and content).

You can set its width, style, and color using properties like border: 2px solid black;

### ****Margin :****

The space **outside** the border.

It separates the element from other elements on the page.

Margins are **transparent**.

**Question 2: What is the difference between border-box and content-box box-sizing in CSS?Which is the default?**

| **Feature** | **content-box (Default)** | **border-box** |
| --- | --- | --- |
| **What width/height includes** | Only the content area | Content + padding + border |
| **Box grows when padding is added?** | ✅ Yes, total size increases | ❌ No, padding fits **within** the width |
| **Usage** | Precise content sizing | Easier layout control |
| **Default** | ✅ Yes (this is the default) | ❌ No (must be set explicitly) |

**Lab Assignment**

 **Task**: Create a profile card layout using the box model. The profile card should

include:

o A profile picture.

o The user’s name and bio.

o A button to "Follow" the user.

**Additional Requirements**:

o Add padding and borders to the elements.

o Ensure the layout is clean and centered on the page using CSS margins.

o Use the box-sizing property to demonstrate both content-box and border-box

ondifferent elements.

1. index.html FILE :

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>Profile Card</title>

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

</head>

<body>

<div class="card">

<img src="./images/duck.jpg" alt="Profile Picture" class="profile-pic">

<h2 class="name">Arjun Prajapati</h2>

<p class="bio">Full Stack developer. Loves building beautiful things with code and creativity.</p>

<button class="follow-btn">Follow</button>

</div>

</body>

</html>

1. style.css FILE :

/\* Center the card on the page \*/

body {

display: flex;

justify-content: center;

align-items: center;

height: 100vh;

margin: 0;

background-color: #f2f2f2;

font-family: Arial, sans-serif;

}

/\* Profile card container \*/

.card {

width: 300px;

background-color: white;

padding: 20px;

margin: 20px;

border: 2px solid #ccc;

border-radius: 10px;

text-align: center;

box-sizing: border-box; /\* Total width includes padding and border \*/

box-shadow: 0 4px 10px rgba(0, 0, 0, 0.1);

}

/\* Profile picture \*/

.profile-pic {

width: 100px;

height: 100px;

border-radius: 50%;

border: 4px solid #4CAF50;

padding: 5px;

box-sizing: content-box; /\* Padding adds to image size \*/

margin-bottom: 15px;

}

/\* User's name \*/

.name {

font-size: 22px;

margin: 10px 0 5px;

}

/\* Bio text \*/

.bio {

font-size: 14px;

color: #555;

margin: 10px 0 20px;

padding: 0 10px;

}

/\* Follow button \*/

.follow-btn {

background-color: #4CAF50;

color: white;

padding: 10px 20px;

border: none;

border-radius: 20px;

cursor: pointer;

transition: background-color 0.3s;

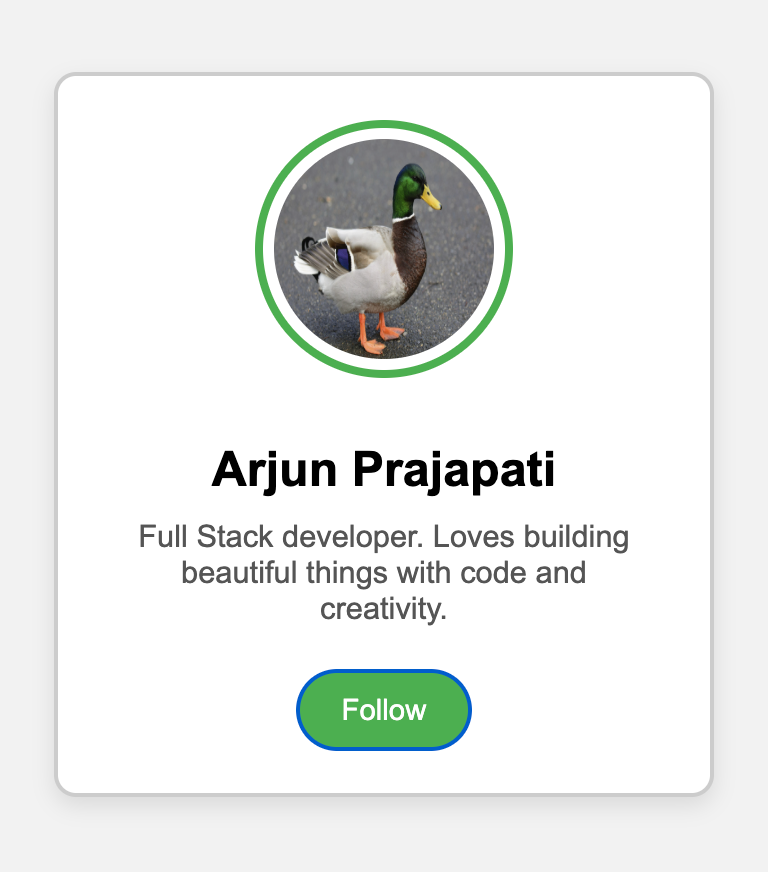
}

.follow-btn:hover {

background-color: #45a049;

}

OUTPUT :



**6. CSS Flexbox**

**Theory Assignment**

**Question 1: What is CSS Flexbox, and how is it useful for layout design? Explain the terms flex-container and flex-item.**

**CSS Flexbox** (Flexible Box Layout) is a one-dimensional layout system in CSS used to arrange elements in rows or columns. It provides a more efficient way to distribute space and align items within a container, especially when building responsive and dynamic layouts.

#### 🎯 ****Key Benefits:****

Easy alignment of elements horizontally or vertically.

Dynamically adjusts layout based on available space.

Simplifies creating responsive designs without using floats or positioning hacks.

Flexbox Terms:

**Flex Container**:

The parent element that has display: flex or display: inline-flex.

Example:

.container {

display: flex;

}

All direct children of this container become **flex items**.

**Flex Item**:

The child elements of a flex container.

Example:

<div class="container">

<div class="item">Item 1</div>

<div class="item">Item 2</div></div>

**Question 2: Describe the properties justify-content, align-items, and flex-direction used inFlexbox.**

These properties are applied to the **flex container** and control the layout behavior of its child flex items:

#### 1. flex-direction:

Defines the direction of the main axis (i.e., how the flex items are placed).

| **Value** | **Description** |
| --- | --- |
| row | Items laid out horizontally (default) |
| row-reverse | Items in reverse horizontal order |
| column | Items laid out vertically |
| column-reverse | Items in reverse vertical order |

.container {

flex-direction: row;

}

#### 2. justify-content:

Aligns flex items along the **main axis** (horizontal in row, vertical in column).

| **Value** | **Description** |
| --- | --- |
| flex-start | Items start at the beginning |
| flex-end | Items at the end |
| center | Items centered |
| space-between | Equal space **between** items |
| space-around | Equal space **around** items |
| space-evenly | Equal space **between and around** items |

.container {

justify-content: center;

}

#### 3. align-items:

Aligns items along the **cross axis** (perpendicular to the main axis).

| **Value** | **Description** |
| --- | --- |
| stretch | Stretches items to fill container height |
| flex-start | Aligns items to the start of the cross axis |
| flex-end | Aligns items to the end |
| center | Centers items vertically (if flex-direction: row) |

.container {

align-items: center;

}

**Lab Assignment**

 **Task**: Create a simple webpage layout using Flexbox. The layout should include:

o A header.

o A sidebar on the left.

o A main content area in the center.

o A footer.

**Additional Requirements**:

o Use Flexbox to position and align the elements.

o Apply different justify-content and align-items properties to observe their effects.

o Ensure the layout is responsive, adjusting forsmaller screens.

1. **index.html FILE :**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>Flexbox Layout</title>

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

</head>

<body>

<header class="header">Header</header>

<div class="container">

<aside class="sidebar">Sidebar</aside>

<main class="main-content">Main Content</main>

</div>

<footer class="footer">Footer</footer>

</body>

</html>

1. **style.css FILE :**

/\* Global styles \*/

\* {

box-sizing: border-box;

margin: 0;

padding: 0;

}

body {

font-family: Arial, sans-serif;

background-color: #f5f5f5;

}

/\* Header \*/

.header {

background-color: #4CAF50;

color: white;

padding: 20px;

text-align: center;

}

/\* Footer \*/

.footer {

background-color: #333;

color: white;

text-align: center;

padding: 15px;

margin-top: auto;

}

/\* Flex container for sidebar and main content \*/

.container {

display: flex;

justify-content: space-between; /\* spread elements \*/

align-items: flex-start; /\* align to top \*/

padding: 20px;

flex-wrap: wrap; /\* allows wrapping on smaller screens \*/

}

/\* Sidebar \*/

.sidebar {

flex: 1 1 200px;

background-color: #ddd;

padding: 20px;

margin: 10px;

min-width: 180px;

}

/\* Main content \*/

.main-content {

flex: 2 1 400px;

background-color: #fff;

padding: 20px;

margin: 10px;

min-width: 300px;

}

/\* Responsive design for smaller screens \*/

@media (max-width: 768px) {

.container {

flex-direction: column;

align-items: stretch;

}

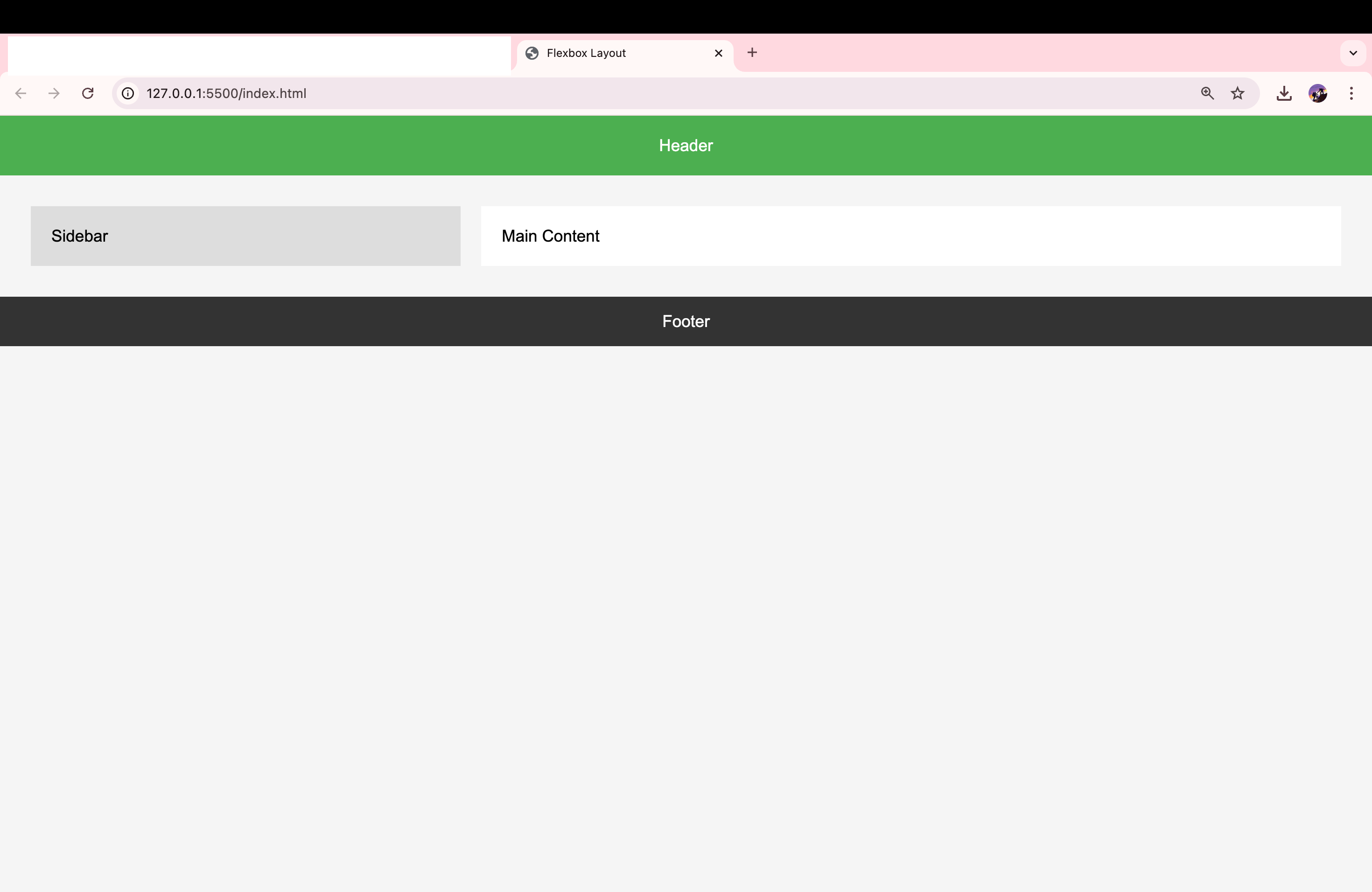
.sidebar, .main-content {

width: 100%;

}

}

**Output :**



**7. CSS Grid**

**Theory Assignment**

**Question 1: Explain CSS Grid and how it differs from Flexbox. When would you use Grid over Flexbox?**

#### ****What is CSS Grid?****

**CSS Grid** is a two-dimensional layout system that allows you to layout items in **rows and columns**. It gives you fine-grained control over positioning and aligning elements in a structured grid.

#### ****CSS Grid vs. Flexbox****

| **Feature** | **CSS Grid** | **Flexbox** |
| --- | --- | --- |
| **Layout type** | Two-dimensional (rows + columns) | One-dimensional (row **or** column) |
| **Best for** | Entire page or large-scale layouts | Small components or linear flows |
| **Item placement** | Allows precise row/column placement | Order and alignment along one axis |
| **Overlap** | Supports overlapping items | No overlapping |

#### ✅ ****When to Use Grid Over Flexbox:****

When you need a **structured layout** with **rows and columns** (e.g., a photo gallery, dashboard, full-page layout).

When content needs to **align both horizontally and vertically**.

Use **Flexbox** when you only need layout in a **single direction** (e.g., a nav bar, list of buttons).

**Question 2: Describe the grid-template-columns, grid-template-rows, and grid-gap properties. Provide examples of how to use them.**

These properties are used to define the **structure** and **spacing** of a CSS Grid layout.

### grid-template-columns

Defines the number and width of columns in the grid.

.grid {

display: grid;

grid-template-columns: 200px 1fr 2fr;

}

This creates **3 columns**:

1st = 200px

2nd = 1 part of remaining space

3rd = 2 parts of remaining space

#### Example:

grid-template-columns: repeat(3, 1fr); /\* 3 equal columns \*/

### grid-template-rows

Defines the number and height of rows in the grid.

.grid {

display: grid;

grid-template-rows: 100px auto 50px;

}

This creates 3 rows with specific heights:

1st = 100px

2nd = automatic height (based on content)

3rd = 50px

### grid-gap (now split as row-gap and column-gap in modern CSS)

Adds spacing **between** rows and columns.

.grid {

display: grid;

grid-template-columns: repeat(2, 1fr);

grid-template-rows: auto;

grid-gap: 20px;

}

This adds **20px of space** between both rows and columns.

### ✅ Example: Grid Layout

<div class="grid">

<div>Item 1</div>

<div>Item 2</div>

<div>Item 3</div>

<div>Item 4</div></div>

.grid {

display: grid;

grid-template-columns: repeat(2, 1fr); /\* 2 equal columns \*/

grid-template-rows: auto auto;

gap: 15px; /\* shorthand for row-gap and column-gap \*/

}

**Result**: A 2x2 grid with equal columns and 15px gaps between items.

**Lab Assignment**

 **Task**: Create a 3x3 grid of product cards using CSS Grid. Each card should contain:

o A product image.

o A product title.

o A price.

**Additional Requirements**:

o Use grid-template-columns to create the grid layout.

o Use grid-gap to add spacing between the grid items.

o Apply hover effects to each card for better interactivity.

1. index.html FILE :

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>Product Grid</title>

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

</head>

<body>

<h1 class="page-title">Our Products</h1>

<div class="grid-container">

<!-- Product Card Example (repeat this 9 times for 3x3) -->

<div class="product-card">

<img src="./images/noise .jpeg" alt="Product" class="product-img">

<h2 class="noise buds vs102">noise buds vs102</h2>

<p class="price">$19.99</p>

</div>

<div class="product-card">

<img src="./images/boAt Rockerz 425.jpeg" alt="Product" class="product-img">

<h2 class="product-title">boAt Rockerz 425</h2>

<p class="price">$29.99</p>

</div>

<!-- ...repeat until 9 cards total -->

<div class="product-card">

<img src="./images/AULA S505.jpeg" alt="Product" class="product-img">

<h2 class="product-title">AULA S505</h2>

<p class="price">$39.99</p>

</div>

<div class="product-card">

<img src="./images/Sex On The Beach.jpeg" alt="Product" class="product-img">

<h2 class="product-title">Sex On The Beach</h2>

<p class="price">$24.99</p>

</div>

<div class="product-card">

<img src="./images/Creatine Monohydrate (122g, 33 Servings).webp" alt="Product" class="product-img">

<h2 class="product-title">Creatine Monohydrate (122g, 33 Servings)</h2>

<p class="price">$44.99</p>

</div>

<div class="product-card">

<img src="./images/MuscleBlaze High Protein Oats.jpeg" alt="Product" class="product-img">

<h2 class="product-title">MuscleBlaze High Protein Oats</h2>

<p class="price">$15.99</p>

</div>

<div class="product-card">

<img src="./images/download.jpeg" alt="Product" class="product-img">

<h2 class="product-title">Titan Skeletal Automatic Blue Dial Brown Leather Strap Watch for Men</h2>

<p class="price">$22.49</p>

</div>

<div class="product-card">

<img src="./images/Coral (9613) 6 Hands Chronograph Watch.jpeg" alt="Product" class="product-img">

<h2 class="product-title">Coral (9613) 6 Hands Chronograph Watch</h2>

<p class="price">$31.75</p>

</div>

<div class="product-card">

<img src="./images/PIX\_2222.webp" alt="Product" class="product-img">

<h2 class="product-title">Calcetto Lightweight Casual Men'S Sneakers</h2>

<p class="price">$17.00</p>

</div>

</div>

</body>

</html>

1. style.css FILE :

body {

font-family: Arial, sans-serif;

background-color: #f9f9f9;

margin: 0;

padding: 20px;

}

.page-title {

text-align: center;

margin-bottom: 30px;

color: #333;

}

/\* Grid container \*/

.grid-container {

display: grid;

grid-template-columns: repeat(3, 1fr); /\* 3 columns \*/

gap: 20px; /\* space between grid items \*/

max-width: 1000px;

margin: auto;

}

/\* Product card \*/

.product-card {

background-color: white;

border: 1px solid #ddd;

border-radius: 10px;

padding: 15px;

text-align: center;

transition: transform 0.3s, box-shadow 0.3s;

}

/\* Product image \*/

.product-img {

width: 100%;

height: auto;

border-radius: 5px;

}

/\* Product title \*/

.product-title {

font-size: 18px;

margin: 10px 0;

color: #333;

}

/\* Price \*/

.price {

color: #4CAF50;

font-weight: bold;

font-size: 16px;

}

/\* Hover effect \*/

.product-card:hover {

transform: translateY(-5px);

box-shadow: 0 8px 16px rgba(0, 0, 0, 0.1);

}

/\* Responsive layout \*/

@media (max-width: 768px) {

.grid-container {

grid-template-columns: repeat(2, 1fr); /\* 2 columns on medium screens \*/

}

}

@media (max-width: 480px) {

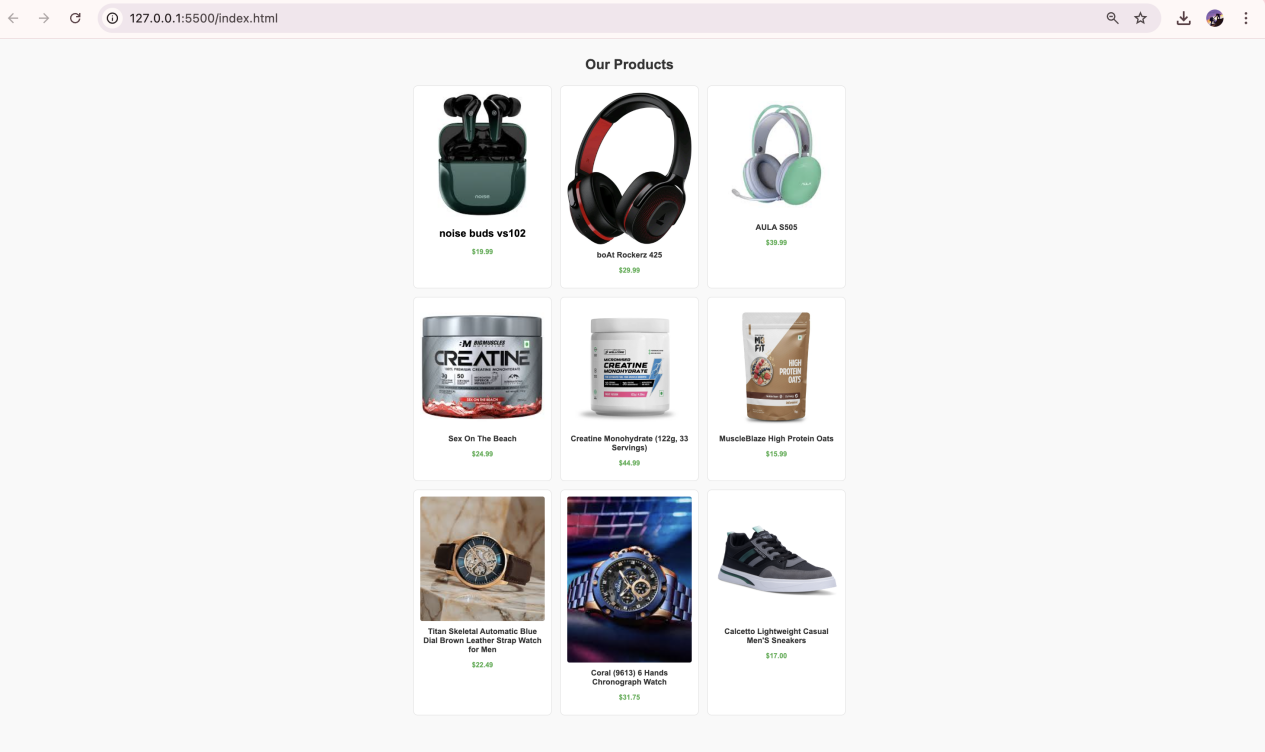
.grid-container {

grid-template-columns: 1fr; /\* 1 column on small screens \*/

}

}

OUTPUT :



**8. Responsive Web Design with Media Queries**

**Theory Assignment**

**Question 1: What are media queries in CSS, and why are they important for responsive design?**

#### ****What are Media Queries?****

**Media queries** are a CSS feature used to apply styles **conditionally** based on the **device’s characteristics**, such as screen width, height, orientation, resolution, etc.

They allow web developers to create **responsive designs** that look good on all screen sizes (desktops, tablets, phones).

#### ****Why Are Media Queries Important?****

They **adapt layouts** to different devices (mobile, tablet, desktop).

Improve **user experience** and accessibility.

Allow **fine-tuned control** over design without duplicating code.

Essential for **mobile-first** or **responsive web development**.

**Question 2: Write a basic media query that adjusts the font size of a webpage for screens smaller than 600px.**

@media (max-width: 600px) {

body {

font-size: 14px;

}

}

**Lab Assignment**

** Task: Build a responsive webpage that includes:**

**o A navigation bar.**

**o A contentsection with two columns.**

**o A footer.**

**Additional Requirements:**

**o Use media queries to make the webpage responsive for mobile devices.**

**o On smaller screens (below 768px),stack the columns vertically.**

**o Adjust the fontsizes and padding to improve readability on mobile.**

1. index.html FILE :

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>Responsive Webpage</title>

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

</head>

<body>

<!-- Navigation Bar -->

<nav class="navbar">

<h1 class="logo">MySite</h1>

<ul class="nav-links">

<li><a href="#">Home</a></li>

<li><a href="#">About</a></li>

<li><a href="#">Contact</a></li>

</ul>

</nav>

<!-- Content Section -->

<section class="content">

<div class="column left">

<h2>Left Column</h2>

<p>This is the left content area. Add any text, images, or features here.</p>

</div>

<div class="column right">

<h2>Right Column</h2>

<p>This is the right content area. You can place related information here.</p>

</div>

</section>

<!-- Footer -->

<footer class="footer">

<p>&copy; 2025 MySite. All rights reserved.</p>

</footer>

</body>

</html>

1. style.css FILE :

/\* Base styles \*/

\* {

box-sizing: border-box;

margin: 0;

padding: 0;

}

body {

font-family: Arial, sans-serif;

line-height: 1.6;

background-color: #f4f4f4;

}

/\* Navigation Bar \*/

.navbar {

background-color: #333;

color: white;

display: flex;

justify-content: space-between;

align-items: center;

padding: 15px 30px;

flex-wrap: wrap;

}

.logo {

font-size: 24px;

}

.nav-links {

list-style: none;

display: flex;

gap: 20px;

}

.nav-links a {

color: white;

text-decoration: none;

font-size: 16px;

}

/\* Content Section \*/

.content {

display: flex;

padding: 30px;

gap: 20px;

max-width: 1000px;

margin: auto;

}

.column {

flex: 1;

padding: 20px;

background-color: white;

border-radius: 8px;

box-shadow: 0 2px 8px rgba(0, 0, 0, 0.05);

}

/\* Footer \*/

.footer {

background-color: #333;

color: white;

text-align: center;

padding: 20px;

margin-top: 30px;

}

/\* ✅ Responsive Media Query \*/

@media (max-width: 768px) {

.content {

flex-direction: column;

padding: 20px;

}

.nav-links {

flex-direction: column;

gap: 10px;

margin-top: 10px;

}

.logo {

font-size: 20px;

}

.nav-links a {

font-size: 14px;

}

.column {

padding: 15px;

}

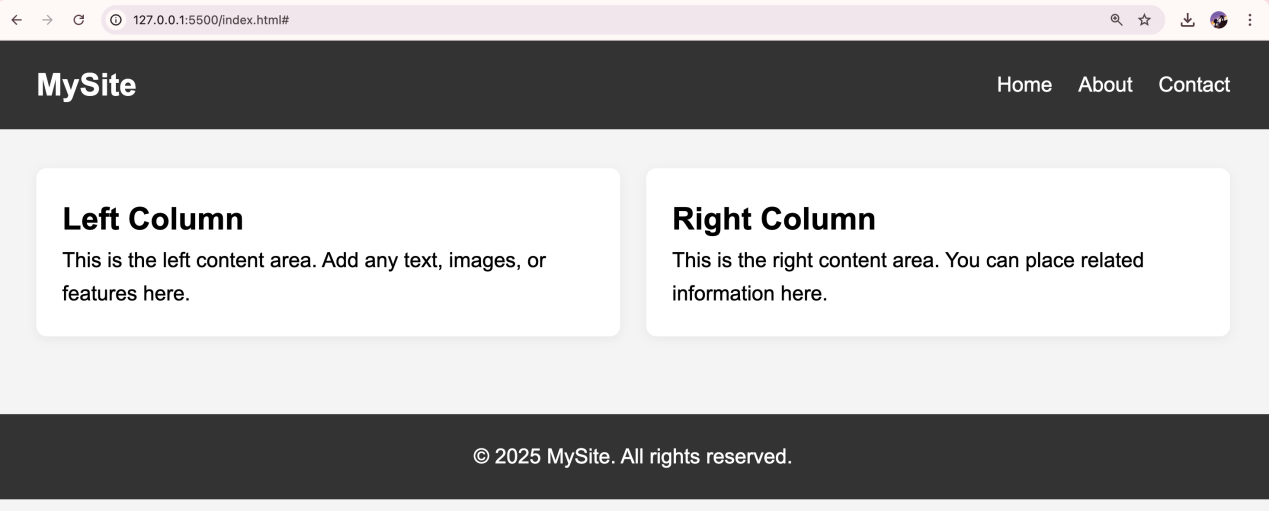
body {

font-size: 15px;

}

}

OUTPUT :



**9. Typography and Web Fonts**

**Theory Assignment**

**Question 1: Explain the difference between web-safe fonts and custom web fonts. Why might you use a web-safe font over a custom font?**

#### ****Web-safe fonts:****

These are **pre-installed fonts** found on most operating systems (Windows, macOS, Linux).

Examples: Arial, Verdana, Times New Roman, Courier New, Georgia, Trebuchet MS.

Because they're already installed, **no download is required**—they **load quickly and consistently** across devices.

#### ****Custom web fonts:****

#### Fonts that are ****not pre-installed****; they are downloaded from the web, often via services like ****Google Fonts****, ****Adobe Fonts****, or self-hosted.

They allow **greater design flexibility and branding**, e.g., Roboto, Lato, Poppins, etc.

#### ****Why use web-safe fonts?****

**Faster page loads** (no external font requests).

**Better compatibility** on older devices and browsers.

**No dependency** on internet availability or font CDN issues.

**Question 2: What is the font-family property in CSS? How do you apply a custom Google Font to a webpage?**

#### ****font-family:****

It defines the **font style** for text in an element.

You can list **multiple fonts** as fallbacks:

p {

font-family: 'Arial', 'Helvetica', sans-serif;

}

The browser uses the first available font in the list.

**Applying a custom Google Font:**

**Step 1: Get the link from Google Fonts:**

Example: For Poppins font

<!-- In your <head> section --><link href="https://fonts.googleapis.com/css2?family=Poppins&display=swap" rel="stylesheet">

**Step 2: Use it in your CSS:**

body {

font-family: 'Poppins', sans-serif;

}

✅ Always include a fallback font (like sans-serif) in case the custom font fails to load.

**Lab Assignment**

** Task: Create a blog post layout with the following:**

**o A title, subtitle, and body content.**

**o Use at least two different fonts (one for headings, one for body content).**

**o Style the text to be responsive and easy to read.**

**Additional Requirements:**

**o Use a custom font from Google Fonts.**

**o Adjust line-height, font-size, and spacing for improved readability.**

1. index.html FILE :

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>My Blog Post</title>

<!-- Google Fonts -->

<link href="https://fonts.googleapis.com/css2?family=Playfair+Display&family=Open+Sans&display=swap" rel="stylesheet">

<!-- Link to External CSS -->

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

</head>

<body>

<div class="blog-container">

<h1 class="title">The Journey Begins</h1>

<h2 class="subtitle">How I Started My Creative Writing Path</h2>

<p class="body-text">

Writing has always been a passion of mine, but it wasn’t until recently that I found the courage to share my stories with the world. In this post, I’ll take you through the experiences that shaped my voice and the lessons I’ve learned along the way...

</p>

<p class="body-text">

The blank page used to scare me. Now, it's an invitation to explore, to wonder, and to create. Every word I write brings me closer to the version of myself I want to be — expressive, authentic, and bold.

</p>

</div>

</body>

</html>

1. style.css FILE :

/\* Base settings \*/

body {

margin: 0;

padding: 0;

background-color: #f4f4f4;

font-family: 'Open Sans', sans-serif;

line-height: 1.6;

font-size: 16px;

color: #333;

}

.blog-container {

max-width: 800px;

margin: 40px auto;

padding: 20px;

background-color: #ffffff;

box-shadow: 0 2px 8px rgba(0, 0, 0, 0.1);

}

/\* Heading Font \*/

.title {

font-family: 'Playfair Display', serif;

font-size: 2.5rem;

margin-bottom: 10px;

color: #222;

}

.subtitle {

font-family: 'Playfair Display', serif;

font-size: 1.5rem;

margin-bottom: 25px;

color: #555;

}

/\* Body Text \*/

.body-text {

margin-bottom: 20px;

font-size: 1.125rem;

line-height: 1.8;

}

/\* Responsive Font Sizes \*/

@media (max-width: 600px) {

.title {

font-size: 2rem;

}

.subtitle {

font-size: 1.25rem;

}

.body-text {

font-size: 1rem;

line-height: 1.7;

}

}

OUTPUT :

