

INSTITUTE :- TOPS TECHNOLOGIES (BARODA)

NAME :- SRUJAL SOLANKI M

COURSE :- JAVA FULL STACK DEVELOPER

Module: CSS and CSS3

CSS Selectors & Styling

Theory Assignment

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

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

Definition:

A CSS selector is a pattern used to select the HTML elements you want to style. CSS selectors target elements based on their tag name, class, ID, attribute, etc.

Examples:

Element Selector:

Selects all elements of a given type.

p {

color: blue;

}

This styles all <p> tags with blue text.

Class Selector:

Targets elements with a specific class.

.contains {

background-color: yellow;

}

ID Selector:

Targets a specific element with a unique ID.

#placeholder {

font-size: 24px;

}

 Question 2: Explain the concept of CSS specificity. How do conflicts between multiple styles

get resolved?

Definition:

CSS Specificity is a set of rules that determines which style rule wins when multiple rules target the same element.

Selector Type Specificity

Universal selector \* 0,0,0

Element selector p, div 0,0,1

Class selector .class, attribute [type="text"], pseudo-class :hover 0,1,0

ID selector #id 1,0,0

Inline styles style="" 1,0,0,0 or even higher

!important (overrides all if used) Highest (but bad practice)

 Question 3: What is the difference between internal, external, and inline CSS? Discuss the

advantages and disadvantages of each approach.

| Type | Where it's written | Example | Advantages | Disadvantages

| ---------------- | ------------------------------------------------- | ------------------------------------------ | ------------------------------------------------------------------- | |

| Inline CSS\*\* | Directly inside an HTML tag using `style=""` | `<h1 style="color:red;">Hello</h1>` | - Quick for testing<br>- Highest specificity | - Hard to maintain<br>- Not reusable

|Internal CSS | Inside `<style>` tag in the HTML `<head>` | `<style> p {color: blue;} </style>` | - Better than inline<br>- Useful for single pages | - Not reusable across pages

|External CSS\*\* | In a separate `.css` file and linked via `<link>` | `<link rel="stylesheet" href="style.css">` | - Reusable<br>- Clean structure<br>- Faster page loading with cache | - Requires separate file<br>- Extra HTTP request

Lab Assignment

 Task: Style the contact form (created in the HTML Forms lab) using external CSS. The

following should be implemented:

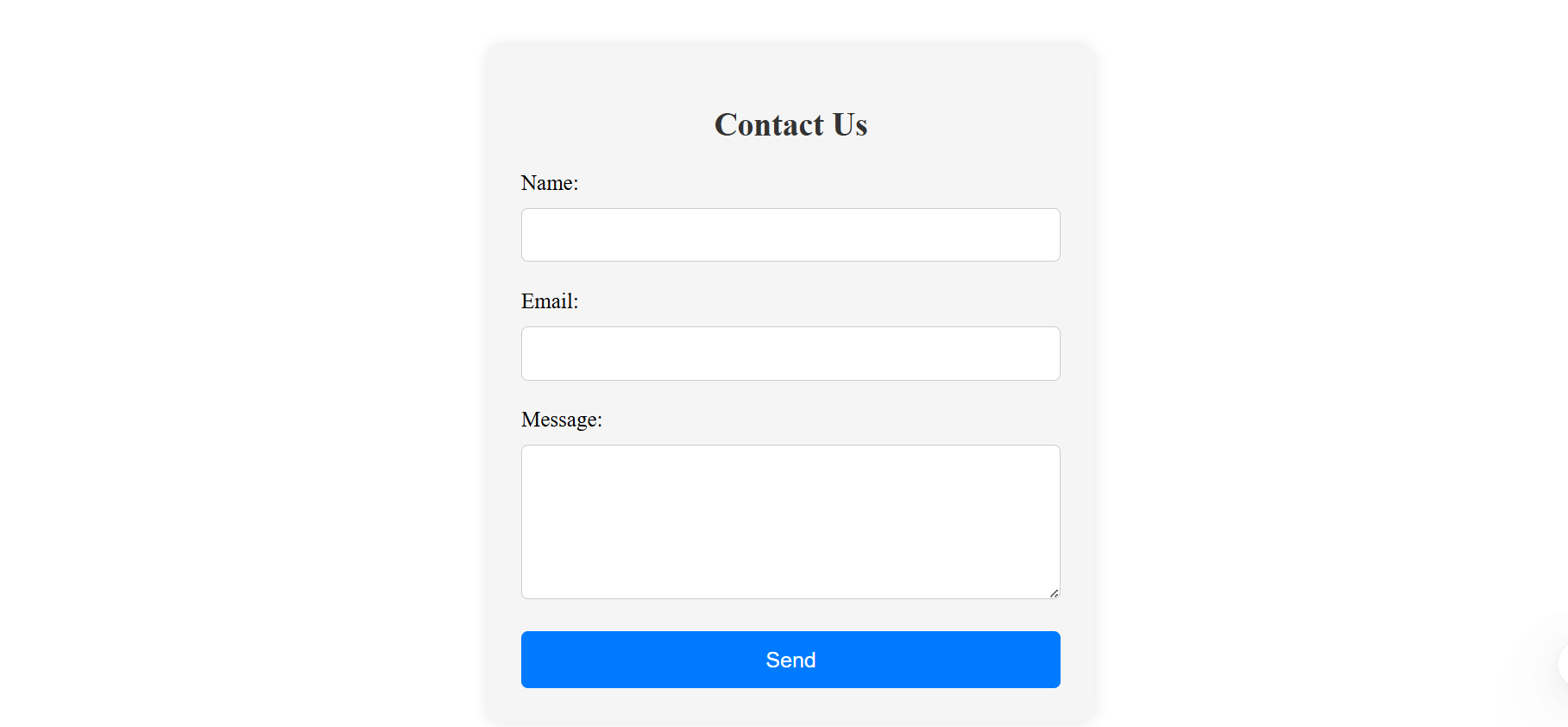
o Change the background color of the form.

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.



5. 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** is the foundation of layout in web design. Every HTML element is treated as a rectangular **box**, and this box is made up of four parts (from inside to outside):

**1. Content**

* This is where your actual text, image, or other content appears.
* The **width** and **height** of the element usually apply to this area (depending on box-sizing).

**2. Padding**

* The space between the content and the border.
* It pushes the content **inward**, creating breathing room around it.
* Padding is **transparent** and part of the box size unless overridden by box-sizing.

**3. Border**

* A visible line around the padding (and content).
* Its **thickness** and **style** (e.g., solid, dashed) are customizable.
* Also affects the element’s size unless using box-sizing: border-box.

**4. Margin**

* The space **outside** the border.
* It separates this element from **other elements**.
* Margins **do not add to the box’s size**, but they **affect layout spacing**.

 Question 2: What is the difference between border-box and content-box box-sizing in

CSS?Which is the default?

CSS uses the box-sizing property to control how the total size of an element is calculated.

**1. content-box**

* The width and height apply **only to the content**.
* Padding and border are **added on top** of the width and height.

Example:

box-sizing: content-box;

width: 200px; /\* applies only to content \*/

padding: 10px;

border: 5px;

Total width = 200 + 10*2 + 5*2 = **230px**  
 Total height = same calculation.

**2. border-box**

* The width and height include **content + padding + border**.
* The content area **shrinks automatically** to fit padding and border inside.

Example:

box-sizing: border-box;

width: 200px; /\* total size including padding and border \*/

padding: 10px;

border: 5px;

Content width = 200 - 10*2 - 5*2 = **170px**  
 Total width = still **200px**

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.

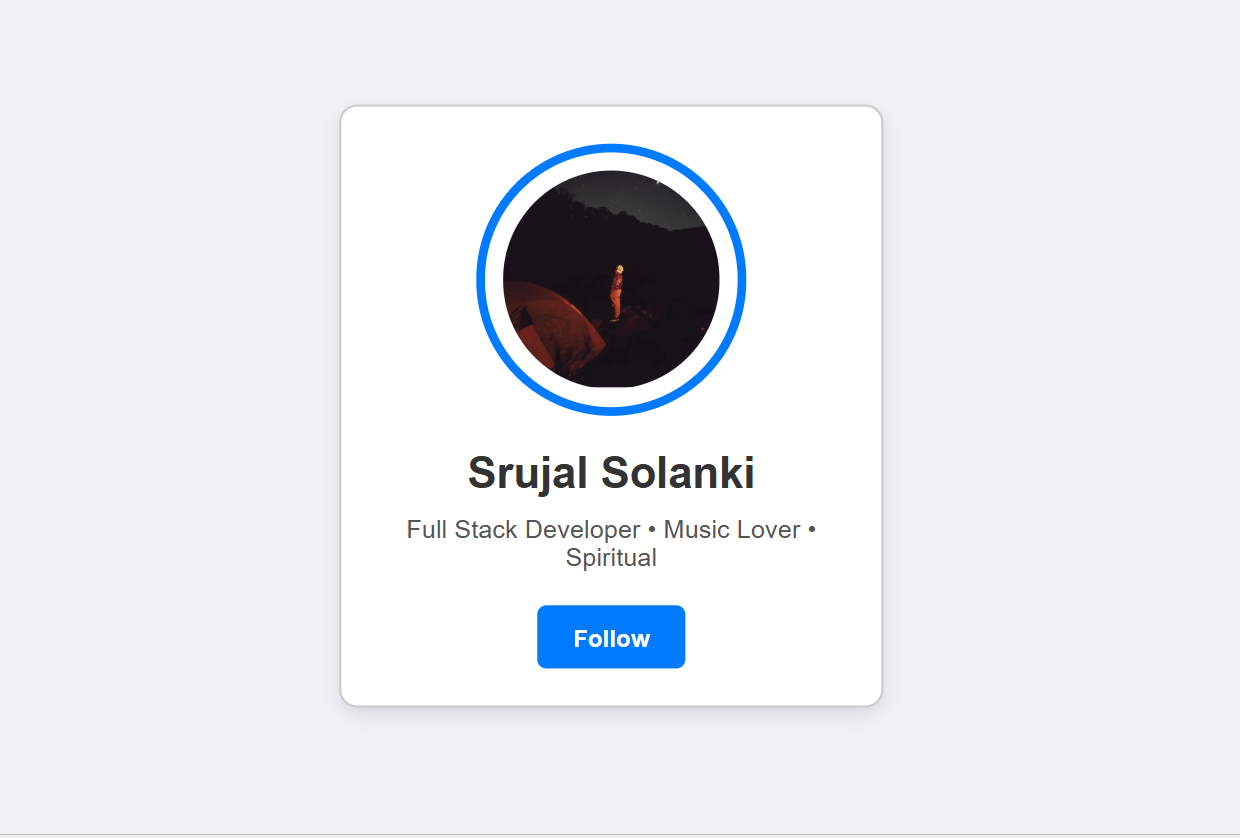
Additional Requirements:

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.



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 layout model designed to make it **easier to arrange items in a one-dimensional row or column**, and distribute space dynamically — **even when screen sizes change**.

It solves common layout problems like:

* Vertical & horizontal centering
* Equal spacing between elements
* Responsively adjusting item sizes

**Flexbox Key Components:**

**flex-container:**

* The **parent** element where display: flex is applied.
* It becomes a flex context for its direct children.

**flex-item:**

* The **children** of the flex-container.
* These are the elements that get laid out and controlled by flex properties.

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

inFlexbox.

**flex-direction**

* Controls the **main axis** direction (row or column).
* Values:
  + row (default): Left to right (horizontal)
  + row-reverse: Right to left
  + column: Top to bottom (vertical)
  + column-reverse: Bottom to top

**justify-content**

* Controls **horizontal alignment** (if flex-direction: row) along the **main axis**.
* Values:
  + flex-start → Items at the start
  + flex-end → Items at the end
  + center → Centered
  + space-between → Equal space *between* items
  + space-around → Equal space *around* items
  + space-evenly → Equal space *between and at ends*

**align-items**

* Controls **vertical alignment** (if flex-direction: row) along the **cross axis**.
* Values:
  + stretch (default) → Items stretch to fill container
  + flex-start → Items align at top
  + flex-end → Items align at bottom
  + center → Items centered vertically
  + baseline → Align text baselines

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.

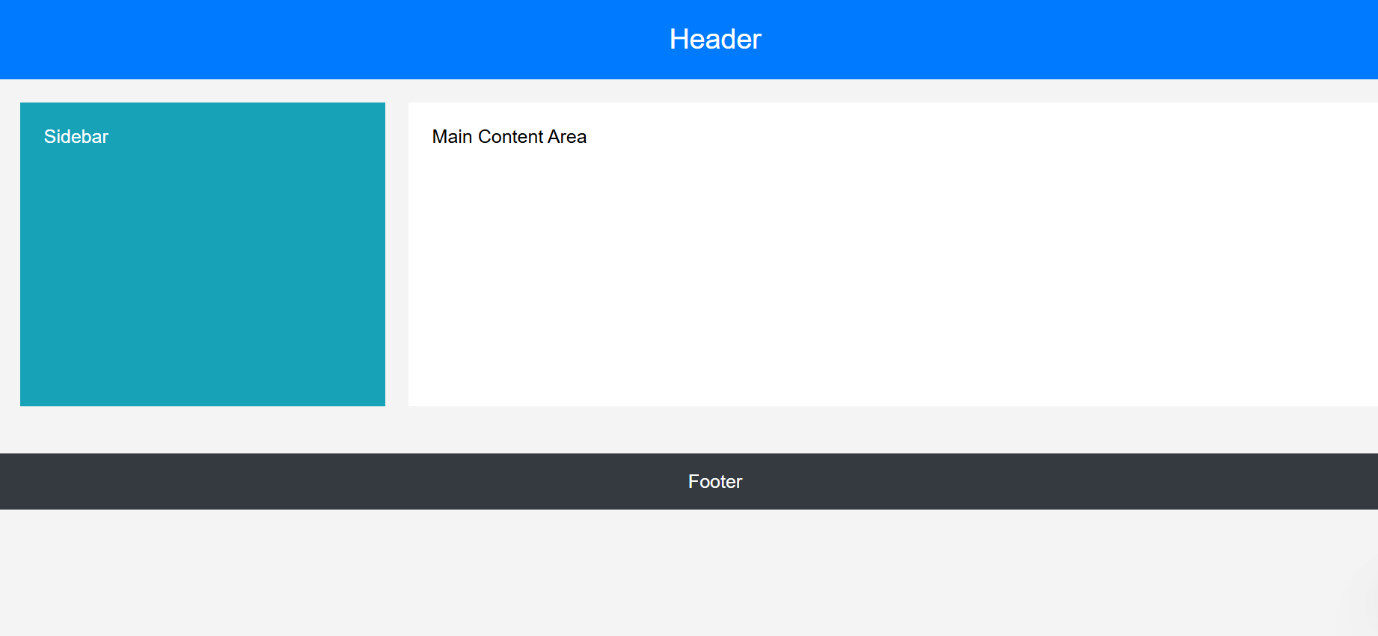
o A footer.

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 for smaller screens.



7. CSS Grid

Theory Assignment

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

Flexbox?

 Question 2: Describe the grid-template-columns, grid-template-rows, and grid-gap

properties. Provide examples of how to use them.

**CSS Grid** is a **two-dimensional layout system** in CSS that allows you to design web pages by placing elements in **rows and columns**.

Think of Grid as designing layouts like a table — with control over both **horizontal (columns)** and **vertical (rows)** at once.

**Flexbox vs Grid: Key Differences**

| **Feature** | **Flexbox** | **CSS Grid** |
| --- | --- | --- |
| Layout Direction | One-dimensional (row **or** column) | Two-dimensional (row **and** column) |
| Parent is | Flex container (display: flex) | Grid container (display: grid) |
| Use Case | Components in a single direction | Full page or complex 2D layouts |
| Content-based layout | Better | Good, but more structure-based |
| Overlap Elements | Not easily | Can overlap elements easily (using grid lines) |

 Question 2: Describe the grid-template-columns, grid-template-rows, and grid-gap

properties. Provide examples of how to use them.

**1. grid-template-columns**

Defines how many **columns** you want in the grid and how wide they should be.

.container {

display: grid;

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

}

This creates **3 columns**:

* First column: fixed width 200px
* Second and third columns: take equal remaining space (fractional units)

You can also use:

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

**2. grid-template-rows**

Defines the number and height of **rows** in the grid.

.container {

display: grid;

grid-template-rows: 100px 300px;

}

This makes two rows:

* First row: 100px
* Second row: 300px

Or dynamic rows:

grid-template-rows: auto 1fr;

**3. grid-gap (or gap)**

Adds spacing **between rows and columns** in the grid (no need for margins).

.container {

display: grid;

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

grid-gap: 20px;

}

You can also set them separately:

gap: 10px 30px; /\* 10px row gap, 30px column gap \*/

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.

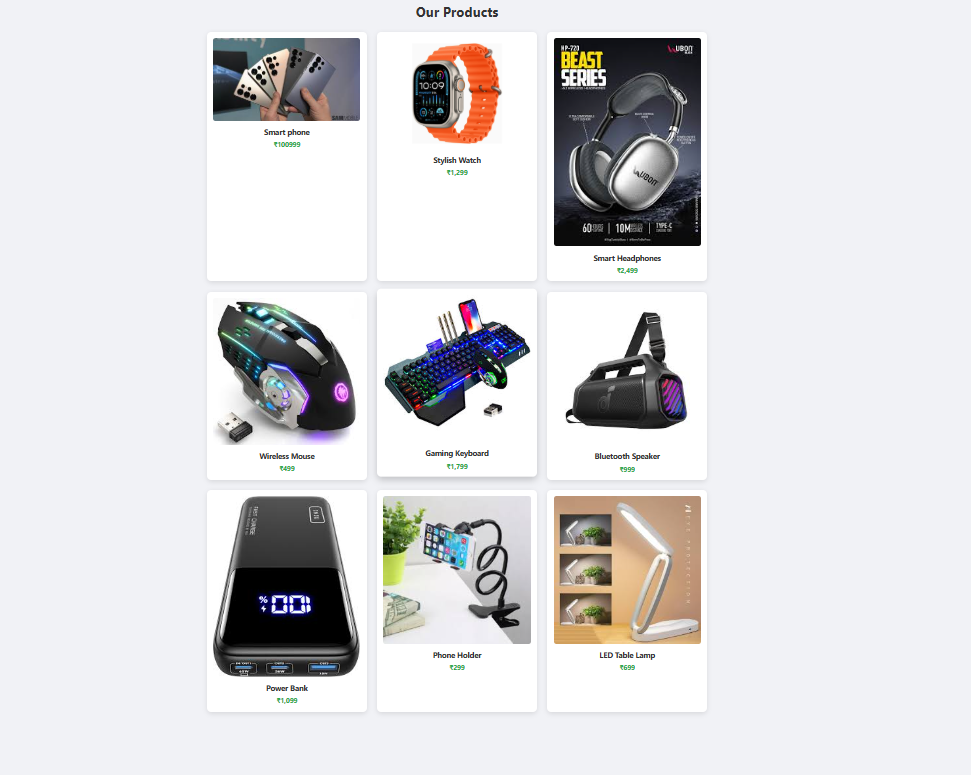
o A price.

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.



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?

**Media Queries** in CSS are used to apply different styles to different devices or screen sizes (like mobile, tablet, desktop).

They **"query" the screen width, height, resolution, etc.**, and apply CSS styles **only when the conditions are true**.

**Why Are Media Queries Important?**

✅ **Responsive Design:** They help make websites adjust automatically to **any screen size** (mobile, tablet, desktop).  
✅ **User Experience:** Improve readability and usability across devices.  
✅ **Performance:** You can hide or modify heavy layouts for smaller screens.  
✅ **Customization:** You can show different styles on different screen widths (like larger fonts on mobile).

 Question 2: Write a basic media query that adjusts the font size of a webpage for screens

@media (max-width: 600px) {

body {

font-size: 14px;

}

}

 @media (max-width: 600px) means:  
Apply the styles **only if the screen width is 600px or less** (like smartphones).

Inside it, we are setting:

smaller than 600px.

Lab Assignment

 Task: Build a responsive webpage that includes:

o A navigation bar.

o A content section with two columns.

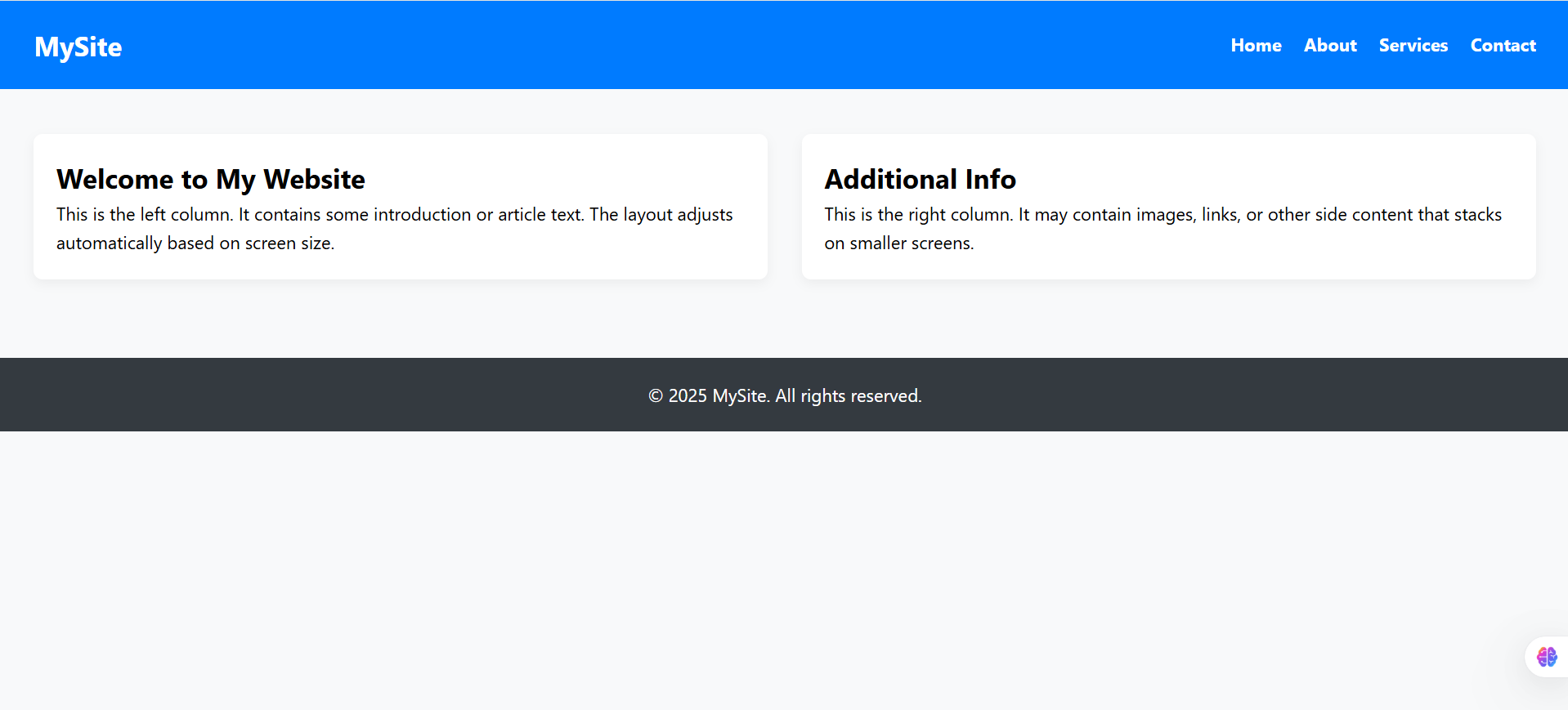
o A footer.

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 font sizes and padding to improve readability on mobile.



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:**

Web-safe fonts are **standard system fonts** that are pre-installed on almost **all operating systems** (Windows, macOS, Linux, Android, iOS).

Examples: Arial, Verdana, Times New Roman, Georgia, Courier New, Tahoma, Helvetica

These fonts **don't need to be downloaded** — they load instantly and are reliable across all browsers and platforms.

**Custom Web Fonts:**

Custom web fonts are **external fonts** (not installed on the user’s system) that are loaded from **web font services** like:

* Google Fonts
* Adobe Fonts
* Fonts.com

Examples: Roboto, Poppins, Montserrat, Lobster, Oswald, etc.

These provide a **richer visual appearance** and branding options, but require:

* Internet connection
* Extra HTTP requests (can slow down page slightly)

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

Font to a webpage?

The font-family CSS property defines the **font style** for your text. It allows you to specify a **font stack** — a list of fonts to try in order.

body {

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

}

In the above:

* Browser tries Arial
* If not available, it tries Helvetica
* If neither is available, it uses default sans-serif font

**How to Apply a Google Font to a Webpage?**

**Step 1: Copy the <link> from** [**https://fonts.google.com**](https://fonts.google.com)

Example for **Poppins**:

<link href="https://fonts.googleapis.com/css2?family=Poppins&display=swap" rel="stylesheet">

<head>

<link href="https://fonts.googleapis.com/css2?family=Poppins&display=swap" rel="stylesheet">

</head>

**Step 2: Apply the Font in CSS using font-family**

body {

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

}

This tells the browser:

* Use 'Poppins' as the primary font (loaded from Google)
* If it fails, fall back to a sans-serif system font

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.

