WD CSS and CSS3

1) CSS Selectors & Styling

Theory Assignment :

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

Answer :-

- A CSS selector is a pattern used to select and style specific elements in an HTML document. It allows applying styles to elements based on their type, class, ID, or other attributes.
- A) Element Selector:
  - Targets all instances of a specific HTML element.
  - EX:-

```
h1 {
    color: blue;
}
```

- B) Class Selector:
  - Targets elements with a specific class attribute.
  - EX:-

```
.highlight {
    background-color: yellow;
}
```

- C) ID Selector:
  - Targets a single element with a specific ID attribute.
  - EX:-

```
#header {
    font-size: 24px;
}
```

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

ANSWER:-

- CSS specificity determines which style rule is applied when multiple rules target the same element. Specificity is calculated based on the type of selectors used:

A) Inline styles (style="...") have the highest specificity.

B) ID selectors are more specific than class, attribute, or pseudo-class selectors.

C) Class selectors are more specific than element selectors.

- EX:-

```
h1 {
    color: blue; /* Specificity: 1 */
  }

.title {
    color: red; /* Specificity: 10 */
  }

#main-title {
    color: green; /* Specificity: 100 */
  }
```

- For an <h1> element with id="main-title" and class="title", the text color will be green because the ID selector has the highest specificity.

- Conflict Resolution:

When specificity is the same, the later rule in the CSS file takes precedence (last rule wins).

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

ANSWER :-

A) INLINE CSS

:- Styles are applied directly to an HTML element using the style attribute.

:- EX

```
<p style="color: red;">This is a red paragraph.</p>
```

:- Advantages

- Quick and easy to apply for small changes.
- Useful for overriding other styles.

:- Disadvantages

- Difficult to maintain or reuse.
- Reduces separation of concerns (HTML and CSS are mixed).

B) Internal CSS

:- Styles are written inside a <style> tag in the <head> section of the HTML document.

:- EX

```
<style>
    p {
        color: blue;
      }
</style>
```

:- Advantages

- Useful for styling a single document.
- Keeps styles scoped to the page.

:- Disadvantages

- Not reusable across multiple pages.
- Can lead to larger HTML files.

C) External CSS

Styles are written in a separate .css file and linked using the <link> tag.

:- EX

```
<link rel="stylesheet" href="styles.css">
```

:- Advantages

- Promotes reusability across multiple pages.
- Enhances maintainability and keeps HTML clean.
- Allows caching of CSS files for faster loading.

:- Disadvantages

- Requires an additional HTTP request to load the CSS file.
- Dependency on the external file—styles break if the file is unavailable.

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 select

Answer :-

```
/* html code */
 <!DOCTYPE html>
                <html lang="en">
                <head>
                <meta charset="UTF-8">
                <meta name="viewport" content="width=device-width, initial-
    scale=1.0">
                <title>Contact Form</title>
                <link rel="stylesheet" href="style.css">
                </head>
```

```
<body>
      <form id="contact-form">
         <h2>Contact Us</h2>
            <div class="form-group">
             <label for="name">Name:</label>
 <input type="text" id="name" class="input-field" placeholder=" name">
                 </div>
           <div class="form-group">
            <label for="email">Email:</label>
<input type="email" id="email" class="input-field" placeholder=“email">
                          </div>
                          <div class="form-group">
                              <label for="email">Mobile</label>
                              <input type="tel" id="mobile" class="input-field"
placeholder="Enter your Mobile NUmber">
                          </div>
                          <div class="form-group">
                              <label for="message">Message:</label>
                              <textarea id="message" class="input-field"
placeholder="Your message"></textarea>
                          </div>
                              <button type="submit" id="submit-
button">Submit</button>
                       </form>
                    </body>
                 </html>
```

```
/* css /*
#contact-form {
                    background-color:pink;
                    border: 1px solid #b882b0;
                    padding: 20px;
                    margin: 50px auto;
                    width: 50%;
                    border-radius: 8px;
                    box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);
                  }
        .form-group {
                    margin-bottom: 15px;
                  }
    .input-field {
                    width: 100%;
                    padding: 10px;
                    margin-top: 5px;
                    border: 1px solid rgb(206, 122, 136);
                    border-radius: 4px;
                    font-size: 16px;
                    box-sizing: border-box;
                  }
#submit-button {
                    background-color: #007bff;
                    color: white;
                    border: none;
```

```
padding: 10px 20px;
                        font-size: 16px;
                        border-radius: 4px;
                        cursor: pointer;
                        transition: background-color 0.3s ease;
                    }

        #submit-button:hover {
                            background-color: #0056b3;
                        }
```

2. 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?

ANSWER :-

- The CSS box model describes how elements are structured and how their sizes are calculated in a web page layout. Each element is a rectangular box comprising the following components:

- A) Content

  - The innermost part of the box, where the text, images, or other content resides.

  - The size of the content can be set using width and height.

B) Padding:

- The space between the content and the border.

- It adds space inside the element but does not include any visible styling.

- Increasing padding increases the total size of the element.

C) Border:

- Surrounds the padding and content.
- Defines the width, style, and color of the border (e.g., border: 2px solid black).
- The border width contributes to the total size of the element.

D) Margin:

- The outermost layer that creates space between the element and neighboring elements.
- Margins do not affect the element's internal dimensions but influence spacing in the layout.

- How Each Affects the Element's Size:
- The element's total size (box size) is calculated as:

Total Width = Content Width + Padding (left + right) + Border (left + right) + Margin (left + right)

Total Height = Content Height + Padding (top + bottom) + Border (top + bottom) + Margin (top + bottom)

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

ANSWER:-

- Box-sizing is a CSS property that determines how the total width and height of an element are calculated.
- A) Content-box (Default):
  - The width and height properties apply only to the content.
  - Padding and border are added outside the specified width and height, increasing the element's total size.

-EX:

```
div {
  box-sizing: content-box; /* Default */
  width: 200px;
  padding: 20px;
  border: 10px solid black;
}
```

- B) Border-box:
  - The width and height properties include the content, padding, and border.
  - Padding and border are subtracted from the specified width and height, keeping the total size constant.

- EX:

```
div {
  box-sizing: border-box;
  width: 200px;
```

padding: 20px;

border: 10px solid black;

}

- Which is the default?
  - content-box is the default box-sizing value.

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.

ANSWER :-

```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Profile Card</title>
  <link rel="stylesheet" href="style.css">
</head>
<body>
  <div class="card">
    <img src="../pAnime/04.jpg" alt="Profile Picture" class="profile-pic">
    <h2 class="user-name">Harry Potter</h2>
    <p class="user-bio">Web Developer | Python | Fullstack | Lifelong
Learner</p>
    <button class="follow-button">Follow</button>
  </div>
</body>
</html>
```

```
body {
    display: flex;
    justify-content: center;
    align-items: center;
    min-height: 100vh;
    margin: 0;
    background-color: #f4f4f4;
    font-family: Arial, sans-serif;
}
.card {
    background-color: #fff;
    width: 300px;
    border: 1px solid #ddd;
    border-radius: 10px;
    padding: 20px;
    box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);
    text-align: center;
    box-sizing: border-box;
}
.profile-pic {
    width: 100px;
    height: 100px;
    border-radius: 50%;
    border: 3px solid #007bff;
    margin-bottom: 15px;
    box-sizing: content-box;
}
```

```
.user-name {
    font-size: 1.5rem;
    color: #333;
    margin: 10px 0;
}
.user-bio {
    font-size: 1rem;
    color: #666;
    margin: 10px 0 20px;
    padding: 10px;
    border: 1px solid #ddd;
    border-radius: 5px;
    box-sizing: border-box;
}
.follow-button {
    background-color: #007bff;
    color: white;
    padding: 10px 20px;
    font-size: 1rem;
    border: none;
    border-radius: 5px;
    cursor: pointer;
    transition: background-color 0.3s;
    box-sizing: content-box;
}
.follow-button:hover {
    background-color: #0056b3;
}
```

3) 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.

ANSWER:-

- CSS Flexbox (Flexible Box Layout) is a layout module in CSS designed to create flexible and efficient alignment and distribution of items within a container, even when their sizes are dynamic. It provides control over spacing, alignment, and the order of items, making it particularly useful for responsive designs.

- Simplifies the layout of items in one-dimensional space (row or column).

- Aligns and distributes items dynamically based on available space.

- Terms:

A) Flex-container:

- The parent element where the Flexbox layout is applied.
- Defined by setting display: flex or display: inline-flex on an element.
- Contains the child elements (flex-items) that are laid out using Flexbox properties.
- Ex:

```
.container {
        display: flex;
    }
```

B) Flex-item:

- The child elements of a flex-container.
- Their size and positioning are controlled using Flexbox properties.
- Ex:

```
<div class="container">
    <div class="item">1</div>
    <div class="item">2</div>
    <div class="item">3</div>
</div>
```

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

ANswer :-

A) justify-content:

- Controls the horizontal alignment of flex-items along the main axis.
- Applies spacing between or around items.
- Common values:

flex-start: Items align to the start of the container (default).

flex-end: Items align to the end of the container.

center: Items are centered along the main axis.

space-between: Items are evenly spaced, with no space at the ends.

space-around: Items are evenly spaced, with half-sized spaces at the ends.

space-evenly: Items are evenly spaced, including equal space at the ends.

- Example:

```
.container {
    justify-content: space-between;
}
```

B) align-items:

- Controls the vertical alignment of flex-items along the cross axis.

- Common values:

stretch: Items stretch to fill the container (default).

flex-start: Items align at the start of the cross axis.

flex-end: Items align at the end of the cross axis.

center: Items are centered along the cross axis.

baseline: Items align based on their baseline.

- Example:

```
.container {
    align-items: center;
}
```

C) flex-direction:

- Specifies the direction of the main axis, determining how flex-items are placed.

- Common values:

row: Items are placed from left to right (default).

row-reverse: Items are placed from right to left.

column: Items are placed from top to bottom.

column-reverse: Items are placed from bottom to top.

- Example:

```
.container {
    flex-direction: column;
}
```

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.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Flexbox Layout</title>
    <link rel="stylesheet" href="style.css">
</head>
<body>
    <header class="header">
        <h1>My Flexbox Layout</h1>
    </header>
    <div class="container">
        <aside class="sidebar">
            <h2>Sidebar</h2>
            <ul>
                <li>Link 1</li>
                <li>Link 2</li>
                <li>Link 3</li>
            </ul>
        </aside>
        <main class="main-content">
            <h2>Main Content</h2>
            <p>
                This is the main content area. Here you can place articles, images, or any other content.
            </p>
```

```
 </main>
    </div>
    <footer class="footer">
        <p>&copy; 2024 My Website</p>
    </footer>
</body>

</html>
```

```
* {
    margin: 0;
    padding: 0;
    box-sizing: border-box;
    font-family: Arial, sans-serif;
}
body {
    display: flex;
    flex-direction: column;
    min-height: 100vh;
}
.header {
    background-color: #333;
    color: white;
    padding: 1rem;
    text-align: center;
}
.container {
    display: flex;
    flex: 1;
    flex-wrap: wrap;
}
.sidebar {
    background-color: #f4f4f4;
    padding: 1rem;
    flex: 1 1 20%;
```

```
 min-width: 200px;
    display: flex;
    flex-direction: column;
    align-items: center;
}
.main-content {
    background-color: #fff;
    padding: 1rem;
    flex: 2 1 60%;
    min-width: 300px;
}
.footer {
    background-color: #333;
    color: white;
    text-align: center;
    padding: 1rem;
}
@media (max-width: 768px) {
    .container {
        flex-direction: column;
    }
}
```

4. CSS Grid

Theory Assignment

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

ANSWER:-

- CSS Grid is a powerful layout system that allows developers to create grid-based designs for web pages. It enables precise control over rows and columns, making it easier to align and position items.

- Difference Between CSS Grid and Flexbox

Aspect CSS Grid	Flexbox	
Layout Direction	Two-dimensional (rows + columns)	One-dimensional (row or column)
Alignment	Aligns items across multiple axes	Aligns items in a single axis
Complexity	Better for complex grid-based layouts	Better for simpler, linear layouts

Control Provides explicit control over rows and columns Controls layout item by item

- When to Use Grid Over Flexbox:

Use CSS Grid when you need a two-dimensional layout, such as arranging rows and columns.

Use Flexbox for simpler, single-axis layouts where elements flow in a row or column.

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

ANswer:-

A) grid-template-columns:

- Defines the number and size of columns in a grid.
- Ex:-

```
.grid-container {
    display: grid;
    grid-template-columns: 100px 200px auto;
}
```

B) grid-template-rows:

- Defines the number and size of rows in a grid.
- Ex:-

```
.grid-container {
    display: grid;
    grid-template-rows: 150px 100px;
}
```

C) grid-gap (or gap):

- Adds spacing between rows and columns.
- Example:

```
.grid-container {
    display: grid;
    grid-gap: 10px;
    grid-template-columns: 1fr 1fr 1fr;
```

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.

ANSWER :-

```
 <!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>3x3 Product Grid</title>
    <link rel="stylesheet" href="style.css">
</head>
<body>
    <h1 class="title">Product Grid</h1>
    <div class="grid-container">
        <div class="card">
            <img src="04.jpg" alt="Product 1">
            <h3>Product 1</h3>
            <p>$19.99</p>
        </div>
        <div class="card">
            <img src="wp12621902-dark-anime-4k-pc-wallpapers.jpg" alt="Product 2">
            <h3>Product 2</h3>
            <p>$29.99</p>
        </div>
        <div class="card">
            <img src="wp5136647-dark-anime-theme-wallpapers.jpg" alt="Product 3">
            <h3>Product 3</h3>
            <p>$39.99</p>
        </div>
        <div class="card“>
```

```
<img src="wp6335580-dark-anime-ultra-hd-wallpapers.jpg" alt="Product 4">
            <h3>Product 4</h3>
            <p>$49.99</p>
        </div>
        <div class="card">
            <img src="04.jpg" alt="Product 5">
            <h3>Product 5</h3>
            <p>$59.99</p>
        </div>
        <div class="card">
            <img src="wp12621902-dark-anime-4k-pc-wallpapers.jpg" alt="Product 6">
            <h3>Product 6</h3>
            <p>$69.99</p>
        </div>
        <div class="card">
            <img src="wp5136647-dark-anime-theme-wallpapers.jpg" alt="Product 7">
            <h3>Product 7</h3>
            <p>$79.99</p>
        </div>
        <div class="card">
            <img src="wp6335580-dark-anime-ultra-hd-wallpapers.jpg" alt="Product 8
            <h3>Product 8</h3>
            <p>$89.99</p>
        </div>
        <div class="card">
            <img src="04.jpg" alt="Product 9">
            <h3>Product 9</h3>
```

```
 <p>$99.99</p>
        </div>
    </div>
</body>

</html>
```

```
* {
    margin: 0;
    padding: 0;
    box-sizing: border-box;
    font-family: Arial, sans-serif;
}
body {
    background-color: #f4f4f4;
    padding: 20px;
    text-align: center;
}
.title {
    margin-bottom: 20px;
}
```

```
.grid-container {
    display: grid;
    grid-template-columns: repeat(3, 1fr);
    grid-gap: 20px;
    max-width: 1200px;
    margin: 0 auto;
}
.card {
    background: #fff;
    border: 1px solid #ddd;
    border-radius: 8px;
    box-shadow: 0 4px 6px rgba(0, 0, 0, 0.1);
    overflow: hidden;
    transition: transform 0.3s, box-shadow 0.3s;
}
.card img {
    width: 100%;
    height: auto;
}

.card h3 {
    margin: 10px 0;
    font-size: 1.2em;
}
```

```
.card p {
    color: #666;
    margin-bottom: 10px;
}
.card:hover {
    transform: translateY(-10px);
    box-shadow: 0 8px 16px rgba(0, 0, 0, 0.2);
}
@media (max-width: 768px) {
    .grid-container {
        grid-template-columns: 1fr 1fr;
        }
}

@media (max-width: 480px) {
    .grid-container {
        grid-template-columns: 1fr;
    }
}
```

5. Responsive Web Design with Media Queries

Theory Assignment

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

ANSWER:-

- Media queries are a feature in CSS that allow developers to apply styles based on specific conditions, such as screen width, height, device type, or orientation. Media queries enable responsive web design, where the layout and appearance of a webpage adapt dynamically to various devices and screen sizes.

- Importance of Media Queries in Responsive Design:

- Device Compatibility: Websites can adapt to desktops, tablets, and mobile devices, ensuring a better user experience.

- Improved Readability: Text size, margins, and padding can be adjusted to ensure content remains legible on smaller screens.

- Performance Optimization: Specific styles can be applied only when needed, reducing unnecessary rendering.

- Enhanced User Experience: By tailoring the layout to different screen sizes, users enjoy a consistent and optimized
experience across devices.

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

ANSWER:-

- Below is an example of a media query that reduces the font size for smaller screens

```
body {
    font-size: 18px;
}
@media (max-width: 600px) {
  body {
    font-size: 14px; /* Adjust the font size for small screens */
  }
}
```

- In this example:
  - @media checks if the screen width is 600px or less.
  - If the condition is met, the font size is reduced to 14px for better readability on smaller devices.

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.

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Responsive Web Design</title>
    <link rel="stylesheet" href="style.css">
</head>

<body>
    <nav class="navbar">
        <div class="logo">MyWebsite</div>
        <ul class="nav-links">
            <li><a href="#">Home</a></li>
            <li><a href="#">About</a></li>
            <li><a href="#">Services</a></li>
            <li><a href="#">Contact</a></li>
        </ul>
    </nav>
    <div class="content">
        <div class="column column-1">
            <h2>Column 1</h2>
            <p>This is the first column. Content can go here to describe products,
services, or any topic of interest.
            </p>
        </div>
        <
```

```
div class="column column-2">
            <h2>Column 2</h2>
            <p>This is the second column. Additional information or supporting
content can go here.</p>
        </div>
    </div>
    <footer class="footer">
        <p>&copy; 2024 My Responsive Website</p>
    </footer>
</body>

</html>
```

```
* {
    margin: 0;
    padding: 0;
    box-sizing: border-box;
    font-family: Arial, sans-serif;
}
body {
    line-height: 1.6;
    background-color: #f9f9f9;
}
```

```
.navbar {
    background-color: #333;
    color: #fff;
    display: flex;
    justify-content: space-between;
    align-items: center;
    padding: 10px 20px;
}
.navbar .logo {
    font-size: 1.5rem;
    font-weight: bold;
}
.navbar .nav-links {
    list-style: none;
    display: flex;
}
.navbar .nav-links li {
    margin-left: 20px;
}

.navbar .nav-links a {
    text-decoration: none;
    color: #fff;
    font-size: 1rem;
    transition: color 0.3s;
}
.navbar .nav-links a:hover {
    color: #ff9800;
```

```
}
.content {
    display: flex;
    justify-content: space-between;
    margin: 20px;
    gap: 20px;
}
.column {
    background-color: #fff;
    padding: 20px;
    border: 1px solid #ddd;
    border-radius: 8px;
    box-shadow: 0 2px 5px rgba(0, 0, 0, 0.1);
    flex: 1;
}
.column h2 {
    margin-bottom: 10px;
}
.footer {
    background-color: #333;
    color: #fff;
    text-align: center;
    padding: 10px 0;
    margin-top: 20px;
}
```

```
@media (max-width: 768px) {
    .content {
        flex-direction: column;
 }
    .navbar {
        flex-direction: column;
        align-items: flex-start;
    }
    .navbar .nav-links {
        flex-direction: column;
        margin-top: 10px;
    }
    .navbar .nav-links li {
        margin-left: 0;
        margin-bottom: 5px;
    }
}
@media (max-width: 600px) {
    body {
        font-size: 16px;
    }
    .navbar .logo {
        font-size: 1.2rem;
    }
    .column {
        padding: 10px;
    }
}
```

6. 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?

ANSwer :-

- A) Web-Safe Fonts:

- Web-safe fonts are pre-installed on most devices (Windows, Mac, Android, iOS).

- They ensure text appears consistently across all browsers and operating systems.

- Ex:- Arial, Times New Roman, Verdana, Georgia, and Courier New.

- B) Custom Web Fonts:

- Custom fonts (like those from Google Fonts or Adobe Fonts) need to be loaded from external resources.

- They allow developers to use unique fonts, enhancing the design and branding of a website.

- Example: Roboto, Open Sans, Lato, or Poppins.

- Why Use Web-Safe Fonts Over Custom Fonts?:

- Performance: Web-safe fonts load faster as they do not require external downloads.

- Reliability: They are always available, ensuring consistent display across devices.

- Fallback: In cases where internet speed is slow, web-safe fonts act as reliable fallbacks.

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

Answer:-

- The font-family property in CSS specifies the font to be used for an element.
- It can accept a list of fonts as a fallback mechanis :

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

- How to Apply a Custom Google Font:
  - Go to Google Fonts.
  - Choose a font and copy its <link> tag or @import rule.
  - Add the link to your HTML file.
- EXAMPLE

Step 1 :-

- Include the Google Font in the <head> of your HTML:
- <link href="https://fonts.googleapis.com/css2?family=Roboto:wght@400;700&family=Playfair+Display:wg

Step 2 :-

- Apply the font using font-family in CSS
- body {

font-family: 'Roboto', sans-serif; /* Body content 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

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Responsive Blog Post</title>
    <link

href="https://fonts.googleapis.com/css2?family=Roboto:wght@400;700&family=Pla
yfair+Display:wght@700&display=swap"
        rel="stylesheet">
    <link rel="stylesheet" href="style.css">
</head>

<body>
    <div class="blog-post">
        <h1 class="title">The Art of Web Typography</h1>
        <h2 class="subtitle">Enhancing readability and design through fonts</h2>
        <div class="content">
            <p>
                Typography plays a crucial role in web design. Choosing the right fonts
can elevate a website's
                aesthetics while ensuring content remains easy to read. In modern web
development, custom fonts like
                those from Google Fonts have made it simple to implement beautiful
typography without compromising on
                performance.
```

```
 </p>
            <p>
                However, it's essential to balance creativity and practicality.
Combining too many fonts can make the
                design look chaotic. Typically, designers choose one font for headings
and another for body text to
                maintain harmony.
            </p>
            <p>
                Using web-safe fonts as fallbacks ensures your site looks consistent,
even if custom fonts fail to load.
                Responsiveness is also key—adjusting font sizes, line spacing, and
margins for smaller screens improves
                readability on all devices.
            </p>
        </div>
    </div>
</body>

</html>
```

```
* {
    margin: 0;
    padding: 0;
    box-sizing: border-box;
}
body {
    font-family: 'Roboto', sans-serif;
    line-height: 1.6;
    background-color: #f8f9fa;
    color: #333;
    padding: 20px;
}
.blog-post {
    max-width: 800px;
    margin: 0 auto;
    background: #fff;
    padding: 20px;
    border-radius: 8px;
    box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);
}
.title {
    font-family: 'Playfair Display', serif;
       font-size: 2.5rem;
    font-weight: 700;
    margin-bottom: 10px;
    color: #2c3e50;
    text-align: center;
```

```
subtitle {
    font-family: 'Roboto', sans-serif;
    font-size: 1.5rem;
    font-weight: 400;
    margin-bottom: 20px;
    color: #7f8c8d;
    text-align: center;
}
.content p {
    font-size: 1.1rem;
    line-height: 1.8;
    margin-bottom: 20px;
    text-align: justify;
}
@media (max-width: 768px) {
    .title {
        font-size: 2rem;
    }

    .subtitle {
        font-size: 1.2rem;
    }

    .content p {
        font-size: 1rem;
    }
}
```

```
@media (max-width: 480px) {
    body {
        padding: 10px;
    }

    .title {
        font-size: 1.8rem;
    }

    .subtitle {
        font-size: 1rem;
    }

    .content p {
        font-size: 0.9rem;
    }
}
```