Module 16 - CSS in Full Stack Course

CSS Selectors & Styling

Theory Assignment

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

Ans: CSS selectors are used to select the content you want to style. Selectors are the part of CSS rule set. CSS selectors select HTML elements according to its id, class, type, attribute etc.

- **Element Selector**
  - The element selector selects the HTML element by name.
  - **Example:**

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport"
content="width=device-width, initial-scale=1.0">
    <title>Document</title>
    <style>
        p{
          text-align:center;
          font-family:Times New Roman;
          font-weight:600;
          color:pink;
        }
    </style>
</head>
<body>
    <p>India is Best Country in the World</p>
```

```
</body>
</html>
```

➢ Class Selector

- This selector targets elements that have a specific class attribute. It begins with a dot (.).
- **Example:**

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport"
content="width=device-width, initial-scale=1.0">
    <title>Document</title>
    <style>
        .vel{
            text-align:center;
            font-family:Times New Roman;
            font-weight:600;
            color:pink;
        }
        p.demo{
            text-align:center;
            font-family:Times New Roman;
            font-weight:500;
            color:blue;
        }
    </style>
</head>
<body>
    <p class=”vel”>India is Best Country in the
World</p>
    <p class=”demo”>India and Russia is Good
Friend</p>
```

```
</body>
</html>
```

➢ Id Selector

- The id selector selects the id attribute of an HTML element to select a specific element. An id is always unique within the page so it is chosen to select a single, unique element. It is written with the hash character (#), followed by the id of the element. It is highest priority of the selector.
- **Example:**

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport"
content="width=device-width, initial-scale=1.0">
<title>Document</title>
<style>
#first{
text-align:center;
font-family:Times New Roman;
font-weight:500;
color:yellow;
}
</style>
</head>
<body>
<p id=”first”>India and Russia Both are Good Friendly
Countries </p>
</body>
</html>
```

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

Ans: CSS specificity is a crucial concept that determines which CSS rules apply to an element when multiple styles could potentially affect it. It is based on the principle that some selectors are more specific than others, allowing for a hierarchy of importance when it comes to styling elements.

❖ How Specificity Works:

- Specificity is calculated based on the components used in a CSS selector. The more specific the selector, the higher the specificity.

1. Inline Styles:

- Inline styles Directly applied in the HTML element using the style attribute. These have the first highest specificity. Inline styles are applied directly to the element in the HTML using the style attribute. The inline CSS is also a method to insert style sheets in HTML document
- **Example: Hello, world! **

2. Id Selectors:

- Second highest priority, identified by the unique id attribute of an element
- **Example: #vel{ color: red;} Hello,World **

3. Class Selector:

- Third highest priority, targeted using class names.
- **Example: .vel{ color: pink;} Hello Wolrld **

❖ Resolving Conflicts:

- **Inline styles** will always override other rules.
- If there are no inline styles, the rule with the **highest specificity** wins.
- ID selector is present, it will override any class, attribute, or element selectors.
- If multiple class selectors are present, the one with the highest specificity will apply.

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

Ans:

❖ **Difference Between Internal, External, And Inline CSS**

Feature	Internal CSS	External CSS	Inline CSS
Location	It is used within <head> section of HTML document.	It is used in a separate .css file.	It is used within HTML tag using the style attribute.
Sector Scope	Affects multiple elements within the same HTML element.	Affects multiple HTML documents or an entire website.	Affects a single element or a group of elements.
Priority	Medium priority. Overrides external styles but can be overridden by inline styles.	Lowest priority. Can be overridden by both inline and internal styles.	Highest priority. Overrides internal and external styles.
Reusability	Can be reused on multiple elements within the same HTML document.	Can be reused on multiple HTML documents or an entire website.	Not reusable. Styles need to be repeated for each element.
File Size	Internal styles are part of the HTML file, which increases the file size.	External styles are in a separate file, which reduces the HTML file size and can be cached for faster page loads.	Inline styles increase the HTML file size, which can affect the page load time.

- **Advantages of Inline CSS**
  - Inline CSS Overrides external and internal styles with higher specificity.
  - Using style attributes we can provide styles directly to our HTML elements.
  - Inline styles don't require separate CSS files, potentially reducing HTTP requests.
  - No need to create and upload a separate document as in the external style
- **Disadvantages of Inline CSS**
  - Inline CSS does not provide browser cache advantages.
  - Inline CSS styles cannot be reused anywhere else.
  - Adding style attributes to every HTML element is time-consuming.
  - Styling multiple elements can increase your pages size and download time, impacting overall page performance.
- **Advantages of Internal CSS**
  - You need not upload several files because the code will only be added to one HTML file.
  - It is used multiple elements within the same HTML element.
  - You can include CSS in HTML texts by utilising a type of CSS known as internal CSS. A single HTML web page's layout can be designed, and styles can be modified within the HTML code.
  - You need not upload several files because the code will only be added to one HTML file.
  - No dependency on external resources ➔ No issues with broken file links or incorrect paths.
- **Disadvantages of Internal CSS**
  - Limited Reusability ➔ Styles are restricted to one page.
  - Larger page sizes ➔ Embedding CSS in HTML increases file size.
  - Difficult to Maintain for Large Projects
  - Performance Issues on Larger Sites

- Styles are not cached, slowing down page loading.

➢ **Advantages of External CSS**

- A significant advantage of external CSS is its reusability, as a single stylesheet can be linked across multiple web pages, significantly reducing repetition and redundancy.
- A performance advantage of external CSS is that it uses browser caching to improve loading times after the initial visit.
- **Easier Maintenance and Scalability** ➔ Centralized styling for all pages.
- **Flexibility** ➔ Ability to use multiple CSS files for different purposes or devices.
- **Collaboration-friendly** ➔ Multiple developers can work simultaneously on different aspects of the project.
- **SEO benefits** ➔ Cleaner HTML and faster loading improve search engine rankings.

➢ **Disadvantages of External CSS**

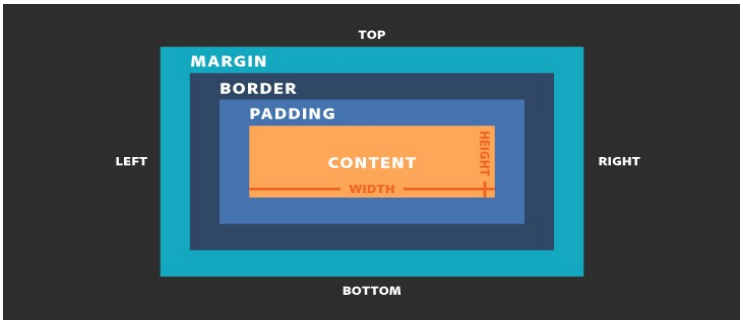
- Caching issues may result in outdated content being displayed to users.
- File management complexity, especially for large projects.
- Caching issues may result in outdated content being displayed to users.
- Dependency on external servers introduces risks like downtime or broken links.
- Increased HTTP requests can slow down the initial page load.
- Slower rendering due to blocking external resources.

5. CSS Box Model

Theory Assignment

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

Ans: The CSS box model is a fundamental concept in web design, and it defines how elements are structured and how their sizes are calculated on a webpage.



- **Content**
  - The content of the box, where text and images appear.
  - The content area is the central part of the CSS box model, containing the main content (e.g., text, images, videos, or elements like <p> or). It can be styled with CSS properties like height and width.
- **Padding**
  - The padding area is the space between the content and the border of an element. The padding is transparent.
  - It includes the areas highlighted in light green and skin color in the example.
  - The distance between the content edge and the border is the padding.
  - The border marks the end of the padding area.
  - The padding area contributes to the element's total dimensions.
  - Padding can be adjusted using CSS properties.
- **Border**
  - A border that goes around the padding and content.
  - The area that marks the end of an element is called as the border it is the outer fencing for the element.
  - The CSS border is a shorthand property used to set the border on an element.

- **Margin**
    - The area outside the border element is called margin. The margin is completely transparent and doesn't have any background color. It clears an area around the element.
    - Basically this area depends on the parent of the element.
    - Top, bottom, left and right margin can be changed independently using separate properties. You can also change all properties at once by using shorthand margin property.

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

Ans: In CSS, the box-sizing property controls how the total width and height of an element are calculated, especially when padding and borders are added. The two most commonly used values are content-box and border-box.

Difference Between Border-Box and Content-Box in CSS

Border-Box	Content-Box
The width includes content, padding, and borders.	Width applies to the content only, padding and borders are added outside.
The element's total width/height remains as specified, with padding and borders included inside.	The element's total width/height becomes larger than the specified value due to padding and borders.
More predictable, as the element stays within the defined size.	Can lead to unexpected overflow if you add padding/borders.
border-box is used for precise layout control (often in modern CSS resets).	content-box is the default value.
The width/height you set includes the content, padding, and borders, so the total size remains exactly as specified.	The width/height you set is just for the content axrea, and padding/borders are added outside, increasing the total size.
If you set a table width of 300px, the total table width remains 300px, and padding/borders are inside.	If you set a table width of 300px, the table may be wider than 300px due to padding and borders.

6. CSS Flexbox

Theory Assignment

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

Ans: CSS Flexbox (Flexible Box Layout) is a layout model in CSS that allows you to design complex layouts more efficiently and with less code, especially when it comes to aligning and distributing space between items in a container.

- It is a one-dimensional layout model, meaning it can handle layouts in a **row** (horizontal axis) or **column** (vertical axis), but not both simultaneously.
- Flexbox provides powerful and flexible tools for distributing space and aligning elements, and it automatically adjusts the layout based on the available space.

How is it useful for layout design

- **Alignment**: Flexbox makes aligning items (both vertically and horizontally) easier than using traditional methods like floats or positioning.
- **Space Distribution**: You can distribute space evenly between items and align them in a way that adapts to the container's size.
- **Responsive Design**: Flexbox allows the layout to automatically adjust as the viewport or container size changes, making it ideal for responsive design.
- **Simplified Layouts**: Flexbox simplifies the creation of complex layouts without relying on floats, grids, or clearfix hacks.
- **Handling Dynamic Content**: Flexbox is great for dealing with dynamic or unpredictable content. Items can adjust automatically when the content size changes, making it very useful when you're dealing with dynamic elements that change in size (like images, text blocks, or buttons).

- **Compatibility with Modern Browsers:** Flexbox is supported by all modern browsers, and it's considered a reliable and powerful layout tool. It’s even designed to handle cases where traditional CSS methods might have issues, such as dealing with different screen sizes or unknown content sizes.

- **Flex Container**
  - The flex container can manage the direction, alignment, spacing, and wrapping behavior of its items, making it a powerful tool for building flexible, responsive, and complex layouts with minimal effort.
  - A **flex container** is an element that enables the Flexbox layout model. By setting display: flex; you turn a container into a flex container, allowing you to control the layout and alignment of its child elements (flex items).
- **Flex Item**
  - A flex item is any direct child of a flex container. When you apply display: flex; (or display: inline-flex;) to a parent element, all of its direct children automatically become flex items. These flex items are then controlled by the flex container, and their layout, alignment, and spacing can be adjusted using various Flexbox properties.

2. Describe the properties justify-content, align-items, and flex-direction used in Flexbox.

Ans: In Flexbox, the properties justify-content, align-items, and flex-direction are fundamental for controlling the positioning, alignment, and layout of flex items within a flex container. These properties help you easily control the alignment and distribution of items both along the main axis and the cross axis.

1. **justify-content**
   - The justify-content property aligns flex items along the main axis (the axis defined by flex-direction). This property is used to control the distribution of space between and around flex items in the container.

- **Justify-Content Values:**
    - **flex-start (default):** Items are aligned at the start of the main axis (left for row, top for column).
    - **flex-end**: Items are aligned at the **end** of the main axis (right for row, bottom for column).
    - **center**: Items are aligned at the **center** of the main axis.
    - **space-between**: Items are evenly distributed along the main axis, with **no space at the ends**.
    - **space-around**: Items are evenly distributed along the main axis, with **equal space around each item**.
    - **space-evenly**: Items are evenly distributed with **equal space between and around** the items.

2. align-items

- The align-items property specifies the default alignment for items inside a flexbox or grid container.
- In a flexbox container, the flexbox items are aligned on the cross axis, which is vertical by default (opposite of flex-direction).
- **Align-Items Values:**
    - **flex-start:** Items are aligned at the start of the cross axis (top for row, left for column).
    - **flex-end:** Items are aligned at the end of the cross axis (bottom for row, right for column).
    - **center**: Items are aligned at the center of the cross axis.
    - **baseline**: Items are aligned along their baseline (usually the baseline of text).
    - **stretch (default):** Items are stretched to fill the container along the cross axis.

3. flex-direction

- The flex-direction property defines the direction of the main axis in the flex container, which determines how the flex items are arranged. It can be used to change the orientation of the flex container and the direction in which the items are placed.
- **Flex-Direction Values:**

- **row (default):** Flex items are arranged horizontally, from left to right.
- **row-reverse**: Flex items are arranged **horizontally** in reverse order, from right to left.
- **column**: Flex items are arranged **vertically**, from top to bottom.
- **column-reverse**: Flex items are arranged **vertically** in reverse order, from bottom to top.

7.CSS Grid

Theory Assignment

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

Ans: CSS Grid Layout is a two-dimensional layout system that allows web designers and developers to create complex layouts using rows and columns. Flexbox, which is designed for one-dimensional layouts (either rows or columns), **CSS Grid** provides control over both dimensions at the same time. This makes it especially powerful for designing more complex, structured, and responsive web layouts.

- **Grid Container:** You create a grid container and place items within it. The items can be placed in specific grid cells, spanning across multiple rows or columns, or follow a more flexible layout.
- **Defining Grid:** You define a grid by setting display: grid on the container and specifying the rows and columns using properties like grid-template-rows, grid-template-columns, or grid-template-areas.

❖ Defferences Between Grid And Flexbox in CSS

	CSS Grid	Flexbox CSS

Purpose	**Grid** is for complex layouts with multiple rows and columns.(Mandatory)	**Flexbox** is for simpler, one-dimensional layouts where items are arranged in a row or column.
Dimensionality	CSS Grid is two-dimensional: It allows you to control both rows and columns at the same time.	Flexbox is one-dimensional: You can only align items in a single row or column.
Control	**Grid** offers more precise control over both dimensions (rows and columns) and allows you to position items at specific grid locations.	**Flexbox** provides control over the arrangement of items in a single dimension (either a row or column), with flexible alignment and spacing.
Use Cases	**CSS Grid** is ideal for larger-scale layouts, like grid-based page layouts where you need control over both axes (e.g., creating a complete webpage layout, grids for articles, sidebars, etc.).	**Flexbox** is best suited for smaller-scale layouts, where you're dealing with a single row or column of items (e.g., navigation bars, toolbars, or any simple linear layout).
Mandatory	**CSS Grid** is mandatory of row and column.	**Flexbox** is not mandatory of row and column.
Item Placement	**CSS Grid** you can precisely place items into specific rows and columns or even make them span multiple rows or columns.	**Flexbox** the items are laid out based on the order of elements and flow in a row or column. You can control alignment and distribution, but it's more about managing a line of content.
Flexibility	**CSS Grid** provides more advanced control, allowing items to be flexible and positioned in a two-dimensional space.	**Flexbox** excels in distributing items along a single axis (either row or column) and making them flexible within that context.
Responsiveness	**Both** Flexbox and CSS Grid are responsive tools, but CSS Grid often offers more powerful tools for creating complex responsive layouts due to its ability to define grid areas and track sizes based on media queries.	**Both** Flexbox and CSS Grid are responsive tools, but Flexbox is cannot be create complex responsive layouts as well as cannot be track sized based on media queries.

❖ <u>When to Use Grid Over Flexbox:</u>

- **Complex Layouts:** When you need to create a layout with both rows and columns (like a traditional grid system), CSS Grid is the

better choice. It gives you more control over placement in two dimensions.

- **Item Placement**: If you need to place items in specific grid areas or make items span multiple rows and columns, CSS Grid is ideal.
- **Two-Dimensional Control**: If your layout is more complex and requires control of both **horizontal** and **vertical** placements at the same time, Grid is far superior.

❖ **When to Use Flexbox Over Grid:**

- **One-Dimensional Layouts:** Flexbox is the go-to choice for simpler layouts where items align in a single row or column (e.g., horizontal navigation bar or a column of items).
- **Alignment and Spacing:** If you need to easily distribute space or align items along a single axis, Flexbox shines.
- **Dynamic/Responsive Designs:** Flexbox is great for responsive designs where elements need to adjust dynamically in one direction.

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

Ans: In CSS Grid, the properties grid-template-columns, grid-template-rows, and grid-gap are essential for defining and controlling the layout of grid items.

➢ grid-template-columns

- The grid-template-columns property specifies the number (and the widths) of columns in a grid layout. The values are a space separated list, where each value specifies the size of the respective column.
- **Syntax: grid-template-columns: <value> <value> ...;**

➢ grid-template-rows

- The grid-template-rows property specifies the number (and the heights) of the rows in a grid layout. The values are a space-separated list, where each value specifies the height of the respective row.
- **Syntax: grid-template-columns: value1 value2 value3 ...;**

- **grid-gap:**
  - This property defines the spacing between rows and columns in a grid. It creates a gap between grid items, both horizontally (columns) and vertically (rows).
  - **Syntax: grid-gap: <row-gap> <column-gap>;**
- **Example:**

```
<!DOCTYPE html>
<html lang="en">
        <head>
                <meta charset="UTF-8">
                <meta name="viewport" content="width=device-width,
initial-scale=1.0">
                <title>Document</title>
  <style>
        .container {
                        display: grid;
                        grid-template-columns: 200px 1fr;
                        grid-template-rows: 100px 200px;
                        gap: 10px;
                   }
                .item1 {
                         background: lightblue;
                }
        .item2 {
                        background: lightgreen;
                         }
     .item3 {
                background: lightcoral;
                  }
   .item4 {
     background: lightyellow;
    }
   </style>
</head>
```

```
<body>
        <div class="container">
          <div class="item1">Item 1</div>
          <div class="item2">Item 2</div>
          <div class="item3">Item 3</div>
          <div class="item4">Item 4</div>
         </div>
</body>
</html>
```

8-Responsive Web Design with Media Queries

Theory Assignment

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

Ans: **Media queries** in CSS are a powerful feature that allows you to apply different styles to a webpage depending on the characteristics of the device or viewport. These characteristics can include things like the screen width, height, resolution, orientation (landscape or portrait), and more. Media queries enable the creation of responsive designs, where the layout and style of a webpage adjust based on the user's device, ensuring an optimal viewing experience across different screen sizes.

- **Why are Media Queries Important for Responsive Design?**
  - Responsive design is all about ensuring that a website looks good and is usable on a wide variety of devices, from small smartphones to large desktop monitors. Media queries are key to this because they allow developers to write conditional CSS rules that only apply under specific conditions. This helps websites adapt to different screen sizes and resolutions, making them user-friendly across all devices.
  - Without media queries, you'd have to create separate stylesheets for each device or rely on fixed layouts, which could make your

website difficult to navigate or look unappealing on certain screens.

- **How Do Media Queries Work?**
  - A media query consists of a media type and one or more expressions that test the features of the device (such as screen width or resolution). You can think of a media query as a way to ask the browser, "Is the screen width less than 600px?" and, if so, apply a particular style.
- **Syntax**: **@media media-type and (condition) { /* CSS styles here */ }**

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

Ans: It uses the @media rule to include a block of CSS properties only if a certain condition is true.

- **Example:** If the browser window is 600px or smaller, the background color will be lightgreen.

```
<!DOCTYPE html>
<html lang="en">
<head>
        <meta charset="UTF-8">
              <meta name="viewport" content="width=device-
       width,  initial-scale=1.0">
              <title>Media Query</title>
  <style>
              body{
                    background-color: lightblue;
       }
       @media only screen and (max-width: 600px)
             {
              body {
                    background-color: lightgreen;
             }
```

```
            }
        </style>
    </head>
    <body>
        <p>Resize the browser window. When the width of this
        document is 600 pixels or less, the background-color is
        "lightblue", otherwise it is "lightgreen".</p>

</body>

</html>
```

9. Typography and Web Fonts

Theory Assignment

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

Ans: Difference Between Web-safe and Custom Web Fonts

Web-Safe Fonts	Custom Web Fonts
Web-safe fonts are a set of fonts that are pre-installed across most operating systems and browsers.	Custom web fonts are fonts that are not typically pre-installed on a user's device.
These fonts are universally supported and are reliable for rendering text consistently across different devices and platforms.	They are loaded from external sources (like Google Fonts, Adobe Fonts, or other font hosting services) when the web page is loaded. Web developers can specify custom fonts by linking to them in their CSS.
Example: Arial, Times New Roman, Courier New, Verdana, Georgia.	**Example: Google Fonts:** Roboto, Open Sans, Lora **Adobe Fonts:** Proxima Nova, Futura, Avenir

Cross-platform compatibility: Since these fonts are universally available on most devices, they ensure that the text appears consistently across all platforms without any issues.	**Design flexibility:** Custom fonts allow designers to create a unique look for a website that aligns with branding and design aesthetics.
Faster loading times: Web-safe fonts don't need to be downloaded from the web, which can improve page load speed, especially for users with slow internet connections.	**Variety and creativity:** You have access to a wide range of styles and fonts that are not typically found in standard web-safe fonts, making your site look more distinct and professional.
Simplicity: These fonts are simple and have been used for a long time, meaning they are highly readable and widely accepted for web content.	**Improved typography:** Custom fonts can enhance readability and user experience by providing more appropriate choices for different types of content, such as headings, body text, or callouts.

❖ Why might you use a web-safe font over a custom font

- Web-safe fonts load faster because they don't need to be fetched from an external server. Custom fonts can increase load times due to the need for additional HTTP requests and downloading font files.
- Web-safe fonts are widely supported and don't rely on users having a specific font installed or successfully downloading a custom font file. Custom fonts might not load properly on some devices or browsers if there are issues with the internet connection or font delivery.
- If you're working on a simple website or a project that doesn't require unique design elements, sticking with web-safe fonts can save you time and avoid unnecessary complexity.
- Some custom fonts might not be as legible or readable across different devices, especially on smaller screens or in low-resolution settings. Web-safe fonts are often more optimized for readability.

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

Ans: The font-family property specifies the font for an element. The font-family property can hold several font names as a "fallback" system. If the browser does not support the first font, it tries the next font.

- There are two types of font family names:
  - ✓ **family-name**: The name of a font-family, like "times", "courier", "arial", etc.
  - ✓ **Generic-family**: The name of a generic-family, like "serif", "sans-serif", "cursive", "fantasy", "monospace".
    - Start with the font you want, and always end with a generic family, to let the browser pick a similar font in the generic family, if no other fonts are available.
- **Syntax: font-family: "Font Name", fallback-font, generic-family;**
- **Example: p { font-family: "Arial", "Helvetica", sans-serif; }**

❖ How to Apply a Custom Google Font to a Webpage

- ✓ To apply a custom Google Font, you can follow several steps:
  - **Go to Google Fonts**
  - **Get the Embed Code**
  - **Add the Embed Code to Your HTML**
  - **Apply the Font in CSS**

❖ **Example:**

```
<!DOCTYPE html>
<html lang="en">
<head>
 <meta charset="UTF-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <title>Custom Font Example</title>
      <link href="https://fonts.googleapis.com/css2?family=Roboto:wght@400;700&display=swap" rel="stylesheet">
 <style>
```

```
                body {
                  font-family: "Roboto", sans-serif;
                }
  </style>
</head>
<body>
  <h1>Welcome to My Website</h1>
  <p>This text uses the Roboto font from Google Fonts</p>
</body>
</html>
```