Module 3 – Mernstack – CSS and CSS3

CSS Selectors & Styling:

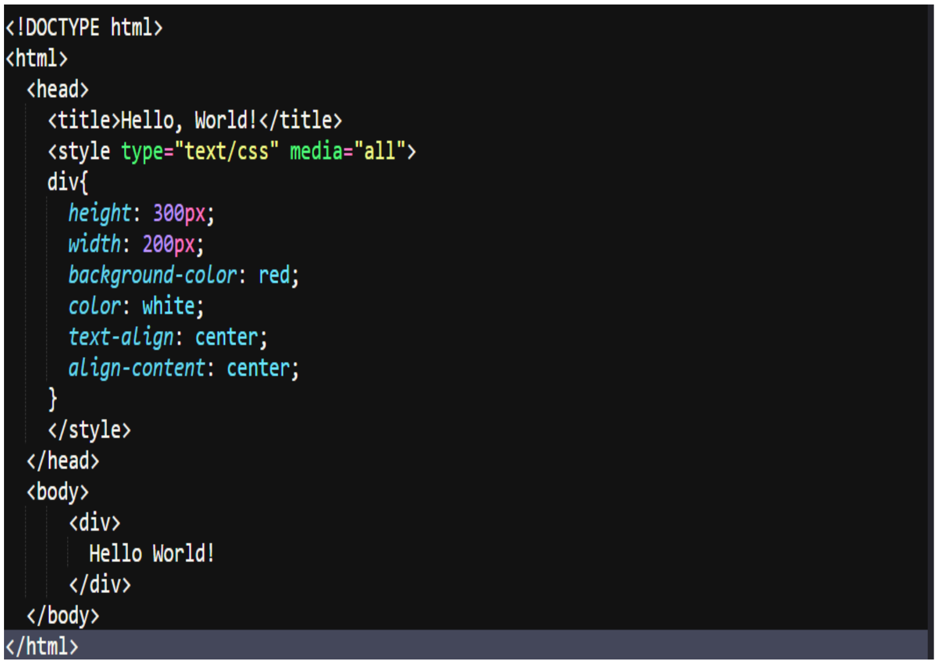
Theory Assignment:

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

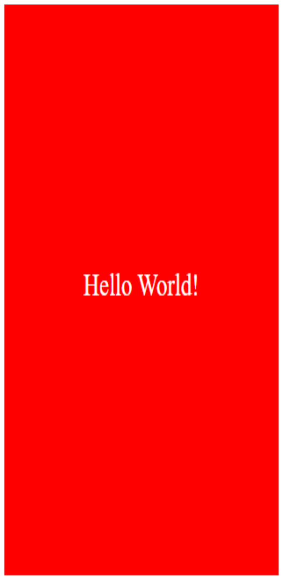
A **CSS selector** is a pattern used to select and style HTML elements. It defines which part of the HTML a style should apply to.

Examples:

1. **Element Selector**: Selects elements by their tag name.

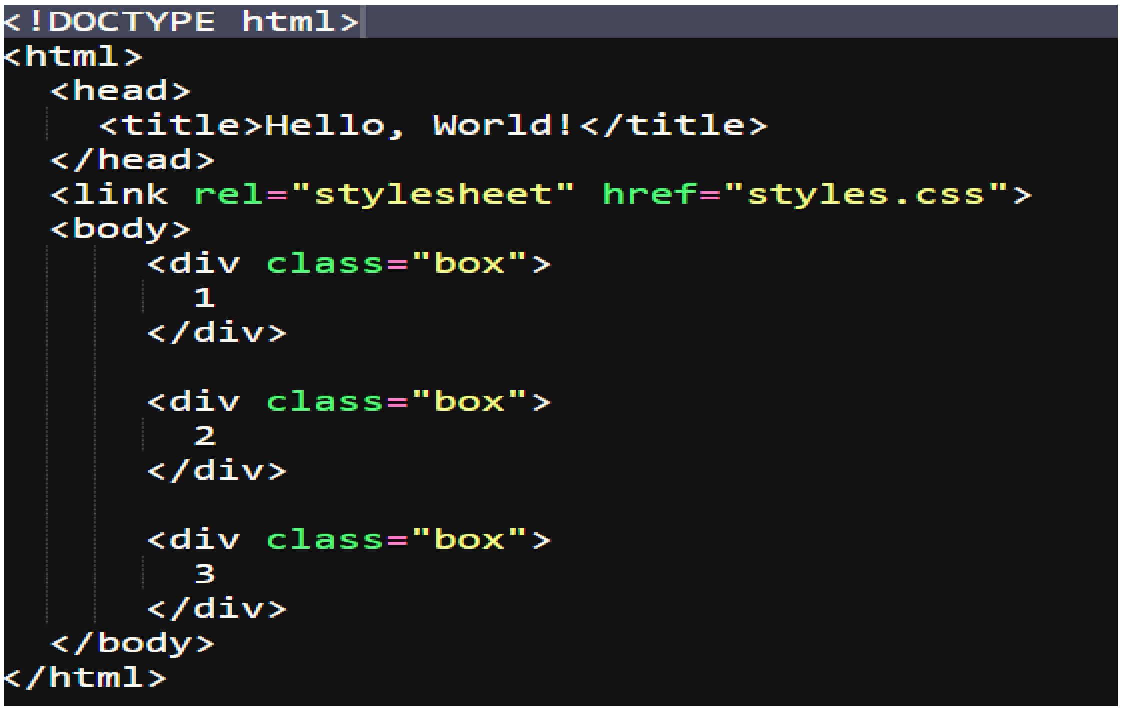


```
<!DOCTYPE html>
<html>
  <head>
    <title>Hello, World!</title>
    <style type="text/css" media="all">
    div{
      height: 300px;
      width: 200px;
      background-color: red;
      color: white;
      text-align: center;
      align-content: center;
    }
    </style>
  </head>
  <body>
      <div>
        Hello World!
      </div>
  </body>
</html>
```

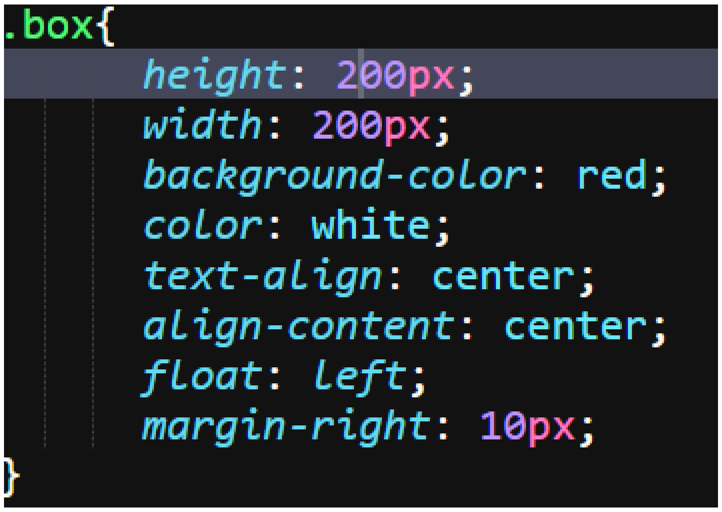


This styles all `<div>` elements.

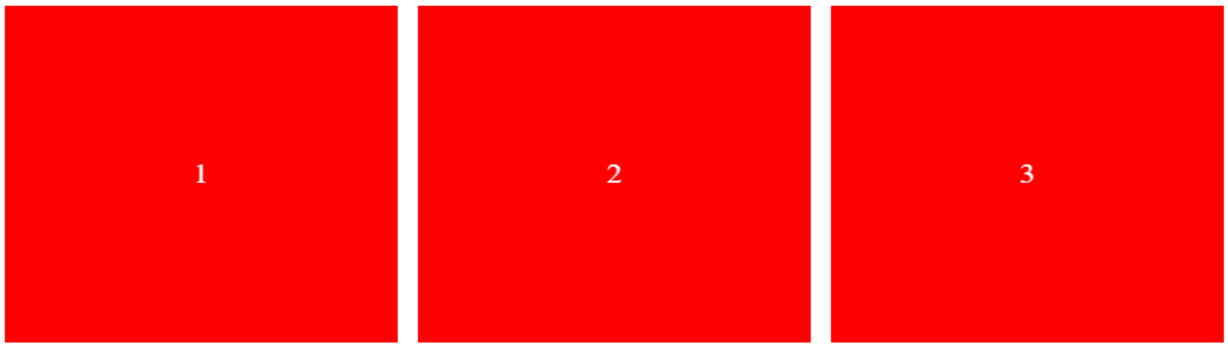
2. **Class Selector**: Selects elements with a specific class.



```
<!DOCTYPE html>
<html>
    <head>
        <title>Hello, World!</title>
    </head>
    <link rel="stylesheet" href="styles.css">
    <body>
        <div class="box">
            1
        </div>

        <div class="box">
            2
        </div>

        <div class="box">
            3
        </div>
    </body>
</html>
```

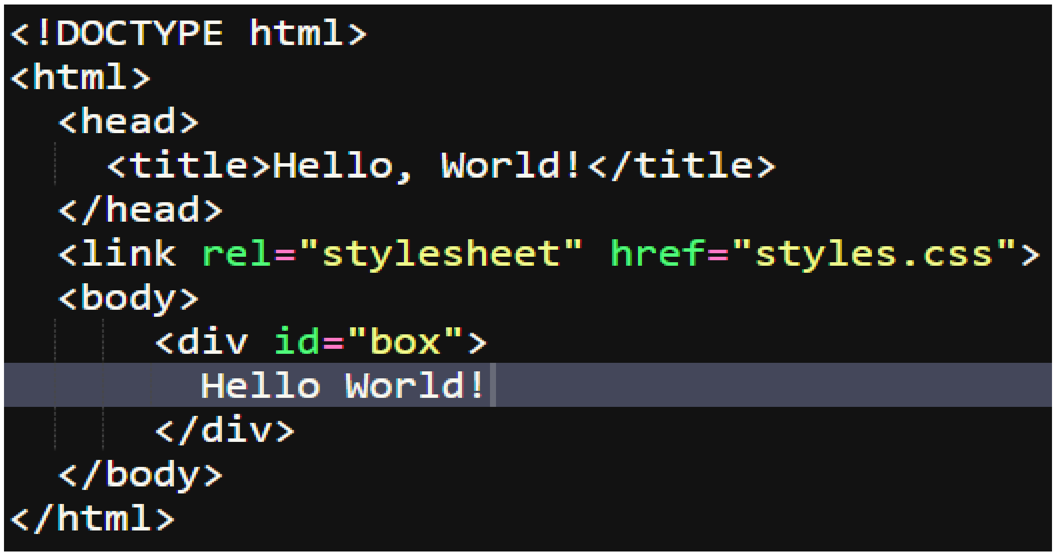


```
.box{
        height: 200px;
        width: 200px;
        background-color: red;
        color: white;
        text-align: center;
        align-content: center;
        float: left;
        margin-right: 10px;
}
```

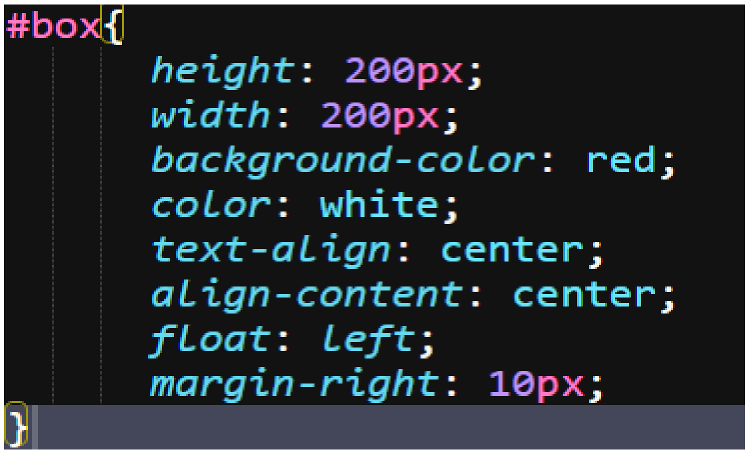


This styles all elements with the class `box`.

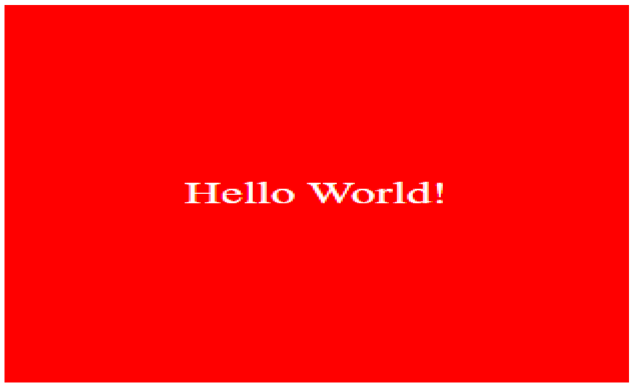
3. **ID Selector**: Selects a specific element by its ID.



```
<!DOCTYPE html>
<html>
  <head>
    <title>Hello, World!</title>
  </head>
  <link rel="stylesheet" href="styles.css">
  <body>
      <div id="box">
        Hello World!
      </div>
  </body>
</html>
```



```
#box{
      height: 200px;
      width: 200px;
      background-color: red;
      color: white;
      text-align: center;
      align-content: center;
      float: left;
      margin-right: 10px;
}
```



This styles the element with the ID `box`.

These selectors help target and style specific parts of a webpage efficiently.

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

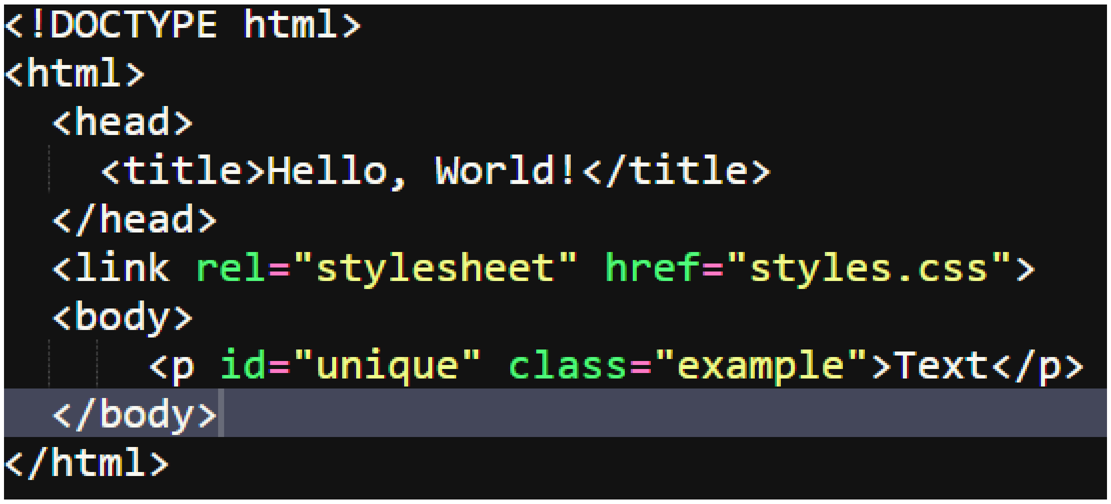
CSS specificity determines which style rules are applied when there are conflicting styles. It's a way of deciding which CSS rule takes precedence.

How Specificity Works:

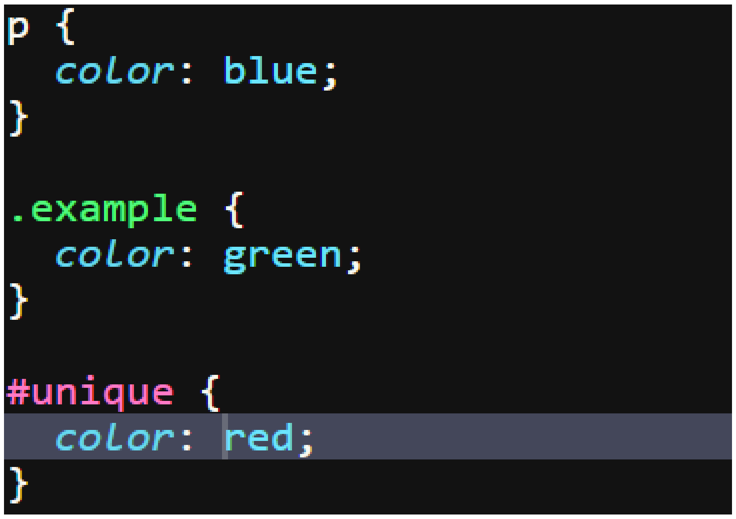
Each type of selector has a specificity value:

1. **Inline styles** (e.g., `style="color: red;"`) - Highest specificity.
2. **ID selectors** (`#id`) - Higher specificity.
3. **Class selectors**, **attributes**, and **pseudo-classes** (`.class`, `[attr]`, `:hover`) - Medium specificity.
4. **Element selectors** and **pseudo-elements** (`div`, `p`, `::before`) - Lower specificity.

Example:



```
<!DOCTYPE html>
<html>
  <head>
    <title>Hello, World!</title>
  </head>
  <link rel="stylesheet" href="styles.css">
  <body>
      <p id="unique" class="example">Text</p>
  </body>
</html>
```



```
p {
  color: blue;
}

.example {
  color: green;
}

#unique {
  color: red;
}
```

Text

In this case, the text will be red because the ID selector has the highest specificity.

Resolving Conflicts:

When multiple styles target the same element:

- The rule with the highest specificity wins.
- If specificity is the same, the rule defined later in the CSS takes precedence.
- Inline styles override both external and internal styles unless overridden by `!important`.

Understanding specificity helps in writing more efficient and conflict-free CSS.

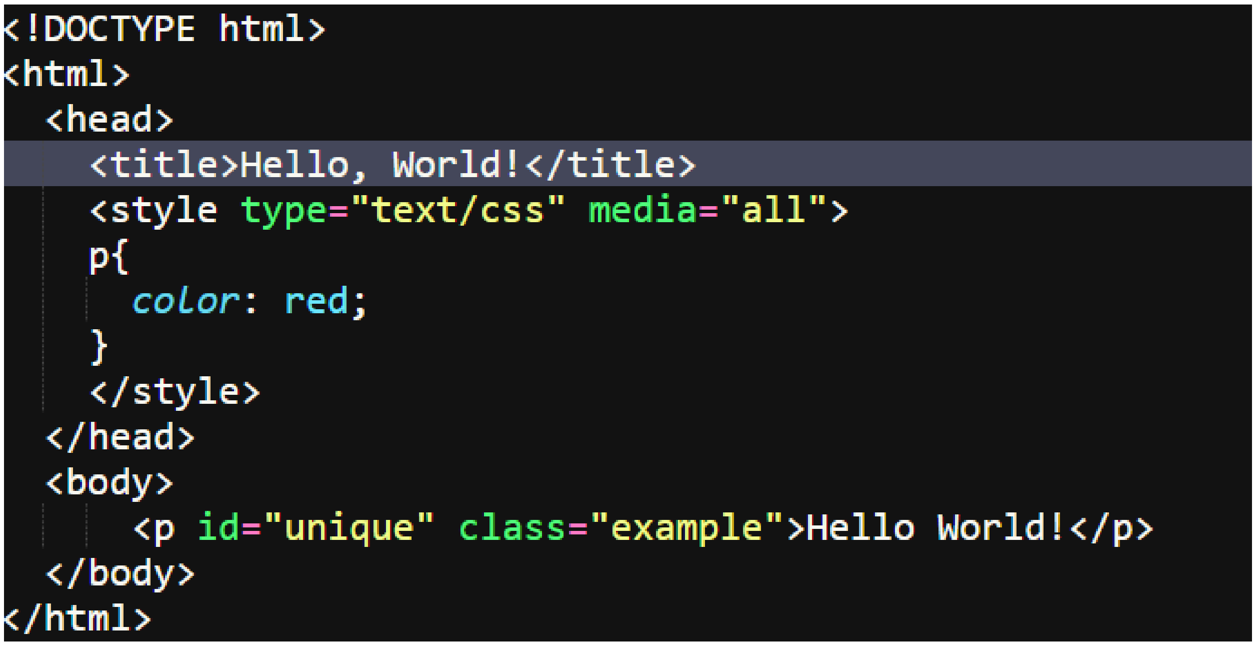
Question 3: What is the difference between internal, external, and inline CSS? Discuss the advantages and disadvantages of each approach.

Differences Between Internal, External, and Inline CSS:

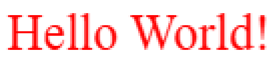
1. **Internal CSS**:

Defined within a `<style>` tag inside the `<head>` section of an HTML document.

Example:



```
<!DOCTYPE html>
<html>
  <head>
    <title>Hello, World!</title>
    <style type="text/css" media="all">
    p{
      color: red;
    }
    </style>
  </head>
  <body>
      <p id="unique" class="example">Hello World!</p>
  </body>
</html>
```



Hello World!

Advantages:

- Easy to manage for single-page styles.
- No need for external files.

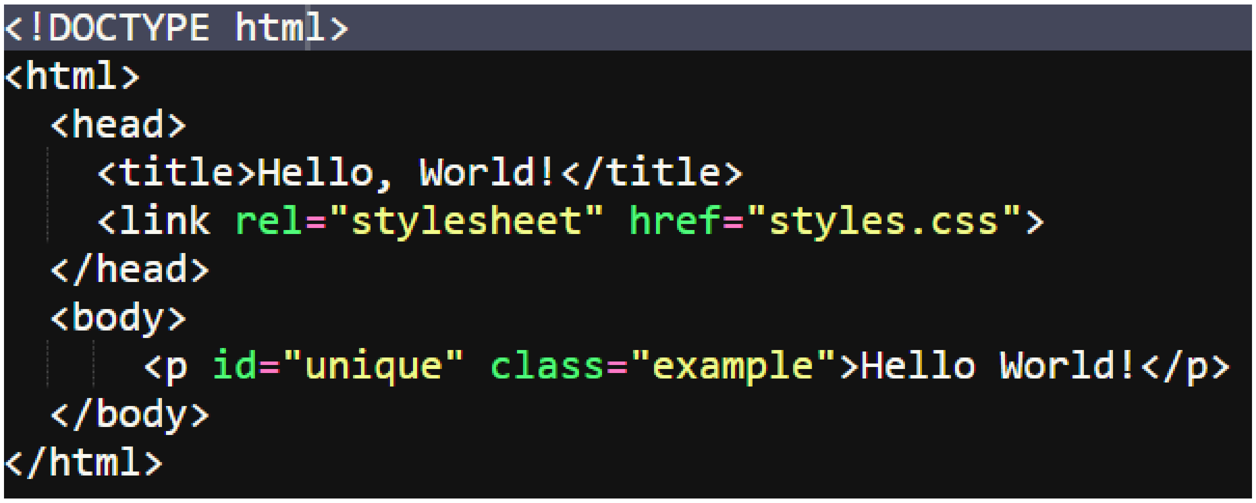
Disadvantages:

- Not reusable across multiple pages.
- Can clutter the HTML file.

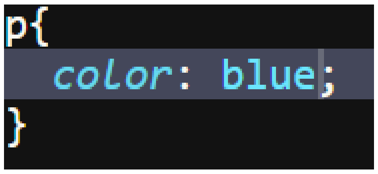
2. External CSS:

Written in a separate `.css` file and linked to the HTML document.

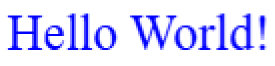
Example:



```html
<!DOCTYPE html>
<html>
  <head>
    <title>Hello, World!</title>
    <link rel="stylesheet" href="styles.css">
  </head>
  <body>
      <p id="unique" class="example">Hello World!</p>
  </body>
</html>
```



```css
p{
  color: blue;
}
```



Hello World!

Advantages:

- Reusable across multiple pages.
- Keeps HTML cleaner and separates content from design.

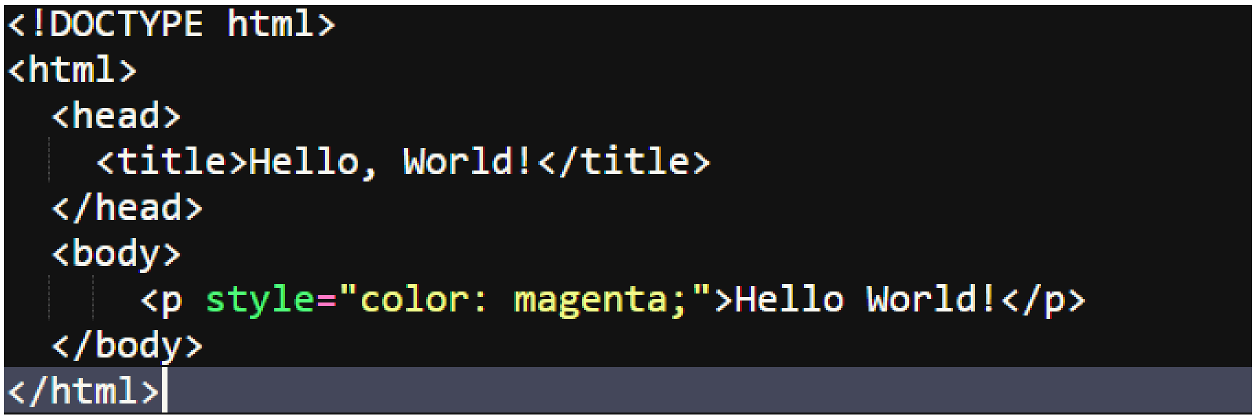
Disadvantages:

- Requires an extra HTTP request to load the CSS file.
- Changes in the external file affect all linked pages.

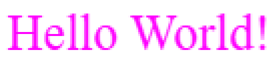
3. **Inline CSS**:

Added directly to an HTML element using the `style` attribute.

Example:



```
<!DOCTYPE html>
<html>
  <head>
    <title>Hello, World!</title>
  </head>
  <body>
      <p style="color: magenta;">Hello World!</p>
  </body>
</html>
```



Hello World!

Advantages:

- Useful for quick, small changes.
- Does not require external files.

Disadvantages:

- Hard to maintain and manage.
- Not reusable, leading to potential duplication of styles.

Summary:

- **External CSS** is best for large websites for reusability and cleaner code.
- **Internal CSS** works well for single-page applications.

- **Inline CSS** is useful for minor tweaks but should be used sparingly to avoid messy code.

Lab Assignment:

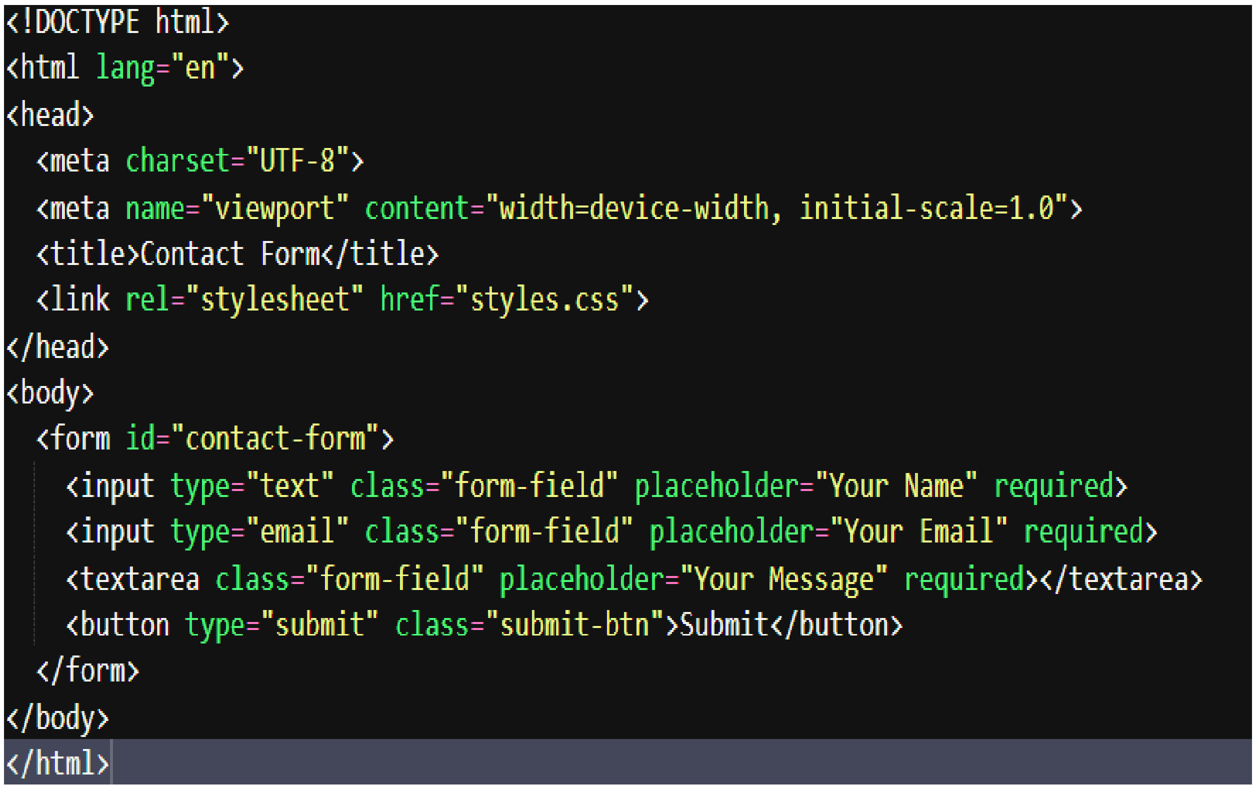
Task: Style the contact form (created in the HTML Forms lab) using external CSS. The following should be implemented:

• Change the background color of the form.

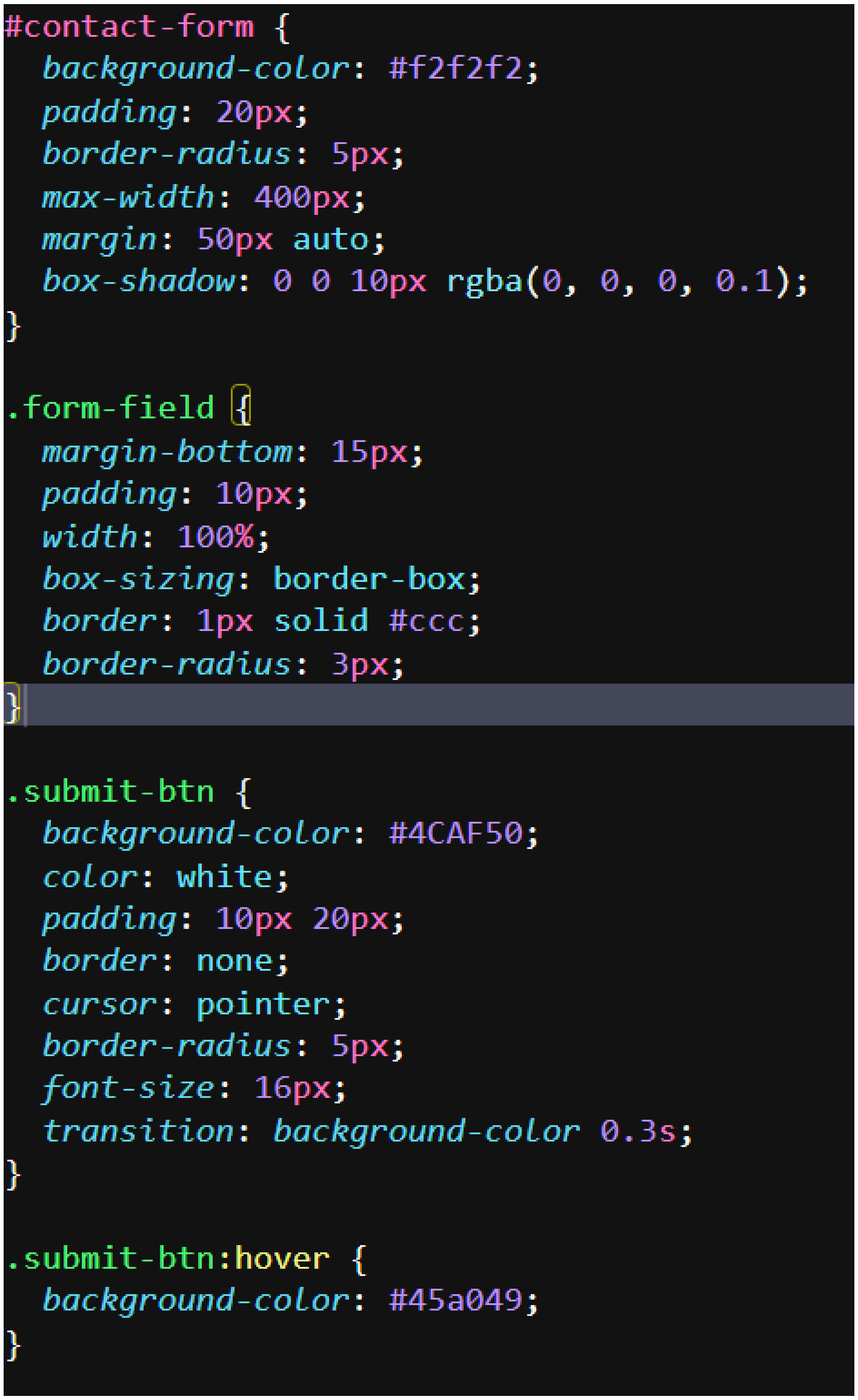
• Add padding and margins to form fields.

• Style the submit button with a hover effect.

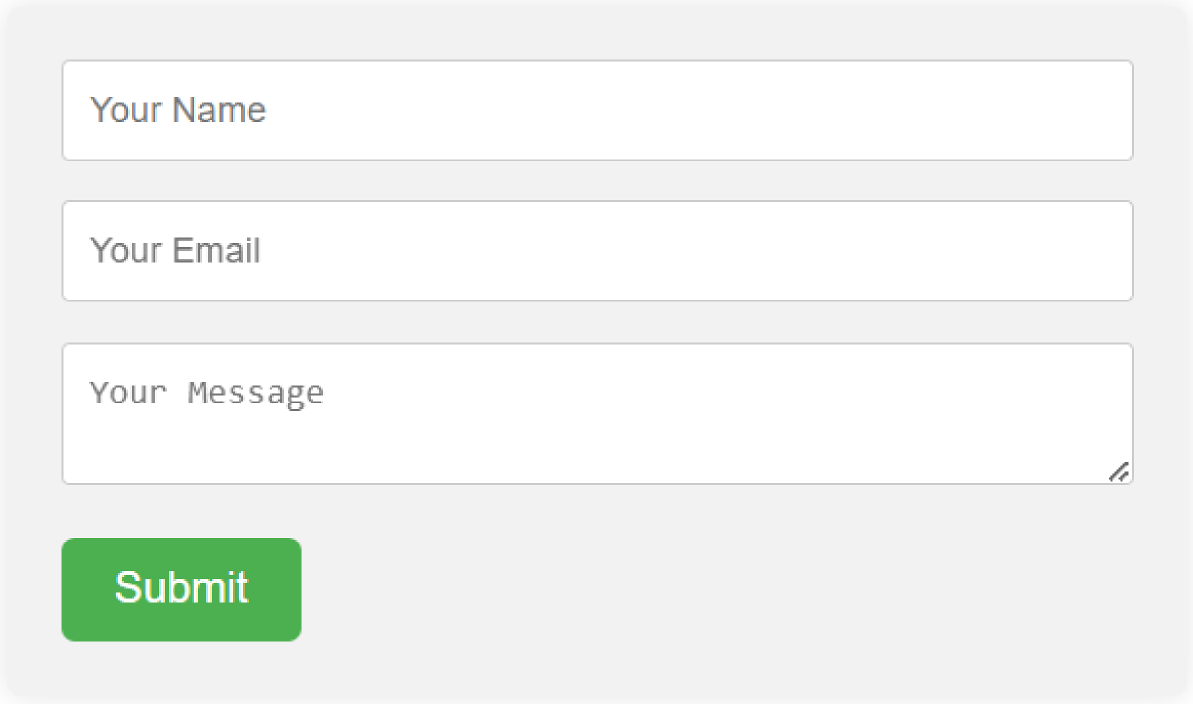
• Use class selectors for styling common elements and ID selectors for unique elements.



```
<!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="styles.css">
</head>
<body>
  <form id="contact-form">
    <input type="text" class="form-field" placeholder="Your Name" required>
    <input type="email" class="form-field" placeholder="Your Email" required>
    <textarea class="form-field" placeholder="Your Message" required></textarea>
    <button type="submit" class="submit-btn">Submit</button>
  </form>
</body>
</html>
```



```css
#contact-form {
  background-color: #f2f2f2;
  padding: 20px;
  border-radius: 5px;
  max-width: 400px;
  margin: 50px auto;
  box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
}

.form-field {
  margin-bottom: 15px;
  padding: 10px;
  width: 100%;
  box-sizing: border-box;
  border: 1px solid #ccc;
  border-radius: 3px;
}

.submit-btn {
  background-color: #4CAF50;
  color: white;
  padding: 10px 20px;
  border: none;
  cursor: pointer;
  border-radius: 5px;
  font-size: 16px;
  transition: background-color 0.3s;
}

.submit-btn:hover {
  background-color: #45a049;
}
```



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** defines how elements are structured on a webpage, consisting of:

1. **Content**: The actual text or images inside the element.
2. **Padding**: Space between the content and the border, increases element size.
3. **Border**: Surrounds the padding and content, adds to the element size.

4. **Margin**: Space outside the border, creates distance from other elements.

Each component affects the total size of the element, with padding, border, and margin adding extra space around the content.

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

The difference between `border-box` and `content-box` in CSS lies in how the total width and height of an element are calculated:

1.`content-box` (Default):

- The width and height include only the content
- Padding and border are added outside the specified width/height, increasing the total size.

2.`border-box`:

- The width and height include content, padding, and border.
- The total size of the element stays fixed, with padding and border inside the specified width/height.

Example:

- **`content-box`**: `width: 100px` + padding + border = total size > 100px.
- **`border-box`**: `width: 100px` includes padding and border = total size is 100px.

Lab Assignment:

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

• A profile picture.

• The user’s name and bio.

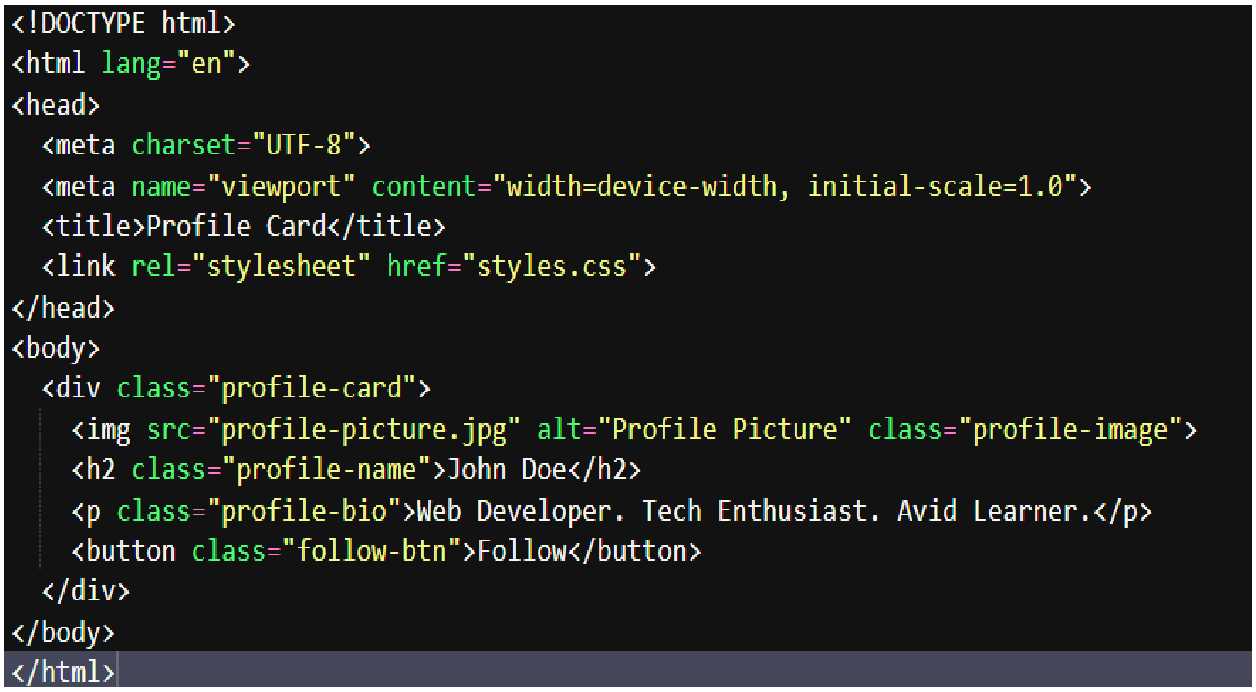
• A button to "Follow" the user.

Additional Requirements:

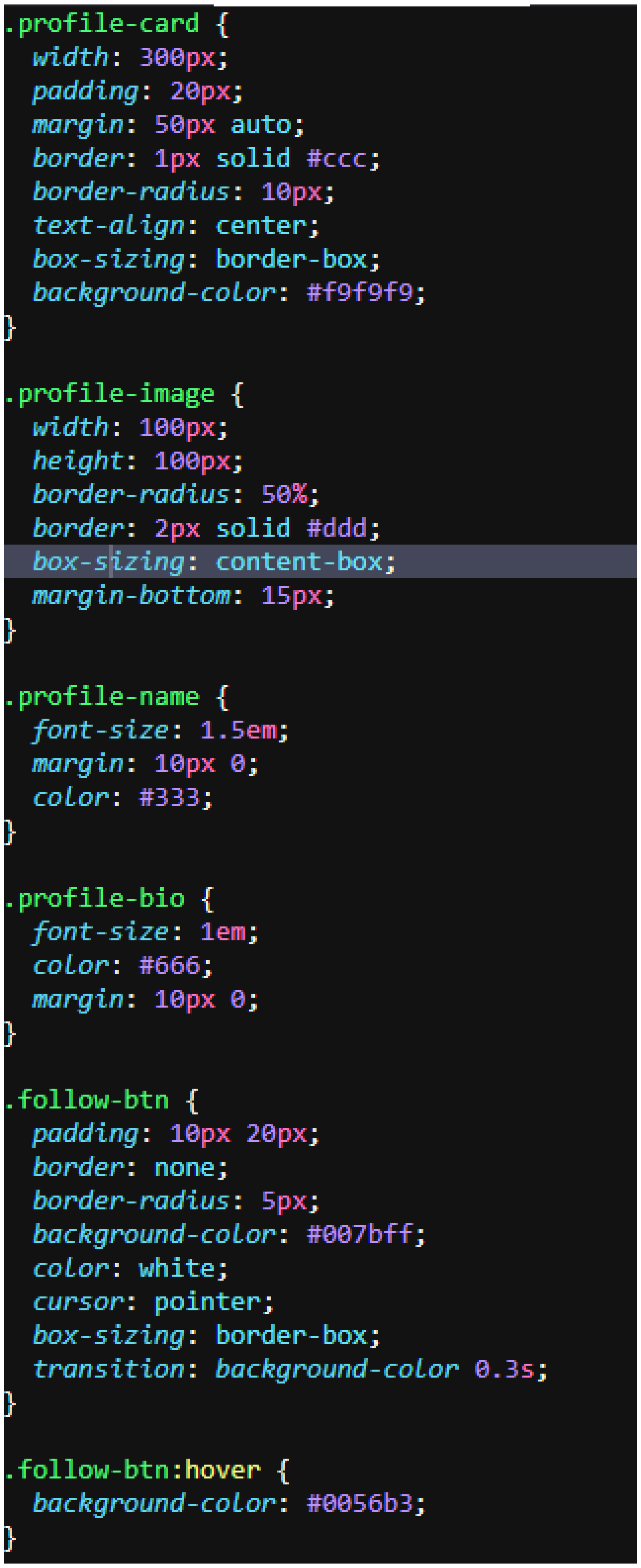
• Add padding and borders to the elements.

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

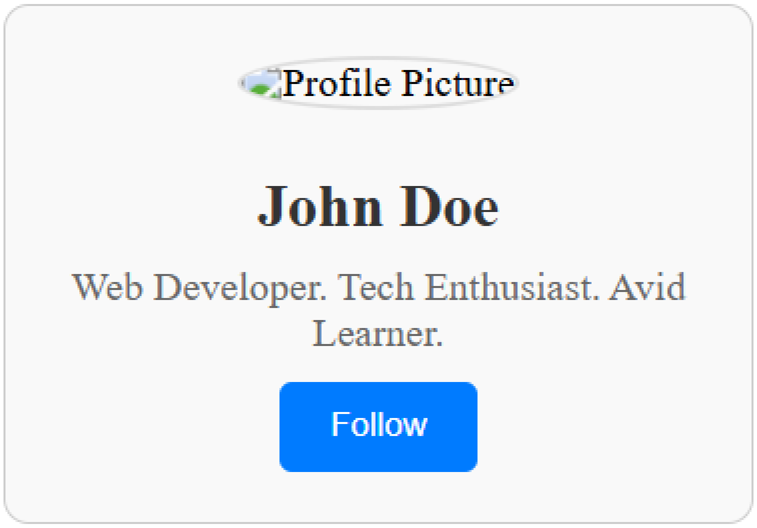
• Use the box-sizing property to demonstrate both content-box and border-box on different elements.



```
<!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="styles.css">
</head>
<body>
  <div class="profile-card">
    <img src="profile-picture.jpg" alt="Profile Picture" class="profile-image">
    <h2 class="profile-name">John Doe</h2>
    <p class="profile-bio">Web Developer. Tech Enthusiast. Avid Learner.</p>
    <button class="follow-btn">Follow</button>
  </div>
</body>
</html>
```



```css
.profile-card {
  width: 300px;
  padding: 20px;
  margin: 50px auto;
  border: 1px solid #ccc;
  border-radius: 10px;
  text-align: center;
  box-sizing: border-box;
  background-color: #f9f9f9;
}

.profile-image {
  width: 100px;
  height: 100px;
  border-radius: 50%;
  border: 2px solid #ddd;
  box-sizing: content-box;
  margin-bottom: 15px;
}

.profile-name {
  font-size: 1.5em;
  margin: 10px 0;
  color: #333;
}

.profile-bio {
  font-size: 1em;
  color: #666;
  margin: 10px 0;
}

.follow-btn {
  padding: 10px 20px;
  border: none;
  border-radius: 5px;
  background-color: #007bff;
  color: white;
  cursor: pointer;
  box-sizing: border-box;
  transition: background-color 0.3s;
}

.follow-btn:hover {
  background-color: #0056b3;
}
```



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 is a layout system that helps in creating flexible and responsive designs by aligning and distributing space among elements within a container.

Key Terms:

1. **Flex-container**: The parent element that holds flex items, defined with `display: flex;`.
2. **Flex-item**: The child elements inside the flex-container, which can be adjusted in size and alignment based on the container's properties.

Flexbox simplifies complex layouts, making it easier to align elements both horizontally and vertically.

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

Flexbox Properties:

1. `justify-content`:

- Controls the alignment of flex items **horizontally** (along the main axis).
- Options:
  1. `flex-start`: Aligns items to the start.
  2. `flex-end`: Aligns items to the end.
  3. `center`: Centers items.
  4. `space-between`: Distributes items evenly, with space between them.
  5. `space-around`: Distributes items evenly with space around them.

2. `align-items`:

- Controls the alignment of flex items **vertically** (along the cross axis).
- Options:
  1. `flex-start`: Aligns items to the top.
  2. `flex-end`: Aligns items to the bottom.
  3. `center`: Centers items vertically.
  4. `stretch`: Stretches items to fill the container.
  5. `baseline`: Aligns items to their baseline.

3. `flex-direction`:

- Defines the **main axis** (the direction in which flex items are laid out).
- Options:
    1. `row` (default): Aligns items horizontally (left to right).
    2. `row-reverse`: Aligns items horizontally (right to left).
    3. `column`: Aligns items vertically (top to bottom).
    4. `column-reverse`: Aligns items vertically (bottom to top).

These properties help control the positioning and alignment of elements in a flexbox layout.

Lab Assignment:

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

• A header.

• A sidebar on the left.

• A main content area in the center.

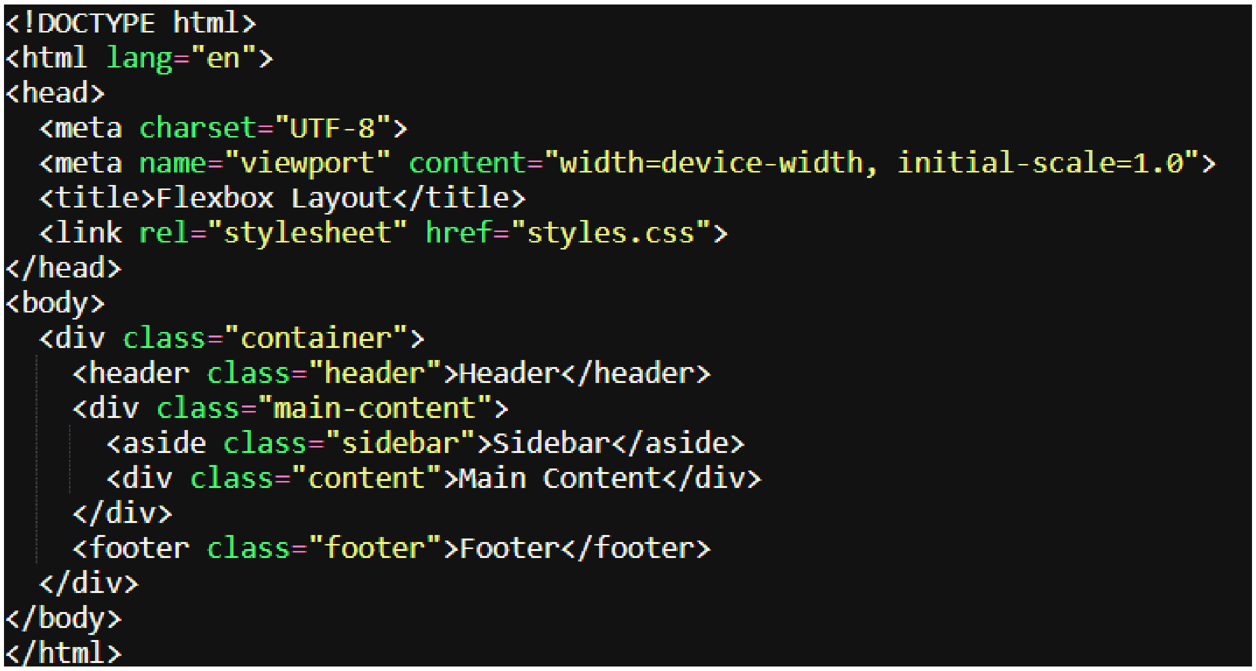
• A footer.

Additional Requirements:

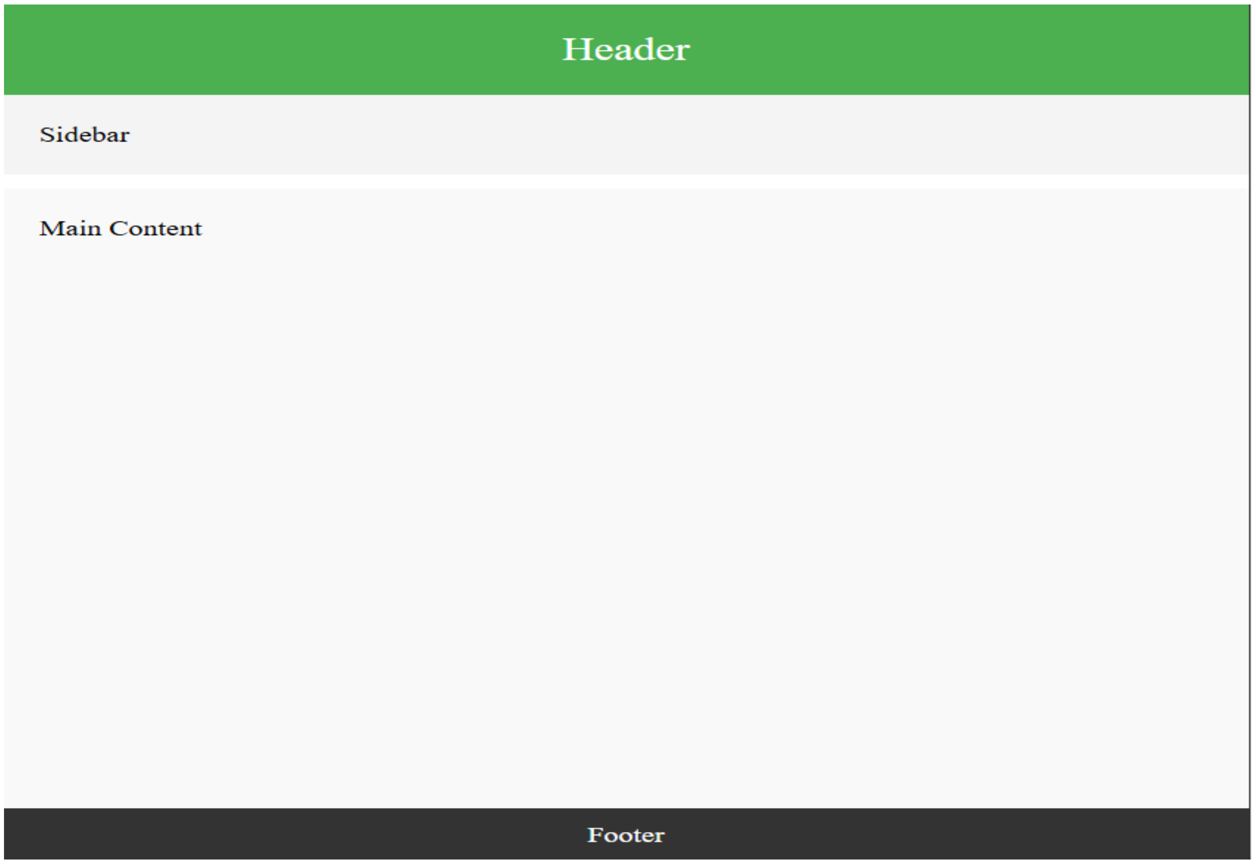
• Use Flexbox to position and align the elements.

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

- Ensure the layout is responsive, adjusting for smaller screens.



```html
<!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="styles.css">
</head>
<body>
  <div class="container">
    <header class="header">Header</header>
    <div class="main-content">
      <aside class="sidebar">Sidebar</aside>
      <div class="content">Main Content</div>
    </div>
    <footer class="footer">Footer</footer>
  </div>
</body>
</html>
```

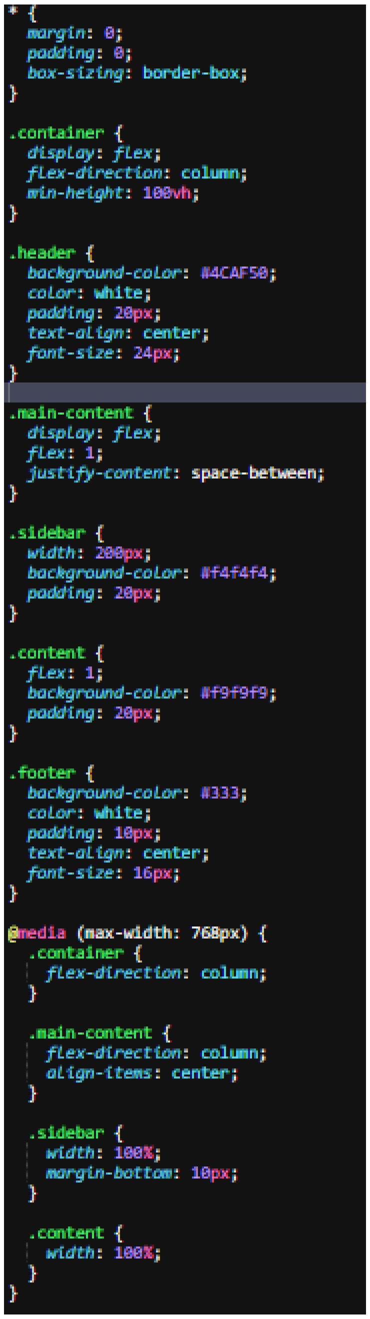


Header

Sidebar

Main Content

Footer



```css
* {
  margin: 0;
  padding: 0;
  box-sizing: border-box;
}

.container {
  display: flex;
  flex-direction: column;
  min-height: 100vh;
}

.header {
  background-color: #4CAF50;
  color: white;
  padding: 20px;
  text-align: center;
  font-size: 24px;
}

.main-content {
  display: flex;
  flex: 1;
  justify-content: space-between;
}

.sidebar {
  width: 200px;
  background-color: #f4f4f4;
  padding: 20px;
}

.content {
  flex: 1;
  background-color: #f9f9f9;
  padding: 20px;
}

.footer {
  background-color: #333;
  color: white;
  padding: 10px;
  text-align: center;
  font-size: 16px;
}

@media (max-width: 768px) {
  .container {
    flex-direction: column;
  }

  .main-content {
    flex-direction: column;
    align-items: center;
  }

  .sidebar {
    width: 100%;
    margin-bottom: 10px;
  }

  .content {
    width: 100%;
  }
}
```

<u>CSS Grid:</u>

Theory Assignment:

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

CSS Grid is a layout system designed for creating two-dimensional layouts (rows and columns). It allows precise control over both axes, making it ideal for complex grid-based designs.

Differences from Flexbox:

- **Flexbox**: One-dimensional, aligns items in a single row or column.
- **Grid**: Two-dimensional, manages both rows and columns simultaneously.

When to Use:

- **Use Grid**: When designing complex layouts with multiple rows and columns, like a webpage layout.
- **Use Flexbox**: For simpler, one-dimensional layouts, like aligning items in a navbar or centering elements.

Grid provides more control for complex designs, while Flexbox is better for simpler, linear arrangements.

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

CSS Grid Properties:

1. `grid-template-columns`:

- Defines the column structure of a grid.
- Example: `grid-template-columns: 1fr 2fr;` creates two columns, the second twice as wide as the first.

2. `grid-template-rows`:

- Defines the row structure of a grid.
- Example: `grid-template-rows: 100px auto;` creates a grid with a fixed height row and a flexible one.

3. `grid-gap`:

- Adds space between rows and columns.
- Example: `grid-gap: 10px;` creates a 10px gap between all grid items.

These properties help structure and space grid layouts efficiently.

Lab Assignment :

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

• A product image.

• A product title.

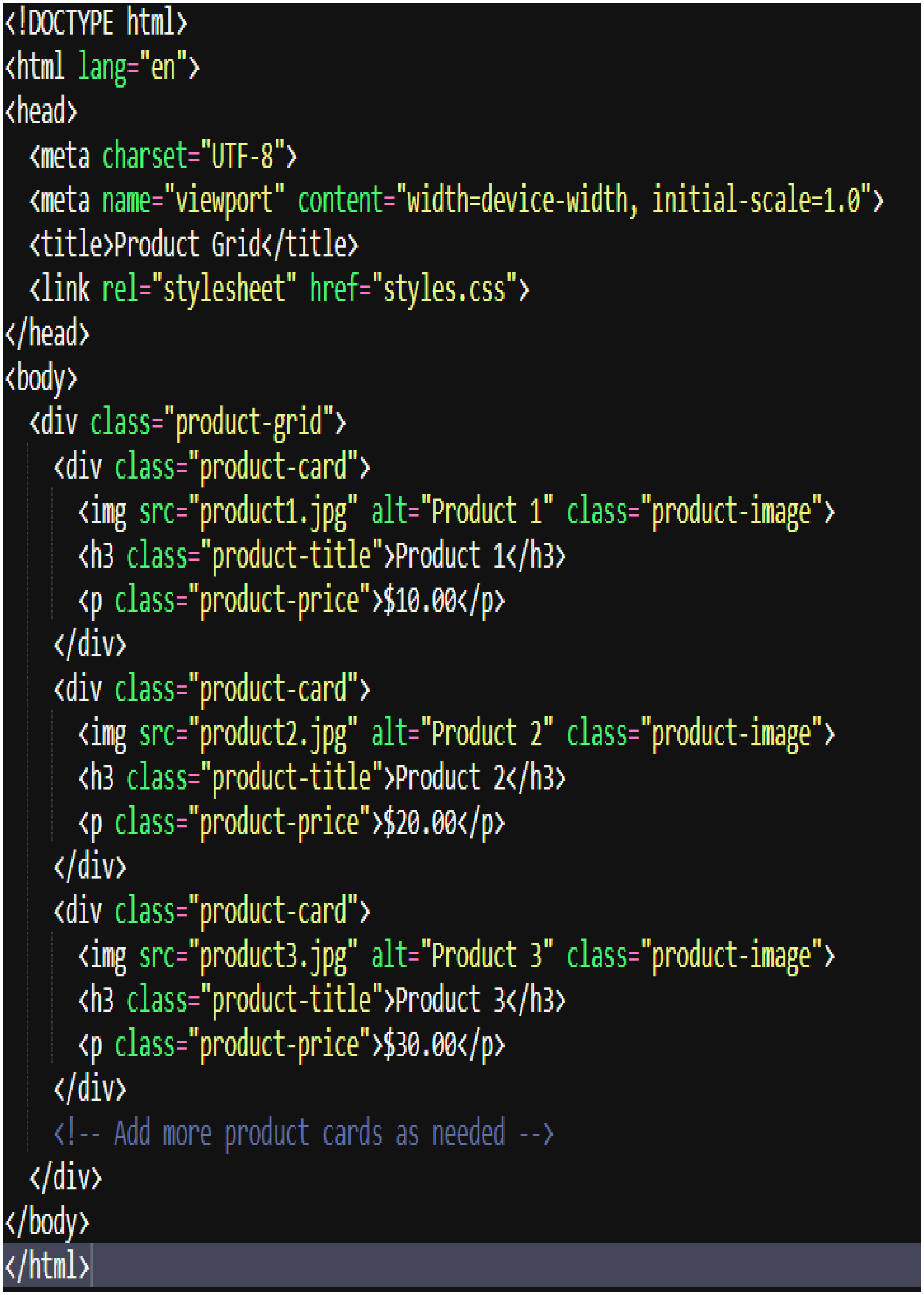
• A price.

Additional Requirements:

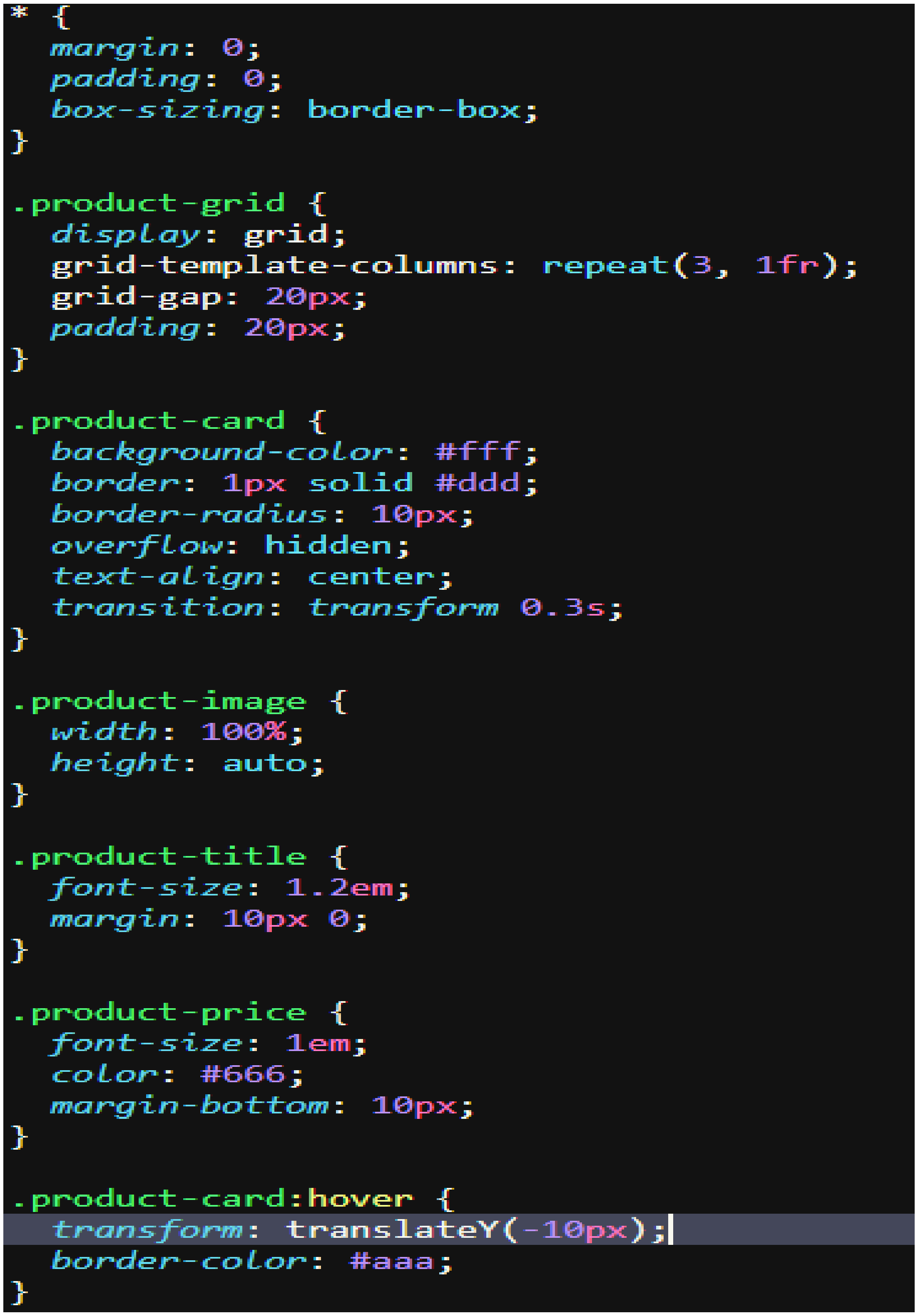
• Use grid-template-columns to create the grid layout.

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

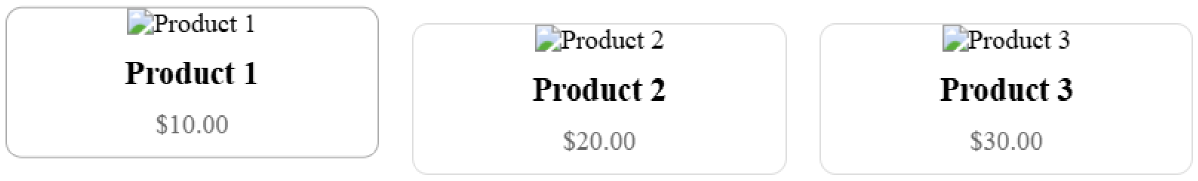
• Apply hover effects to each card for better interactivity.



```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Product Grid</title>
  <link rel="stylesheet" href="styles.css">
</head>
<body>
  <div class="product-grid">
    <div class="product-card">
      <img src="product1.jpg" alt="Product 1" class="product-image">
      <h3 class="product-title">Product 1</h3>
      <p class="product-price">$10.00</p>
    </div>
    <div class="product-card">
      <img src="product2.jpg" alt="Product 2" class="product-image">
      <h3 class="product-title">Product 2</h3>
      <p class="product-price">$20.00</p>
    </div>
    <div class="product-card">
      <img src="product3.jpg" alt="Product 3" class="product-image">
      <h3 class="product-title">Product 3</h3>
      <p class="product-price">$30.00</p>
    </div>
    <!-- Add more product cards as needed -->
  </div>
</body>
</html>
```



```css
* {
  margin: 0;
  padding: 0;
  box-sizing: border-box;
}

.product-grid {
  display: grid;
  grid-template-columns: repeat(3, 1fr);
  grid-gap: 20px;
  padding: 20px;
}

.product-card {
  background-color: #fff;
  border: 1px solid #ddd;
  border-radius: 10px;
  overflow: hidden;
  text-align: center;
  transition: transform 0.3s;
}

.product-image {
  width: 100%;
  height: auto;
}

.product-title {
  font-size: 1.2em;
  margin: 10px 0;
}

.product-price {
  font-size: 1em;
  color: #666;
  margin-bottom: 10px;
}

.product-card:hover {
  transform: translateY(-10px);
  border-color: #aaa;
}
```



Product 1

Product 1

$10.00

Product 2

Product 2

$20.00

Product 3

Product 3

$30.00

Responsive Web Design with Media Queries:

Theory Assignment:

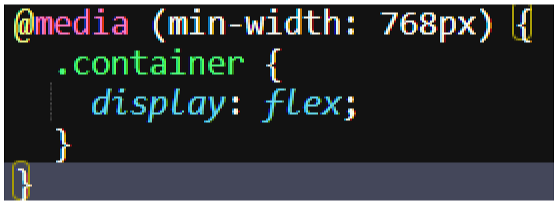
Question 1: What are media queries in CSS, and why are they important for responsive design?

Media queries in CSS allow you to apply styles based on the characteristics of the user's device, such as screen width, height, orientation, or resolution. They are crucial for creating **responsive designs**, ensuring that a website looks and functions well on different devices (desktops, tablets, smartphones).

Importance:

1. **Adaptability**: Media queries enable a website to adapt its layout and styling to various screen sizes and devices.
2. **Improved User Experience**: Ensures content is accessible and easy to navigate on any device.
3. **Mobile-First Design**: Allows developers to design for mobile devices first and progressively enhance for larger screens.

Example:

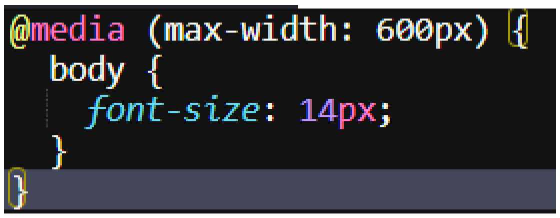


```css
@media (min-width: 768px) {
  .container {
    display: flex;
  }
}
```

This applies specific styles when the screen width is 768px or more.

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

Basic Media Query for Adjusting Font Size:



```css
@media (max-width: 600px) {
  body {
    font-size: 14px;
  }
}
```

This media query reduces the font size to `14px` for screens with a width of `600px` or less, enhancing readability on smaller devices.

Lab Assignment :

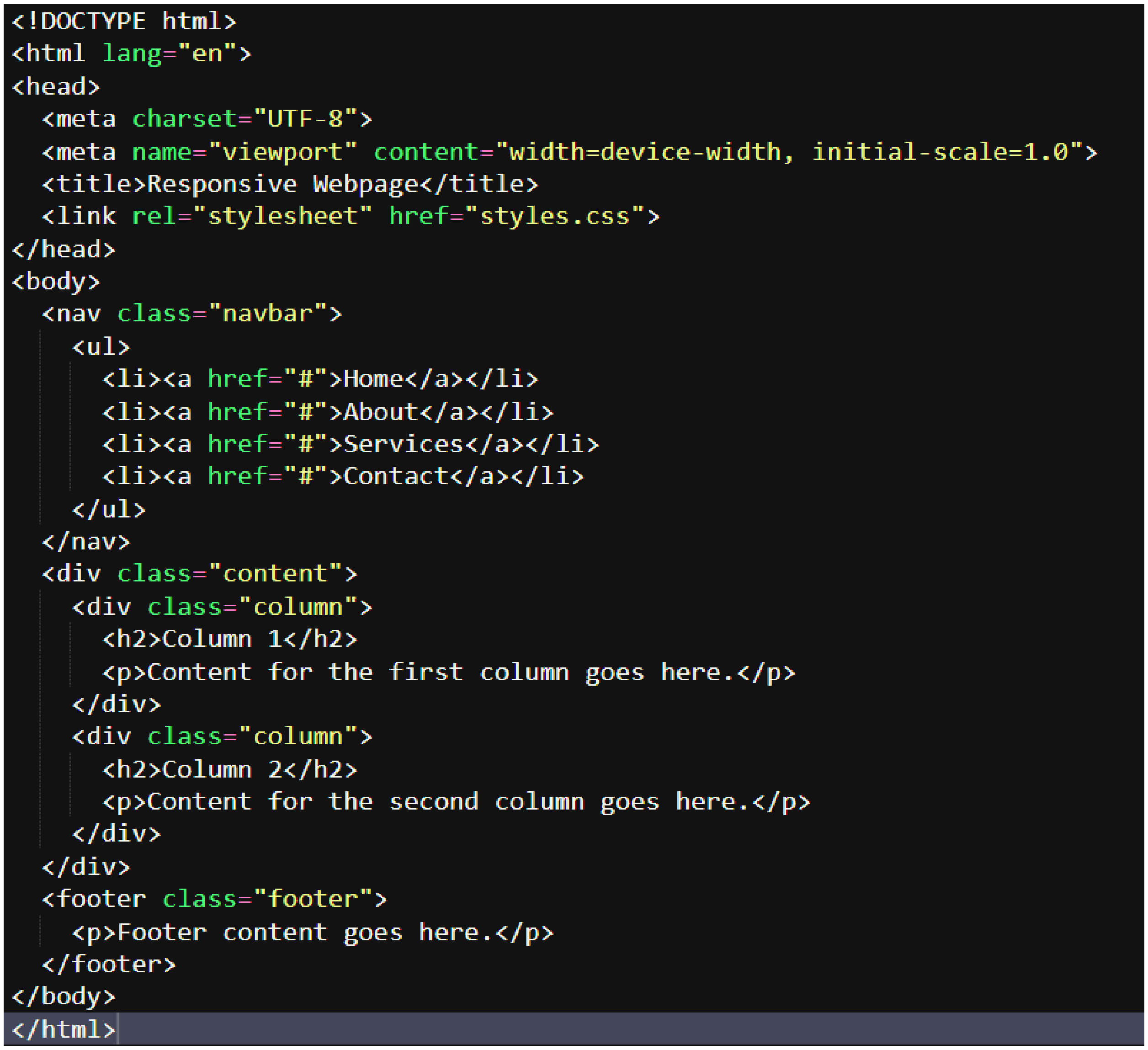
- Task: Build a responsive webpage that includes:
- A navigation bar.
- A content section with two columns.
- A footer.

Additional Requirements:

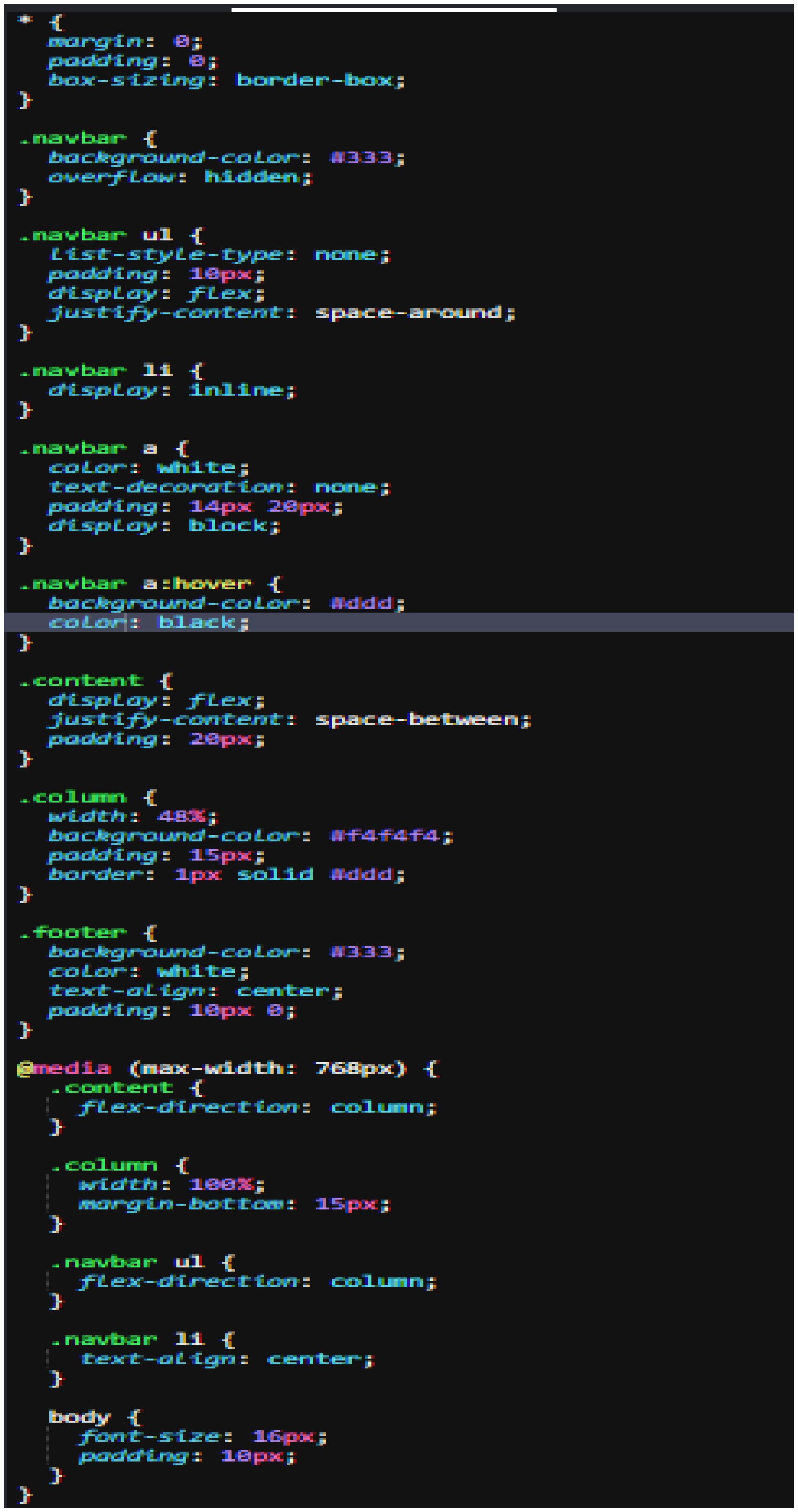
• Use media queries to make the webpage responsive for mobile devices.

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

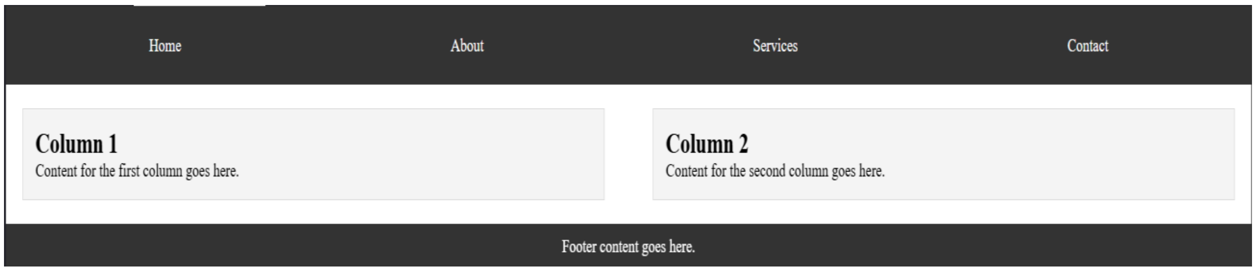
• Adjust the font sizes and padding to improve readability on mobile.



```html
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Responsive Webpage</title>
  <link rel="stylesheet" href="styles.css">
</head>
<body>
  <nav class="navbar">
    <ul>
      <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">
      <h2>Column 1</h2>
      <p>Content for the first column goes here.</p>
    </div>
    <div class="column">
      <h2>Column 2</h2>
      <p>Content for the second column goes here.</p>
    </div>
  </div>
  <footer class="footer">
    <p>Footer content goes here.</p>
  </footer>
</body>
</html>
```



```
* {
  margin: 0;
  padding: 0;
  box-sizing: border-box;
}

.navbar {
  background-color: #333;
  overflow: hidden;
}

.navbar ul {
  list-style-type: none;
  padding: 10px;
  display: flex;
  justify-content: space-around;
}

.navbar li {
  display: inline;
}

.navbar a {
  color: white;
  text-decoration: none;
  padding: 14px 20px;
  display: block;
}

.navbar a:hover {
  background-color: #ddd;
  color: black;
}

.content {
  display: flex;
  justify-content: space-between;
  padding: 20px;
}

.column {
  width: 48%;
  background-color: #f4f4f4;
  padding: 15px;
  border: 1px solid #ddd;
}

.footer {
  background-color: #333;
  color: white;
  text-align: center;
  padding: 10px 0;
}

@media (max-width: 768px) {
  .content {
    flex-direction: column;
  }

  .column {
    width: 100%;
    margin-bottom: 15px;
  }

  .navbar ul {
    flex-direction: column;
  }

  .navbar li {
    text-align: center;
  }

  body {
    font-size: 16px;
    padding: 10px;
  }
}
```



Column 1
Content for the first column goes here.

Column 2
Content for the second column goes here.

Footer content goes here.

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 are standard fonts pre-installed on most devices, ensuring consistent appearance across different platforms. Examples include Arial, Times New Roman, and Courier New.

Custom web fonts are fonts not pre-installed on devices. They are loaded from external sources (like Google Fonts) and offer greater design flexibility and uniqueness.

Differences:

1. **Compatibility**: Web-safe fonts are universally supported, ensuring consistent display. Custom fonts might not render properly if not loaded correctly.
2. **Performance**: Web-safe fonts load faster since they're already on the user's device. Custom fonts can slow down page load times as they require additional resources.

Why Use Web-Safe Fonts:

- **Reliability**: Guaranteed to work across all browsers and devices.
- **Performance**: Faster loading times, improving user experience, especially on slower connections.
- **Fallback**: As a fallback option when custom fonts fail to load, ensuring readability.

Web-safe fonts are a safer choice for performance and compatibility, while custom fonts offer more unique design options.

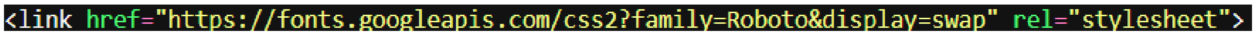
Question 2: What is the font-family property in CSS? How do you apply a custom Google Font to a webpage?

`font-family` is a CSS property used to specify the font of text. It defines a prioritized list of font names and fallback options to ensure text displays properly if the preferred font is unavailable.

Applying a Custom Google Font:

1.Import the Font:

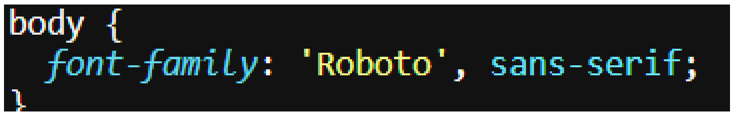
- Use the `<link>` tag in the `<head>` section of your HTML to import the font



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

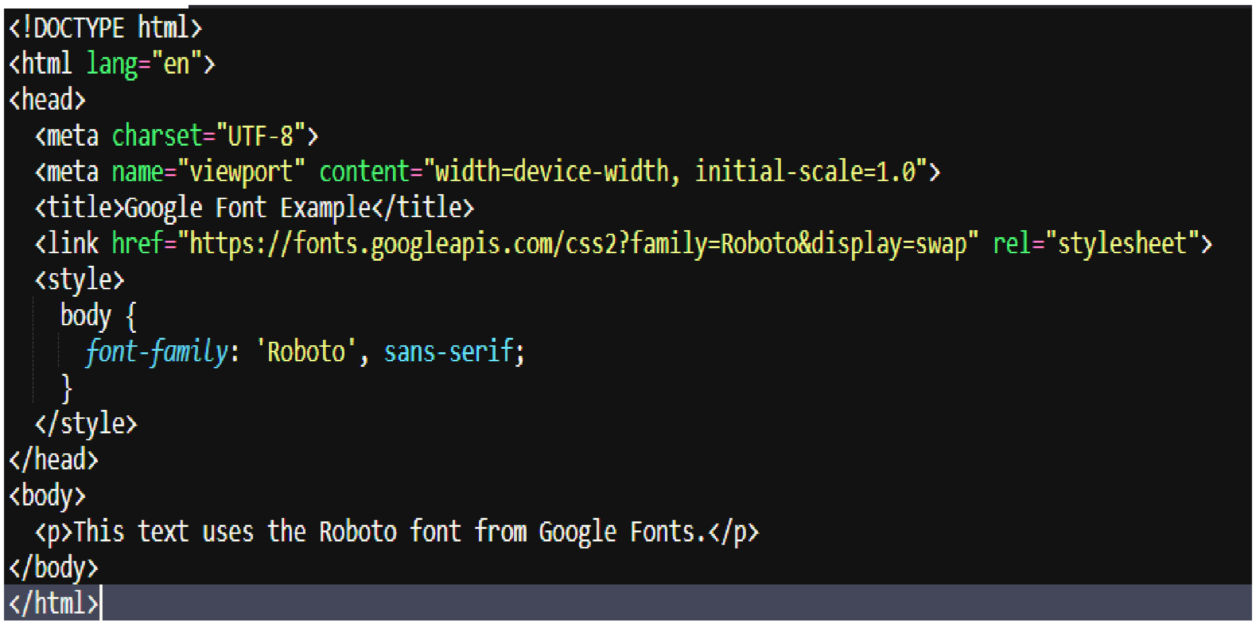
2.Apply the Font in CSS:

- Use the `font-family` property to apply the imported font.



```css
body {
  font-family: 'Roboto', sans-serif;
}
```

Example:



```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Google Font Example</title>
  <link href="https://fonts.googleapis.com/css2?family=Roboto&display=swap" rel="stylesheet">
  <style>
    body {
      font-family: 'Roboto', sans-serif;
    }
  </style>
</head>
<body>
  <p>This text uses the Roboto font from Google Fonts.</p>
</body>
</html>
```

`font-family` ensures the correct font is used, and importing Google Fonts adds a custom style to the webpage.

Lab Assignment:

Task: Create a blog post layout with the following:

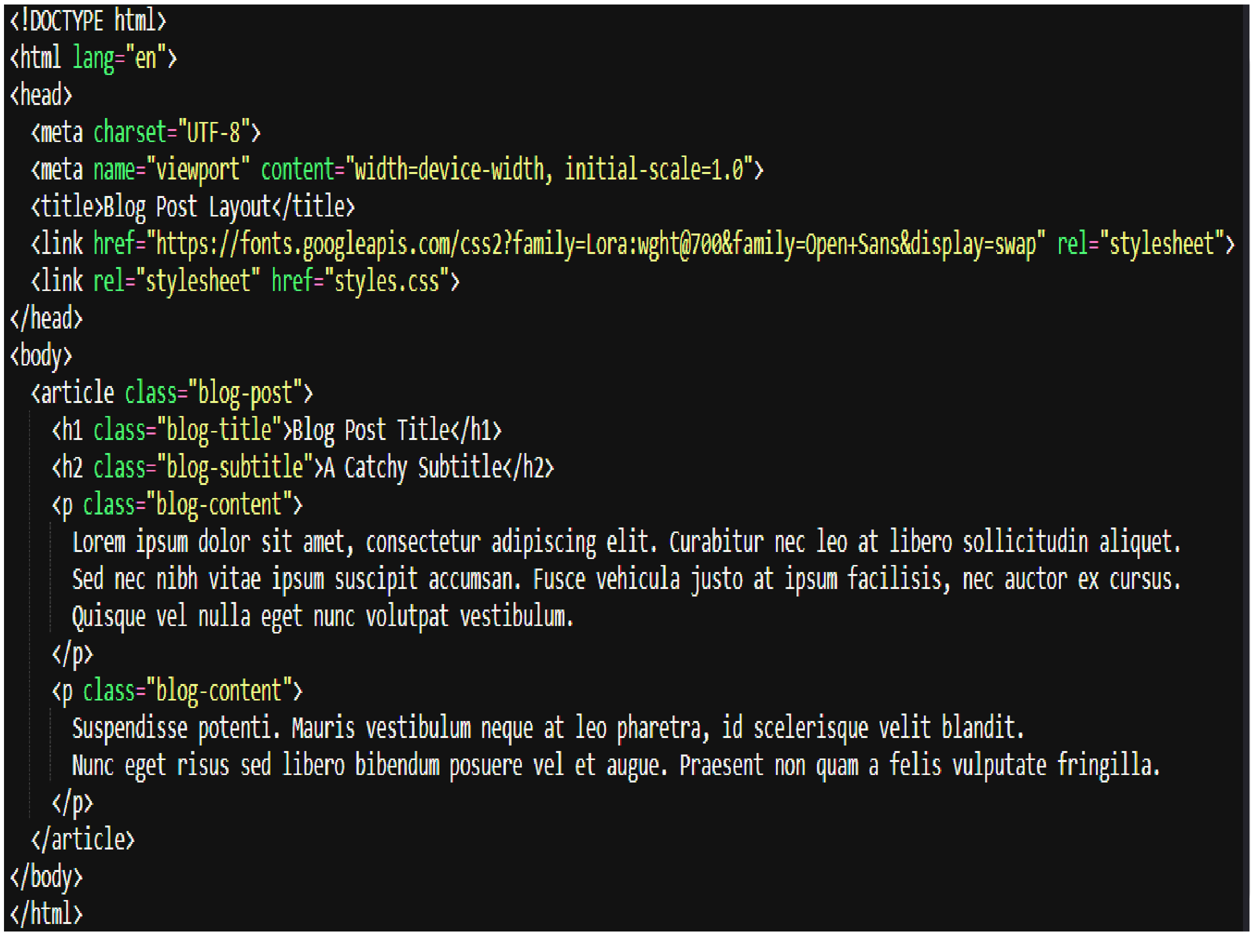
• A title, subtitle, and body content.

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

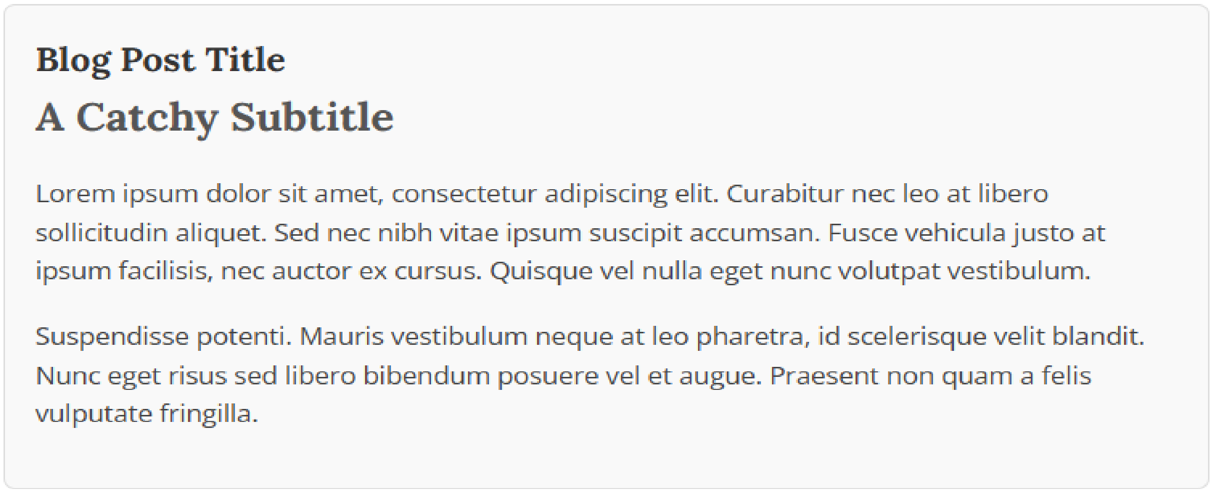
• Style the text to be responsive and easy to read.

Additional Requirements:

- Use a custom font from Google Fonts.
- Adjust line-height, font-size, and spacing for improved readability.



```html
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Blog Post Layout</title>
  <link href="https://fonts.googleapis.com/css2?family=Lora:wght@700&family=Open+Sans&display=swap" rel="stylesheet">
  <link rel="stylesheet" href="styles.css">
</head>
<body>
  <article class="blog-post">
    <h1 class="blog-title">Blog Post Title</h1>
    <h2 class="blog-subtitle">A Catchy Subtitle</h2>
    <p class="blog-content">
      Lorem ipsum dolor sit amet, consectetur adipiscing elit. Curabitur nec leo at libero sollicitudin aliquet.
      Sed nec nibh vitae ipsum suscipit accumsan. Fusce vehicula justo at ipsum facilisis, nec auctor ex cursus.
      Quisque vel nulla eget nunc volutpat vestibulum.
    </p>
    <p class="blog-content">
      Suspendisse potenti. Mauris vestibulum neque at leo pharetra, id scelerisque velit blandit.
      Nunc eget risus sed libero bibendum posuere vel et augue. Praesent non quam a felis vulputate fringilla.
    </p>
  </article>
</body>
</html>
```

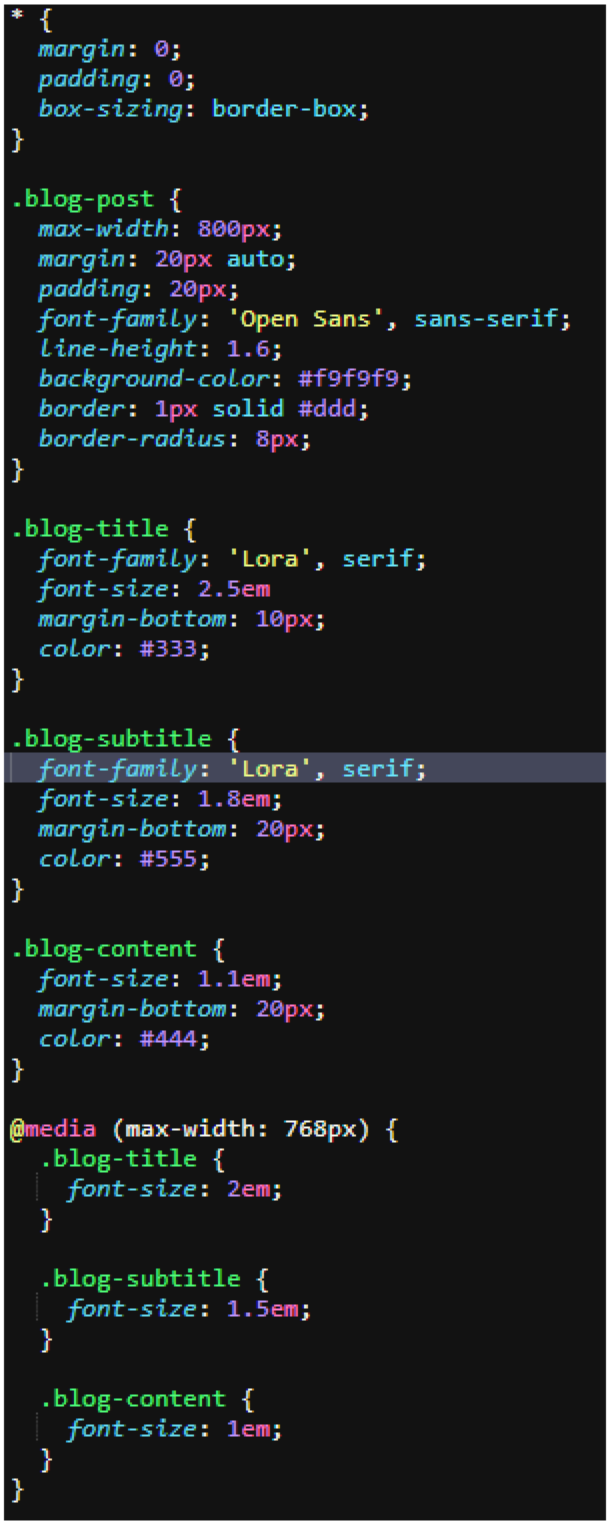


Blog Post Title

A Catchy Subtitle

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Curabitur nec leo at libero sollicitudin aliquet. Sed nec nibh vitae ipsum suscipit accumsan. Fusce vehicula justo at ipsum facilisis, nec auctor ex cursus. Quisque vel nulla eget nunc volutpat vestibulum.

Suspendisse potenti. Mauris vestibulum neque at leo pharetra, id scelerisque velit blandit. Nunc eget risus sed libero bibendum posuere vel et augue. Praesent non quam a felis vulputate fringilla.



```css
* {
  margin: 0;
  padding: 0;
  box-sizing: border-box;
}

.blog-post {
  max-width: 800px;
  margin: 20px auto;
  padding: 20px;
  font-family: 'Open Sans', sans-serif;
  line-height: 1.6;
  background-color: #f9f9f9;
  border: 1px solid #ddd;
  border-radius: 8px;
}

.blog-title {
  font-family: 'Lora', serif;
  font-size: 2.5em
  margin-bottom: 10px;
  color: #333;
}

.blog-subtitle {
  font-family: 'Lora', serif;
  font-size: 1.8em;
  margin-bottom: 20px;
  color: #555;
}

.blog-content {
  font-size: 1.1em;
  margin-bottom: 20px;
  color: #444;
}

@media (max-width: 768px) {
  .blog-title {
    font-size: 2em;
  }

  .blog-subtitle {
    font-size: 1.5em;
  }

  .blog-content {
    font-size: 1em;
  }
}
```