**Module 3 – Mernstack – CSS and CSS3**

# CSS Selectors & Styling:

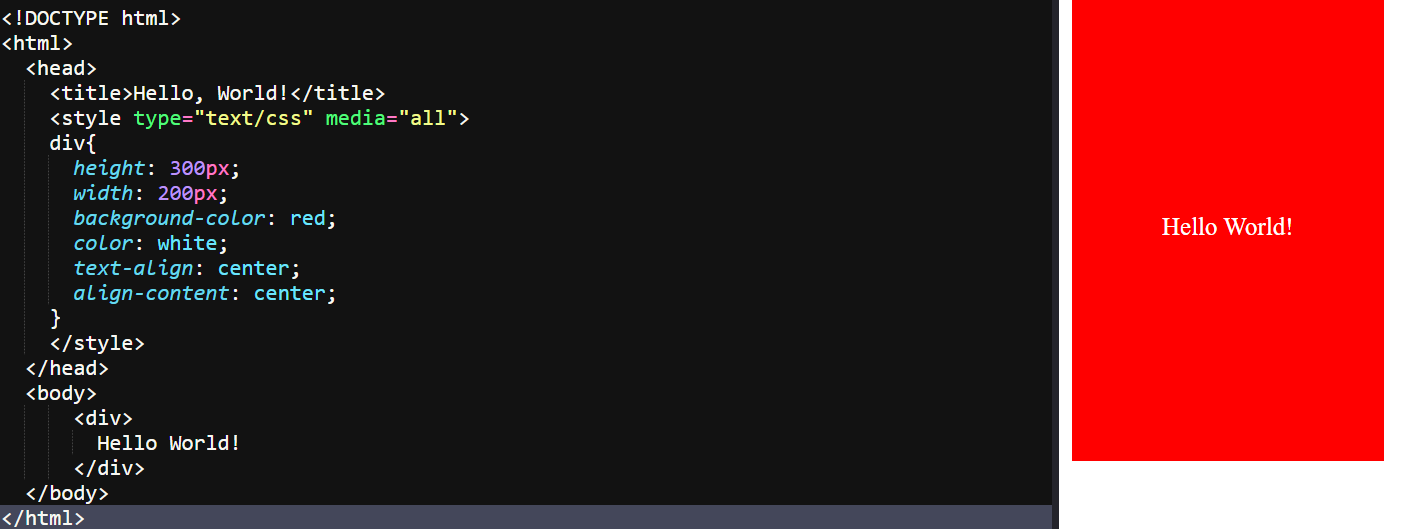
# Theory Assignment:

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.

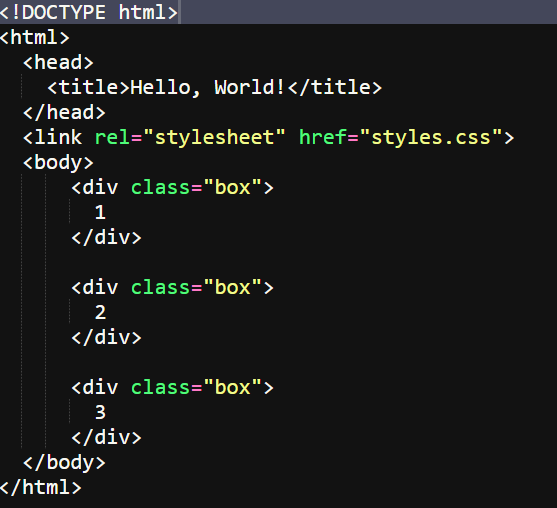
# Examples:

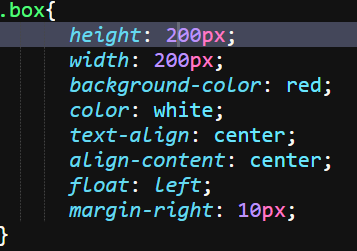
1. **Element Selector**: Selects elements by their tag name.

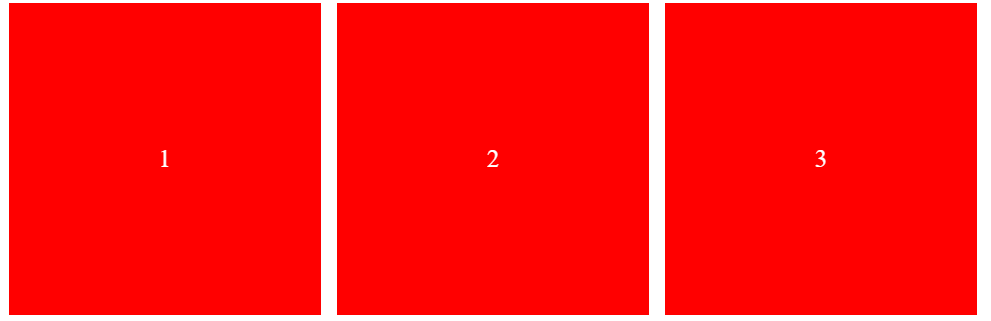


This styles all <div> elements.

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

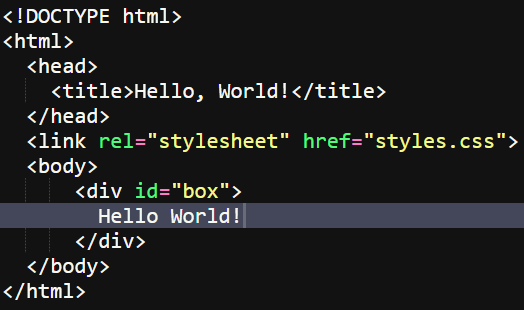


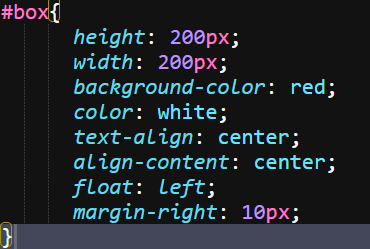


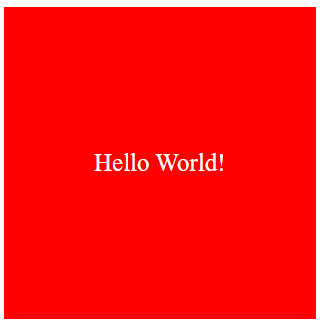


This styles all elements with the class box.

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







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?

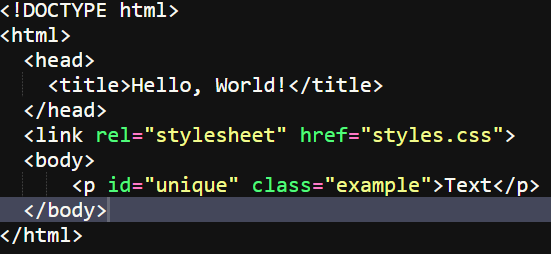
**CSS specificity** determines which style rules are applied when there are conflicting styles. It's a way of deciding which CSS rule takes precedence.

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:







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.

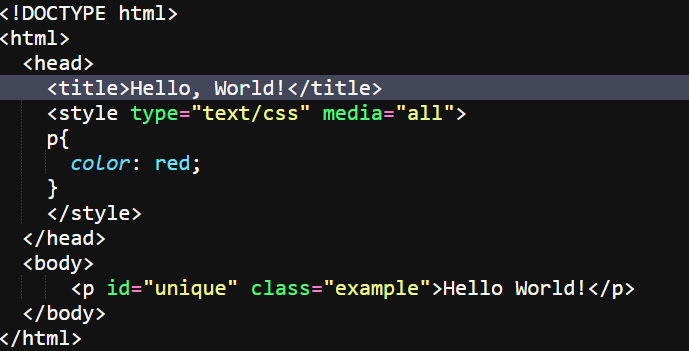
Question 3: What is the difference between internal, external, and inline CSS? Discuss the advantages and disadvantages of each approach.

Differences Between Internal, External, and Inline CSS:

1. **Internal CSS**:

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

# Example:





# Advantages:

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

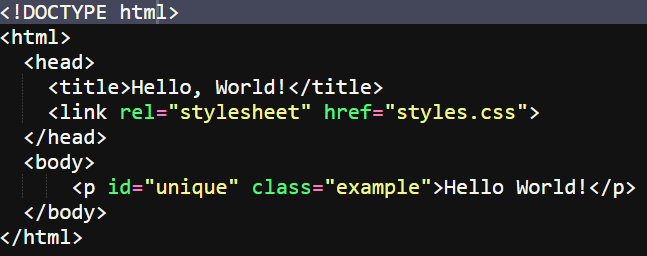
# Disadvantages:

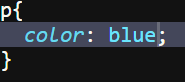
* Not reusable across multiple pages.
* Can clutter the HTML file.

1. **External CSS**:

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

# Example:







# Advantages:

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

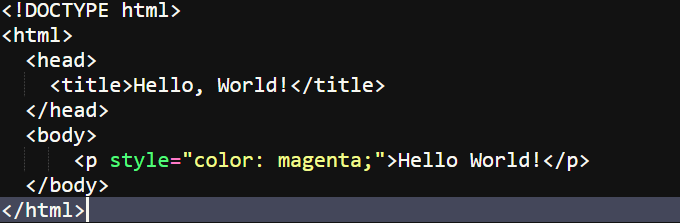
# Disadvantages:

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

1. **Inline CSS**:

Added directly to an HTML element using the style attribute.

# Example:





# 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:

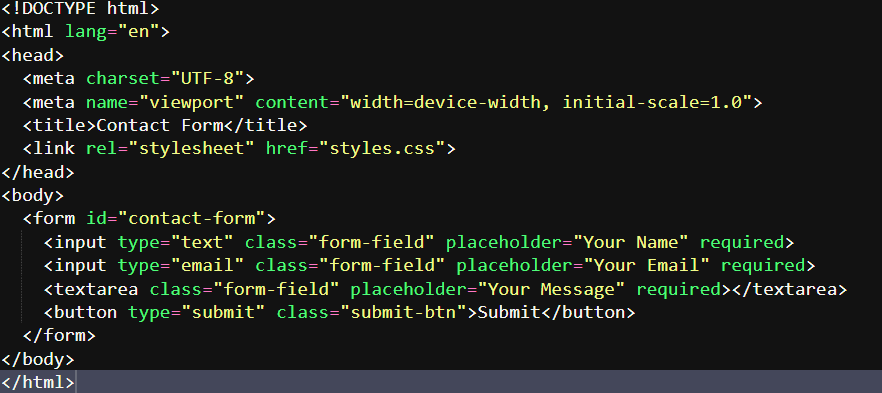
Task: Style the contact form (created in the HTML Forms lab) using external CSS. The following should be implemented:

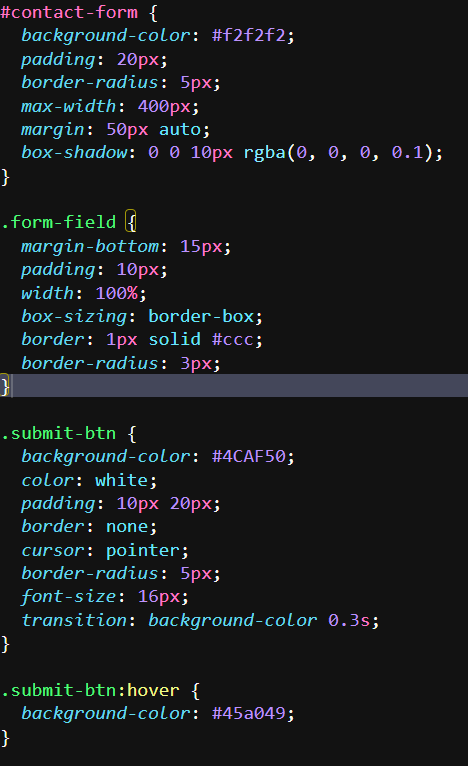
• Change the background color of the form.

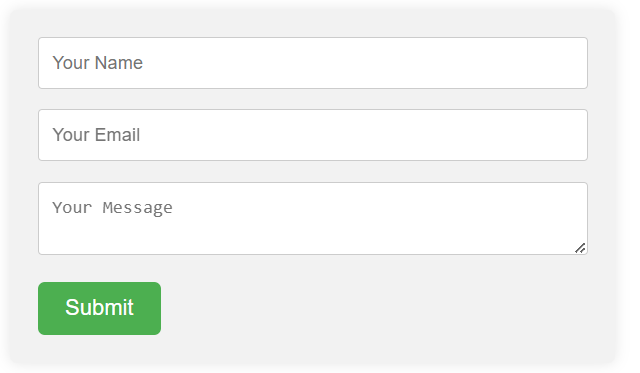
• Add padding and margins to form fields.

• Style the submit button with a hover effect.

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







# 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.

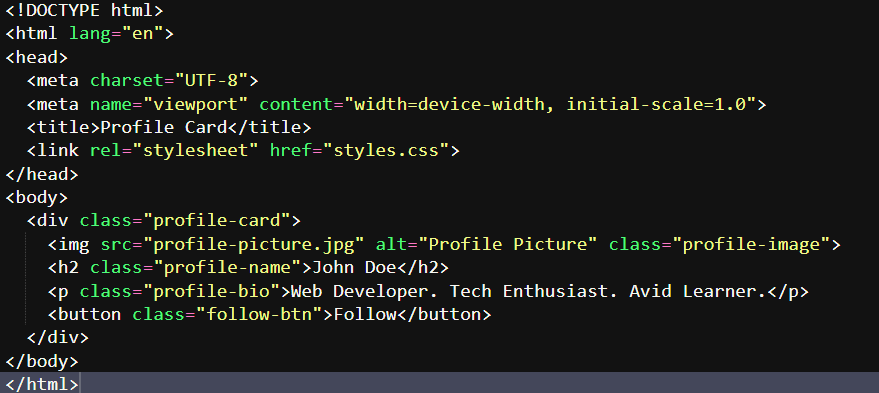
• A button to "Follow" the user.

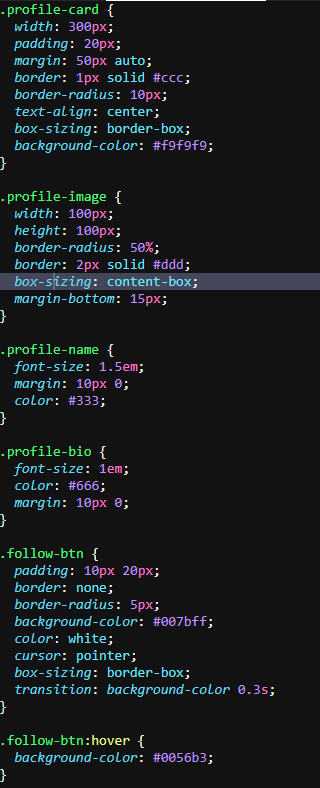
Additional Requirements:

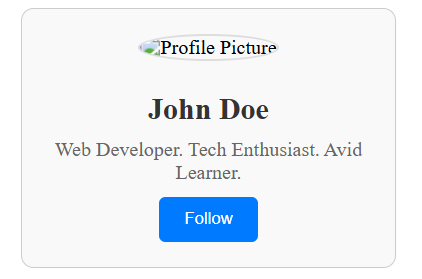
• Add padding and borders to the elements.

• 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.







# 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.

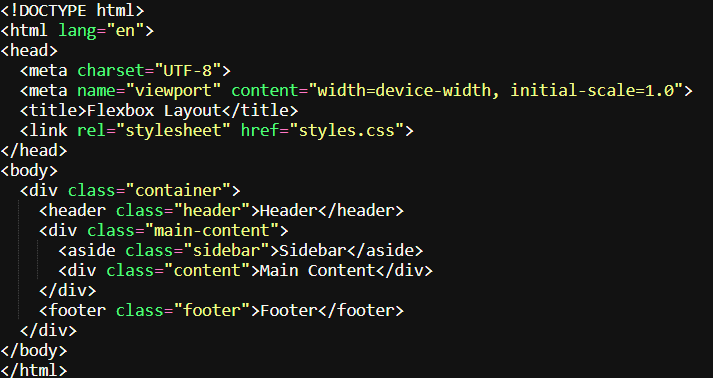
• A footer.

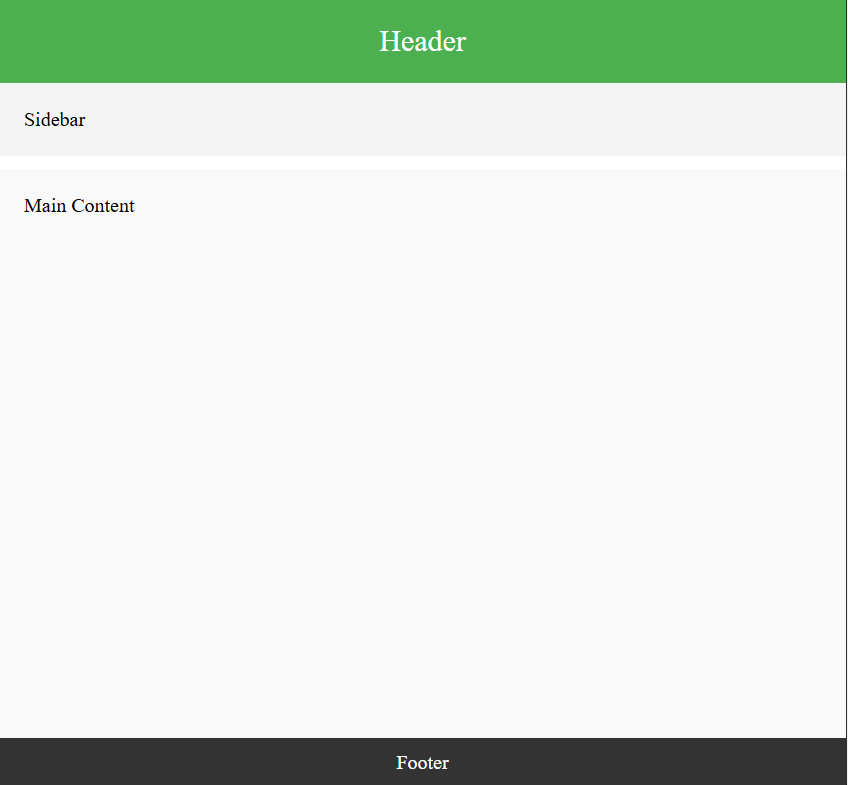
Additional Requirements:

• 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.







# CSS Grid:

# 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.

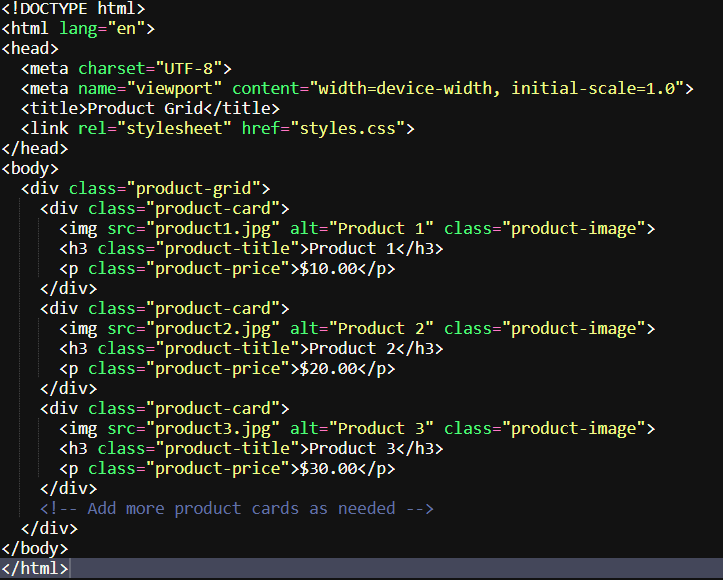
• A price.

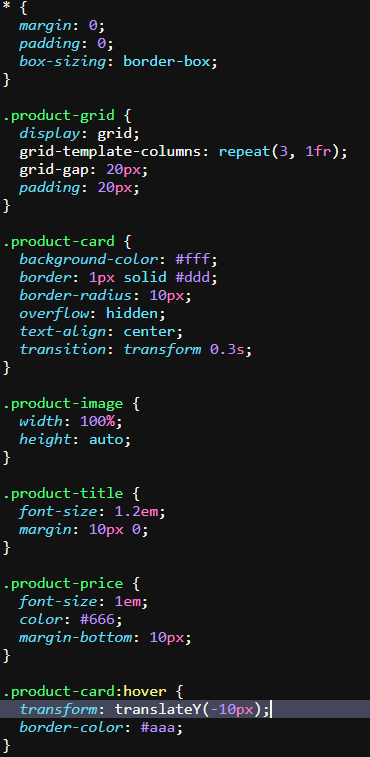
Additional Requirements:

• 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.







# Responsive Web Design with Media Queries:

# Theory Assignment:

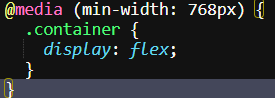
Question 1: What are media queries in CSS, and why are they important for responsive design?

**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.

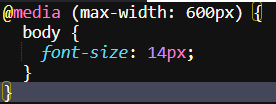
# Example:



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:



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.

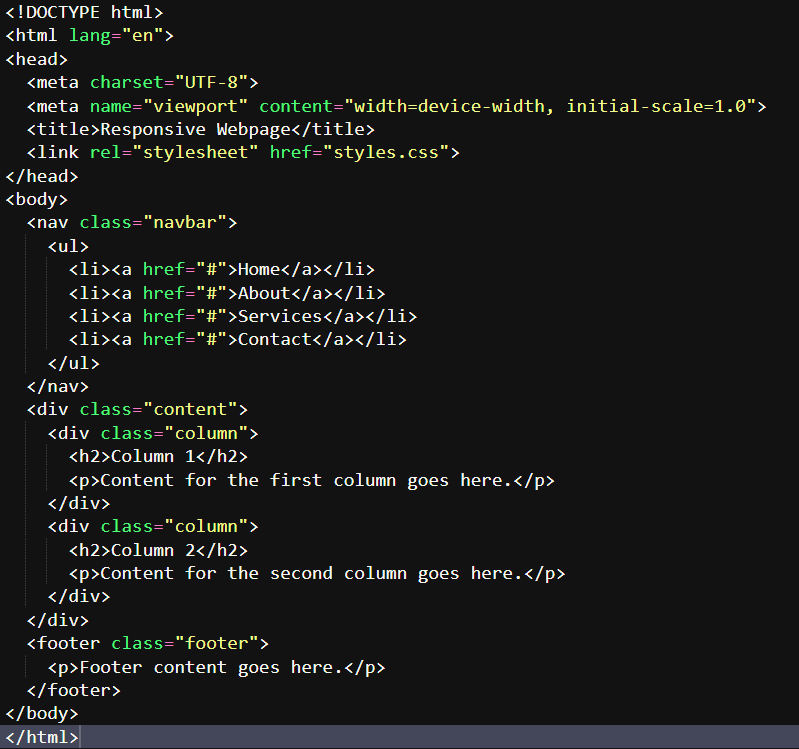
• A footer.

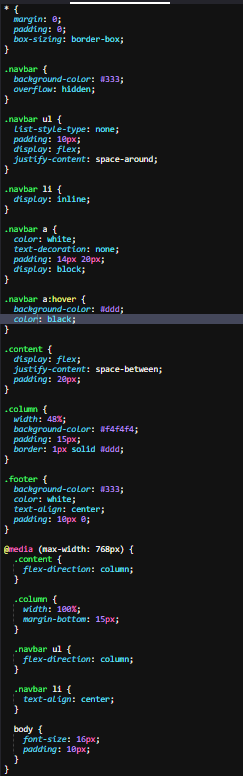
Additional Requirements:

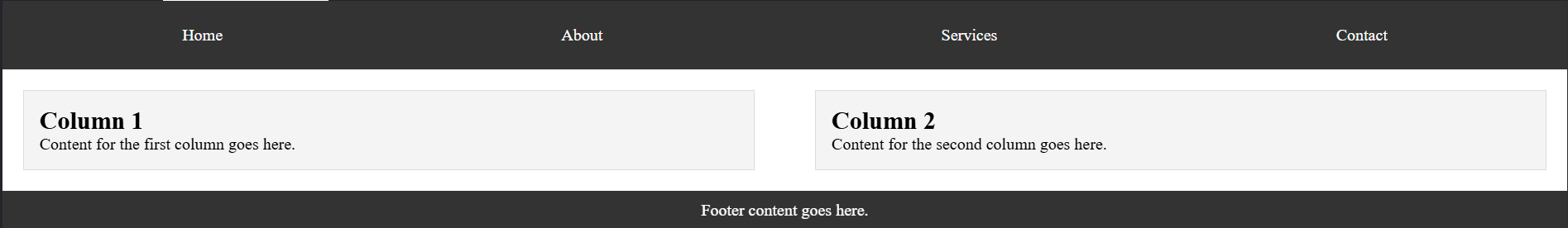
• Use media queries to make the webpage responsive for mobile devices.

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

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







# 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

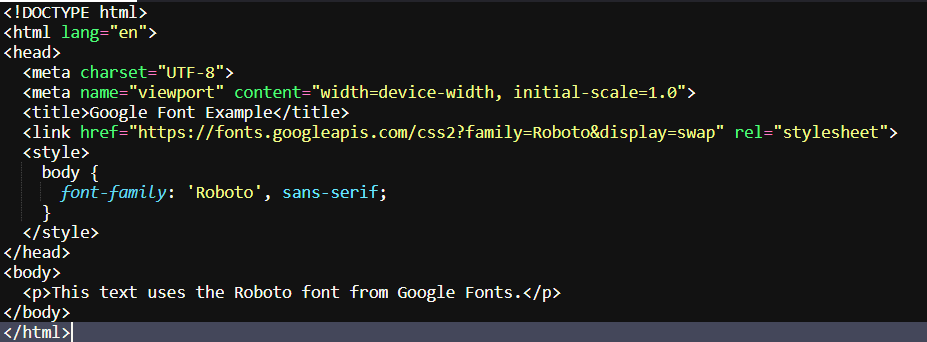


**2.Apply the Font in CSS**:

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



# Example:



**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.

