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 specific elements in an HTML document. It defines the elements to which a set of CSS rules will apply.

Types of CSS Selectors and Examples

1. Element Selector

Selects all HTML elements of a specific type.

Example:

HTML Code:

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
    <link rel="stylesheet" href="style.css">
</head>
<body>
    <p>this is a new paragraph</p>
    <p>this is a second paragraph</p>
</body>
</html>
```

CSS Code:

```css
p
{
    color: blue;
    background-color: brown;
}
```

2. Class Selector

Targets elements with a specific class attribute. Classes are reusable across multiple elements.

Example:

HTML Code:

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
    <link rel="stylesheet" href="style.css">
</head>
<body>
    <div class="highlight">This is highlighted text.</div>
    <span class="highlight">This is also highlighted text.</span>

</body>
</html>
```

CSS Code:

```css
.highlight {
    background-color: yellow;
    font-weight: bold;
}
```

3. ID Selector

Targets a single element with a specific id attribute. IDs should be unique within a document.

Example:

HTML Code:

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
    <link rel="stylesheet" href="style.css">
</head>
<body>
    <h1 id="main-title">Welcome to My Website</h1>
</body>
</html>
```

CSS Code:

```css
#main-title {
    color: green;
    text-align: center;
}
```

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

CSS specificity is a set of rules used by browsers to determine which CSS rule to apply when multiple rules target the same element. Specificity calculates the importance of a rule based on the types of selectors used.

1. Specificity

- Styles with higher specificity take precedence over those with lower specificity.
- Order of specificity (from highest to lowest):

1. Inline styles (style="...").
2. ID selectors (#id).
3. Class, attribute, and pseudo-class selectors (.class, [attr], :hover).
4. Element and pseudo-element selectors (p, ::before).

2. Source Order (Cascade)

- If two rules have the same specificity, the rule that appears later in the CSS takes precedence.

Example:

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

3. !important Declaration

- A rule with !important overrides all other rules, regardless of specificity or source order.

Example:

```
p {
    color: blue !important;
}
p {
    color: red;
}
```

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

1. Inline CSS

- Styles are applied directly to individual elements using the style attribute.
- Used for quick, one-time styling changes on specific elements.

Advantages:

- Ideal for quick, single-element styling.
- Does not rely on external files or <style> tags.

Disadvantages:

- Clutters the HTML structure, making it harder to read and maintain.

- Not reusable, leading to inconsistent styles across pages.

2. Internal CSS

- Styles are placed directly within the HTML document, inside a <style> tag located in the <head> section.
- Typically used for applying styles to a single page.

Internal CSS:

Advantages:

- Simple and quick to apply for single-page styling.
- No need for additional files.

Disadvantages:

- Not reusable across multiple pages.
- Makes the HTML document larger and harder to maintain.

3. External CSS

- Styles are written in a separate file with a .css extension and linked to the HTML document.
- Used for applying consistent styles across multiple pages.

Advantages:

- Promotes consistency by reusing styles across multiple pages.
- Keeps HTML files clean and organized.
- Easier to maintain and update styles for larger projects.

Disadvantages:

- Requires an additional HTTP request to load the stylesheet, which may affect performance.
- Testing and debugging may require additional tools.

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.

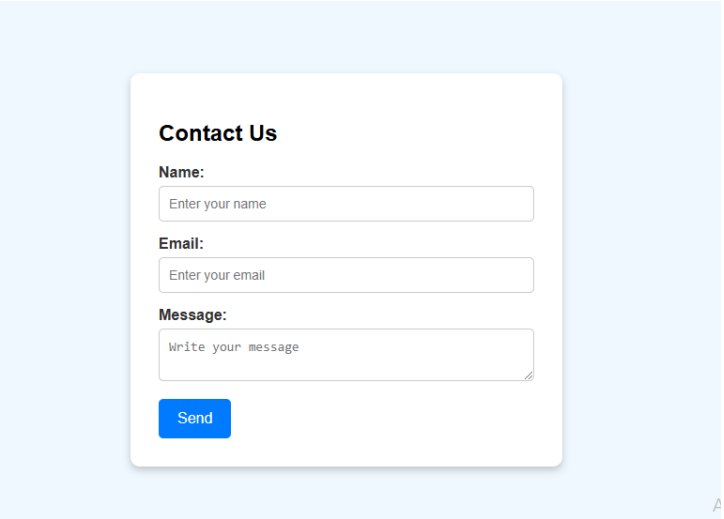
HTML Code:

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Styled Contact Form</title>
    <link rel="stylesheet" href="css 1.css">
</head>
<body>
    <form id="contact-form">
        <h2>Contact Us</h2>

        <label for="name">Name:</label>
        <input type="text" id="name" class="form-field" placeholder="Enter your name" required>

        <label for="email">Email:</label>
        <input type="email" id="email" class="form-field" placeholder="Enter your email" required>

        <label for="message">Message:</label>
        <textarea id="message" class="form-field" placeholder="Write your message" required></textarea>

        <button type="submit" id="submit-button">Send</button>
    </form>
</body>
</html>
```

CSS Code:

```css
body {
    font-family: Arial, sans-serif;
    background-color: #f0f8ff;
    margin: 0;
    padding: 0;
    display: flex;
    justify-content: center;
    align-items: center;
    height: 100vh;
}

#contact-form {
    background-color: #ffffff;
    padding: 30px;
    border-radius: 10px;
    box-shadow: 0 4px 8px rgba(0, 0, 0, 0.2);
    max-width: 400px;
    width: 100%;
}

label {
    display: block;
    margin-bottom: 5px;
    font-weight: bold;
    color: #333;
}

.form-field {
    width: 100%;
    padding: 10px;
    margin-bottom: 15px;
    border: 1px solid #ccc;
    border-radius: 5px;
    font-size: 14px;
    box-sizing: border-box;
}

#submit-button {
    background-color: #007BFF;
    color: #ffffff;
    border: none;
    padding: 12px 20px;
    font-size: 16px;
    border-radius: 5px;
    cursor: pointer;
    transition: background-color 0.3s ease;
}
#submit-button:hover {
    background-color: #0056b3;
}
```

Output:



• 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 describes how the size of an element is calculated, including its content, padding, border, and margin:

1. **Content:** The main area where text or images are displayed. Its size is defined by the width and height properties.
2. **Padding:** The space between the content and the border. It increases the element's size by adding extra space **inside** the border.
3. **Border:** A line surrounding the padding. It adds to the total size of the element, depending on its width.
4. **Margin:** The space **outside** the border, creating distance between the element and other elements. It does not affect the element's size but influences its placement.

Total Size (Standard):
Total Width = Content + Padding + Border
Total Height = Content + Padding + Border

Using box-sizing: border-box; , padding and border are included in the specified width and height.

• 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.

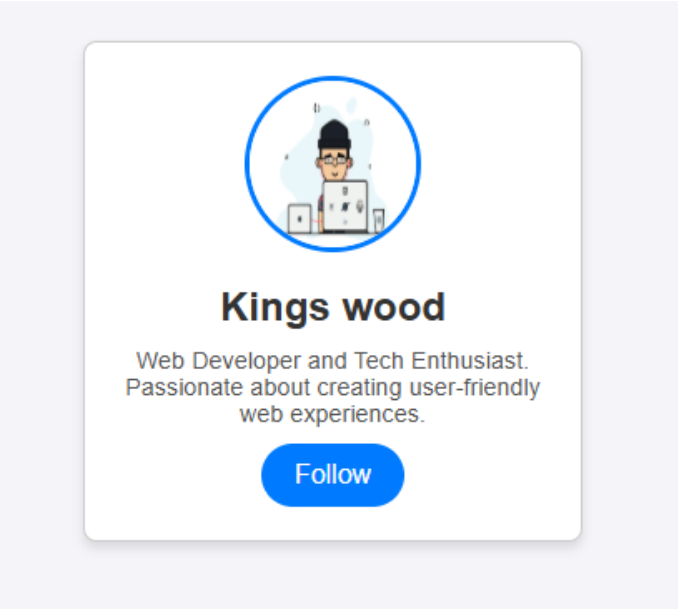
• Ensure the layout is clean and centered on the page using CSS margins.

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

```html
<!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="css 1.css">
</head>
<body>
    <div class="profile-card">
        <img src="download 45.jpeg" alt="Profile Picture" class="profile-picture">
        <h2 class="user-name">Lord Johnnie</h2>
        <p class="user-bio">Web Developer, Tech Enthusiast, and cricket lover</p>
        <button class="follow-button">Follow</button>
    </div>
</body>
</html>
```

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

body {
    font-family: Arial, sans-serif;
    background-color: #f4f4f9;
    display: flex;
    justify-content: center;
    align-items: center;
    height: 100vh;
    margin: 0;
}

.profile-card {
    width: 300px;
    background-color: #ffffff;
    border: 1px solid #ccc;
    border-radius: 10px;
    padding: 20px;
    text-align: center;
    box-shadow: 0 4px 6px rgba(0, 0, 0, 0.1);
}

.profile-picture {
    width: 150px;
    height: 150px;
    border-radius: 50%;
    margin-bottom: 15px;
    border: 3px solid #007BFF;
    box-sizing: content-box;
}

.user-name {
    font-size: 1.5rem;
    font-weight: bold;
    color: #333;
    margin-bottom: 10px;
}

.user-bio {
    font-size: 1rem;
    color: #666;
    margin-bottom: 20px;
}

.follow-button {
    background-color: #007BFF;
    color: #ffffff;
    border: none;
    padding: 10px 20px;
    border-radius: 5px;
    cursor: pointer;
    transition: background-color 0.3s ease;
    box-sizing: border-box;
}

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

OUTPUT:



• CSS Flexbox

Theory Assignment

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

CSS Flexbox (Flexible Box Layout) is a layout model in CSS designed to efficiently align and distribute space among items within a container, even when their sizes are dynamic. It simplifies creating responsive designs.

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?

1.justify-content:
Aligns items along the **main axis** (horizontal by default).

- Values:
    1. flex-start (default): Items align at the start.
    2. flex-end: Items align at the end.
    3. center: Items align in the center.
    4. space-between, space-around, space-evenly: Distribute space between/around items.

2.align-items:

- Controls the alignment of flex items **vertically** (along the cross axis).
- Values:
    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:**
   Sets the direction of the **main axis**.

- Values:

    1. row (default): Items go left-to-right.
    2. row-reverse: Items go right-to-left.
    3. column: Items stack top-to-bottom.
    4. column-reverse: Items stack bottom-to-top.

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.

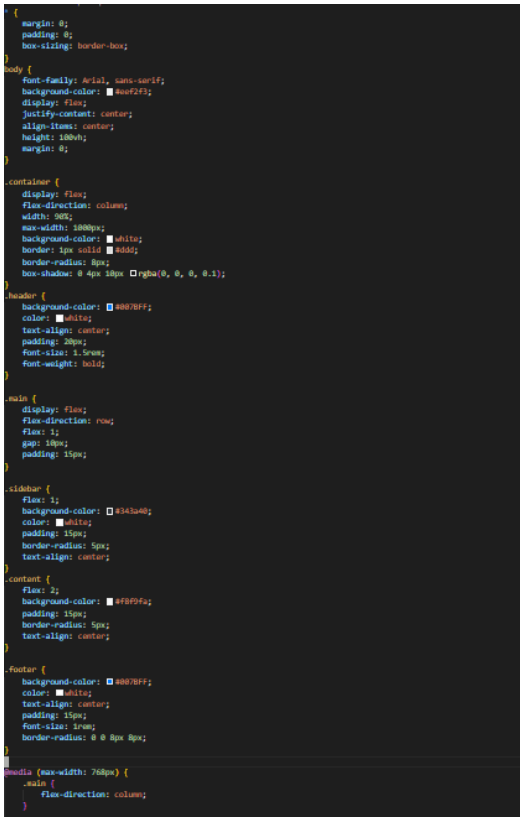
Additional Requirements:

• Use Flexbox to position and align the elements.

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

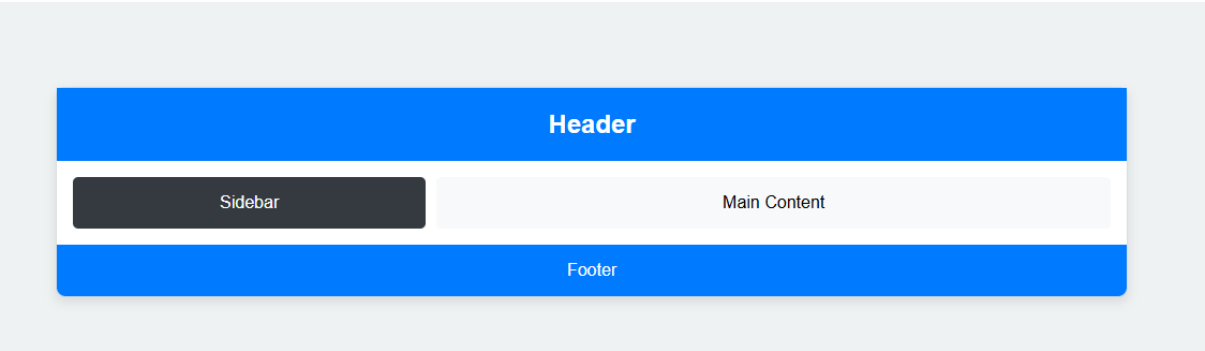
• Ensure the layout is responsive, adjusting for smaller screens.

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Simple Flexbox Layout</title>
    <link rel="stylesheet" href="css 1.css">
</head>
<body>
    <div class="container">
        <header class="header">Header</header>
        <div class="main">
            <aside class="sidebar">Sidebar</aside>
            <section class="content">Main Content</section>
        </div>
        <footer class="footer">Footer</footer>
    </div>
</body>
</html>
```



```
{
    margin: 0;
    padding: 0;
    box-sizing: border-box;
}
body {
    font-family: Arial, sans-serif;
    background-color: #eef2f3;
    display: flex;
    justify-content: center;
    align-items: center;
    height: 100vh;
    margin: 0;
}

.container {
    display: flex;
    flex-direction: column;
    width: 90%;
    max-width: 1000px;
    background-color: white;
    border: 1px solid #ddd;
    border-radius: 8px;
    box-shadow: 0 4px 10px rgba(0, 0, 0, 0.1);
}
.header {
    background-color: #007BFF;
    color: white;
    text-align: center;
    padding: 20px;
    font-size: 1.5rem;
    font-weight: bold;
}

.main {
    display: flex;
    flex-direction: row;
    flex: 1;
    gap: 10px;
    padding: 15px;
}

.sidebar {
    flex: 1;
    background-color: #343a40;
    color: white;
    padding: 15px;
    border-radius: 5px;
    text-align: center;
}
.content {
    flex: 2;
    background-color: #f8f9fa;
    padding: 15px;
    border-radius: 5px;
    text-align: center;
}

.footer {
    background-color: #007BFF;
    color: white;
    text-align: center;
    padding: 15px;
    font-size: 1rem;
    border-radius: 0 0 8px 8px;
}

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

OUTPUT:



• 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.

```html
<!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="css 1.css">
</head>
<body>
    <div class="product-grid">
        <!-- Product Card 1 -->
        <div class="product-card">
            <img src="download.jpg" alt="Product 1" class="product-image">
            <h3 class="product-title">Product 1</h3>
            <p class="product-price">$10.00</p>
        </div>
        <!-- Product Card 2 -->
        <div class="product-card">
            <img src="images 123.jpeg"alt="Product 2" class="product-image">
            <h3 class="product-title">Product 2</h3>
            <p class="product-price">$20.00</p>
        </div>
        <!-- Product Card 3 -->
        <div class="product-card">
            <img src="images 8999.jpeg" alt="Product 3" class="product-image">
            <h3 class="product-title">Product 3</h3>
            <p class="product-price">$30.00</p>
        </div>
    </div>
</body>
</html>
```

```css
body {
    font-family: Arial, sans-serif;
    background-color: #f4f4f9;
    display: flex;
    justify-content: center;
    align-items: center;
    min-height: 100vh;
    margin: 0;
}

.product-grid {
    display: grid;
    grid-template-columns: repeat(3, 1fr);
    gap: 20px;
    width: 90%;
    max-width: 1200px;
}

.product-card {
    background-color: #ffffff;
    border: 1px solid #ddd;
    border-radius: 10px;
    box-shadow: 0 4px 6px rgba(0, 0, 0, 0.1);
    text-align: center;
    padding: 15px;
    transition: transform 0.3s ease, box-shadow 0.3s ease;
}

.product-card:hover {
    transform: translateY(-5px);
    box-shadow: 0 8px 12px rgba(0, 0, 0, 0.2);
}
.product-image {
    width: 100%;
    border-radius: 5px;
    margin-bottom: 10px;
}

.product-title {
    font-size: 1.2rem;
    margin-bottom: 5px;
    color: #333;
}

.product-price {
    font-size: 1rem;
    font-weight: bold;
    color: #007BFF;
}
```

OUTPUT:

Product 1

$10.99

Product 2

$17.99

Product 3

$25.99