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.

Answer:

What is a CSS selector?

- A CSS selector is a **pattern used to select (target) HTML elements** so you can apply styles to them.
- It tells the browser **which element(s)** the CSS rules should affect.

Types of Selectors with Examples

1. Element Selector

- Selects all HTML elements of a specific type.
- **Example:**
  h1 { } → applies style to all <h1> elements.

2. Class Selector

- Used to select elements with a **class attribute**.
- A class can be used on **multiple elements**.
- **Example:**
  .title { } → targets any element with class="title".

3. ID Selector

- Selects a single element with a specific **unique ID**.
- IDs should be used **only once** on a page.
- **Example:**
  #header { } → targets the element with id="header".

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

Answer:

What is CSS Specificity?

- CSS specificity is a **score (priority system)** that decides **which style rule will be applied** when multiple rules target the same element.
- In simple words:
  Higher specificity = higher priority.

Specificity Order (Highest to Lowest)

1. **Inline Styles**
   (written directly inside the HTML element)
   Example: <p style="color:red;">
2. **ID Selectors**
   Example: #header
3. **Class, Attribute, and Pseudo-Class Selectors**
   Example: .title, [type="text"], :hover
4. **Element Selectors**
   Example: h1, p, div
5. **Universal Selector (*)**
   Lowest priority
6. **Inline > ID > Class > Element**

How conflicts are resolved?

If two or more styles target the same element:

- **The rule with higher specificity wins.**

Example:

- #box (ID) beats .box (class)
- .box (class) beats div (element)

If specificity (stronger) is the same:

- The **rule written later** in the CSS file wins (last rule overrides earlier one).
- p { color: blue; }
- p { color: red; }
- Last rule wins

If both have equal priority and timing:

- The browser uses the last one it reads.
- .title { font-size: 20px; }
- .title { font-size: 30px; }
- Same selector, same priority. Last rule wins.

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

Answer:

1. Inline CSS

Definition:

CSS written directly inside an HTML tag using the style attribute.

Example:
<h1 style="color:red;">Hello</h1>

Advantages:

- Easy to apply quick styles.
- Good for testing or very small changes.
- Overrides internal and external CSS (high specificity).

Disadvantages:

- Not reusable (must repeat style again and again).
- Makes HTML messy and hard to maintain.
- Not recommended for large projects.

2. Internal CSS

Definition:

CSS written inside the <style> tag in the <head> section of an HTML file.

Advantages:

- Good for styling a **single page**.
- Styles stay in one place (head section).
- Easier to manage than inline CSS.

Disadvantages:

- Not reusable across multiple pages.
- Increases page size.
- Still mixes HTML and CSS in one file.

3. External CSS

Definition:

CSS written in a separate .css file and linked with <link>.

Advantages:

- Best for large projects and multiple pages.
- Reusable: one file can style the whole website.
- Cleaner HTML and easier maintenance.
- Loads faster due to caching.

Disadvantages:

- Requires extra file linking.
- Page may load unstyled for a moment if CSS file loads slowly.

Note: If inline, internal, and external CSS are used together, **inline CSS has the highest priority, followed by internal CSS, and then external CSS**.

➢ CSS Box Model - Theory Assignment:

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

Answer:

The CSS Box Model describes how every HTML element is displayed as a box on a webpage. It is made up of four layers: **content → padding → border → margin**

1. Content

- The actual text, image, or data inside the box.
- **Affects size:** Changing width/height changes the size of the content area.

2. Padding

- Space inside the box, between the content and the border.
- **Affects size:** Adding padding increases the total size of the element because the box grows outward.

3. Border

- A line surrounding the padding and content.
- **Affects size:** Border thickness increases the total size of the element.

4. Margin

- Space outside the box, creating distance between elements.
- **Does NOT change the element's size**, but affects the space around it.

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

Answer:

1. content-box (default)

- Width and height apply **only to the content area**.
- Padding and border are **added on top**, which **increases the total size** of the element.
- This is the **default** box-sizing in CSS.

Example meaning:
If width = 200px and padding/border are added, the final size becomes more than 200px.

2. border-box

- Width and height include **content + padding + border**.
- The total size of the element **stays the same**, even if padding or border increases.
- Makes layouts easier to control.

Example meaning:
If width = 200px, the final size remains **200px**, no matter the padding or border.

Short Difference

- **content-box (default):** padding and border increase total size.
- **border-box:** total size stays fixed; padding and border fit inside the given width/height.

➢ CSS Flexbox - Theory Assignment:

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

Answer:

What is CSS Flexbox?

- CSS Flexbox (Flexible Box Layout) is a layout system that helps you arrange elements easily in a row or a column.
- It automatically adjusts spacing, alignment, and size of items to fit different screen sizes.

Why is Flexbox useful?

- Makes responsive design easier
- Automatically handles spacing between items
- Helps align items (center, left, right, top, bottom)
- Items can grow, shrink, or wrap based on available space
- Great for modern layouts like navbars, cards, forms, and grids

1. Flex-container

- The parent element where **display: flex** is applied.
- It controls the layout direction, spacing, alignment, and wrapping of its children.

Example meaning:
If you apply display: flex on a , that becomes the **flex-container**.

2. Flex-item

- The child elements inside a flex-container.
- These items follow the rules of Flexbox (grow, shrink, align).

Example meaning:
If a has 3 boxes inside it, those 3 boxes are **flex-items**.

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

Answer:

1. flex-direction

This property decides **the direction in which flex items are arranged**.

- **row** → items go left to right
- **column** → items go top to bottom

How it affects the axes

- **Main Axis** = direction items flow
- **Cross Axis** = opposite direction

Example 1: flex-direction: row

- Items arranged **left → right**
- **Main axis = horizontal**
- **Cross axis = vertical**

So here:

- **justify-content** controls **left–right** alignment
- **align-items** controls **top–bottom** alignment

Example 2: flex-direction: column

- Items arranged **top → bottom**
- **Main axis = vertical**
- **Cross axis = horizontal**

So here:

- **justify-content** controls **top–bottom** alignment
- **align-items** controls **left–right** alignment

2. justify-content

Controls **how items are aligned on the main axis**.
(Left–right when row, top–bottom when column)

Common values:
flex-start, center, flex-end, space-between, space-around, space-evenly

3. align-items

Controls **how items are aligned on the cross axis**.
(Top–bottom when row, left–right when column)

Common values:
flex-start, center, flex-end, stretch

- **CSS Grid - Theory Assignment:**

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

Answer:

What is CSS Grid?

- CSS Grid is a **two-dimensional layout system** used to design webpages using **rows and columns** at the same time.
- It is best for creating **full page layouts** and complex designs.

How CSS Grid differs from Flexbox

Feature	CSS Grid	Flexbox
Layout type	Two-dimensional (rows **and** columns)	One-dimensional (row **or** column)
Best for	Page layout, complex structures	Small components, alignment
Control	Controls rows and columns together	Controls items in one direction
Usage level	Layout of whole page	Layout of sections or items

When to use Grid over Flexbox

Use **CSS Grid** when:

- You want to design a **full webpage layout**
- You need **both rows and columns** control
- The layout is **complex** (dashboard, gallery, grid system)
- Precise placement of items is needed

Use **Flexbox** when:

- You are aligning items in **one direction**
- Creating **navbars, cards, buttons, forms**
- Simple and flexible layouts are required

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

Answer:

1. grid-template-columns

- Defines the **number and width of columns** in a grid layout.
- You can set fixed sizes, flexible sizes, or auto sizes.

Example use:
Create **3 columns**, each 200px wide:

grid-template-columns: 200px 200px 200px;

Create **3 equal columns**:

grid-template-columns: 1fr 1fr 1fr;

2. grid-template-rows

- Defines the **number and height of rows** in the grid.
- Works the same way as columns but in vertical direction.

Example use:
Create **2 rows**, each 150px high:

grid-template-rows: 150px 150px;

3. grid-gap (now called gap)

- Controls the **space between rows and columns**.
- Makes spacing clean and easy.

Example use:
Add space between grid items:
grid-gap: 20px;

Different spacing for rows and columns:
grid-gap: 10px 20px; /* row gap | column gap */

Summary

- **grid-template-columns** → sets column sizes
- **grid-template-rows** → sets row sizes
- **grid-gap (gap)** → sets space between grid items

➢ Responsive Web Design with Media Queries - Theory Assignment:

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

Answer:

What are media queries?

- Media queries are a **CSS feature used to apply styles based on screen size, device type, or resolution**.
- They allow a website to change its layout and design for different devices like mobile, tablet, and desktop.

Why are media queries important?

- They help create **responsive websites**.
- The webpage adjusts automatically to different screen sizes.
- Improve user experience on mobile, tablet, and desktop.
- Avoids zooming and horizontal scrolling.
- Makes websites look good on all devices.

Simple Example (meaning)

- One design for mobile screens
- Another design for larger screens

CSS decides which style to apply using media queries.

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

Answer:

A **basic media query example** that changes font size for screens smaller than **600px**:

Example:

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

Explanation:

- @media (max-width: 600px) → applies styles when screen width is **600px or less**
- body { font-size: 14px; } → reduces font size for small screens (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?

Answer:

Web-safe Fonts

- Web-safe fonts are **already installed on most devices and browsers**.
- They display the same on all systems without extra downloads.
- Examples: Arial, Times New Roman, Verdana, Georgia.

Custom Web Fonts

- Custom web fonts are **not pre-installed** on user devices.
- They are downloaded from the internet (Google Fonts, Adobe Fonts, etc.).
- Provide more design and branding options.
- Examples: Poppins, Roboto, Montserrat.

Why use web-safe fonts over custom fonts?

- Faster page loading (no font download needed).
- Better performance on slow internet.
- Guaranteed compatibility across all browsers and devices.
- Simple and reliable for basic websites.

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

Answer:

What is font-family in CSS?

- The font-family property is used to **set the font style of text** on a webpage.
- You can give a **list of fonts** so that if one font is not available, the next one is used.

Example meaning:
If the first font is not found, the browser uses the next available font.

How to apply a custom Google Font to a webpage

Step 1: Select a font from Google Fonts

- Go to **fonts.google.com**
- Choose a font (for example: Poppins, Roboto)

Step 2: Link the Google Font in HTML

Add the link inside the <head> section:

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

Step 3: Apply the font using CSS

Use font-family in your CSS:

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