Theory

Module 3 – Mernstack – CSS and CSS3

CSS Selectors & Styling

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

   Ans:

   A **CSS selector** is a pattern used to select and style HTML elements. It tells the browser which elements the CSS rules should apply to.

   1. Element Selector

   Selects all elements of a given type.

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

   2. Class Selector

   Selects elements with a specific class attribute. Use a. before the class name.

   ```
   .card {
     border: 1px solid gray;
   }
   ```

   3. ID Selector

   Selects a single element with a specific ID. Use # before the ID name.

   ```
   #header {
     background-color: lightgray;
   }
   ```

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

   Ans:

   CSS specificity is the set of rules that browsers use to determine which style rules are applied to an element when there are conflicting rules.

   It works like a ranking system: each selector has a specificity value based on what kind of selector it is.

Specificity Breakdown:

- Inline styles (e.g., style="color: red"): highest specificity.
- ID selectors (e.g., #header): high specificity.
- Class selectors, attributes, and pseudo-classes (e.g., .nav, [type="text"], :hover): medium specificity.
- Element selectors and pseudo-elements (e.g., div, h1, ::before): low specificity.

How Conflicts Are Resolved:

1. Compare Specificity – The rule with the highest specificity wins.
2. Source Order – If specificity is the same, the last rule defined in the CSS wins.
3. **!**important – Overrides all normal rules, but can still be overridden by another !important rule with higher specificity.

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

Ans:

1. Inline CSS

CSS is written directly in the HTML element using the style attribute.

```
<p style="color: red;">Hello</p>
```

Advantages:

- Quick to apply styles to a single element.
- Useful for testing or small changes.

Disadvantages:

- Difficult to maintain or update.
- Breaks separation of content and design.
- Low reusability.

2. Internal CSS

CSS is written inside a <style> tag within the <head> of the HTML file.

```
<head>
  <style>
    p { color: blue; }
  </style>
</head>
```

Advantages:

- Styles are kept in one place for a single HTML file.
- Better than inline for medium-sized projects.

Disadvantages:

- Not reusable across multiple pages.
- **Can make the HTML file larger and harder to manage.**

3. External CSS
CSS is written in a separate .css file and linked using the <link> tag.

<link rel="stylesheet" href="styles.css">

Advantages:

- Promotes reusability across multiple pages.
- Easier to maintain and scale.
- Keeps HTML cleaner and focused on content.

Disadvantages:

- Requires an extra HTTP request (though it's often cached).
- Won't work if the external file fails to load.

CSS Box Model

1. Explain the CSS box model and its components (content, padding, border, margin). How does each affect the size of an element?
   Ans:
   CSS box model is a core concept in web design that describes how elements are structured and how their size is calculated. Every HTML element is considered a box consisting of four main parts:

❖ Content
- The actual content of the element (text, images, etc.).
- Size is set using width and height.

❖ Padding
- Space inside the element, between the content and the border.
- Increases the space around the content, without affecting the border.
- Adds to the total size of the element.

❖ Border

- The edge around the padding and content.
- Thickness is controlled using border-width (and style/color as well).
- Also adds to the total size.

❖ Margin

- Space outside the element, between this box and surrounding elements.
- Does not affect the size of the element itself but affects layout spacing.

How the Box Size Is Calculated:

❖ Default Behavior: box-sizing: content-box

- width and height apply only to the content.
- Padding and border are added on top of that.

  Example:
  width: 200px;
  padding: 10px;
  border: 5px solid;
  Total width = 200 + 10×2 (padding) + 5×2 (border) = 230px

❖ Alternative: box-sizing: border-box

- The width and height include the padding and border.
- Makes layout more predictable and easier to manage.

  Same example with border-box:
  box-sizing: border-box;
  width: 200px;
  padding: 10px;
  border: 5px solid;
  Now the actual content area shrinks to fit inside the 200px.

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

Ans:

content-box (Default):

❖ How it works:

- The width and height apply only to the content area.

- Padding and border are added outside the content, increasing the total size of the element.

➢ Total Size Calculation:

Total width = width + padding + border
Total height = height + padding + border

➢ Example:

box-sizing: content-box;
width: 200px;
padding: 20px;
border: 5px solid;
Total width = 200 + 40 + 10 = 250px

border-box:

❖ How it works:

- The width and height include padding and border.
- The content area shrinks to make space for them.

➢ Total Size Calculation:

Total width = declared width (includes padding + border)

➢ Example:

box-sizing: border-box;
width: 200px;
padding: 20px;
border: 5px solid;
Total width = 200px
The content area becomes 200 - 40 - 10 = 150px

Which is the Default?

- content-box is the default value in CSS.

CSS Flexbox

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

Ans:

CSS Flexbox (short for *Flexible Box Layout*) is a CSS layout model that makes it easy to design flexible and responsive layout structures without using floats or positioning. It helps you align, space, and distribute items inside a container — even when their size is unknown or dynamic.

Flexbox is super useful because it:

- Can easily center items vertically and horizontally.

- Automatically adjusts the size of items to fill available space.
- Makes it easier to create layouts that adapt to different screen sizes (responsive design).

❖ **Key Terms:**

- Flex-container:
  This is the parent element that has display: flex; (or display: inline-flex;) applied to it. It defines a flex context for its direct children, making them "flex items."
  Example:

```
.container {
  display: flex;
}
```

- Flex-item:
  These are the direct children of a flex-container. They are automatically laid out according to flexbox rules.
  Example:

```
<div class="container">
  <div class="flex-item">Item 1</div>
  <div class="flex-item">Item 2</div>
</div>
```

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

Ans:

1. justify-content
   - What it does:
     Controls how flex items are spaced along the main axis (horizontal by default).
   - Common values:
     - flex-start → Items packed at the start
     - flex-end → Items packed at the end
     - center → Items centered
     - space-between → Equal space between items
     - space-around → Equal space around items
     - space-evenly → Equal space between and around items

   Example:

```
.container {
display: flex;
justify-content: center;
}
```

2. align-items

- What it does:
  Controls how flex items are aligned along the cross axis (vertical by default).
- Common values:
  - stretch → Items stretch to fill the container (default)
  - flex-start → Items align to the top
  - flex-end → Items align to the bottom
  - center → Items align at the center
  - baseline → Items align along their text baseline

Example:

```
.container {
display: flex;
 align-items: center;
}
```

3. flex-direction

- What it does:
  Defines the direction the main axis runs — thus the direction the flex items are placed in.
- Common values:
  - row → Left to right (default)
  - row-reverse → Right to left
  - column → Top to bottom
  - column-reverse → Bottom to top

Example:

```
.container {
display: flex;
flex-direction: column;
}
```

CSS Grid

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

Ans:

CSS Grid

CSS Grid is a layout system that allows you to build *two-dimensional* layouts — meaning you can control layout in rows and columns at the same time.

- With Grid, you can design complex web pages with precise control over both horizontal and vertical placement of elements.
- You define a grid container and set up grid tracks (rows and columns) inside it.

Example:

```
.container {
  display: grid;
  grid-template-columns: 200px 1fr 1fr;
  grid-template-rows: 100px 200px;
}
```

When to use Grid over Flexbox

- Use Grid when you need a full-page layout or a structured grid with both rows and columns.
  - Example: designing a webpage layout with a header, sidebar, main content, and footer.
- Use Flexbox when you only need to arrange items in a single direction.
  - Example: building a navbar, stacking buttons in a row, or aligning a few cards.

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

Ans:

1. grid-template-columns

- What it does:
  Defines the number and size of columns in your grid.

Example:

```
.container {
  display: grid;
  grid-template-columns: 200px 1fr 2fr;
}
```

- This sets up 3 columns:
  - 1st column → 200px wide
  - 2nd column → takes up 1 part of available space
  - 3rd column → takes up 2 parts of available space

2. grid-template-rows

- What it does:
  Defines the number and size of rows in your grid.

Example:

```
.container {
  display: grid;
  grid-template-rows: 100px auto 50px;
}
```

- This sets up 3 rows:
  - 1st row → 100px height
  - 2nd row → automatically stretches to fill available space
  - 3rd row → 50px height

3. grid-gap (now commonly written as gap)

- What it does:
  Adds spacing between rows and columns without needing margins.

Example:

```
.container {
  display: grid;
  grid-template-columns: 1fr 1fr 1fr;
  grid-gap: 20px;
}
```

- This creates 3 equal columns with 20px of space between each row and column.

Responsive Web Design with Media Queries

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

Ans:

Media queries are special CSS techniques that allow your web page to change its style depending on things like:

- screen size
- device type (mobile, tablet, desktop)
- orientation (portrait or landscape)
- resolution

In short, they make your website responsive — meaning it looks good and works well on all devices!

Why Media Queries are Important for Responsive Design

- Adapt Layouts: You can create different layouts for mobile phones, tablets, laptops, and desktops.
- Improve User Experience: Ensures your content is easy to read and navigate on any screen size.
- Save Resources: You can hide or shrink images, buttons, or content that's unnecessary on smaller screens.
- Follow Best Practices: Responsive design is now essential — most users browse on mobile!

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

Ans:

Here's a basic media query that adjusts the font size when the screen is smaller than 600px:

```
body {
  font-size: 18px; /* Default font size for larger screens */
}

@media (max-width: 600px) {
  body {
    font-size: 14px; /* Smaller font size for small screens */
  }
}
```

❖ What this does:

- Normally, the font size is 18px.
- But when the screen is 600px wide or smaller, the font size changes to 14px — making it more readable on small devices!

Typography and Web Fonts

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:

❖ **Web-Safe Fonts**

- Definition:
  Web-safe fonts are fonts that are already installed on most computers, tablets, and smartphones.
- Examples:
  Arial, Times New Roman, Courier New, Verdana, Georgia, Tahoma.
- Why they're called "safe":
  Because you can trust that almost all users will have them, and your website will look consistent without extra effort.
- No downloading needed — they load instantly.

❖ **Custom Web Fonts**

- Definition:
  Custom web fonts are external fonts that you include in your website using services like Google Fonts, Adobe Fonts, or by uploading font files yourself.
- Examples:
  Roboto, Open Sans, Lato, Montserrat, Poppins, etc.
- How they work:
  The browser downloads the font file when loading the page, so you can use beautiful or unique typography not available by default.

- Usage example (Google Fonts):

```
<link href=https://fonts.googleapis.com/css2?family=Roboto&display=swap
rel="stylesheet">
body {
 font-family: 'Roboto', sans-serif;
 }
```

Why you might use a web-safe font instead of a custom font

1. Faster loading:
   No extra font files to download → your website loads quicker.
2. Better reliability:
   No risk of fonts failing to load if the user has slow internet.

3. Compatibility:
   Works on very old browsers and devices without issues.
4. Simplicity:
   Easy to set up without linking to external services.

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

Ans:

- ❖ **Definition:**
  The font-family property sets the font used for text in an HTML element.
- ❖ **How it works:**
  You can list one or more fonts in order of preference — if the first font isn't available, the browser tries the next one.

Example:

```
p {
  font-family: "Arial", "Helvetica", sans-serif;
}
```

- The paragraph text will use Arial.
- If Arial isn't available, it tries Helvetica.
- If neither is available, it falls back to any sans-serif font.

How to Apply a Custom Google Font to a Webpage

Steps:

1. Go to Google Fonts.
2. Pick a font you like (e.g., "Poppins").
3. Copy the <link> tag they give you.

Example:

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

- Place this inside the <head> section of your HTML.

4. Use the font in your CSS with font-family.

Example:

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

- Now your whole webpage will use the Poppins font!