Module 3 – Frontend – CSS and CSS3

CSS Selectors & Styling

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

➢ CSS selectors are used to "find" (or select) the HTML elements you want to style.

🔸 **Types of CSS Selectors with Examples:**

1. **Element Selector**
   - Targets HTML elements by their tag name.

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

➡️ This rule changes the text color of all <p> (paragraph) elements to blue.

2. **Class Selector**
   - Targets elements with a specific class name.
   - Starts with a dot (.).

```
.box {                          //css code
   background-color: yellow;
}
```

➡️ This applies a yellow background to any element with class="box".

```
<div class="box">Hello</div>    //Html code
```

3. **ID Selector**
   - Targets a single element with a specific ID.
   - Starts with a hash (#).

Module 3 – Frontend – CSS and CSS3

```
#header {                    //css code
    font-size: 24px;
}
```

➡️ This sets the font size of the element with id="header".

```
<h1 id="header">Welcome</h1>          //Html code
```

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

➢ **1. Inline CSS**

❖ **Definition:**

- CSS is written directly inside the HTML element using the style attribute.

Example:

```
<h1 style="color: red;">Welcome</h1>
```

✅ **Advantages:**

- Quick and easy to apply for **small changes**.
- Useful for **testing or overriding** other styles.

❌ **Disadvantages:**

- Hard to maintain and read if overused.
- Cannot apply to multiple elements at once (not reusable).
- Makes HTML messy and long.

2. Internal CSS

❖ **Definition:**

- CSS is written **within the <style> tag** inside the <head> section of the HTML file.

Example:

```
<head>
    <style>
     h1 {
          color: blue;
     }
    </style>
</head>
```

✅ **Advantages:**

- Styles are kept in one place for that specific page.
- Easier to manage than inline if used on one page only.

❌ **Disadvantages:**

- Styles are limited to **only that page**.
- Increases page size if many pages use similar styles (not reusable).

3. External CSS

❖ **Definition:**

- CSS is written in a **separate .css file** and linked to the HTML using a <link> tag.

Example (style.css):

```
h1 {
  color: green;
}
```

HTML

```
<head><link rel="stylesheet" href="style.css"></head>
```

✅ **Advantages:**

- **Reusable** across multiple pages (saves time).
- Keeps HTML clean and separates structure from style.
- Easier to maintain for large websites.

❌ **Disadvantages:**

- Requires an **extra HTTP request** to load the CSS file (may slightly slow load time).
- Styles won't apply if the external file fails to load

Module 3 – Frontend – CSS and CSS3

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?

- The CSS Box Model is a fundamental concept in web design and layout. It describes how every HTML element is essentially a box, and how the **size** and **spacing** of that box are calculated. The box model consists of **four main components**:

1. Content

- This is the innermost part of the box.
- It contains text, images, or other elements.
- You can set the size of the content using the width and height properties.
- Example: width: 200px; height: 100px;

2. Padding

- Padding is the space between the content and the border.
- It adds space inside the element, but does not affect the actual content.
- It can be set for each side: padding-top, padding-right, padding-bottom, padding-left.
- Example: padding: 10px; adds 10px on all sides inside the border.

3. Border

- The border wraps the padding (if any) and the content.

- It's the line around the element.
- You can control its thickness, style, and color.
- Example: border: 2px solid black;

4. Margin

- Margin is the space outside the element.
- It creates space between this element and others.
- Example: margin: 20px; adds 20px on all sides outside the border.

How These Affect the Total Size of an Element

```
width: 200px;
padding: 10px;                    // css code
border: 2px solid;
margin: 20px;
```

Then the actual size of the **element box** (not including margin) is:

- **Width**: 200 (content) + 10**2 (left + right padding) + 2*+2 (left + right border) = **224px**
- **Height**: same way if height is set

If you add margins, they go **outside** this, so the element will take more **space on the page**, but margin is not part of the box size.

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

- box-sizing Property
  - The box-sizing property controls **how the total width and height of an element are calculated**.

1. content-box (Default)

- Only the width and height apply to the content.
- Padding and border are added outside the content box.
- This increases the total size of the element.

Example:

```
box-sizing: content-box;  /* default */
width: 200px;                                    // css code
padding: 10px;
border: 5px solid;
```

Total element width =200 (content) + 10**2 (padding) + 5*2 (border) =**230px**

2. border-box

- The width and height include the content, padding, and border.
- The total element size stays fixed, and the content area adjusts accordingly.

Example:

```
box-sizing: border-box;
width: 200px;                                    // css code
padding: 10px;
border: 5px solid;
```

Total element width = 200px (fixed) Here, the content area =200 – 10**2 (padding) – 5*2 (border) = **170px**

Module 3 – Frontend – CSS and CSS3

CSS Flexbox

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

- CSS Flexbox (short for *Flexible Box Layout*) is a modern CSS layout model that makes it easier to design flexible and responsive layouts without using floats or positioning hacks.

It allows elements to:

- Automatically **adjust their size and position**
- **Align** items vertically and horizontally

Why Flexbox is Useful for Layout Design

Flexbox is perfect when:

- You want elements to align in a row or column
- You need to center elements easily
- You want flexible spacing between items
- You're building **responsive web designs**

Example situations:

- Navigation menus
- Cards in a grid
- Responsive forms and sidebars

Flexbox Terminology

1. **Flex Container**
   - The parent element that holds flex items.
     You make any container a flex container by setting:
     display: flex;

   This enables Flexbox layout behavior inside it.

Properties for Flex Container:

- flex-direction: row | column | row-reverse | column-reverse
- justify-content: aligns items horizontally (main axis)
- align-items: aligns items vertically (cross axis)
- flex-wrap: allows wrapping to the next line
- align-content: space between multiple rows or columns

2. Flex Items

- The child elements inside a flex container.

Properties for Flex Items:

- flex: shorthand for flex-grow, flex-shrink, and flex-basis
- align-self: overrides align-items for individual item
- order: changes the visual order of items

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

1. Here's a clear explanation of the Flexbox properties: justify-content, align-items, and flex-direction. These are used on the flex container to control the layout of the flex items inside it.

1. **flex-direction**

   This property defines the main axis — the direction in which the flex items are placed.

   Values:

   - row (default): Left to right (horizontal)
   - row-reverse: Right to left
   - column: Top to bottom (vertical)
   - column-reverse: Bottom to top

Example:

```
.flex-container {
display: flex;                    // css code
flex-direction: row; /* or column */
}
```

2. justify-content

This property aligns flex items along the main axis (horizontal by default).

Used to control spacing between items.

Common values:

- flex-start: Items align to the start of the main axis
- flex-end: Items align to the end
- center: Items are centered
- space-between: Equal space between items
- space-around: Equal space around items
- space-evenly: Equal space between and around

Example:

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

3. align-items

This property aligns flex items along the cross axis (perpendicular to the main axis).

Common values:

- stretch (default): Items stretch to fill the container height
- flex-start: Align to the top (if row) or left (if column)
- flex-end: Align to the bottom (or right)
- center: Vertically or cross-axis centered
- baseline: Align items based on their text baseline

Example:

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

Module 3 – Frontend – CSS and CSS3

CSS Grid

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

2. **CSS Grid** is a layout system that helps you **place items in rows and columns** — just like a table.

👉 It's good when you want to design **full page layouts** or **grids** (like photo galleries)

What is Flexbox?

Flexbox is also a layout system, but it works in one direction only — row (left to right) or column (top to bottom).

👉 It's great for **small parts** of a page like menus, buttons, and cards.

Grid vs Flexbox (Simple Table)

Feature	Grid	Flexbox
Direction	Row **and** Column (2D)	Only Row **or** Column (1D)
Layout Type	Full layout (like a table)	Line layout (like a row/column)
Control	Exact placement (row, column)	Items flow automatically
Best For	Web page layout, image gallery	Navbar, buttons, cards

When to Use Grid:

- **You want to design a full page with rows and columns**
- **You want to place items in specific areas**
- **Example: Header, Sidebar, Main Content, Footer**

When to Use Flexbox:

- You need items in a line (row or column)
- You want simple alignment and spacing
- Example: Menu bar, card list, button group

Easy Example:

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

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

- **1. 🧱 grid-template-columns**

  This sets how many columns you want and their width.

  Example:

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

✅ This creates 3 columns:

- 1st column: 100px
- 2nd column: 200px
- 3rd column: 100px

You can also use **fr** unit:

```
grid-template-columns: 1fr 2fr;
```

✅ This means:

- 1st column = 1 part
- 2nd column = 2 parts (twice as big)

- **2. 📏 grid-template-rows**

  This sets the **height** of each **row**.

  Example:

```
.container {
  display: grid;
  grid-template-rows: 100px 150px;
}
```

  ✅ This makes:

  - 1st row = 100px
  - 2nd row = 150px
  - You can also use auto or %, like:

```
grid-template-rows: auto 50%;
```

3. ➖ grid-gap (or gap)

This adds **space between grid items** (like margins between boxes).

Example:

```
.container {
  display: grid;
  grid-template-columns: 1fr 1fr;
  grid-template-rows: auto auto;
  grid-gap: 10px;
}
```

✅ Adds **10px gap** between all rows and columns.

You can also write:

```
gap: 10px 20px;
```

- 10px = row gap (top-bottom)
- 20px = column gap (left-right)

Module 3 – Frontend – CSS and CSS3

Responsive Web Design with Media Queries

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

➢ What Are Media Queries in CSS?

❖ Media queries are like rules in CSS that help your website look good on all devices — like mobile phones, tablets, and computers.

❖ They check the **screen size** and **change the style** if needed.

Example:

```
@media (max-width: 600px) {
  body {
    background-color: lightblue;
  }
}
```

❖ **Why Are Media Queries Important for Responsive Design?**

Media queries are important because they:

- Make your website fit nicely on small or big screens
- Help people read and use your website easily on mobile
- Change layout or style when screen size changes
- You can use one website for all devices (no need to create separate mobile sites)

Common Use Cases:

- Changing font sizes on smaller screens

- Hiding/showing elements (like a sidebar)
- Switching from horizontal to vertical layouts
- Adjusting padding and margins

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

➢ Here is a basic media query that adjusts the font size of the webpage when the screen is smaller than 600px:

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

Explanation in Easy Words:

- @media (max-width: 600px) → This means "if the screen is 600 pixels wide or smaller"
- body { font-size: 14px; } → This changes the text size to 14px on small screens like mobile phones

Module 3 – Frontend – CSS and CSS3

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?**

➢ **<u>Difference Between Web-Safe Fonts and Custom Web Fonts</u>**

1. <u>Web-Safe Fonts:</u>
   - These are common fonts that are already installed on most computers and devices.
     - ❖ Examples:
       - Arial
       - Times New Roman
       - Verdana
       - Georgia
       - Courier New
   - They are considered "safe" because they will look the **same across different devices and browsers**.
2. <u>Custom Web Fonts:</u>
   - These are **special fonts** that are **not installed** on all devices.
   - They are loaded using services like **Google Fonts** or **@font-face** in CSS.
     - ❖ Examples:
       - Roboto
       - Open Sans
       - Lato
       - Poppins
     - **Why Use Web-Safe Fonts Over Custom Fonts?**

       You might choose web-safe fonts because:

1. **Faster Loading:**
- No extra font files need to be downloaded — improves page speed.
2. **Better Compatibility:**
- Works on all browsers and devices without any issue.
3. **No Internet Needed for Fonts:**
- Works even if there's no internet or the font CDN fails.
4. **Simple Projects:**
- Good for emails, small websites, or when speed and reliability are more important than style.

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

➢ The font-family property in CSS is used to specify the typeface (font) for text content on a webpage. It allows you to set one or more font names, and the browser will use the first one available on the user's device.

Syntax:

```
selector {
  font-family: "Font Name", fallback-font, generic-family;
}
```

Example:

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

If **Arial** is not available, the browser will try a similar **sans-serif** font.

Module 3 – Frontend – CSS and CSS3

How to Apply a Custom Google Font

To use a Google Font on your website, follow these steps:

1. Go to Google Fonts
   - Visit https://fonts.google.com and choose a font.
2. Copy the Embed Link
   - After selecting a font, Google will give you a <link> tag like this:

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

   - Place it inside the <head> section of your HTML document.
3. Use the Font in CSS
   - Apply the font using the font-family property:

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