Module 3 – Frontend – CSS and CSS3

Theory Assignment

CSS Selectors & Styling

Question 1: What is a CSS selector?

A CSS selector is a pattern used to select and apply styles to HTML elements. It determines which elements on a webpage will be affected by specific CSS rules.

Examples of Selectors:

1. Element Selector: Targets all instances of a specific HTML element.

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

This applies to all <p> elements in the document.

2. Class Selector: Targets elements with a specific class attribute.

```
css
CopyEdit
.highlight {
background-color: yellow;
```

}

This applies to all elements with class="highlight".

3. ID Selector: Targets a specific element with a unique ID.

css

CopyEdit

```
#main-title {
font-size: 24px;
}
```

This applies only to the element with id="main-title".

Question 2: Explain the concept of CSS specificity.

CSS specificity is a set of rules that determine which CSS rule is applied when multiple styles target the same element.

Specificity Calculation:

Each type of selector has a different weight:

- Inline styles (style attribute) → Highest specificity (1000)
- ID selectors (#id) → High specificity (100)
- Class, attribute, and pseudo-class selectors (.class, [attr], :hover) → Medium specificity (10)
- Element and pseudo-element selectors (div, h1, ::before) → Lowest specificity (1)

Conflict Resolution Example:

css

CopyEdit

```
p {
color: blue; /* Specificity: 1 */
}

.highlight {
color: red; /* Specificity: 10 */
}

#main-text {
color: green; /* Specificity: 100 */
}
```

If an element has all three styles applied (<p id="main-text" class="highlight">), the final color will be green because the ID selector has the highest specificity.

If specificity is the same, the last rule in the CSS file takes precedence.

Question 3: Difference Between Internal, External, and Inline CSS

Type	Description	Advantages	Disadvantages
Inline CSS	CSS is applied directly inside an	- Quick and easy for small	- Not reusable. - Difficult to

Type	Description	Advantages	Disadvantages
	HTML element using the style attribute.	changes. - Highest specificity.	maintain. - Increases HTML file size.
Internal CSS	CSS is written inside a <style> tag within the <head> section of the HTML file.	- Easier to manage than inline CSS. - No need for an external file.	- Still not reusable across multiple pages. - Can make the HTML file bulky.
External CSS	CSS is written in a separate .css file and linked to the HTML file using <link>.	- Reusable across multiple pages. - Easier to maintain and update. - Keeps HTML cleaner.	- Requires an additional HTTP request to load the CSS file. - Styles might not load immediately.

In most cases, external CSS is the preferred approach for scalability and maintainability.

<u>CSS Box Model</u>

Question 1: CSS Box Model and Its Components

The CSS Box Model describes how elements are structured and sized in a webpage. Every HTML element is treated as a rectangular box consisting of the following layers:

1. Content – The actual content inside the element (text, images, etc.).
2. Padding – The space between the content and the border.
3. Border – The boundary that wraps around the padding and content.
4. Margin – The space outside the border that separates the element from others.

Visual Representation:

```
+---------------------------+  ← Margin
          |                    |
|  +---------------------+  |  ← Border
|  |  +--------------+   |  |  ← Padding
      |  |  |  Content    |   |  |
      |  |  +--------------+   |  |
      |  +---------------------+  |
          |                    |
      +---------------------------+
```

How Each Affects the Size of an Element:

The total width and height of an element are calculated as:

Total Width = Content Width + Padding (Left & Right) + Border (Left & Right) + Margin (Left & Right)
Total Height = Content Height + Padding (Top & Bottom) + Border (Top & Bottom) + Margin (Top & Bottom)

For example, if an element has:

css

width: 200px;

padding: 10px;

border: 5px solid black;

margin: 20px;

Its total width will be:
200 + (10 * 2) + (5 * 2) + (20 * 2) = 270px

Question 2: Difference Between border-box and content-box Box Sizing

The box-sizing property controls how the total size of an element is calculated.

1. content-box (Default)

- Only the content width/height is defined.
- Padding and border are added to the total size.
- Example:

css

CopyEdit

```
div {
width: 200px;
padding: 10px;
border: 5px solid black;
box-sizing: content-box;
}
```

Total width = 200 + 10*2 + 5*2 = 230px

2. border-box

- The defined width/height includes padding and border.
- The content shrinks to fit inside the total size.
- Example:

css

```
div {
width: 200px;
padding: 10px;
border: 5px solid black;
box-sizing: border-box;
}
```

Total width = 200px (padding and border included)

Key Difference:

- content-box expands the total size when adding padding/border.

- border-box keeps the total size fixed and adjusts content accordingly.

CSS Flexbox

Question 1: What is CSS Flexbox, and How is it Useful for Layout Design?

CSS Flexbox (Flexible Box Layout) is a layout model designed to make it easier to align and distribute space among items in a container, even when their sizes are unknown or dynamic. It is particularly useful for creating responsive layouts, centering elements, and managing spacing efficiently.

Key Components of Flexbox:

1. Flex Container
    - The parent element that holds the flex items.
    - Defined using display: flex; or display: inline-flex;.
    - Controls how child elements are positioned.

Example:

css

```
.container {
display: flex;
background-color: lightgray;
}
```

2. Flex Items

- The child elements inside the flex container.
- These items respond to flexbox properties like flex-grow, flex-shrink, and flex-basis.

Example:

css

```
.item {
flex: 1; /* Makes items flexible */
padding: 10px;
background-color: lightblue;
border: 1px solid blue;
}
```

Why Flexbox is Useful:
✓ Makes responsive design easier.
✓ Aligns items both horizontally and vertically effortlessly.
✓ Eliminates the need for floats and positioning hacks.
✓ Automatically adjusts item sizes based on available space.

Question 2: Describe justify-content, align-items, and flex-direction in Flexbox

1. justify-content (Horizontal Alignment)

Controls how flex items are aligned along the main axis (left to right for row, top to bottom for column).

Common Values:

- flex-start → Items align at the start (default).
- flex-end → Items align at the end.
- center → Items are centered.
- space-between → Items are spaced with no gaps at the ends.
- space-around → Equal space around each item.
- space-evenly → Equal space between and around items.

Example:

css

```
.container {
display: flex;
justify-content: center; /* Centers items horizontally */
}
```

2. align-items (Vertical Alignment)

Controls how flex items align along the cross axis (top to bottom for row, left to right for column).

Common Values:

- stretch → Items stretch to fill the container height (default).
- flex-start → Items align at the top.
- flex-end → Items align at the bottom.
- center → Items are centered.

- baseline → Aligns items based on text baselines.

Example:

css

```
.container {
  display: flex;
  align-items: center; /* Centers items vertically */
}
```

3. flex-direction (Main Axis Direction)

Defines whether items are arranged horizontally (row) or vertically (column).

Common Values:

- row → Items placed left to right (default).
- row-reverse → Items placed right to left.
- column → Items placed top to bottom.
- column-reverse → Items placed bottom to top.

Example:

css

```
.container {
  display: flex;
  flex-direction: column; /* Stacks items vertically */
```

}

Key Takeaways:

- justify-content → Controls horizontal alignment.
- align-items → Controls vertical alignment.
- flex-direction → Sets layout direction (row/column).

CSS Grid

Question 1: What is CSS Grid, and How Does It Differ from Flexbox?

CSS Grid is a two-dimensional layout system that allows for precise placement of elements along both rows and columns. Unlike Flexbox, which is a one-dimensional layout system (either row or column), Grid provides greater control over complex layouts.

Differences Between Grid and Flexbox:

Feature	CSS Grid	Flexbox
Layout Type	Two-dimensional (rows & columns)	One-dimensional (row or column)
Main Use	Complex page layouts (grids, dashboards)	Aligning and distributing items (navigation bars, buttons, etc.)

Feature	CSS Grid	Flexbox
Alignment Control	Precise control over rows & columns	Focuses on alignment along one axis
Example Use Cases	Web page layouts, image galleries, dashboards	Navbars, buttons, cards, simple components

When to Use Grid Over Flexbox?

- Use Grid when designing entire page layouts with rows and columns.
- Use Flexbox for smaller, dynamic components like navbars, buttons, or centering items.
- In many cases, Grid and Flexbox can be used together for better flexibility.

Question 2: grid-template-columns, grid-template-rows, and grid-gap

1. grid-template-columns

Defines the number and size of columns in a grid.

Example:

```
css
.container {
display: grid;
grid-template-columns: 200px 200px 200px; /* Three
columns of 200px each */

}
```

or

css

```
.container {

display: grid;

grid-template-columns: 1fr 2fr 1fr; /* First & third columns
take 1 fraction, middle column takes 2 */

}
```

2. grid-template-rows

Defines the number and size of rows in a grid.

Example:

css

CopyEdit

```
.container {

display: grid;

grid-template-rows: 100px 150px; /* Two rows: first is 100px,
second is 150px */

}
```

or

css

CopyEdit

```
.container {
display: grid;
grid-template-rows: auto auto; /* Rows adjust based on content */
}
```

3. grid-gap (or gap)

Controls the spacing between grid items (both rows & columns).

Example:

css

```
.container {
display: grid;
grid-template-columns: repeat(3, 1fr);
grid-template-rows: repeat(2, 150px);
grid-gap: 20px; /* Adds 20px spacing between items */
}
```

Shorthand Variations:

css

CopyEdit

```
gap: 10px;      /* Equal row & column gap */
```

row-gap: 10px;   /* Gap only between rows */

column-gap: 15px; /* Gap only between columns */

Key Takeaways:

- grid-template-columns defines the number & size of columns.
- grid-template-rows defines the number & size of rows.
- grid-gap (or gap) controls spacing between grid items.

<u>Responsive Web Design with Media Queries</u>

Question 1: What Are Media Queries in CSS, and Why Are They Important for Responsive Design?

Media queries are a feature in CSS that allow styles to be applied conditionally based on a device’s screen size, resolution, or other characteristics. They enable responsive design, ensuring websites look good on all devices (desktops, tablets, mobiles).

Why Media Queries Are Important:

Create flexible layouts that adapt to different screen sizes.

Enhance user experience by optimizing design for different devices.

Reduce the need for separate mobile & desktop websites.
Improve accessibility by adjusting font sizes, spacing, and layouts dynamically.

Question 2: Basic Media Query for Screens Smaller Than 600px

The following media query reduces the font size when the screen width is 600px or smaller:

css

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

Explanation:

- @media (max-width: 600px): Applies styles only when the screen width is 600px or less.
- body { font-size: 14px; }: Changes the default font size for better readability on small screens.

Tip: You can use multiple media queries for different screen sizes to make your website fully responsive!

Typography and Web Fonts

Question 1: Difference Between Web-Safe Fonts and Custom Web Fonts

1. Web-Safe Fonts

Web-safe fonts are **pre-installed** on most operating systems (Windows, macOS, Linux), ensuring that they display consistently across different devices and browsers.

Examples of Web-Safe Fonts:

- Arial
- Times New Roman
- Verdana
- Georgia
- Courier New

Advantages:

- Loads **faster** since no external files are required.
- Ensures **consistent appearance** across all devices.

Disadvantages:

- Limited choices, making designs less unique.

2. Custom Web Fonts

Custom fonts (e.g., Google Fonts, Adobe Fonts) are **not pre-installed** on devices and must be downloaded from an external source before rendering.

Examples of Custom Web Fonts:

- Roboto (Google Fonts)
- Open Sans (Google Fonts)
- Lora (Adobe Fonts)

Advantages:

- **More design flexibility** with unique typography.
- **Brand consistency** across platforms.

Disadvantages:

- **Slightly slower loading times** due to external requests.
- **Fallback fonts needed** in case the custom font fails to load.

When to Use Web-Safe Fonts Over Custom Fonts?

- When performance and **fast loading times** are critical.
- When designing for **email templates** (since custom fonts may not render in all email clients).
- When **font consistency** across all devices is a priority.

Question 2: What is the font-family Property in CSS?

The **font-family** property specifies which font should be used for text in an element. It allows specifying multiple fonts as fallbacks in case the preferred font is unavailable.

Example:

css

CopyEdit

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

- If **Arial** is available, it will be used.
- If not, **Helvetica** will be used.
- If neither is available, a generic **sans-serif** font will be displayed.

How to Apply a Custom Google Font to a Webpage

Step 1: Import the Font in the <head>

Add the following <link> tag inside the HTML <head> section:

html

CopyEdit

```
<link
href="https://fonts.googleapis.com/css2?family=Roboto:wght@400;700&display=swap" rel="stylesheet">
```

Step 2: Use the Font in CSS

css

CopyEdit

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

Alternative: Use @import in CSS

css

CopyEdit

```
@import
url('https://fonts.googleapis.com/css2?family=Roboto:wght@
400;700&display=swap');


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

Key Takeaways:

- **Web-safe fonts** ensure consistency but are limited in style.
- **Custom web fonts** provide better typography but may impact performance.
- The font-family property allows specifying multiple fonts with fallbacks.
- **Google Fonts** can be easily integrated using <link> or @import