ASSIGNMENT

CSS & CSS 3

KASVALA BRIJESH
TOPS
Nikol - Ahmedabad

Module 3 – Frontend – CSS and CSS3

Theory

CSS Selectors & Styling

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

- A CSS selector is a pattern used to select and apply styles to specific elements in an HTML document.
- CSS selectors define which HTML elements the rules should apply to.

Examples of Selectors:

1. **Element Selector:**

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

2. **Class Selector:**

```
.highlight {
   background-color: yellow;
}
```

3. **ID Selector:**

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

2) Explain the concept of CSS specificity. How do conflicts between multiple stylesget resolved?

- CSS specificity determines which style rule is applied when multiple rules target the same element.
- Specificity is calculated based on the type of selectors used, as follows:

a) Inline styles (e.g., style="color: red;") have the highest specificity.
b) ID selectors (#id) are more specific than class or element selectors.

c) Class selectors, attributes, and pseudo-classes (e.g., .class, [attr], :hover) are less specific than ID selectors but more specific than element selectors.
d) Element selectors and pseudo-elements (e.g., p, ::before) have the lowest specificity.

Resolving Conflicts:

- Higher specificity wins.
- If specificity is equal, the later rule in the CSS overrides the earlier one.
- If both fail to resolve the conflict, the browser's default styles apply.

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

➢

1. Internal CSS

- Written within a `<style>` tag inside the `<head>` of an HTML document.

  EX:-

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

- Advantages:
  - Useful for single-page styling.
  - Keeps styles centralized in one place within the HTML document.
- Disadvantages:
  - Cannot reuse styles across multiple pages.
  - Can make the HTML file harder to read and maintain.

2. External CSS

- Defined in a separate .css file and linked to the HTML using a <link> tag.

  EX:-

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

- Advantages:
  - Encourages reusability and separation of concerns.
  - Allows centralized management of styles across multiple pages.
  - Reduces HTML file size.

➢ Disadvantages:
  - Requires an additional HTTP request to load the CSS file, which might affect performance if not optimized.

3. Inline CSS

- **Applied directly within an HTML element using the style attribute.**

EX:-

```
<p style="color: blue;">Hello, world!</p>
```

- **Advantages:**
  - **Useful for quick fixes or unique styles for specific elements.**
  - **Does not require a separate file or <style> block.**
- **Disadvantages:**
  - **Violates the principle of separation of concerns.**
  - **Makes the code harder to maintain and reuse.**
  - **Leads to cluttered HTML.**

CSS Box Model

4) 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 that describes how elements are structured and sized on a web page. It consists of four main components:
  I. **Content**:
    - This is the innermost part of the box, where the text, images, or other content of the element is displayed.
    - The width and height of the content are defined by the **width** and **height** properties in CSS.
  II. **Padding**:
    - The padding is the space between the content and the border.
    - It increases the space around the content, effectively expanding the size of the box.
    - Padding is **inside the border** and can be set individually for each side (top, right, bottom, left).
  III. **Border**:
    - The border wraps around the padding and content, forming a visible edge or boundary.
    - The size of the border contributes to the overall dimensions of the element.
  IV. **Margin**:
    - The margin is the outermost layer of the box, creating space between the element and adjacent elements.
    - It does not affect the size of the box itself but affects the spacing between boxes.
- **How these affect the size of an element**:
  - 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)

5) What is the difference between border-box and content-box box-sizing inCSS? Which is the default?

-

A. **Content-box** (Default):
  - The width and height properties **only include the content**.

- Padding and borders are **added outside** the specified width and height, which means the total size of the element increases when padding or border is added.

EX:-

box-sizing: content-box;

width: 100px;

padding: 10px;

border: 5px solid;

B. **Border-box**:

- The width and height properties **include the content, padding, and border**.
- The total size of the element remains the same, regardless of the padding or border values, as these are adjusted within the specified width and height.

EX:-

box-sizing: border-box;

width: 100px;

padding: 10px;

border: 5px solid;

➢ **Default Box-Sizing**:

- The default box-sizing is **content-box**, but many developers prefer border-box because it simplifies layout calculations. It's common to reset the box-sizing for all elements using:

EX:-

```
* {
  box-sizing: border-box;
}
```

CSS Flexbox

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

- **CSS Flexbox (Flexible Box Layout)** is a layout model in CSS designed to make it easier to arrange items within a container, especially when their size and position need to adapt to different screen sizes or dynamic content.
- **Key Terms:**
  a) **Flex-container**:
  - The parent element where the `display: flex` or `display: inline-flex` property is applied.
  - It defines a flexible context for its children, allowing them to align and distribute according to the Flexbox rules.

  EX:-

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

  b) **Flex-item**:
  - direct child elements of a flex-container.
  - These items follow the rules and properties defined by the flex-container, such as alignment, order, and spacing.

  EX:-

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

7) Describe the properties justify-content, align-items, and flexdirection used in Flexbox.

- a. **`justify-content`**:
  - Defines how flex-items are aligned **along the main axis** (horizontal by default).
  - Useful for controlling the horizontal spacing between items.
  - Values:
    - `flex-start` (default): Items align at the start of the main axis.
    - `flex-end`: Items align at the end of the main axis.

- `center`: Items are centered along the main axis.
- `space-between`: Evenly distributes items with space only between them.
- `space-around`: Distributes items with equal space around each item.
- `space-evenly`: Equal space between and around items.

EX:-.container {

justify-content: center;

}

b. **align-items**:
- Defines how flex-items are aligned **along the cross axis** (vertical by default).
- Useful for controlling vertical alignment within the container.

o Values:
- stretch (default): Items stretch to fill the container's height.
- flex-start: Items align at the start of the cross axis.
- flex-end: Items align at the end of the cross axis.
- center: Items are centered along the cross axis.
- baseline: Items align along their text baselines.

EX:-.container {

align-items: center;

}

c. **flex-direction**:
- Defines the direction of the main axis, determining how flex-items are placed in the container.

o Values:
- row (default): Items are placed in a horizontal row, left to right.
- row-reverse: Items are placed in a horizontal row, right to left.
- column: Items are placed in a vertical column, top to bottom.
- column-reverse: Items are placed in a vertical column, bottom to top.

EX:-.container {

flex-direction: column;

}

CSS Grid

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

- **What is CSS Grid?**

  CSS Grid is a two-dimensional layout system in CSS that allows you to create complex and precise layouts.

- **Key Differences Between CSS Grid and Flexbox:**

Feature	CSS Grid	Flexbox
Layout type	Two-dimensional (manages rows and columns).	One-dimensional (manages rows or columns).
Use case	Best for grid-based layouts with defined rows/columns.	Best for aligning items along a single axis (horizontal or vertical).
Parent-child relationship	Defines layout for both container and items explicitly.	Focuses on the arrangement of items in the container.
Alignment	Offers precise control over both horizontal and vertical alignment for items in the grid.	Focuses on aligning items along the main and cross axis.
Auto placement	Automatically places items based on grid lines.	Items flow sequentially based on content size and available space.

- **When to Use CSS Grid Over Flexbox?**

- **Use CSS Grid when:**
  - **You need a structured, grid-like layout.**
  - **You want precise control over rows, columns, and item placement.**
  - **Your design involves both horizontal and vertical alignments.**

- **Use Flexbox when:**
  - **You need a simple layout (e.g., navigation bar, buttons).**
  - **You're working with a single axis (row or column).**
  - **Items need to dynamically align and distribute space.**

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

-

i. `grid-template-columns:-`

- Defines the number and size of columns in the grid.
- You can specify values like fixed widths, percentages, `auto`, or the `fr` unit (fraction of the available space).

EX:-

```
.container {
  display: grid;
  grid-template-columns: 100px 200px 1fr; /* Three columns: fixed, fixed, and flexible */
}
```

ii. grid-template-rows:-

- **Defines the number and size of rows in the grid.**
- **Similar to columns, you can use fixed sizes, percentages, auto, or fr units.**

EX:-

```
.container {
  display: grid;
  grid-template-rows: 50px auto 2fr; /* Three rows: fixed, flexible, and larger flexible */
}
```

iii. grid-gap (or gap):-

- **Defines the spacing between rows and columns in the grid.**
- **You can set the gap for both rows and columns with gap or individually with row-gap and column-gap.**

EX:-

```
.container {
  display: grid;
  grid-template-columns: 1fr 1fr 1fr;
  grid-template-rows: auto auto;
  gap: 20px; /* Adds 20px spacing between rows and columns */
}
```

➢ Full Example Combining All Three Properties:-

```
<div class="container">
  <div>Item 1</div>
  <div>Item 2</div>
  <div>Item 3</div>
```

```
  <div>Item 4</div>
</div>

<style>
  .container {
    display: grid;
    grid-template-columns: 150px 1fr 2fr; /* Three columns */
    grid-template-rows: 100px auto;       /* Two rows */
    gap: 10px;                            /* Space between items */
  }
</style>
```

Responsive Web Design with Media Queries

10) What are media queries in CSS, and why are they important for responsive design?

- **What are Media Queries?**
  - Media queries are a feature of CSS that allow you to apply different styles depending on the characteristics of the user's device, such as screen size, resolution, orientation, or even device type. They enable your web page to adapt and respond to varying screen sizes and environments.
- **Why are Media Queries Important for Responsive Design?**
  - **Device Adaptability**: They allow websites to adjust their layout and design to look good on devices of all sizes (e.g., phones, tablets, desktops).
  - **Improved User Experience**: Ensures content is easy to read and interact with on any screen.
  - **Performance Optimization**: You can load lighter or simpler styles for smaller devices, improving performance.
  - **Future-Proofing**: Helps handle a wide range of current and future devices with varying resolutions and aspect ratios.

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

- **Here's an example of a media query that adjusts the font size for screens smaller than 600px:**

EX:-

```
/* Default styles for larger screens */
body {
  font-size: 16px;
}

/* Media query for smaller screens */
@media (max-width: 600px) {
  body {
    font-size: 14px; /* Reduces the font size for smaller screens */
  }
}
```

Typography and Web Fonts

12) Explain the difference between web-safe fonts and custom web fonts. Why might you use a web-safe font over a custom font?

- **Web-Safe Fonts:-**
  - **Definition**: Web-safe fonts are fonts that are pre-installed on most operating systems (Windows, macOS, Linux, etc.), ensuring they display consistently across different devices without requiring additional downloads.
  - **Examples**: Arial, Times New Roman, Verdana, Georgia, Courier New, etc.
  - **Advantages**:
    - No additional load time since they are already available on the user's device.
    - Reliable and consistent rendering across different browsers and platforms.
- **Custom Web Fonts:-**
  - **Definition**: Custom web fonts are fonts not installed on the user's device but are downloaded from the web. These include fonts from services like Google Fonts, Adobe Fonts, or self-hosted fonts.
  - **Advantages**:
    - Allows for unique and branded typography.
    - Provides a wider variety of styles and design options.
- **Why Use a Web-Safe Font Over a Custom Font?**
  - **Performance**: Web-safe fonts don't require downloading, resulting in faster page load times.
  - **Fallback**: They ensure compatibility for users with limited internet access or devices that don't support custom fonts.
  - **Simplicity**: For simple projects or minimalistic designs, web-safe fonts are often sufficient

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

- **`font-family` Property in CSS:-**
  - The `font-family` property specifies the font(s) for text in an element.
  - You can specify multiple fonts as a fallback list. If the first font is unavailable, the browser tries the next one.

Syntax:-

font-family: "Primary Font", "Fallback Font", generic-family;

EX:-

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

- **How to Apply a Custom Google Font to a Webpage**

1. Choose a Font from Google Fonts:

- **Visit Google Fonts and select a font you want to use.**

2. Embed the Font in Your HTML:

- Google Fonts provides an <link> tag to include in your <head> section. For example:

EX:-

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

3. Apply the Font Using font-family:

- Use the font in your CSS by specifying its name.

EX:-

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