CSS Assignment

CSS Selectors & Styling

Que 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.

Types Of Selectors:

1. **Element Selector** - Selects elements based on the HTML tag name.
   Ex:
   ```
   p {
   color: blue;
   }
   ```
2. **Class Selector** - Selects elements with a specific class.
   Ex:
   ```
   .highlight {
           background-color: yellow;
                 }
   ```
3. **ID Selector** - Selects a specific element using an ID.
   Ex:
   ```
   #header {
     font-size: 20px;
   }
   ```

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

Ans: **CSS specificity** determines which style is applied when multiple styles target the same element.

Specificity Calculation:

- **Inline styles** (style="color: red;") → **Highest specificity** (1,0,0,0)
- **ID selector** (#id) → Specificity (0,1,0,0)
- **Class selector, attribute selector, pseudo-class** (.class, [type="text"], :hover) → Specificity (0,0,1,0)
- **Element selector, pseudo-element** (h1, ::before) → Lowest specificity (0,0,0,1)

Conflict Resolution:

1. Higher specificity wins (e.g., an ID selector overrides a class selector).
2. If specificity is the same, the later rule in the CSS file applies.

3. Inline styles override external or internal CSS.
4. Using !important overrides all styles but should be avoided when possible.

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

Ans:

1. Inline Css: Inline CSS is applied directly within an HTML element using the `style` attribute.
   Example: ```html <p style="color: red; font-size: 16px;">This is a paragraph.</p>

   Advantages:
   - Quick to apply for small styling changes.
   - No need for an external file.

   Disadvantages:
   - Difficult to maintain for large projects.
   - Increases HTML file size and reduces reusability.
2. Internal CSS:
   Definition: Internal CSS is written within a <style> tag inside the <head> section of an HTML document.
   Example:
   ```
   <head>
     <style>
       p {
         color: blue;
         font-size: 18px;
       }
     </style>
   </head>
   ```

   Advantages:
   - Useful for single-page designs.
   - Keeps styles separate from content.

   Disadvantages:
   - Not reusable across multiple pages.
   - Increases the size of the HTML file.
3. External CSS:
   Definition: External CSS is stored in a separate .css file and linked to an HTML document using the <link> tag.
   Example: <link rel="stylesheet" href="styles.css">

   Advantages:
   - Best for large projects and multiple-page websites.
   - Keeps HTML clean and easy to maintain.

Disadvantages:

- Requires an additional HTTP request, which might slightly slow down the page load.
- Styles won't apply if the CSS file is missing or not loaded properly.

Conclusion:

- **Inline CSS** is useful for quick fixes but not recommended for large projects.
- **Internal CSS** is best for styling a single page but lacks reusability.
- **External CSS** is the most efficient approach for large websites, ensuring better maintainability and performance.

CSS Box Model

Question 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 and Its Components**

The **CSS Box Model** is a fundamental concept that describes how elements are structured and how their size is calculated in a webpage. It consists of four main components:

1. **Content**
   - This is the actual content inside the element, such as text, images, or other elements.
   - The width and height properties define the size of the content area.
2. **Padding**
   - Padding is the space between the content and the border.
   - It increases the overall size of the element without affecting its position relative to other elements.
   - Example: padding: 10px; adds 10 pixels of space inside the border around the content.
3. **Border**
   - The border surrounds the padding and content.
   - It can have different styles (solid, dashed, double, etc.), thickness, and colors.
   - Example: border: 2px solid black; adds a 2-pixel black border around the element.
4. **Margin**

- The margin is the space outside the border, creating distance between the element and its neighbors.
- It does not affect the element's size but influences the layout by controlling spacing.
- Example: margin: 20px; adds 20 pixels of space outside the border.

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

Ans : **Difference Between border-box and content-box in box-sizing**

The **box-sizing** property in CSS controls how the total width and height of an element are calculated. There are two main values:

1. **content-box (Default)**
    - The width and height properties **only** include the content.
    - **Padding and border are added separately**, increasing the total size of the element.
    - This is the **default behavior** of all elements in CSS.

    EXAMPLE:

    ```
    .content-box-example {
            width: 200px;
            height: 100px;
            padding: 10px;
            border: 5px solid black;
            box-sizing: content-box; /* Default */
        }
    ```

2. **border-box**

- The width and height **include** the content, padding, and border.
- This makes it easier to control layouts since the total size remains fixed.

  Example:

  ```
  .border-box-example {
    width: 200px;
    height: 100px;
    padding: 10px;
    border: 5px solid black;
    box-sizing: border-box;
  }
  ```

Which One to Use?

- **Use content-box** when you want precise control over content size and don't mind the extra calculations for spacing.
- **Use border-box** when designing layouts because it keeps the total size fixed, preventing unexpected size increases.

CSS Flexbox

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

Ans: **What is CSS Flexbox?**

Flexbox (Flexible Box Layout) is a way to arrange items inside a container so they adjust automatically to different screen sizes. It helps in making websites responsive and layouts easy to manage.

Why Use Flexbox?

- Helps in aligning items easily (horizontally & vertically).
- Makes items resize automatically to fit different screens.
- Allows even spacing between items without extra effort.
- Helps rearrange items without changing the HTML structure.

Key Terms: flex-container and flex-item

1. Flex Container (Parent Element)

A flex container is the main box that holds all the items. You make an element a flex container by using:

```
display: flex;
```

Important Flex Container Properties

- flex-direction: Sets the layout direction (row, column).
- justify-content: Controls horizontal alignment (left, center, right).
- align-items: Controls vertical alignment (top, center, bottom).
- flex-wrap: Decides if items wrap when they don't fit in one line.

Example:

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

```
    align-items: center;
}
```

2. Flex Item (Child Elements)

A **flex item** is any item inside the flex container. You can control its size and position.

Important Flex Item Properties

- flex-grow: Controls how much an item should expand.
- flex-shrink: Controls how much an item should shrink.
- flex-basis: Sets the default size of the item.
- order: Changes the order of items.

Example:

```
.item {
    flex-grow: 1;
    flex-basis: 100px;
}
```

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

Ans : **Flexbox Properties: justify-content, align-items, and flex-direction**

Flexbox helps arrange items inside a container. These three properties control how items are placed inside the container

1. justify-content (Align Items Side to Side)

This property controls how items are arranged horizontally (left to right).

Common Options:

- flex-start : Items stick to the left (default).
- flex-end : Items move to the right.
- center : Items move to the middle.
- space-between : First item at the left, last at the right, and space between others.
- space-around : Items have equal space around them.
- space-evenly : Even space between all items.

Example:

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

```
  justify-content: center;
}
```

2. align-items (Align Items Up and Down)

This property controls how items are arranged vertically (top to bottom).

Common Options:

- flex-start → Items stick to the top.
- flex-end → Items move to the bottom.
- center → Items move to the **middle**.
- stretch → Items stretch to fill the container.
- baseline → Items align based on text inside.

Example:

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

3. flex-direction (Row or Column Layout)

This property controls whether items are placed in a row or column.

Common Options:

- row → Items are placed side by side (default).
- row-reverse → Items are side by side but in reverse order.
- column → Items are placed one below the other.
- column-reverse → Items are stacked but in reverse order.

Example:

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

justify-content → Moves items left, right, or center.
align-items → Moves items up, down, or center.
flex-direction → Puts items in a row or column.

CSS Grid

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

Ans : **CSS Grid vs. Flexbox: Understanding the Differences and Use Cases**

What is CSS Grid?

CSS Grid is a two-dimensional layout system designed for creating complex web page layouts. It allows you to define both rows and columns, making it ideal for designing entire web pages or large sections of a page.

Key Features of CSS Grid:

- Works in both rows and columns (2D layout).
- Uses grid containers (display: grid;) and grid items.
- Provides control over spacing using grid gaps (gap, row-gap, column-gap).
- Can define explicit and implicit grids.
- Supports grid template areas for layout design.

What is Flexbox?

Flexbox is a one-dimensional layout system that is used for aligning and distributing space among items within a container. It works either in a row or a column, but not both simultaneously.

Key Features of Flexbox:

- Works in one direction at a time (row or column).
- Uses flex containers (display: flex;) and flex items.
- Distributes space efficiently with justify-content and align-items.
- Allows flexible resizing of elements with flex-grow, flex-shrink, and flex-basis.
- Great for creating responsive layouts.

When to Use CSS Grid vs. Flexbox?

Use CSS Grid when:

- You need a two-dimensional layout (both rows & columns).
- You are designing a complex web page structure.
- You need to control the placement of elements precisely.
- You want to use grid-template-areas for semantic layout design.

Use Flexbox when:

- You need a one-dimensional layout (row OR column).
- You are designing small UI components like buttons, navigation bars, or form elements.

- You want to distribute space between elements dynamically.
- You need to align elements quickly and efficiently.

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

Ans: **CSS Grid Properties: grid-template-columns, grid-template-rows, and grid-gap**

1. grid-template-columns

The grid-template-columns property defines the number of columns in a grid layout and their respective sizes.

Example Usage:

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

2. grid-template-rows

The grid-template-rows property defines the number of rows and their heights.

Example Usage:

```
.container {
  display: grid;
  grid-template-rows: 100px 1fr 2fr;
}
```

3. gap (Previously grid-gap)

The gap property sets spacing between rows and columns in a grid. It is shorthand for row-gap and column-gap.

Example Usage:

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

Conclusion

- grid-template-columns: Defines column sizes.

- grid-template-rows: Defines row sizes.
- gap: Controls spacing between grid items.

Responsive Web Design with Media Queries

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

Ans : **Media Queries in CSS**

Media queries are a feature in CSS that allow developers to apply different styles based on the characteristics of the user's device, such as screen size, resolution, orientation, and more. They are a core part of responsive web design.

Syntax of Media Queries

A basic media query looks like this:

```
@media (max-width: 768px) {
  body {
    background-color: lightgray;
  }
}
```

Why Are Media Queries Important for Responsive Design?

1. **Ensure a Better User Experience** – Media queries adjust styles for different devices, making websites look good on desktops, tablets, and mobile phones.
2. **Improve Accessibility** – Users with different screen sizes or devices (like those using assistive technologies) get an optimized experience.
3. **Optimize Performance** – Loading styles specific to a device reduces unnecessary resource usage.
4. **Support Different Orientations** – Styles can adapt when a device switches between portrait and landscape mode.
5. **Future-Proofing** – Media queries help websites remain compatible with new devices and screen sizes.

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

Ans :

```
body {
  font-size: 18px;
}
@media (max-width: 600px) {
  body {
    font-size: 14px;
  }
}
```

Typography and Web Fonts

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?

Ans : **Difference Between Web-Safe Fonts and Custom Web Fonts**

1. Web-Safe Fonts:

Web-safe fonts are fonts that are pre-installed on most operating systems (Windows, macOS, Linux) and are widely supported across all browsers. They ensure that text appears consistently regardless of the user's device.

Examples of Web-Safe Fonts:

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

Advantages of Web-Safe Fonts:
Faster loading times (no extra download needed).
Consistent appearance across devices and browsers.
No dependency on external font files.

Disadvantages:
Limited design choices.
Lack of uniqueness compared to custom fonts.

2. Custom Web Fonts:

Custom web fonts are fonts that are not pre-installed on a user's system but are loaded via CSS from external sources like Google Fonts, Adobe Fonts, or self-hosted font files.

Examples of Custom Web Fonts:

- Roboto (Google Fonts)
- Open Sans (Google Fonts)
- Lato (Google Fonts)
- Montserrat (Adobe Fonts)

Advantages of Custom Web Fonts:
Greater design flexibility and uniqueness.
Improved branding and aesthetics.
Wide variety of styles and weights available.

Disadvantages:
Slower page load speed (fonts must be downloaded).
Possible fallback issues if the font fails to load.
External dependencies if using a hosted service.

Why Use a Web-Safe Font Over a Custom Font?

You might choose a **web-safe font** over a **custom font** when:

- Performance is a priority (faster page loads).
- You want maximum compatibility across all devices and browsers.
- You don't want to rely on external services (Google Fonts, Adobe Fonts).
- You're designing for older browsers that may not support modern font embedding.

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

Ans: **What is the font-family Property in CSS?**

The font-family property in CSS is used to specify the font of text in a webpage. It allows you to define a primary font along with fallback fonts in case the primary font is not available.

Syntax:

```
selector {
  font-family: "Primary Font", "Fallback Font", generic-family;
}
```

Example with Web-Safe Fonts:

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

How to Apply a Custom Google Font to a Webpage?

You can add a **Google Font** to your webpage using two methods:

Method 1: Using <link> in HTML

1 **Go to** **Google Fonts** and choose a font.
2 Copy the <link> tag provided by Google Fonts.
3 Paste it inside the <head> of your HTML file.
4 Use font-family in CSS to apply the font.

Example :

```
<!DOCTYPE html>

<html lang="en">

<head>

  <meta charset="UTF-8">

  <meta name="viewport" content="width=device-width, initial-scale=1.0">

  <title>Google Font Example</title>

  <!-- Link to Google Font -->

  <link href="https://fonts.googleapis.com/css2?family=Poppins:wght@300;400;700&display=swap"
rel="stylesheet">

  <style>

    body {

      font-family: "Poppins", sans-serif;

    }

  </style>

</head>

<body>

  <h1>Custom Google Font Example</h1>

  <p>This text is using the Poppins font from Google Fonts.</p>

</body>

</html>
```