**CSS THEORY**

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

**Ans.CSS selector** is a rule or pattern used in CSS (Cascading Style Sheets) to target and apply styles to specific HTML elements on a webpage.

* It **selects** the elements you want to style.
* Then you define the **styles** (like color, size, layout) that should be applied to those selected elements.

**1. Element Selector**

* Selects HTML elements by their **tag name**.
* Applies the style to **every** occurrence of that element.
* Example:  
  If you select the p element, it styles **all** <p> paragraphs on the page.

**2. Class Selector**

* Selects elements with a specific **class attribute**.
* A class can be assigned to **multiple elements**, so this selector styles a group of elements at once.
* Class selectors start with a **dot .** before the class name.
* Example:  
  A class called .card will apply the same style to every element with class="card".

**3. ID Selector**

* Selects **one unique element** with a specific **id attribute**.
* IDs must be **unique** within a page (one ID = one element).
* ID selectors start with a **hash #** before the ID name.
* Example:  
  An ID called #header will apply styles to only the element with id="header".

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

**Ans.** If there are two or more CSS rules that point to the same element, the selector with the highest specificity will "win", and its style declaration will be applied to that HTML element. Think of specificity as a hierarchy that determines which style declaration is ultimately applied to an element.

The order in which stylesheets are linked in the HTML document influences the application of styles. Styles from a stylesheet loaded later can override those from an earlier one. By organizing the order of stylesheet inclusion, you control which styles have priority, helping to manage and resolve conflicts.

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

**Ans. 1. Inline CSS**

**Example**

<p style="color: red; font-size: 18px;">Hello!</p>

**Advantages**

* Quick and easy for small tweaks.
* Useful for testing or dynamic changes with JavaScript.

**Disadvantages**

* Hard to maintain (especially with many elements).
* Violates **separation of concerns** (HTML & CSS mixed).
* No reusability — you have to repeat styles.

**2. Internal CSS**

**Example**

<head>

<style>

p {

color: green;

font-size: 16px;

}

</style>

</head>

**Advantages**

* Keeps CSS in one place within the same file.
* Good for **small websites or single-page apps**.
* No need for external requests (faster load for small pages).

**Disadvantages**

* Styles not reusable across multiple HTML files.
* Makes the HTML file heavier and less organized.
* Less efficient for large projects.

**3. External CSS**

**Example**

<head>

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

</head>

**styles.css**

p {

color: blue;

font-size: 14px;

}

**Advantages**

* Best for **maintainability** and **reusability**.
* Clean separation between content (HTML) and style (CSS).
* Cached by browsers — improves performance for larger sites.

**Disadvantages**

* One more HTTP request (can affect load time if not optimized).
* Won’t work without internet if the CSS is hosted remotely.

**4. 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 fundamental concept in web design and layout. It describes how the size of every HTML element is calculated and how elements are spaced and layered on a page. Each element is represented as a rectangular box made up of four areas:

**CSS Box Model Components**

1. **Content**
   * This is the innermost part of the box.
   * It contains the actual text or images.
   * Its size is set using width and height properties.
   * **Does it affect element size?** Yes — it’s the base size of the element.
2. **Padding**
   * This space surrounds the content.
   * It creates space **inside the element**, between the content and the border.
   * Set using the padding property (can be set on each side individually: padding-top, padding-right, etc.).
   * **Affects size?** Yes — it adds to the total size of the element **unless** box-sizing: border-box is used.
3. **Border**
   * This wraps around the padding and content.
   * Controlled using properties like border-width, border-style, and border-color.
   * **Affects size?** Yes — adds to the overall width and height.
4. **Margin**
   * This is the outermost layer.
   * It creates space **between** the current element and neighboring elements.
   * Set using the margin property.
   * **Affects size?** No — it doesn't affect the box's internal size but affects spacing around it.

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

**Ans.** The box-sizing property in CSS controls **how the total width and height of an element are calculated** — whether the size you set includes the element's border and padding, or if it applies only to the content.

**content-box (Default value)**

* **Definition:** When you use box-sizing: content-box, the width and height properties apply **only to the content area** of the element.
* Padding and border are **not included** in the size.
* This means the actual rendered size of the element will be **larger** than the specified width/height if you add padding or borders.

**Used by default in CSS**, unless explicitly changed.

**border-box**

* **Definition:** When you use box-sizing: border-box, the width and height properties apply to the **entire element**, including content, padding, and border.
* The content area is automatically adjusted to make space for the padding and border **within** the defined size.
* This makes layout calculations simpler and helps avoid elements unintentionally overflowing their containers.

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

**Ans. CSS Flexbox** (short for *Flexible Box Layout*) is a layout model in CSS that makes it easier to design **responsive** and **space-efficient** layouts. It allows you to align and distribute space among items in a container, even when their sizes are unknown or dynamic.

It’s especially useful for:

* Centering items (horizontally and vertically)
* Creating flexible rows or columns
* Managing spacing and alignment across different screen sizes

**1. Flex Container**

* The **parent** element that holds flex items.
* Declared by setting: display: flex or display: inline-flex.
* Controls the layout behavior of its children using properties like:
  + flex-direction
  + justify-content
  + align-items
  + flex-wrap

**2. Flex Items**

* The **children** of a flex container.
* These elements are automatically laid out according to the rules of Flexbox.
* You can control their behavior with properties like:
  + flex-grow, flex-shrink, flex-basis
  + align-self
  + order

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

**Ans. 1. flex-direction**

**What it does:**  
Defines the **main axis** — the direction in which flex items are placed.

**Values:**

* row (default): items are placed **left to right** (horizontal)
* row-reverse: items go **right to left**
* column: items are placed **top to bottom** (vertical)
* column-reverse: items go **bottom to top**

**2. justify-content**

**What it does:**  
Aligns **flex items along the main axis** (defined by flex-direction), controlling **horizontal alignment in a row**, or **vertical alignment in a column**.

**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 spacing **between** items
* space-around: equal spacing **around** items
* space-evenly: equal spacing **between and around**

**3. align-items**

**What it does:**  
Aligns items **along the cross axis** (perpendicular to the main axis), controlling **vertical alignment in a row**, or **horizontal alignment in a column**.

**Values:**

* flex-start: items align to the **start** of the cross axis
* flex-end: items align to the **end**
* center: items are **centered**
* baseline: aligns based on text baselines
* stretch (default): items stretch to fill the container

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

**Ans. CSS Grid Layout** is a two-dimensional layout system in CSS. It allows you to create complex, responsive layouts using **rows and columns**.

With Grid, you define a grid structure on a container, and then place child elements into cells based on row and column positions.

* **Grid Container**: Declared with display: grid.
* **Grid Items**: The direct children of the grid container.
* **Rows and Columns**: Defined using grid-template-rows and grid-template-columns.
* **Placement**: Items can be placed precisely with grid-row, grid-column, or grid-area.

**When to Use Grid**

Use **CSS Grid** when:

* You need **full control over both rows and columns**.
* You're building **complex layouts** like web pages, dashboards, or image galleries.
* You want to **position items explicitly** in both axes.
* You're laying out large sections of the page, not just a row or column of elements.

**When to Use Flexbox**

Use **Flexbox** when:

* You're working with a **single axis layout** (row *or* column).
* You need to **align and distribute space** between items dynamically.
* You're building **navigation bars**, **button groups**, or **form elements**.

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

**Ans. 1. grid-template-columns**

* This property defines the **structure of columns** in a grid layout.
* It specifies **how many columns** there are and how **wide** each column should be.
* You can use units like:
  + px or em for fixed widths,
  + fr for fractional space (e.g., 1fr, 2fr),
  + auto for content-based sizing.
* You can also use the repeat() function to simplify repetitive patterns.

**2. grid-template-rows**

* This property defines the **structure of rows** in the grid.
* It works similarly to grid-template-columns, but controls **height** instead of width.
* You can define fixed heights, automatic sizing, or flexible proportions using the same units (px, fr, auto, etc.).

**3. gap (previously grid-gap)**

* This property defines the **space between rows and columns** inside the grid.
* You can specify one value (applies to both row and column gaps) or two values (row-gap column-gap).
* It does **not** add space outside the grid container, only **between grid items**.

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

**Ans. Media queries** are a CSS feature that allow you to apply styles **conditionally**, based on the characteristics of the user's device — such as screen width, height, resolution, orientation, and more.

They are a key tool in **responsive web design**, enabling websites to **adapt to different screen sizes and devices** (like desktops, tablets, and smartphones).

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

**Ans.** A **media query** in CSS allows you to apply specific styles based on conditions such as the device's screen width, height, or other features. In the context of adjusting the font size for screens smaller than **600px**, you would typically use the max-width condition.

1. **Responsive Design**: Ensures your website is **legible and accessible** across different screen sizes.
2. **User Experience**: Enhances readability on smaller devices by adjusting the font size, so users don't have to zoom in.
3. **Custom Styling**: Provides flexibility to tailor your site’s appearance based on the device, leading to a **better overall design**.

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

**Ans.1. Web-Safe Fonts**

**Definition:**  
Web-safe fonts are **pre-installed fonts** that are available across most devices and operating systems. They are commonly used in web design because they are widely supported and can be rendered without relying on external resources or downloads.

**Characteristics:**

* **Pre-installed:** These fonts are typically built into major operating systems (Windows, macOS, Linux) and browsers, meaning they are readily available without the need for additional downloads.
* **Limited Variety:** There are only a few web-safe fonts available, which limits the diversity in typography and design.
* **No External Requests:** Since the fonts are locally available, they do not require any additional network requests or dependencies to load.

**2. Custom Web Fonts**

**Definition:**  
Custom web fonts are fonts that are **not pre-installed** on devices but can be **downloaded from external sources** (e.g., Google Fonts, Adobe Fonts, etc.) and used on a website. They offer much more variety and flexibility in typography.

**Characteristics:**

* **External Resources:** Custom fonts are typically hosted by a third-party service (like Google Fonts or a self-hosted file) and need to be downloaded when the webpage loads.
* **Wide Variety:** Custom web fonts allow designers to choose from a wide range of fonts that are not part of the web-safe set, offering greater flexibility in design.
* **Potential Loading Delays:** Since the font needs to be downloaded, it can increase page load time, especially if the font files are large or if the user has a slow internet connection.

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

**Ans.** The **font-family** property in CSS allows you to specify which fonts should be used for text, with the ability to provide fallback fonts.

Custom **Google Fonts** are applied by embedding a <link> tag in the HTML <head>, followed by referencing the font in your CSS using the font-family property.