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

* A **CSS selector** is a pattern used in CSS (Cascading Style Sheets) to select and style specific HTML elements on a webpage. It tells the browser **which elements to apply styles to**.

**Types of CSS Selectors with Examples:**

1. **Element Selector**  
   Selects all HTML elements of a specific type.  
   **Example:**

css

*p {*

*color: blue;*

*}*

→ Selects all <p> (paragraph) elements and sets their text color to blue.

1. **Class Selector**  
   Selects all elements that have a specific class attribute.  
   **Syntax:** .className  
   **Example:**

css

*.highlight {*

*background-color: yellow;*

*}*

→ Selects all elements with class="highlight" and sets a yellow background.

1. **ID Selector**  
   Selects a single element that has a specific ID.  
   **Syntax:** #idName  
   **Example:**

css

*#header {*

*font-size: 24px;*

*}*

→ Selects the element with id="header" and sets the font size to 24 pixels.

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

* **CSS specificity** is a set of rules the browser uses to determine **which CSS rule applies** when multiple rules target the same element. When there are conflicts (i.e., more than one rule applies to an element), the rule with **higher specificity** takes precedence.

**How Specificity Works**

* Each CSS selector has a **specificity value**, which is calculated based on the types of selectors used. It's often expressed as a 4-part value: a,b,c,d, where:
* **a** = Inline styles (e.g., style="...") → highest specificity
* **b** = Number of ID selectors (#id)
* **c** = Number of class selectors (.class), attributes ([type="text"]), and pseudo-classes (:hover, :nth-child)
* **d** = Number of element names (div, h1) and pseudo-elements (::before, ::after)
* The higher this value, the more "specific" the rule.

**Example:**

Given this HTML:

html

*<p id="intro" class="highlight">Hello</p>*

And these CSS rules:

css

*p { color: black; } /\* specificity: 0,0,0,1 \*/*

*.highlight { color: green; } /\* specificity: 0,0,1,0 \*/*

*#intro { color: red; } /\* specificity: 0,1,0,0 \*/*

* #intro has the highest specificity, so the paragraph will be **red**.

**Conflict Resolution Rules:**

1. **Higher specificity wins**
2. If specificity is equal, the **later** rule in the CSS (or stylesheet) wins (last rule wins).
3. **Inline styles** (e.g., style="color:blue;") override all external or embedded styles (unless !important is used).
4. !important can override specificity, but should be used sparingly.

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

* The three main types of CSS are **inline**, **internal**, and **external**. They differ in **where the styles are placed** and how they're applied to HTML elements.

**✅ 1. Inline CSS**

**Definition:**  
Styles are written directly in the style attribute of an HTML element.

**Example:**

html

*<p style="color: red;">Hello</p>*

**Advantages:**

* Quick for testing or small tweaks.
* No need for separate CSS files.

**Disadvantages:**

* Poor maintainability — styles are mixed with content.
* Difficult to reuse or update consistently.
* High specificity — can override other styles unintentionally.

**✅ 2. Internal CSS**

**Definition:**  
Styles are placed inside a <style> tag within the <head> section of an HTML document.

**Example:**

html

*<head>*

*<style>*

*p {*

*color: blue;*

*}*

*</style>*

*</head>*

**Advantages:**

* Styles are centralized within a single HTML file.
* Useful for small, standalone documents or demos.

**Disadvantages:**

* Not reusable across multiple pages.
* Adds to page size if repeated across many pages.

**✅ 3. External CSS**

**Definition:**  
Styles are placed in a separate .css file and linked via the <link> tag in the HTML.

**Example:**

html

*<head>*

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

*</head>*

**Advantages:**

* Fully separates content (HTML) from presentation (CSS).
* Easy to reuse across multiple pages → better consistency.
* Reduces page clutter and improves maintainability.
* Browser can cache CSS files → faster loading.

**Disadvantages:**

* Requires an extra HTTP request (though this is minimal with caching).
* Styles won't load if the CSS file is missing or incorrectly linked.

**Here’s a clear table comparing inline, internal, and external CSS:**

| **Feature** | **Inline CSS** | **Internal CSS** | **External CSS** |
| --- | --- | --- | --- |
| **Location** | **Inside an element’s style attribute** | **In <style> tag within HTML <head>** | **In a separate .css file linked via <link>** |
| **Syntax Example** | **<p style="color:red;">Text</p>** | **<style>p { color: red; }</style>** | **In style.css: p { color: red; }** |
| **Reusability** | **❌ No – specific to a single element** | **❌ No – specific to one HTML page** | **✅ Yes – can be used across multiple pages** |
| **Maintainability** | **❌ Poor – hard to manage with many styles** | **😐 Moderate – styles are centralized but inline** | **✅ Excellent – styles are fully separated** |
| **Performance** | **😐 Lower – no caching, mixed content** | **😐 Okay – good for small pages** | **✅ Best – styles cached by browser** |
| **Use Case** | **Quick fixes, testing** | **Single-page websites, embedded apps** | **Large/multi-page websites** |
| **Specificity** | **✅ Highest – overrides other styles** | **Medium – can be overridden by inline or important** | **✅ Lower – easier to manage override hierarchy** |
| **File Size Impact** | **Increases HTML file size** | **Increases HTML file size** | **Reduces HTML file size** |
| **Separation of Concerns** | **❌ Mixed with content** | **😐 Partial separation** | **✅ Full separation of content and style** |