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

A **CSS selector** is a pattern used to select and style specific HTML elements. It tells the browser which HTML elements should be affected by the CSS rules.

**Examples:**

* **Element Selector**: Targets HTML elements by name.

p {

color: blue;

}  
**Class Selector**: Targets elements with a specific class attribute (prefixed with a dot .).  
.highlight {

background-color: yellow;

}  
  
  
**ID Selector**: Targets a single element with a specific ID (prefixed with a #).  
#main-heading {

font-size: 24px;

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

**CSS specificity** is a set of rules that browsers use to determine which style rule applies when multiple rules target the same element.

**Specificity Hierarchy (from low to high):**

1. **Element selectors** (e.g., div, p) – score: 0-0-1
2. **Class selectors, attributes, and pseudo-classes** (e.g., .box, [type="text"], :hover) – score: 0-1-0
3. **ID selectors** (e.g., #header) – score: 1-0-0
4. **Inline styles** (e.g., style="color: red;") – score: 1-0-0-0 (highest specificity)
5. **!important declaration** overrides all other rules (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.

| **CSS Type** | **Description** | **Advantages** | **Disadvantages** |
| --- | --- | --- | --- |
| **Inline CSS** | Applied directly to an HTML element via the style attribute. | - Quick to apply - Useful for small fixes or testing | - Hard to maintain - Not reusable - Lower performance |
| **Internal CSS** | Defined within a <style> tag inside the HTML <head>. | - Useful for single-page styling - No external file needed | - Not reusable across multiple pages - Increases HTML file size |
| **External CSS** | Stored in a separate .css file and linked via <link> in HTML. | - Reusable across multiple pages - Cleaner and more organized | - Requires extra HTTP request - No styling if file fails to load |

**Question 1: 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** describes how elements on a webpage are structured and how their dimensions are calculated. Every HTML element is considered a rectangular **box** that consists of four main parts:

**🔹 Components:**

* **Content**:  
  The actual content like text or images.  
  → This is the area where content is displayed.
* **Padding**:  
  Space between the content and the border.  
  → Adds spacing **inside** the element.
* **Border**:  
  A line surrounding the padding (and content).  
  → Adds a visible frame around the element.
* **Margin**:  
  Space **outside** the border.  
  → Creates distance between elements.

**🔸 How it affects the size:**

* The **total size** of an element depends on whether box-sizing is set to content-box (default) or border-box (more practical—explained in Q2).

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

The box-sizing property determines how the browser calculates the total width and height of an element.

**🔹 content-box (Default):**

* Width and height apply **only to the content**.
* Padding and border are added **outside** the specified width/height.

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

**🔹 What is Flexbox?**

**CSS Flexbox** (Flexible Box Layout) is a layout model that allows you to **design responsive, flexible layouts** without using floats or positioning hacks. It distributes space **dynamically** inside a container, even when the size of the elements is unknown or changes.

It makes it easy to:

* Center elements (horizontally and vertically)
* Create flexible grids
* Manage spacing and alignment in rows or columns
* **Question 2: Describe the properties justify-content, align-items, and flex-direction used in Flexbox.**

These properties are applied to the **flex container**, and they control the layout behavior of the items inside.

**flex-direction:**

* Defines the **main axis direction**

**justify-content:**

* Aligns items **along the main axis** (horizontal if row, vertical if column).

| **Question 1: Explain CSS Grid and how it differs from Flexbox. When would you use Grid over Flexbox?**  **Feature** | **Flexbox** | **Grid** |
| --- | --- | --- |
| Axis | One-dimensional (row or column) | Two-dimensional (rows **and** columns) |
| Use-case | Alignment of items in a row/column | Complex layouts like page templates |
| Layout direction | Main axis (horizontal/vertical) | Works with both axes simultaneously |
| Content placement | Content flows naturally | Items can be precisely positioned |
| Browser Support | Excellent | Excellent |

Q2 . Describe the grid-template-columns, grid-template-rows, and grid-gap properties. Provide examples of how to use them.  
  
.container {

display: grid;

grid-template-columns: 200px 1fr 2fr;

grid-template-rows: 100px 200px;

}  
  
.container {

display: grid;

grid-template-columns: repeat(3, 1fr);

gap: 20px; /\* adds both row and column spacing \*/

}  
  
grid eg  
<div class="grid-container">

<div class="box">1</div>

<div class="box">2</div>

<div class="box">3</div>

<div class="box">4</div>

</div>  
  
**Question 1: 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 styles **based on the characteristics of the user’s device**, such as screen width, height, orientation, resolution, etc.

They make it possible to **create responsive web designs**, which adapt gracefully to different screen sizes (like phones, tablets, and desktops).

**🔸 Why Are They Important?**

* They **enhance user experience** by ensuring your layout looks good on all devices.
* Help maintain **readability** and usability.
* Let you avoid **scrolling or zooming issues** on small screens.
* They are a core part of **mobile-first design** strategies.

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

Here’s how you do it:

css

CopyEdit

body {

font-size: 18px; /\* default font size for larger screens \*/

}

@media (max-width: 600px) {

body {

font-size: 14px; /\* smaller font size for smaller screens \*/

}

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

**🔹 Web-Safe Fonts:**

* These are **pre-installed fonts** that are **commonly available** across most operating systems and devices.
* Examples: Arial, Verdana, Times New Roman, Courier New, Georgia.
* They're **fast** and **don't require internet** to load, since they're already on the user's machine.

**🔸 Custom Web Fonts:**

* These are **downloaded fonts** delivered via CSS using @font-face or services like **Google Fonts**.
* They **enhance design** and branding by allowing more font style options (e.g., Roboto, Poppins, Open Sans).
* Require an internet connection (if not hosted locally) and **may impact load time** slightly.

**Why Use Web-Safe Fonts?**

* They're **lightweight and reliable**—no loading delays.
* Great for **fallback options** when custom fonts fail to load.
* Ideal for **performance-critical** websites or slow connections.

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

* The font-family property specifies the **font to be used** for text content on your webpage.
* You can define a **font stack**: preferred font followed by fallback fonts.

**Example:**

css

CopyEdit

body {

font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;

}