CSS and CSS3

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 HTML elements. It tells the browser which HTML elements the CSS rules should apply to.

Types of CSS Selectors with Examples

1. Element Selector

Selects all HTML elements of a given type (also called a type selector).

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

2. Class Selector

- Selects all elements with a specific class attribute.
- Prefixed with a **dot (.)**.

```
.box {
  border: 1px solid black;
  padding: 10px;
}
<div class="box">This is a box</div>
```

3. ID Selector

- Selects a single element with a specific ID.
- Prefixed with a **hash (#)**.
- IDs must be **unique** in a document.

```
#header {
  background-color: lightgray;
}
```

Header Content

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 to apply when multiple rules target the same element.
It helps resolve conflicts between styles.

Specificity Breakdown

Component	What it Counts	Example
a	Inline styles	style="color: red"
b	ID selectors (#id)	#header
c	Classes (.class), attributes, pseudo-classes	. btn, [type="text"], :hover
d	Element names and pseudo-elements	p, div, ::before

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

Type	Location	Scope
Inline CSS	Inside the HTML element (using style attribute)	Only that specific element
Internal CSS	Inside a <style> tag in the <head> of the HTML file	The entire HTML page
External CSS	In a separate .css file linked to the HTML	Can be reused across multiple pages

1.Inline CSS

Advantages:

- Quick for small changes or testing
- Useful when applying a single, unique style

Disadvantages:

- Hard to maintain for large projects
- Violates separation of content and design
- Cannot be reused across elements or pages
- Increases HTML file size

2.Internal CSS

Advantages:

- Styles are centralized within a single HTML page
- No need to load an external file
- Easier to debug for single-page websites

Disadvantages:

- Cannot be reused across multiple HTML pages
- Increases the size of the HTML file
- Not ideal for large websites

3.External CSS

Advantages:

- Reusable across multiple HTML pages
- Promotes clean separation of content (HTML) and style (CSS)
- Easier to manage and maintain large stylesheets
- Faster loading after the first page (CSS is cached)

Disadvantages:

- Requires an extra HTTP request to load the CSS file
- Styles won't apply if the CSS file fails to load

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 describes how elements are structured and spaced in a web page layout. Every HTML element is considered as a box made up of 4 parts:

1. Content

- The actual content (text, image, etc.) inside the element.
- It is the innermost part and its dimensions are controlled using width and height.

2. Padding

- Space between the content and the border.
- It adds space inside the element, increasing the total size.

3. Border

- The line around the padding and content.
- also increases the element's overall size.

4. Margin

- The space outside the element, separating it from other elements.
- Doesn’t increase the box size, but adds space around it.

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

Feature	content-box	border-box
Width & height include	Only content	Content + padding + border
Padding & border	Added outside the box	Included inside the box
Total element size increases with padding/border	Yes	No
Content area size	Fixed (as per width/height)	Shrinks with added padding/border
Easier for layout control	No	Yes
Common use	Precise content size needed	Responsive layouts, flexible design
CSS syntax	box-sizing: content-box;	box-sizing: border-box;
Default value in browsers	Yes *(default)*	No (must be set manually)

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 CSS layout module that provides a more efficient way to arrange, align, and distribute space among items in a container — even when their size is unknown or dynamic.

Flexbox is Useful for Layout Design

- Makes it easy to create responsive designs

- Aligns items horizontally or vertically
- Handles space distribution and item alignment with minimal code
- Eliminates the need for floats and complex positioning

1. Flex Container

- The **parent element** that holds flex items
- Declared with display: flex or display: inline-flex
- Controls how child items (flex-items) are arranged

2. Flex Item

- The **child elements** inside the flex container
- These items respond to flex properties such as flex-grow, order, align-self, etc.

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

1. flex-direction

This property sets the main axis — the direction in which flex items are placed inside the flex container.

Value	Description
row	Default – left to right (horizontal)
row-reverse	Right to left
column	Top to bottom (vertical)
column-reverse	Bottom to top

2. justify-content

This property aligns flex items along the main axis (horizontal by default).

Value	Description
flex-start	Align items to the start of the main axis

Value	Description
flex-end	Align items to the end of the main axis
center	Center items along the main axis
space-between	Equal space between items
space-around	Equal space around items
space-evenly	Equal space between and around items

3. align-items

This property aligns flex items along the cross axis (perpendicular to the main axis).

Value	Description
stretch	(Default) Stretch items to fill the container
flex-start	Align to the start of the cross axis
flex-end	Align to the end of the cross axis
center	Center items vertically (if row)
baseline	Align items based on their text baseline

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

CSS Grid is a two-dimensional layout system in CSS. It allows you to design web layouts using rows and columns, making it ideal for building complex interfaces like web pages, dashboards, and photo galleries.

CSS Grid vs. Flexbox — Key Differences

Feature	CSS Grid	Flexbox
Layout Direction	Two-dimensional (rows + columns)	One-dimensional (row or column)
Best For	Full-page layouts, grid-based designs	Alignment and spacing of items in a line
Control	Precise control over both axes	More flexible for content flow

Feature	CSS Grid	Flexbox
Overlapping Items	Yes, with grid-area or z-index	Not designed for overlap
Item Placement	Can place items anywhere in grid	Items flow in a single direction
Responsiveness	Good, but more setup needed	Easier for simple responsive layouts
Browser Support	Widely supported in modern browsers	Widely supported

When to Use CSS Grid:

- You need full page layout (e.g., header, sidebar, main, footer)
- You're building asymmetric or complex grid layouts
- You want to control both rows and columns
- You want items to overlap or span multiple areas

When to Use Flexbox:

- You're aligning items in a row or column
- You need a navigation bar, card row or form
- Content is dynamic and flows naturally
- Simpler layouts with single-axis control

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

1. grid-template-columns

Defines the number and size of columns in the grid.

Syntax:

```
grid-template-columns: value1 value2 value3;
```

Examples:

```
.grid {
display: grid;
```

```
grid-template-columns: 100px 200px auto;
}
```

2. grid-template-rows

Defines the number and size of rows in the grid.

Example:

```
.grid {
  display: grid;
  grid-template-rows: 50px 100px auto;
}
```

3. grid-gap / gap

Defines the spacing (gutter) between rows and columns in the grid.

Example:

```
.grid {
  display: grid;
  grid-template-columns: 1fr 1fr 1fr;
  grid-gap: 10px; /* or gap: 10px; */
}
```

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

Media queries in CSS are rules that allow you to apply styles based on the device's characteristics, such as screen width, height, orientation, resolution, and more. They are a core feature of responsive web design, which ensures your website looks and functions well on all devices—desktops, tablets, and smartphones.

Syntax of a Media Query:

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

Why Are Media Queries Important?

1. Responsive Design:
    - Ensures your site adapts to various screen sizes and resolutions.
    - Makes content readable and user-friendly on all devices.
2. Improved User Experience:
    - Hides or adjusts elements for smaller screens (like navigation menus).
    - Changes layout (like from a multi-column grid to a single column on mobile).
3. Performance Optimization:
    - You can choose to load or display lighter resources for mobile devices.
4. Accessibility:
    - Helps users on different devices, including those using assistive technologies.

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

```
body {
        font-size: 18px; /* Default for larger screens */
}
@media (max-width: 600px) {
        body {
                font-size: 14px;
                }
}
```

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

Feature	Web-Safe Fonts	Custom Web Fonts
Definition	Pre-installed fonts on most devices.	Fonts that are downloaded from the web.
Examples	Arial, Times New Roman, Verdana, Georgia	Google Fonts like Roboto, Open Sans, Lato
Availability	Always available—no need to load externally.	Must be linked or embedded via CSS or JavaScript.
Performance	Load instantly—no extra resources needed.	May slightly impact page load speed.
Design Flexibility	Limited variety and styling.	Wide range of styles and designs.
Usage	Used when reliability and speed are key.	Used for branding and modern design appeal.

Why Use a Web-Safe Font Instead of a Custom Font?

1. Faster Page Load:
    - No external files to download; improves performance on slow networks.
2. Greater Compatibility:
    - Works on virtually all devices and browsers by default.
3. Fallback Option:
    - Useful as a backup if a custom font fails to load.
4. Simple Projects:
    - Ideal for basic websites, emails, or internal tools where style is less critical.

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

The font-family property in CSS specifies the typeface (font) to be used for text on a webpage. It can define:

- A specific font (e.g., Arial, Times New Roman)
- A fallback list of fonts in case the preferred one is not available

- A generic font family (like serif, sans-serif, monospace) at the end

How to Apply a Custom Google Font to a Webpage

Step 1: Choose a font from Google Fonts

Example: Let's use Roboto

Step 2: Copy the `link` tag into your `<head>` section of HTML

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

Step 3: Use the font in your CSS with `font-family`

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