CSS Selectors & Styling

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

Ans: A CSS selector is used to select HTML elements so that styles can be applied to them.

1. Element Selector – selects all elements of a type.

```
main {
  color: blue;
}
```

2. Class Selector – selects elements with a class attribute.

```
.main {
  background-color: yellow;
}
```

3. ID Selector – selects an element with a unique ID.

```
#main- {
  text-align: center;
}
```

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

Ans: CSS Specificity is the set of rules that the browser uses to decide which CSS style should be applied when

there are conflicting styles for the same element. It works like a priority system. The more specific the selector, the higher priority it gets.

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

Ans: In CSS, there are three main ways to apply styles to HTML: internal CSS, external CSS, and inline CSS. Each has its own use cases, advantages, and disadvantages.

Inline CSS : CSS is written directly inside the HTML element using the style attribute.

Advantages:

- Quick for testing or applying styles to a single element.
- Does not require a separate CSS file.

Disadvantages:

- Not reusable, must be repeated for each element.
- Makes HTML code messy and harder to maintain.
- Lowers separation of content and design.

Internal CSS: CSS is written inside a <style> tag within the <head> section of an HTML file.

Advantages:

- Useful for styling a single page.

- Keeps CSS separate from element code (unlike inline).

Disadvantages:

- Styles are not reusable across multiple pages.
- Increases page size if repeated on many pages.

External CSS: CSS is written in a separate .css file and linked using <link> inside <head>.

Advantages:

- Reusable across multiple pages.
- Keeps HTML and CSS separate → cleaner and easier to maintain.
- Reduces code duplication and page size.

Disadvantages:

- Requires an additional HTTP request (slightly slower loading).
- Styles won't load if the CSS file is missing or not linked properly.

4. Explain the CSS box model and its components (content, padding, border, margin). How does each affect the size of an element?

- Content =

1 This is where your actual text, images, or other content lives.

2.Controlled by width and height properties.

Example: width: 200px; height: 100px;

- Padding

3. Space between the content and the border.

4. Transparent area that pushes the border outward.

Example: padding: 20px; adds 20px on all sides.

- Border

1. The visible edge around the padding and content.

2. Can be styled with thickness, color, and type.

3. Example: border: 5px solid black;

- Margin

1. Space outside the border, separating the element from others.

2. Also transparent, and doesn't affect the element's internal size.

3. Example: margin: 10px; adds spacing around the element.

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

- Content Box =

1. Default behavior in CSS

2. width and height apply only to the content
3. Padding and border are added outside the specified dimensions

- Border Box =
  1. width and height include content + padding + border
  2. Makes layout more predictable and easier to manage

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

- Flexbox is a one-dimensional layout model that makes it easy to align and distribute space among items in a container—even when their size is unknown or dynamic. It's ideal for laying out items in a row or column, and it adapts beautifully to different screen sizes.

- Flex Container =
  1. The parent element that holds flex items. You turn it into a flex container by setting: display: flex
- Flex Item =

1. Any direct child of a flex container becomes a flex item. These items can grow, shrink, and be aligned using Flexbox properties.

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

- Justify Content = Aligns items along the main axis (horizontal if row, vertical if column).
    1. flex-start = Items align at the start
    2. flex-end = Items align at the end
    3. center = Items are centered
    4. space-between = Equal space between items
    5. space-around = Equal space around items
    6. space-evenly = Equal space between and around items

- Align-items = Aligns items along the cross axis (perpendicular to the main axis).
    1. stretch = Items stretch to fill container (default)
    2. flex-start = Align at the top (or left if column)
    3. flex-end = Align at the bottom (or right if column)
    4. center = Vertically centered

5. baseline = Align based on text baseline

- flex-direction =
    1. row = items go left to right
    2. row-reverse = items go right to left
    3. column = items go top to bottom
    4. column-reverse = items go bottom to top

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

- CSS Grid is designed for layout control in both rows and columns-perfect for full-page or section-based designs.
- Key Features:
    1. Two-dimensional: You can align items both horizontally and vertically.
    2. Explicit layout: You define the grid structure first, then place items.
    3. Precise placement: Use grid-row, grid-column, or named grid areas.

- Flexbox is ideal for aligning items in a single direction-either row or column.
- Key Features:
    1. One-dimensional: Aligns items along one axis at a time.
    2. Content-driven: Items adapt to content size.
    3. Great for components: Navigation bars, button groups, cards.

- Grid Over Flexbox:
- Use Grid when:
    1. You need a structured layout with rows and columns.
    2. You want precise control over placement and spacing.
    3. You're building complex designs like dashboards or galleries.
- Use Flexbox when:
    1. You're aligning items **in** one direction.
    2. You want dynamic sizing based on content.
    3. You're designing components, not full layouts.

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

- Grid-template-columns:
- Defines the number and width of columns in a grid container.
- Example:

1.

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

- Grid-template-rows:
- Defines the height of rows in a grid container
- Example:

1. container {

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

- Grid-gap:

- Adds spacing between grid items. Though grid-gap is shorthand, modern CSS prefers gap.
- Example:

```
1.container {
  display: grid;
  grid-template-columns: repeat (3, 1fr);
  grid-template-rows: 100px 100px;
  grid-gap: 20px 10px;
}
```

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

- Media queries are a CSS3 feature that apply styles conditionally based on the characteristics of the user's device or viewport.
- Important because:
    1. Adapts to any screen size
    2. Improves usability on mobile and desktop
    3. Avoids horizontal scrolling and awkward layouts
    4. Delivers a consistent experience across devices

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

- /* Default font size for larger screens */
    1.

```
body {
font-size: 16px;
}

/* Media query for smaller screens */
@media (max-width: 600px) {
  body {
    font-size: 14px;
  }
}
```

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

-

Reason	Web-Safe	Custom Fonts

	Fonts	
Fast loading	Yes	Can slow down
Offline compatibility	Yes	Requires download
Universal fallback	Yes	Needs backup fonts
Branding flexibility	Limited	High
Accessibility	Often better	Depends on font

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

- The font-family property defines the font stack-a prioritized list of fonts to use. If the first font isn't available, the browser tries the next one, and so on.
- Applying a custom Google Font to a webpage:
  1. Step 1: Link the Font in <head>
  2. Add this inside your HTML <head> section:
- <link href="https://fonts.googleapis.com/css2?family=Roboto&display=swap" rel="stylesheet">

1. Step 2: Use font-family in CSS
2. Now apply it in your stylesheet:

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