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

CSS selectors target the HTML elements on your pages, allowing you to add styles based on their ID, class, type, attribute, and more.

Element Selector:

The element selector selects HTML elements based on the element name for example p, h1, div, span, etc.

Example:

```
h1 {
    color: red;
    font-size: 3rem;
}
p {
    color: white;
    background-color: gray;
}
```

Class Selector:

The class selector selects HTML elements with a specific class attribute.

Example:

```
.paragraph-class {
    color:white;
    font-family: monospace;
    background-color: purple;
}
```

ID selector:

An ID selector targets a single element with a specific ID attribute, using a hash prefix (e.g., #idname).

Example:

```
#div-container{
    color: blue;
    background-color: gray;
}
```

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

Specificity is an advanced algorithm used by HTML browsers to define the CSS declaration which will be the most appropriate to an HTML element. It basically calculates the weights of all the CSS selectors to determine the proper rule. The main goal is when more than one CSS rule is applied to the same element the selector with the highest specificity value will be applied to that element.

Resolving Conflicts

When multiple CSS rules apply to the same element, the rule with the highest specificity will be applied. If rules have the same specificity, the one defined last in the CSS will be applied.

- Inline styles will always override external and internal CSS due to their highest specificity.
- An ID selector will override class selectors, attribute selectors, and element selectors due to its higher specificity.
- If two rules have the same specificity, the one that appears last in the CSS will be applied.

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

Feature	Inline CSS	Internal CSS	External CSS
Location	It is used within HTML tag using the style attribute.	It is used within <head> section of HTML document.	It is used in a separate .css file.
Selector Scope	Affects a single element or a group of elements.	Affects multiple elements within the same HTML element.	Affects multiple HTML documents or an entire website.
Reusability	Not reusable. Styles need to be repeated for each element.	Can be reused on multiple elements within the same HTML document.	Can be reused on multiple HTML documents or an entire website.
Priority	Highest priority. Overrides internal and	Medium priority. Overrides external styles	Lowest priority. Can be overridden by both inline and

	external styles.	but can be overridden by inline styles.	internal styles.
File Size	Inline styles increase the HTML file size, which can affect the page load time.	Internal styles are part of the HTML file, which increases the file size.	External styles are in a separate file, which reduces the HTML file size and can be cached for faster page loads.
Maintainability	Not easy to maintain. Changes need to be made manually to each element.	Relatively easy to maintain. Changes need to be made in one place in the <head> section.	Easiest to maintain. Changes need to be made in one place in the external .css file.

Advantages of Inline Css:

1. **Quick to Implement**- Inline CSS allows you to apply styles directly to an HTML element, making it fast to add or change the design without extra steps.
2. **No additional files** – Since it doesn't require linking to external style sheets, you avoid creating or managing separate CSS files, simplifying the setup.

Disadvantage of Inline css:

1. **Hard to Maintain**:When you mix CSS with HTML, updating and managing the style becomes a big hassle, especially if your website grows or changes often.
2. **Non-reusable styles** – Styles can't be applied to multiple pages or elements without copying and pasting, which is time-consuming and inefficient.
3. **Increases page load time** – Every time a page loads, it has to load the style information too, making the whole website slower to show up on your screen.

Advantages of Internal CSS

- **Simplicity and Portability**: Internal CSS simplifies the process of styling a single web page, making it easier to maintain and modify the design elements without impacting other pages;
- **Isolation of Styles**: By using internal CSS, developers can ensure that the styles defined are specific to the individual web page, maintaining a clear separation of concerns within the codebase;

Disadvantages of Internal CSS

- **Limited Reusability**: Unlike external CSS, internal CSS cannot be reused across multiple web pages, potentially leading to duplication of code and increased maintenance efforts;
- **Complexity with Larger Projects**: In larger web development projects, managing styles within individual HTML files may lead to complexity and hinder the scalability of the application.
- **Advantages of External CSS:**
- Since the CSS code is in a separate document, your HTML files will have a cleaner structure and are smaller in size.
- You can use the same **.css** file for multiple pages.

- **Disadvantages of External CSS:**
- Your pages may not be rendered correctly until the external CSS is loaded.
- Uploading or linking to multiple CSS files can increase your site’s download time.

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

The box model consists of four main components: content, padding, border, and margin.

Components of the CSS Box Model

Content: This is the actual content of the element, such as text, images, or other media.

Impact on Size: The size of the content is determined by its width and height properties. The content size directly affects the overall size of the element.

Padding: Padding is the space between the content and the border. It creates an inner margin around the content.

Impact on Size: Padding adds extra space inside the element, increasing the element's overall size.

Border: The border wraps around the padding (if any) and content. It is a visible outline of the element.

Impact on Size: The border adds to the element's size.

Margin: Margin is the space outside the element's border. It creates an outer margin, separating the element from other elements

Impact on Size: Margins do not increase the size of the element itself but affect the spacing around the element.

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

The box-sizing property in CSS determines how the total width and height of an element are calculated. There are two main values for box-sizing: content-box and border-box.

- **border-box:** In this value, not only width and height properties are included but you will find padding and border inside of the box for example .box {width: 200px; border: 10px solid black;} renders a box that is 200px wide.
- **content-box:** This is the default value of box-sizing. The dimension of element only includes ‘height’ and ‘width’ and does not include ‘border’ and ‘padding’ given to element. Padding and Border take space outside the element. This is often considered the default value.
- It calculates the total width and height of an element by considering only the content, excluding padding

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

Flexbox is a layout method for arranging items in rows or columns.

Flexbox makes it easier to design a flexible responsive layout structure, without using float or positioning.

CSS Flexbox (Flexible Box Layout) is a layout model that provides an efficient way to design and distribute space among items in a container. It simplifies the process of creating complex layouts, especially for dynamic and responsive designs.

Flexbox is implemented by applying the **display: flex** property to the container and using various flex properties for the items.

Flex Container:

- The **flex container** is the parent element that holds flex items. When you set the display property of an element to flex or inline-flex, you create a flex container. This element is responsible for establishing the flex context for its children.

Flex Item:

- **Flex items** are the direct children of a flex container. These items can be individually controlled and aligned within the flex container using various flex properties.

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

justify-content property aligns the flexible container's items when the items do not use all available space on the main-axis (horizontally).

Value	Description
flex-start	Default value. Items are positioned at the beginning of the container
flex-end	Items are positioned at the end of the container
center	Items are positioned in the center of the container
space-between	Items will have space between them
space-around	Items will have space before, between, and after them
space-evenly	Items will have equal space around them

initial	Sets this property to its default value.
inherit	Inherits this property from its parent element.

Align-Items

The align-items property specifies the default alignment for items inside a flexbox or grid container.

- In a flexbox container, the flexbox items are aligned on the cross axis, which is vertical by default (opposite of flex-direction).
- In a grid container, the grid items are aligned in the block direction. For pages in English, block direction is downward and inline direction is left to right.

• **Value**	• **Description**
• normal	• Default. Behaves like 'stretch' for flexbox and grid items, or 'start' for grid items with a defined block
• stretch	• Items are stretched to fit the container
• center	• Items are positioned at the center of the container
• flex-start	• Items are positioned at the beginning of the container
• flex-end	• Items are positioned at the end of

	the container
• start	• Items are positioned at the beginning of their individual grid cells, in the block direction
• end	• Items are positioned at the end of the their individual grid cells, in the block direction
• baseline	• Items are positioned at the baseline of the container
• initial	• Sets this property to its default value.
• inherit	• Inherits this property from its parent element.

Flex direction: The flex-direction property specifies the direction of the flexible items.

Value	**Description**
row	Default value. The flexible items are displayed horizontally, as a row
row-reverse	Same as row, but in reverse order
column	The flexible items are displayed vertically, as a

	column
column-reverse	Same as column, but in reverse order
initial	Sets this property to its default value.
inherit	Inherits this property from its parent element.

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

CSS Grid

CSS Grid is a two-dimensional layout system, meaning it can handle both rows and columns. It's ideal for complex layouts where you want to control both the horizontal and vertical alignment of elements.

Key Features of CSS Grid:

- Two-dimensional: Can create complex layouts with rows and columns.
- Grid Areas: Allows you to define named areas of the layout.

Flexbox

Flexbox is a one-dimensional layout system, meaning it handles either rows or columns, but not both simultaneously. It's perfect for simpler layouts or for distributing space among items in a single direction.

Key Features of Flexbox:

- **Order:** Allows easy reordering of elements within the container.
- **Flexible items:** Items can grow or shrink to fit the space.

When to Use Grid Over Flexbox

- **Complex Layouts:** Use CSS Grid when you need a complex layout with both rows and columns.

- **Alignment:** Choose CSS Grid if you need precise control over the alignment of items in both directions.
- **Overlapping Elements:** Use CSS Grid when you need elements to overlap.

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

. grid-template-columns: Specifies the number and widths of columns in the grid.

Example

Make a grid with 5 columns:

```
.grid-container {
  display: grid;
  grid-template-columns: auto auto auto auto auto;
}
```

. grid-template-rows: Specifies the number and heights of rows in the grid.

Example

```
.grid-container {
  display: grid;
  grid-template-rows: 80px 200px;
}
```

grid-gap

This property defines the space between rows and columns in the grid. It can be defined using either grid-row-gap and grid-column-gap, or the shorthand grid-gap.

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

Media queries are a CSS feature that allows you to apply styles based on specific conditions, such as screen size, resolution, orientation, and more.

They are essential for creating responsive designs, which adapt to different devices and screen sizes.

Device Adaptability: Media queries allow your website to adapt its layout and style to various devices, ensuring a consistent user experience across desktops, tablets, and smartphones.

Improved Usability: By tailoring the layout to the user's screen size, you enhance the usability and readability of your website. This is crucial for users who access your site on smaller screens.

Performance Optimization: Media queries can help optimize performance by loading only necessary styles for a particular device or condition, reducing the overall load time.

Flexibility and Control: They offer fine-grained control over your design, enabling you to create sophisticated, adaptive layouts that respond to different user contexts.

Future-Proofing: As new devices with varying screen sizes and capabilities emerge, media queries ensure your website remains compatible and visually appealing.

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

```
@media (max-width: 600px) {
Body{
   font-size: 14px;
}
}
```

- The default font size is 16px.
- When the viewport width is 600px or smaller, the font size is adjusted to 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?

Web-safe fonts:

Web-safe fonts are fonts that are universally pre-installed on most devices and operating systems. Examples include Arial, Times New Roman, and Courier New. Since these fonts are already available on users' devices, they do not require additional downloads.

Custom web fonts:

Custom web fonts are not pre-installed on devices. They are downloaded from a web server when the webpage is loaded. Examples include fonts hosted on platforms like Google Fonts (e.g., Roboto, Open Sans) or Adobe Fonts.

Why use a web-safe font over a custom font?

- Performance: Faster loading times and reduced bandwidth usage.
- Fallback: Ensures text is rendered correctly even in environments with limited internet connectivity.
- Compatibility: Guaranteed to work across all devices without requiring external resources.

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

font-family property:

The font-family property in CSS specifies the typeface or a list of fallback fonts for text content. It allows developers to define the preferred font followed by alternative options in case the first choice isn't available.

css

Copy code

```
p {
   font-family: "Roboto", Arial, sans-serif;
}
```

Applying a Custom Google Font

To use a custom Google Font, follow these steps:

1. **Choose a Font from Google Fonts:** Go to Google Fonts and select the font you want to use. For this example, let's choose "Roboto".
2. **Include the Google Font in Your HTML:** Copy the <link> tag provided by Google Fonts and paste it in the <head> section of your HTML.

14.Difference b/w HTML & HTML5?

HTML	HTML5
It didn’t support audio and video without the use of flash player support.	It supports audio and video controls with the use of <audio> and <video> tags.
It uses cookies to store temporary data.	It uses SQL databases and application cache to store offline data.
Does not allow JavaScript to run in the browser.	Allows JavaScript to run in the background. This is possible due to JS Web worker API in HTML5.
Vector graphics are possible in HTML with the help of various technologies such as VML, Silver-light, Flash, etc.	Vector graphics are additionally an integral part of HTML5 like SVG and Canvas.
It does not allow drag and drop effects.	It allows drag and drop effects.

HTML	HTML5
Not possible to draw shapes like circle, rectangle, triangle etc.	HTML5 allows to draw shapes like circle, rectangle, triangle etc.

15.What are the additional tags used in HTML5?

Sectioning Elements

- <section>: Defines a section in a document.
- <article>: Represents an independent piece of content.
- <aside>: Contains content that is tangentially related to the content around it.
- <nav>: Represents a section of navigation links.
- <header>: Defines a header for a document or section.
- <footer>: Defines a footer for a document or section.
- <main>: Represents the main content of the document.

Text-Level Semantics

- <mark>: Highlights text for reference or notation.
- <time>: Represents a specific time or date.
- <progress>: Represents the progress of a task.
- <meter>: Represents a scalar measurement within a known range.

Form Elements

- <input>: Adds new input types like email, url, number, range, date, color, etc.

- <datalist>: Provides a list of predefined options for an input element.
- <output>: Represents the result of a calculation.
- <progress>: Represents the progress of a task.

Multimedia Elements

- <audio>: Embeds sound content.
- <video>: Embeds video content.
- <source>: Specifies multiple media resources for media elements (<audio> and <video>).
- <track>: Specifies text tracks for media elements (<audio> and <video>).

Graphics Elements

- <canvas>: Used for rendering graphics on the fly via JavaScript.
- <svg>: Integrates Scalable Vector Graphics.

Interactive Elements

- <details>: Represents additional details that the user can view or hide.
- <summary>: Represents a summary or label for the <details> element.
- <dialog>: Defines a dialog box or window.
- <template>: Holds client-side content that is not rendered when the page loads but can be instantiated later.