MERN STACK CSS AND CSS3:

CSS Selectors & Styling:

(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** a set of CSS rules should apply to.

Here are the main types with examples:

**1. Element Selector**

* **Targets** all elements of a specific type.

**Syntax:**

element {

property: value;

}

**Example:**

p {

color: blue;

}

This changes the text color of all <p> (paragraph) elements to blue.

**2. Class Selector**

* **Targets** all elements with a specific class attribute.
* **Prefix** with a dot .

**Syntax:**

.class-name {

property: value;

}

**Example:**

.highlight {

background-color: yellow;

}

This applies a yellow background to any element with class="highlight".

**HTML:**

html

<div class="highlight">Important</div>

**3. ID Selector**

* **Targets** one unique element with a specific id.
* **Prefix** with a hash #

**Syntax:**

#id-name {

property: value;

}

**Example:**

#header {

font-size: 24px;

}

This sets the font size of the element with id="header" to 24 pixels.

**HTML:**

html

<h1 id="header">Welcome</h1>

(2) Explain the concept of CSS specificity. How do conflicts between multiple stylesget resolved?

**CSS specificity** is a system used by browsers to determine **which CSS rule should be applied to an element** when multiple rules match the same element. It is a method of resolving conflicts between different CSS declarations.

Each selector has a **specificity value**, which is calculated based on the type of selector used. The browser uses this value to **compare** rules and decide which one has **higher priority**.

**📊 Specificity Hierarchy (from lowest to highest):**

1. **Universal selector (\*), combinators (+, >, ~)**  
   → Very low specificity  
   → Specificity: 0,0,0,0
2. **Element selectors (div, p, h1, etc.) and pseudo-elements (::before, ::after)**  
   → Specificity: 0,0,0,1
3. **Class selectors (.class), attributes ([type="text"]), and pseudo-classes (:hover, :nth-child())**  
   → Specificity: 0,0,1,0
4. **ID selectors (#id)**  
   → Specificity: 0,1,0,0
5. **Inline styles (style="" in HTML)**  
   → Specificity: 1,0,0,0
6. **!important declaration**  
   → Overrides all other rules (but specificity is still considered between conflicting !important rules)

**How Conflicts Are Resolved:**

When multiple rules apply to the same element:

1. **Compare specificity values**:  
   The rule with the highest specificity wins.
2. **If specificity is equal**:  
   The rule that appears **later in the stylesheet** (or in the last linked file) takes precedence.
3. **If one rule uses !important**:  
   That rule wins, regardless of specificity, unless two conflicting rules **both** use !important — then specificity is compared again.

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

**1. Inline CSS**

**✅ What it is:**

CSS is written **directly inside an HTML element** using the style attribute.

**🔤 Example:**

<p style="color: blue; font-size: 16px;">Hello World</p>

**Advantages:**

* Easy and quick to apply styles to a single element.
* Useful for testing or applying one-time styles.
* Overrides other CSS types due to higher specificity.

**Disadvantages:**

* Not reusable — you must repeat the style for each element.
* Makes HTML messy and harder to read.
* Difficult to maintain or update styles for large projects.

**2. Internal CSS**

**✅ What it is:**

CSS is written **inside a <style> tag** in the <head> section of the HTML file.

**🔤 Example:**

html

CopyEdit

<head>

<style>

p {

color: green;

}

</style>

</head>

**Advantages:**

* Good for small projects or single-page websites.
* Keeps styles separate from content (unlike inline).
* Easy to override external CSS for that page.

**Disadvantages:**

* Not reusable across multiple pages.
* Increases page load time if overused.
* Still clutters the HTML file if too large.

**3. External CSS**

**✅ What it is:**

CSS is written in a **separate .css file** and linked to the HTML file using <link>.

**🔤 Example:**

html

CopyEdit

<head>

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

</head>

**styles.css**

css

CopyEdit

p {

color: red;

}

**Advantages:**

* **Reusable** across many HTML pages.
* Keeps HTML clean and organized.
* Best for large websites.
* Faster loading on repeat visits (browser caching).

**Disadvantages:**

* Requires an extra HTTP request to load the CSS file.
* Styles may not show if the link is broken or file is missing.
* Slightly more complex for beginners.

• CSS Box Model:

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

**What is the CSS Box Model?**

The **CSS Box Model** is the way browsers **calculate the size and spacing** of elements on a web page.  
Every HTML element is seen as a **box** made up of **four parts**:

**🧱 The 4 Components:**

**1. Content**

* This is the **actual content** inside the element (text, image, etc.).
* It’s the **innermost** part of the box.
* You can set its width and height using width and height.

**2. Padding**

* The space **inside the border** but **around the content**.
* It creates space **between the content and the border**.
* Affects the total size of the element.

**3. Border**

* The line **around the padding and content**.
* You can change its width, style, and color.
* Also adds to the element’s size.

**4. Margin**

* The space **outside the border**, between the element and others.
* It does **not** increase the size of the box itself, but **adds space around it**.

**📐 How Each Affects Size**

By default, the total size of an element is calculated as:

**Total Width = content width + padding + border + margin**

**Total Height = content height + padding + border + margin**

**Example:**

**width: 200px;**

**padding: 10px;**

**border: 5px solid;**

**margin: 20px;**

(2) What is the difference between border-box and content-box box-sizing in CSS? Which is the default?

**(1)content-box (Default)**

* The width and height you set **only apply to the content inside** the box (like text or images).
* If you add padding (space inside) or border (border line), these get **added outside** the content size.
* So, the **total size of the element becomes bigger** than what you set.

**2. border-box**

* The width and height you set **include the content, padding, and border all together**.
* This means the total size of the element **stays the same** no matter how much padding or border you add.
* The content area gets smaller to fit inside the total size.

**Which is the Default?**

* The default value of box-sizing in CSS is **content-box**.

• CSS Flexbox:

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

**What is CSS Flexbox?**

**CSS Flexbox** is a way to arrange and align items in a container, making it easier to create flexible and responsive layouts.

Imagine you have a row of boxes, and you want them to line up nicely, adjust their size, or wrap onto the next line when the screen gets smaller — Flex box helps you do that easily without using complicated code.

**Why is Flex box useful for layout design?**

* It helps **align items** horizontally or vertically.
* Items can **grow, shrink, or stay the same size** depending on the space available.
* It’s great for making **responsive designs** that look good on all screen sizes.
* It saves a lot of time compared to older layout methods like floats or tables.

**Key terms:**

1. **Flex-container**  
   This is the parent element where you apply display: flex;. It’s the box that holds all the items you want to arrange using Flexbox.
2. **Flex-item**  
   These are the child elements inside the flex-container. They are the individual boxes or elements that you want to layout and control using Flexbox.

**Simple example:**

.container {

display: flex; /\* makes this a flex-container \*/

}

.item {

flex: 1; /\* makes flex-items grow equally \*/

}

* .container is the **flex-container**.
* .item is a **flex-item** inside it.

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

**1. flex-direction**

This sets the **main direction** in which the flex-items are placed inside the container.

**Common values:**

* row → (default) items go **left to right**.
* column → items go **top to bottom**.
* row-reverse → items go **right to left**.
* column-reverse → items go **bottom to top**.

✅ **Example:**

.container {

display: flex;

flex-direction: row;

}

**2. justify-content**

This controls how **items are spaced along the main Common values:**

* flex-start → items start from the beginning (left or top).
* flex-end → items go to the end (right or bottom).
* center → items are centered.
* space-between → space **between** items.
* space-around → space **around** items.
* space-evenly → **equal** space between all items.

✅ **Example:**

.container {

display: flex;

justify-content: center;

}

**3. align-items**

This controls how items are aligned **on the cross axis** (perpendicular to flex-direction).

**Common values:**

* stretch → (default) items stretch to fill container.
* flex-start → items align to the **top** (or left in column).
* flex-end → items align to the **bottom** (or right).
* center → items are centered.

✅ **Example:**

.container {

display: flex;

align-items: center;

}

* CSS GRID:

1. Explain CSS Grid and how it differs from Flexbox. When would you use Grid overFlexbox?

CSS Grid is a layout system designed specifically for creating **two-dimensional** layouts (both rows and columns).

Flexbox is a one-dimensional layout system — it works **in a single direction**: either row or column.

**🔹 Key Features:**

* Designed for content alignment **along one axis**.
* You use justify-content, align-items, flex-direction to control layout.
* Ideal for smaller components like navbars, cards, or toolbars.

| **Use Grid When:** |
| --- |
| You need **rows *and* columns**. |
| Layout is complex (e.g., a webpage structure). |
| You want full control of spacing & placement. |

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

 The grid-template-columns property **sets up the number of columns and their sizes in a grid layout**.

 It controls **how wide each column should be**.

.grid-container {

display: grid;

grid-template-columns: 100px 200px 100px;

}

his means there are **3 columns** in the grid:

* The first column is 100px wide.
* The second column is 200px wide.
* The third column is 100px wide.

**What is grid-template-rows?**

* The grid-template-rows property **sets up the number of rows and their heights in a grid layout**.
* It controls **how tall each row should be**.

.grid-container {

display: grid;

grid-template-rows: 50px 100px 50px;

}

**Explanation:**  
This means there are **3 rows** in the grid:

* The first row is 50px high.
* The second row is 100px high.
* The third row is 50px high.

**🔹 What is grid-gap?**

* The grid-gap property **sets the space or gap between rows and columns in a grid**.
* It adds some space so the grid items aren’t directly touching each other.

**Example:**

css

CopyEdit

.grid-container {

display: grid;

grid-template-columns: 100px 100px 100px;

grid-template-rows: 50px 50px 50px;

grid-gap: 20px;

}

✅ **Explanation:**  
This means there’s a **20px gap between the columns and the rows** in the grid.

* Responsive Web Design with Media Queries:

(1)What are media queries in CSS, and why are they important for responsive design?

* **Why are Media Queries Important for Responsive Design?**
* They help your website look **good on all devices** — mobile, tablet, and desktop — by **changing layout and styles** depending on screen size.
* @media (condition) {
* /\* CSS rules here \*/
* }

EXAMPLE:

/\* Default style for big screens (desktops) \*/

.container {

font-size: 20px;

background: lightblue;

}

/\* For small screens (phones under 600px) \*/

@media (max-width: 600px) {

.container {

font-size: 14px;

background: lightgreen;

}

}

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

/\* Default font size for big screens \*/

body {

font-size: 20px;

}

/\* Font size changes for small screens (less than 600px) \*/

@media (max-width: 600px) {

body {

font-size: 14px;

}

}

| **Line** | **What it Does** |
| --- | --- |
| body { font-size: 20px; } | Normally, text size is 20px on big screens |
| @media (max-width: 600px) | If the screen is **600px wide or smaller** (like a mobile phone), then: |
| body { font-size: 14px; } | → The text size becomes smaller: 14px |

* Typography and Web Fonts:

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

**What are web-safe fonts?**  
Web-safe fonts are fonts that are already installed on almost all computers and devices. This means you can use them safely without needing to download or add anything extra.  
Examples: Arial, Times New Roman, Georgia, Verdana.

**What are custom web fonts?**  
Custom web fonts are fonts that are not already on everyone's computer. They need to be downloaded from the internet when someone visits your website. This lets you use special or stylish fonts that match your design.  
Examples: Google Fonts, fonts you buy online, or your own font files.

**Why might you use a web-safe font instead of a custom font?**  
✅ It’s faster to load because the font is already on the computer.  
✅ Your website will work even if there’s a slow internet connection or if custom fonts are blocked.  
✅ It’s a reliable choice — you know it will look the same for all users.

(2) What is the font-familyproperty in CSS? How do you apply a custom Google Font to a webpage?

**What is font-family in CSS?**

The font-family property specifies the font for text in your webpage.  
If the specified font isn’t available on the user's computer, fallback fonts can be provided.

Example:

body {

font-family: Arial, sans-serif;

}

**How to apply a custom Google Font to a webpage:**

1. **Go to** [**Google Fonts**](https://fonts.google.com/).
2. Select your preferred font (say **Roboto**).
3. Click **Select this style**, then **Copy the embed link**.
4. Insert it into your HTML’s <head>:

Example:

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<link rel="preconnect" href="https://fonts.googleapis.com">

<link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>

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

<title>Google Fonts</title>

</head>

<body>

<h1>Hello with Roboto</h1>

</body>

</html>

Apply it in your CSS:

body {

font-family: 'Roboto', sans-serif;

}