Question 1: **Explain CSS Grid and how it differs from Flexbox. When would you use Grid over Flexbox?**

**What is CSS Grid?**

* **CSS Grid** is a two-dimensional layout system in CSS that allows you to design web layouts using **rows and columns**. It gives you precise control over the placement, size, and alignment of elements across both axes (horizontal and vertical).
* To use Grid, you apply display: grid to a container, and then define rows and columns using properties like grid-template-rows, grid-template-columns, and place child elements into the defined grid areas.

**Key Features of CSS Grid:**

* **Two-dimensional layout**: Controls layout in both rows and columns.
* **Explicit and implicit tracks**: Define exact layout areas or let the browser auto-place items.
* **Grid lines, areas, and gaps**: Offers powerful tools to align, span, and space elements.
* **Precise placement**: Items can span multiple rows/columns or be placed in specific cells.

**Example of a Grid Layout:**

css

*.container {*

*display: grid;*

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

*grid-template-rows: auto;*

*gap: 10px;*

*}*

html

*<div class="container">*

*<div class="item1">1</div>*

*<div class="item2">2</div>*

*<div class="item3">3</div>*

*</div>*

* This creates a grid with 3 columns (first fixed, others flexible) and automatically-sized rows.

**Grid vs. Flexbox: Key Differences**

| **Feature** | **CSS Flexbox** | **CSS Grid** |
| --- | --- | --- |
| **Layout type** | One-dimensional (row OR column) | Two-dimensional (rows AND columns) |
| **Use case** | Align items in a single direction | Complex layouts with rows & columns |
| **Alignment** | Along main and cross axis | Along both axes with precise control |
| **Placement** | Based on content flow | Can define explicit cell positions |
| **Item control** | Order and flexibility of size | Grid areas, line numbers, and spanning |
| **Responsiveness** | Good for small, adaptive sections | Better for full-page or major layouts |

**When to Use Grid Over Flexbox**

Use **CSS Grid** when:

* You need a **full page or major section layout** with both rows and columns.
* You want to place items in **specific cells** or areas (e.g., dashboard layout).
* The design requires **row and column spanning**.
* You need **precise alignment** across both axes.

Use **Flexbox** when:

* You're aligning items **in a single line or direction** (row or column).
* You need **content-based** layouts (e.g., nav bars, toolbars, menus).
* You want **equal spacing or alignment** for items of dynamic sizes.

Question 2: **Describe the grid-template-columns, grid-template-rows, and grid-gap**

**properties. Provide examples of how to use them.**

* Here’s a detailed explanation of the CSS Grid properties: **grid-template-columns**, **grid-template-rows**, and **grid-gap** (now split into row-gap and column-gap in modern CSS).

**1. grid-template-columns**

This property defines the **number and size of columns** in a CSS Grid container.

**Syntax:**

css

*grid-template-columns: <track-size> <track-size> ...;*

**Examples:**

css

*.container {*

*display: grid;*

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

*}*

* First column: 100px
* Second column: 200px
* Third column: adjusts to remaining space

You can also use **fr units**, which divide space fractionally:

css

*grid-template-columns: 1fr 2fr 1fr;*

This creates three columns. The second column is twice as wide as the first and third.

**2. grid-template-rows**

This property defines the **number and size of rows** in the grid container.

**Syntax:**

css

*grid-template-rows: <track-size> <track-size> ...;*

**Example:**

css

*.container {*

*display: grid;*

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

*}*

* First row: 50px
* Second row: 100px
* Third row: fills remaining space

You can also repeat tracks:

css

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

This creates 3 equal-height rows.

**3. grid-gap *(deprecated but still widely used)***

Modern CSS separates this into **row-gap** and **column-gap**.

**Syntax:**

css

*grid-gap: <row-gap> <column-gap>;*

**Example:**

css

*.container {*

*display: grid;*

*grid-template-columns: 1fr 1fr;*

*grid-template-rows: auto auto;*

*grid-gap: 20px 10px; /\* 20px row gap, 10px column gap \*/*

*}*

For modern syntax:

css

*.row-gap: 20px;*

*.column-gap: 10px;*

You can also use:

Css

*gap: 20px 10px; /\* same as above \*/*

**Full Example:**

html

*<style>*

*.container {*

*display: grid;*

*grid-template-columns: 150px 1fr 1fr;*

*grid-template-rows: 100px 200px;*

*gap: 20px 10px;*

*background-color: #eee;*

*padding: 10px;*

*}*

*.item {*

*background-color: #8ac;*

*padding: 20px;*

*color: white;*

*font-weight: bold;*

*}*

*</style>*

*<div class="container">*

*<div class="item">1</div>*

*<div class="item">2</div>*

*<div class="item">3</div>*

*<div class="item">4</div>*

*<div class="item">5</div>*

*<div class="item">6</div>*

*</div>*

This example creates a grid with:

* 3 columns (1 fixed, 2 flexible),
* 2 rows,
* and spacing between both rows and columns.