CSS Grid

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

Ans:- CSS Grid is a **2-dimensional layout system** that allows you to design web layouts by aligning items both **horizontally (rows)** and **vertically (columns)** within a grid container. It provides a structured way to create complex layouts without relying on external frameworks.

Key Features of CSS Grid

1. **2D Layout**: Handles both rows and columns, making it ideal for designing entire web page layouts.
2. **Explicit Control**: Allows precise placement of items in specific grid cells.
3. **Flexible Track Sizing**: Supports flexible and fixed sizing for rows and columns.
4. **Auto-placement**: Automatically positions elements into available cells.

When to Use CSS Grid vs. Flexbox?

- **Use CSS Grid**:
    - When designing complex layouts like web pages or dashboards with multiple sections.
    - When you need precise control over row and column alignment.
    - If you want a clean structure with gaps between rows/columns.
- **Use Flexbox**:
    - When arranging items in a single direction (row or column).
    - For simpler, smaller components like buttons, navbars, or toolbars.
    - When items need to adjust dynamically in one direction.

Grid Example :-

```
<div class="grid-container">
    <div class="item">1</div>
    <div class="item">2</div>
    <div class="item">3</div>
    <div class="item">4</div>
</div>
```

Style.css :-

```
.grid-container {
    display: grid;
    grid-template-columns: 1fr 1fr; /* Two equal-width columns */
    grid-template-rows: auto auto; /* Two rows with automatic height */
    gap: 10px; /* Spacing between items */
    background-color: lightgray;
}

.item {
    background-color: steelblue;
    color: white;
    padding: 20px;
}
```

Flexbox Example :-

```
<div class="grid-container">
    <div class="item">1</div>
    <div class="item">2</div>
    <div class="item">3</div>
    <div class="item">4</div>
</div>
```

Style.css :-

```
.flex-container {
    display: flex;
    flex-wrap: wrap; /* Allows items to wrap to the next line */
    gap: 10px;
    background-color: lightgray;
}

.item {
    background-color: coral;
    color: white;
    padding: 20px;
```

```
  width: 45%; /* Each item takes 45% of the row width */
}
```

Question 2: Describe the grid-template-columns, grid-template-rows, and grid-gap properties. Provide examples of how to use them.

1. grid-template-columns :-

- **Purpose**: Defines the number and size of columns in a grid layout.

  Eg:- grid-template-columns: size1 size2 size3 ...;

  grid-template-columns: 100px 200px 1fr;
  Creates three columns: the first is 100px wide, the second is 200px wide, and the third takes up the remaining available space.

2. grid-template-rows :-

- **Purpose**: Defines the number and size of rows in a grid layout.

  Eg:- grid-template-rows: size1 size2 size3 ...;

  grid-template-rows: auto 150px 1fr;
  Creates three rows: the first adjusts automatically to its content, the second is fixed at 150px, and the third takes the remaining available space.

3. gap **(formerly** grid-gap**) :-**

- **Purpose**: Specifies the spacing between rows and columns in the grid.

  Example:- gap: row-gap column-gap;

  gap: 10px 20px;
  Sets a 10px gap between rows and a 20px gap between columns.

- gap: 15px;

  Sets a uniform 15px gap between both rows and columns.

  Example :-

```
<div class="grid-container">
   <div class="item">1</div>
   <div class="item">2</div>
   <div class="item">3</div>
   <div class="item">4</div>
</div>
```

Style.css :-

```
.grid-container {
   display: grid;
   grid-template-columns: 100px 200px 1fr; /* Three columns */
   grid-template-rows: auto 150px;        /* Two rows */
   gap: 10px 20px;                    /* 10px between rows, 20px between columns */
   background-color: lightgray;
}

.item {
   background-color: steelblue;
   color: white;
   padding: 20px;
   text-align: center;
}
```