📌 Grid Layout & Flexbox in CSS3

Grid Layout and Flexbox are **modern CSS techniques** used for designing responsive web layouts. Both are powerful but serve different purposes.

🟢 1. CSS Grid Layout

CSS Grid is a **two-dimensional** layout system, meaning it controls both **rows and columns**. It's perfect for **complex page structures**.

◇ Basic Grid Structure

```
.container {
    display: grid;
    grid-template-columns: repeat(3, 1fr); /* 3 equal columns */
    grid-template-rows: 100px 200px; /* 2 rows with defined height */
    gap: 10px; /* Space between grid items */
}
```

◇ Example: Grid with Items

```
<div class="container">
    <div class="box">1</div>
    <div class="box">2</div>
    <div class="box">3</div>
    <div class="box">4</div>
    <div class="box">5</div>
    <div class="box">6</div>
</div>
```

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

.box {
    background-color: lightblue;
    text-align: center;
    padding: 20px;
    font-size: 20px;
}
```

◇ Important Grid Properties

Property	Description
`display: grid;`	Defines a grid container
`grid-template-columns`	Defines the number & size of columns
`grid-template-rows`	Defines the number & size of rows
`gap`	Adds spacing between grid items
`grid-column`	Controls column placement of an item
`grid-row`	Controls row placement of an item
`justify-items`	Aligns items horizontally inside grid cells
`align-items`	Aligns items vertically inside grid cells

◇ Example: Merging Cells (Grid Areas)

```
.container {
    display: grid;
    grid-template-columns: 1fr 2fr;
    grid-template-areas:
        "header header"
        "sidebar content"
        "footer footer";
}

.header {
    grid-area: header;
    background: orange;
}
.sidebar {
    grid-area: sidebar;
    background: lightgray;
}
.content {
    grid-area: content;
    background: white;
}
.footer {
    grid-area: footer;
    background: gray;
}
```

🟤 2. CSS Flexbox

Flexbox is a **one-dimensional** layout system, perfect for **aligning items in a row or column**.

◇ Basic Flexbox Structure

```css
.container {
    display: flex;
    justify-content: space-between; /* Distributes space between items */
    align-items: center; /* Aligns items vertically */
}
```

◇ Example: Flexbox in Action

```html
<div class="container">
    <div class="box">Box 1</div>
    <div class="box">Box 2</div>
    <div class="box">Box 3</div>
</div>
```

```css
.container {
    display: flex;
    justify-content: center; /* Centers items */
    align-items: center; /* Aligns items vertically */
    gap: 20px;
}

.box {
    background: lightcoral;
    padding: 20px;
    font-size: 18px;
}
```

◇ Important Flexbox Properties

Property	Description
`display: flex;`	Defines a flex container
`flex-direction`	Controls direction of items (`row` / `column`)
`justify-content`	Aligns items horizontally (`center`, `space-between`, etc.)
`align-items`	Aligns items vertically (`center`, `stretch`, etc.)
`flex-wrap`	Allows items to wrap to the next row
`align-self`	Aligns a specific item differently

◇ Flexbox vs Grid

Feature	Grid	Flexbox

Feature	Grid	Flexbox
Dimension	2D (Rows & Columns)	1D (Row OR Column)
Best For	Complex layouts	Simple & dynamic content
Example Use	Page layout, dashboards	Navigation bars, buttons

🎯 Which One to Use?

- **Use Grid** when designing a **full webpage layout** with multiple rows & columns.
- **Use Flexbox** for **smaller components** like navigation bars, cards, and forms.
- **Both can be combined!** (E.g., use Grid for page layout and Flexbox for menu items.)