Day 8 – Flexbox & CSS Grid Deep Dive

The ultimate guide with visuals and code

---

1. Understanding the Difference

Flexbox = 1D layout (arranges items in a row **OR** column)
Grid = 2D layout (arranges items in rows **AND** columns at the same time)

Quick analogy:

- Flexbox = train (carriages in a line) 🚆
- Grid = chessboard (rows & columns) ♟️

---

2. Flexbox – The 1D Layout System

2.1. Basic Setup

```
<div class="flex-container">
  <div class="item">1</div>
  <div class="item">2</div>
  <div class="item">3</div>
</div>
.flex-container {
  display: flex;
  background: #ddd;
  padding: 10px;
}

.item {
  background: lightcoral;
  padding: 20px;
  color: white;
  font-size: 20px;
}
```

2.2. The Main Axis & Cross Axis

```
Main Axis: →   (controlled by justify-content)
Cross Axis: ↓  (controlled by align-items)
```

Diagram:

```
[ item1 ][ item2 ][ item3 ]
   →  justify-content
   ↓  align-items
```

2.3. Flexbox Properties (Container)

Property	What it does	Example
`flex-direction`	Row or column layout	`row`, `column`
`justify-content`	Align along main axis	`center`, `space-between`
`align-items`	Align along cross axis	`flex-start`, `center`
`flex-wrap`	Allow wrapping	`wrap`

Example:

```
.flex-container {
  display: flex;
  flex-direction: row;         /* row or column */
  justify-content: space-evenly; /* spacing */
  align-items: center;         /* vertical alignment */
  flex-wrap: wrap;             /* items go to next line if needed */
}
```

2.4. Flexbox Properties (Items)

Property	Meaning	Example
`flex-grow`	Share of extra space	`flex-grow: 1;`
`flex-shrink`	How much to shrink	`flex-shrink: 0;`
`flex-basis`	Initial size before grow/shrink	`flex-basis: 150px;`

Example:

```
.item {
  flex-grow: 1;    /* take equal space */
  flex-basis: 150px; /* start at 150px wide */
}
```

📌 Flexbox Flow Diagram:

```
Container
  ├── flex-direction
  │     ├── row → horizontal main axis
  │     └── column → vertical main axis
  ├── justify-content (main axis)
  └── align-items (cross axis)
```

---

3. CSS Grid – The 2D Layout System

3.1. Basic Setup

```
<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>
.grid-container {
  display: grid;
  grid-template-columns: 150px 150px;
  grid-template-rows: 100px 100px;
  gap: 10px;
  background: #ddd;
  padding: 10px;
}

.item {
  background: steelblue;
  color: white;
  font-size: 20px;
  display: flex; /* to center text */
  align-items: center;
  justify-content: center;
}
```

3.2. Understanding the Grid

```
+-----+-----+
|  1  |  2  |
+-----+-----+
|  3  |  4  |
+-----+-----+
```

- **Grid tracks** = rows & columns
- **Grid cells** = individual boxes
- **Grid lines** = borders between cells

3.3. Common Grid Properties

Property	What it does	Example
`grid-template-columns`	Defines column widths	`200px 1fr`
`grid-template-rows`	Defines row heights	`100px auto`
`gap`	Space between cells	`gap: 20px;`
`grid-column`	Span multiple columns	`grid-column: 1 / 3;`
`grid-row`	Span multiple rows	`grid-row: 1 / 2;`

3.4. Auto-fit & Auto-fill

```
.grid-container {
  grid-template-columns: repeat(auto-fit, minmax(200px, 1fr));
}
```

- Makes the layout **responsive**
- Columns resize between **200px** and **fill available space**

📌 **CSS Grid Flow Diagram:**

```
Container
  ├── grid-template-columns
  ├── grid-template-rows
  ├── gap
  └── place-items (align + justify)
```

---

4. Flexbox vs Grid Quick Reference

Feature	Flexbox	Grid
Layout	1D (row or column)	2D (rows & columns)
Best For	Navbars, toolbars, small modules	Whole-page layouts, complex designs
Syntax	Easier	More powerful

---

5. Mini Practice Quiz

1. What property in Flexbox controls horizontal spacing between items?
2. In Grid, how do you make an item span 2 columns?
3. Which layout system is better for a chessboard design?
4. Write CSS to make Flexbox items wrap to the next line.