

Modern CSS Layouts

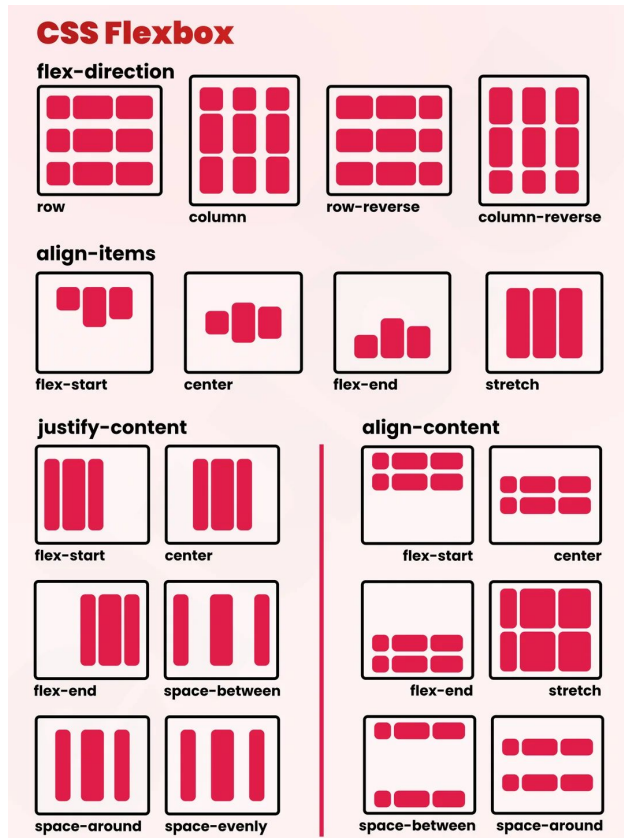
By Naimul Islam Mehedi

Flexbox (1D Layouts)

Flexbox is perfect when you want to align items **in a row OR a column**. Think of it as arranging items like **toys on a shelf (horizontal) or stacked in a box (vertical)**.

Basic Concepts

- **Flex container:** The parent element with `display: flex;`
- **Flex items:** The children elements inside the container.



Example 1: Horizontal Navigation Menu (Flex Row)

```
<nav class="menu">
  <a href="#">Home</a>
  <a href="#">About</a>
  <a href="#">Services</a>
  <a href="#">Contact</a>
</nav>
```

```
.menu {
  display: flex;
  justify-content: center;
  background-color: #333;
  color: white;
  padding: 10px;
  gap: 30px;
}

.menu a {
  color: white;
  text-decoration: none;
}
```

Example 2: Responsive Cards (Flex Wrap)

```
<div class="cards">
  <div class="card">Card 1</div>
  <div class="card">Card 2</div>
  <div class="card">Card 3</div>
  <div class="card">Card 4</div>
</div>
```

```
.cards {
  display: flex;
  flex-wrap: wrap;
  gap: 20px;
  background-color: #aliceblue;
  border-radius: 20px;
}

.card {
  flex: 1 1 200px; /* grow, shrink, base with */
  background: #lightblue;
  text-align: center;
  height: 200px;
}
```

Flexbox Project: What You'll Learn?

- `display: flex;` makes cards sit side by side.
- `flex: 1 1 250px;` makes them **responsive** (minimum 250px).
- `flex-wrap: wrap;` allows them to go to next line on mobile.
- `justify-content: center;` keeps them centered.
- `align-items: stretch;` ensures equal height.

The image displays three pricing plans for a service called 'Flexbox Project'. The 'Basic' and 'Pro' plans are shown in a row at the top, while the 'Enterprise' plan is shown below them. The 'Pro' plan is highlighted with a blue border. Each plan includes a 'Choose Plan' button.


| Plan | Price / month | Projects | Support | Storage |
|------------|---------------|-----------|-----------|---------|
| Basic | \$9 | 5 | Basic | 1 GB |
| Pro | \$29 | 50 | Priority | 50 GB |
| Enterprise | \$99 | Unlimited | Dedicated | 1 TB |



CSS Grid – Grid Container

Before anything else, you must know:

- To use grid, the parent element (container) must have `display: grid;`
- Once you do this, the children (grid items) will automatically follow grid rules.

```
<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;
  background:  lightgray;
  padding: 10px;
}

.item {
  background:  tomato;
  color:  white;
  padding: 20px;
  margin: 5px;
  text-align: center;
}
```

Defining Columns (grid-template-columns)

Now let's learn how to **create columns** inside the grid.

This is one of the most powerful features of CSS Grid.

Property: **grid-template-columns**

- Defines **how many columns** your grid should have
- You can set:
 - Fixed values (200px 200px 200px)
 - Percentages (50% 50%)
 - Flexible fractions (1fr 2fr)

```
.grid-container {
  display: grid;
  grid-template-columns: 200px 200px;
  gap: 10px;
  background-color: lightgray;
  padding: 10px;
}

.item {
  background: tomato;
  color: white;
  padding: 20px;
  margin: 5px;
  text-align: center;
}
```

CSS Grid – : Defining Rows (grid-template-rows)

Now, just like columns, we can control **row heights**.

Property: **grid-template-rows**

- Defines how tall each row should be.
- Accepts fixed values (**100px 200px**), percentages, **fr**, or **auto**.

```
.grid-container {  
  display: grid;  
  grid-template-columns: 1fr 2fr;  
  gap: 10px;  
  background-color: lightgray;  
  padding: 10px;  
  grid-template-rows: 100px 200px;  
}  
  
.item {  
  background-color: orange;  
  color: white;  
  text-align: center;  
  font-size: 20px;  
  display: flex;  
  justify-content: center;  
  align-items: center;  
}
```


CSS Grid – Gaps (row-gap, column-gap, gap)

Just like in Flexbox, we can add spacing between grid items without using margins.

Properties

- **row-gap**: space between rows
- **column-gap**: space between columns
- **gap**: shorthand (sets both at once)

```
.grid-container {  
  display: grid;  
  grid-template-columns: 1fr 2fr;  
  background-color: lightgray;  
  padding: 10px;  
  grid-template-rows: auto auto;  
  
  row-gap: 20px;  
  column-gap: 40px;  
  padding: 10px;  
}  
  
.item {  
  background-color: orange;  
  color: white;  
  text-align: center;  
  padding: 20px;  
}
```

CSS Grid – Spanning Columns & Rows

Sometimes, you want an item to take **more than one column or row** (like a banner spanning across the page or a sidebar stretching multiple rows).

👉 For that, we use:

- `grid-column` → controls horizontal span
- `grid-row` → controls vertical span

```
<div class="grid-container">
  <div class="item header">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: 1fr 2fr;
  background-color: lightgray;
  gap: 10px;
}

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

.header {
  grid-column: 1 / 3;
}
```

CSS Grid – Spanning Columns & Rows

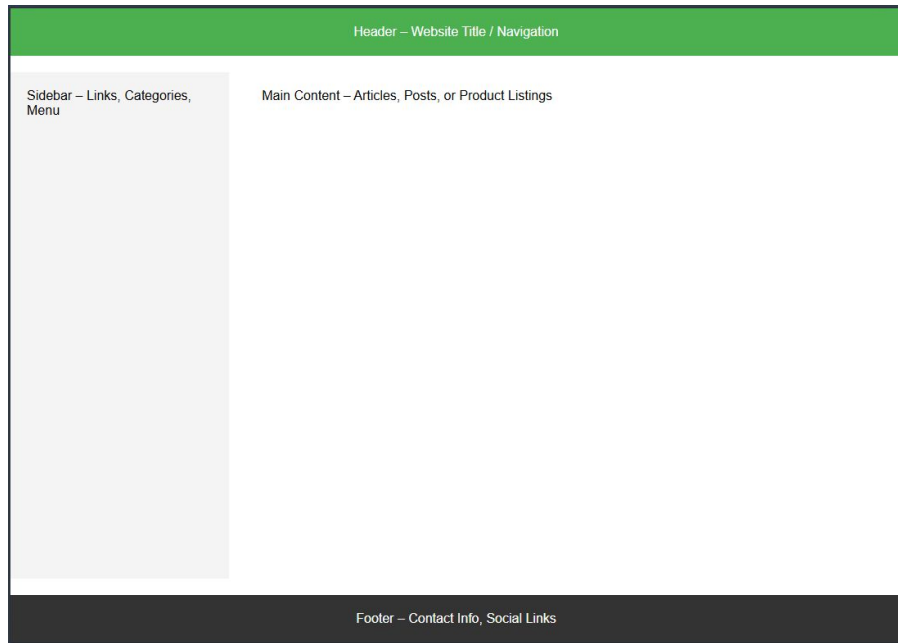
👉 The **Header** will stretch across both columns.

```
.grid-container {  
  display: grid;  
  grid-template-columns: 1fr 2fr;  
  background-color: lightgray;  
  gap: 10px;  
}  
  
.item {  
  background-color: orange;  
  color: white;  
  text-align: center;  
  padding: 20px;  
}  
  
.header {  
  grid-column: 1 / 3;  
}  
  
.item:nth-child(2) {  
  grid-row: 2 / 4; /* takes row 2 and 3 */  
}
```

CSS Grid – Grid Template Areas

This is one of the **most powerful and readable ways** to define layouts.

Instead of using numbers
(`grid-column: 1 / 3;`), you can
name areas and arrange them like a
blueprint.



```
<body>
  <div class="container">
    <header>
      <p>Header - Website Title / Navigation</p>
    </header>
    <aside>
      <p>Sidebar - Links, Categories, Menu</p>
    </aside>
    <main>
      <p>Main Content - Articles, Posts, or
      Product Listings</p>
    </main>
    <footer>
      <p>Footer - Contact Info, Social Links</p>
    </footer>
  </div>
</body>
```

```
body {
  margin: 0;
  font-family: Arial, Helvetica, sans-serif;
}

.container {
  display: grid;
  grid-template-areas:
    "header header"
    "sidebar main"
    "footer footer"
  ;
  grid-template-columns: 1fr 3fr;
  grid-template-rows: auto 1fr auto;
  gap: 20px;
  height: 100vh;
}
```

```
header {  
  grid-area: header;  
  background: #4CAF50;  
  color: white;  
  padding: 20px;  
  text-align: center;  
}  
  
aside {  
  grid-area: sidebar;  
  background: #f4f4f4;  
  padding: 20px;  
}
```

```
main {  
  grid-area: main;  
  background: #fff;  
  padding: 20px;  
}  
  
footer {  
  grid-area: footer;  
  background: #333;  
  color: white;  
  padding: 20px;  
  text-align: center;  
}
```