CSS Grid

CSS Grid Layout is a **two-dimensional** layout system that allows you to design web pages in **rows and columns**. It gives you complete control over both axes, unlike Flexbox which is mostly one-dimensional.

Enabling Grid

Set the container's `display` property to `grid`:

```
.container {
  display: grid;
}
```

All direct children of this container become **grid items**.

Defining Rows and Columns

You use `grid-template-columns` and `grid-template-rows` to define the grid structure:

```
.container {
  display: grid;
  grid-template-columns: 200px 1fr 1fr;
  grid-template-rows: 100px auto;
}
```

- `1fr` means "1 fraction of remaining space"
- You can mix fixed units (e.g. `px`) with flexible ones (`fr`)

Repeat Syntax

To avoid repeating values:

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

Creates 3 equal-width columns.

Grid Gap

Adds spacing between rows and columns:

```
.container {
  gap: 20px; /* shorthand for row-gap and column-gap */
}
```

Placing Items

You can control where an item appears in the grid using `grid-column` and `grid-row`.

```
.item {
  grid-column: 1 / 3; /* spans column 1 to 2 (exclusive of 3) */
  grid-row: 2 / 3;
}
```

You can also use `span`:

```css
.item {
  grid-column: span 2;
}
```

Named Areas (Optional but Powerful)

Define areas using `grid-template-areas`:

```css
.container {
  display: grid;
  grid-template-areas:
    "header header"
    "sidebar content"
    "footer footer";
  grid-template-columns: 1fr 3fr;
  grid-template-rows: auto 1fr auto;
}
```

Then assign each item:

```css
.header { grid-area: header; }
.sidebar { grid-area: sidebar; }
.content { grid-area: content; }
.footer { grid-area: footer; }
```

Auto-Placement

Grid can automatically place items:

```css
.container {
  display: grid;
```

```
  grid-template-columns: repeat(auto-fill, minmax(150px, 1fr));
}
```

This makes the layout responsive, automatically filling space with flexible-width items.

Complete Example

```
<div class="container">
  <div class="item header">Header</div>
  <div class="item sidebar">Sidebar</div>
  <div class="item content">Content</div>
  <div class="item footer">Footer</div>
</div>
```

```
.container {
  display: grid;
  grid-template-areas:
    "header header"
    "sidebar content"
    "footer footer";
  grid-template-columns: 1fr 3fr;
  grid-template-rows: auto 1fr auto;
  gap: 10px;
}

.header  { grid-area: header; background: #ddd; }
.sidebar { grid-area: sidebar; background: #bbb; }
.content { grid-area: content; background: #eee; }
.footer  { grid-area: footer; background: #ccc; }

.item {
  padding: 20px;
}
```

Summary

- CSS Grid is **perfect for page layouts** with rows and columns.
- `grid-template-columns` and `grid-template-rows` define structure.
- Use `grid-column` and `grid-row` to place or span items.
- Named grid areas make your layout more readable and semantic.
- Auto-fill and auto-fit allow responsive grids.