🚀 Say Goodbye to Layout Struggles: CSS Grid Basics

Stop hacking your layouts.

CSS Grid is the most powerful layout system in web design. It allows you to create complex, responsive 2D layouts with ease.

Ready to level up your CSS?

Grid vs. Flexbox: The Difference

Flexbox is 1-Dimensional:

Good for items in a single row OR a single column.

Grid is 2-Dimensional:

Good for items in rows AND columns simultaneously.

Rule: Use Grid for the page structure; use Flexbox for the components inside it.

Step 1: Defining the Grid

Turn any element into a grid container. Define columns with 'grid-template-columns'.

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

The Magic Unit: 'fr'

The 'fr' unit represents a fraction of available space. It calculates perfectly with gaps, unlike percentages.

```
/* Creates 3 equal columns
*/
grid-template-columns: 1fr
1fr 1fr;
```

Rows and Gaps

Define row heights and the spacing between cells easily without margins.

```
grid-template-rows: 100px
auto;
gap: 20px; /* Space between
all cells */
```

Mid-Guide Checkpoint

You have defined the Skeleton (The Container).

The next slides show you how to control the Items (The Children) to make them span across multiple columns or rows.

Placing Items: Line Numbers

Grid lines start at 1. You can tell an item to span across multiple lines.

```
.item-1 {
  /* Start at col line 1,
end at 3 */
  grid-column: 1 / 3;
}
```

The 'Cheat Mode': Grid Areas

You can name your grid sections and arrange them visually in your CSS string!

```
grid-template-areas:
  "header header"
  "sidebar content"
  "footer footer";
```

Responsive Power: repeat()

The ultimate responsive layout one-liner. It creates as many columns as will fit!

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