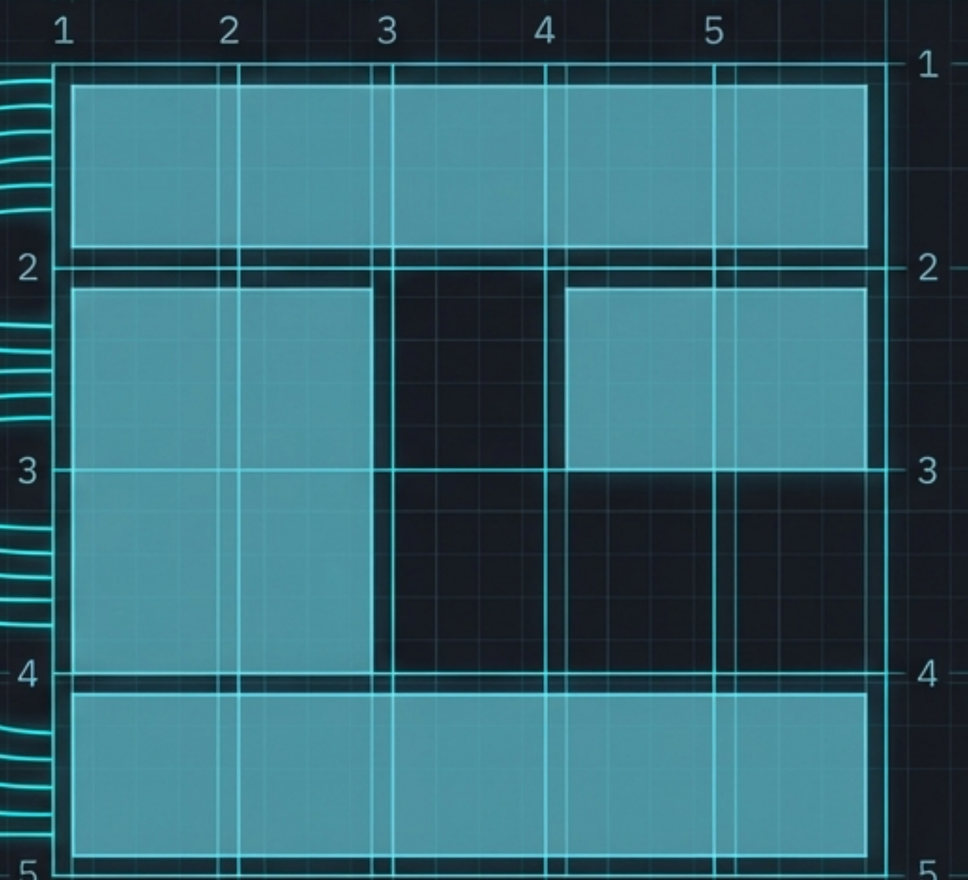
Named Areas

Blueprints, Not Coordinates

Line numbers are precise. But they can also get... gross.

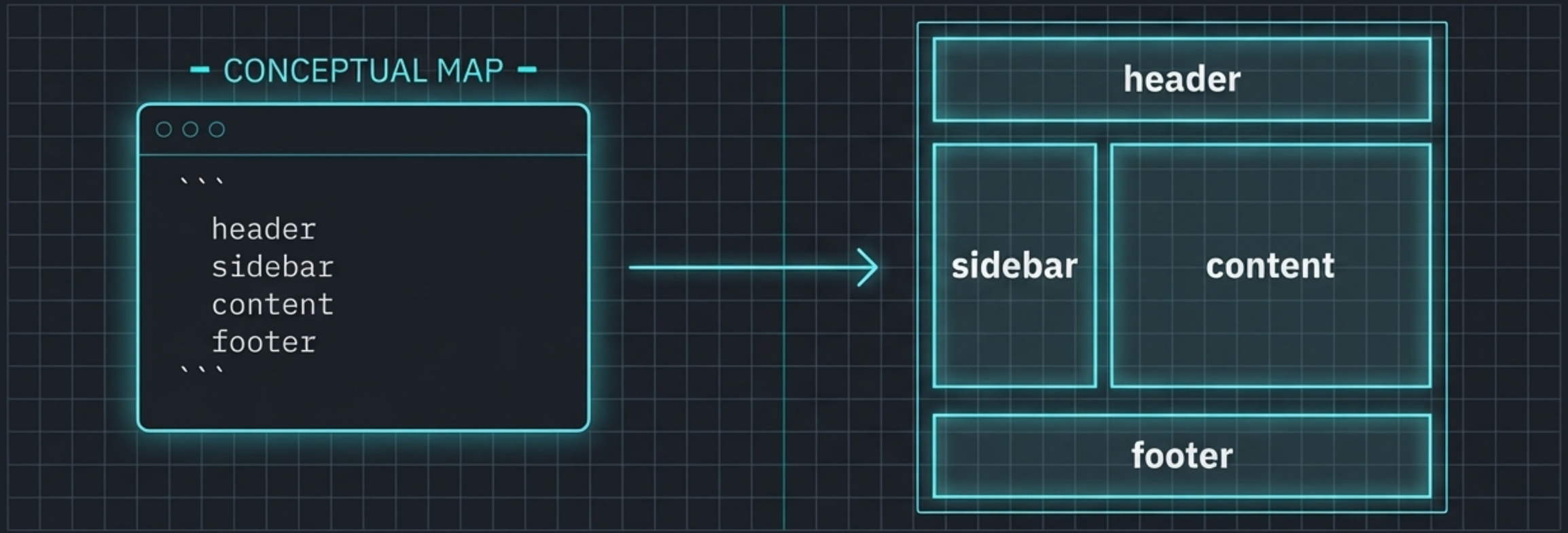
Using line numbers to place grid items works, but it can be difficult to read and even harder to change. The layout's intent is hidden in a series of coordinates.

```
.header {
  grid-column: 1 / 5;
  grid-row: 1 / 2;
}
.item-a {
  grid-column: 1 / 3;
  grid-row: 2 / 4;
}
.item-b {
  grid-column: 3 / 5;
  grid-row: 2 / 3;
}
.footer {
  grid-column: 1 / 5;
  grid-row: 4 / 5;
}
```



A layout described as a map.

`grid-template-areas` lets you design a layout visually, right in your CSS. It’s for creating readable layout intent.



Step 1: Define Areas on the Container

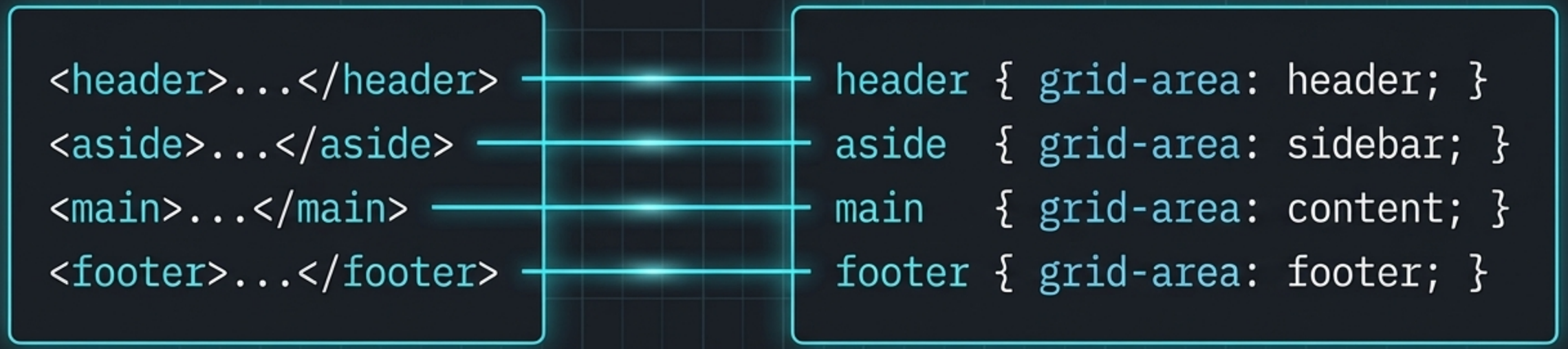
On the grid container, you use the `grid-template-areas` property. Each string defines a row, and the names you use create the structure of your layout map.

```
.layout {
  display: grid;
  grid-template-columns: 14rem 1fr;
  grid-template-rows: auto 1fr auto;
  grid-template-areas:
    "header   header"
    "sidebar  content"
    "footer   footer";
}
```

Row 1
Row 2
Row 3

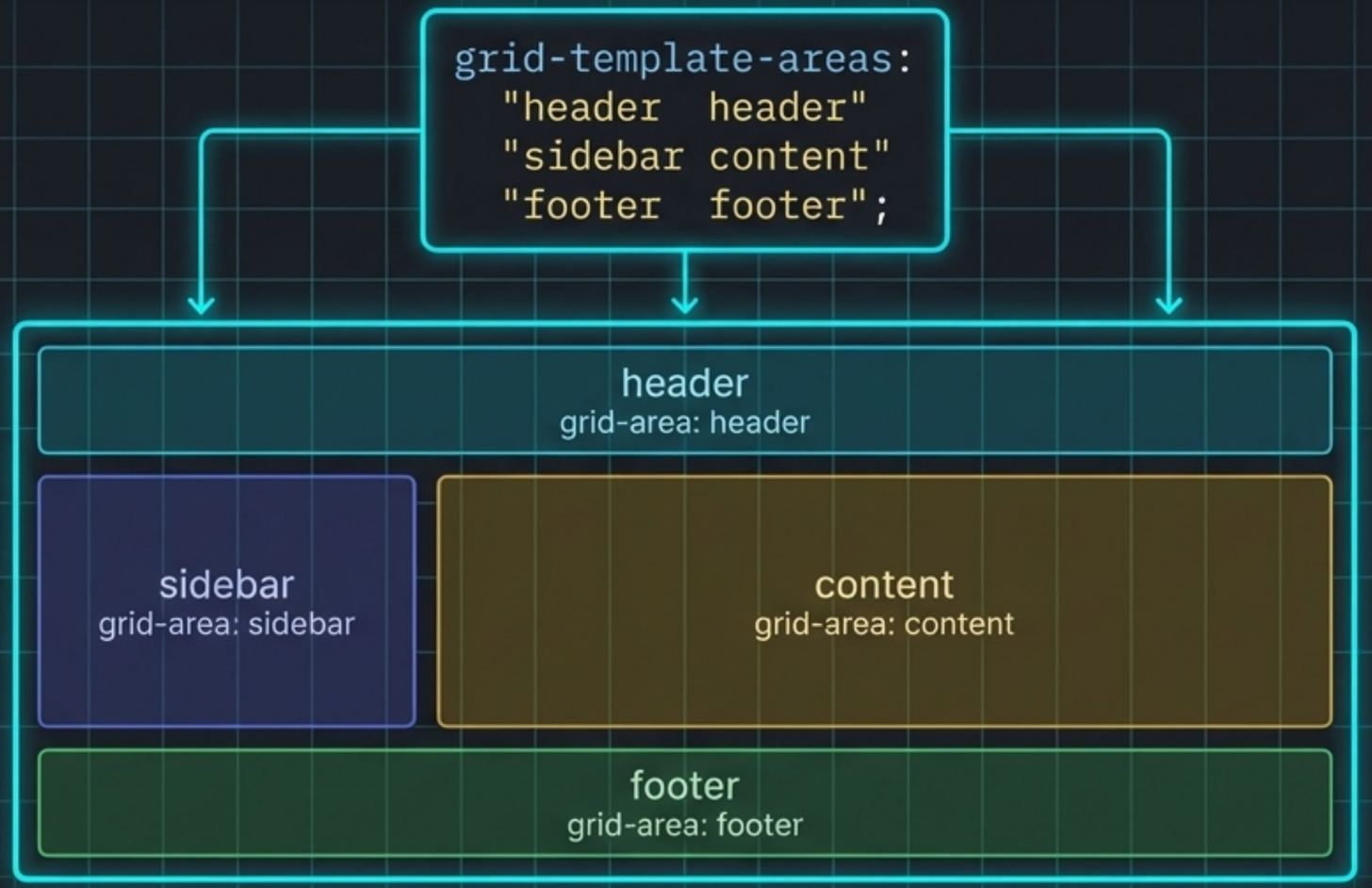
Step 2: Assign Items to Areas

With the blueprint defined, you assign each grid item to a named area using the `grid-area` property. Now your HTML can stay semantic, and the grid does the layout work.



The Blueprint and the Blocks

You’re not “placing items”. You’re declaring a blueprint — then assigning blocks to named areas.

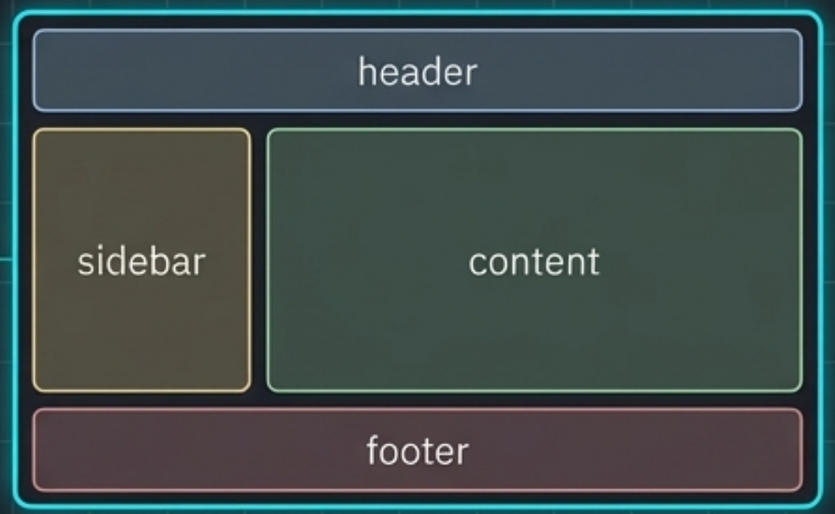


Dots Mean “Empty”

You don’t have to use every cell in your grid. A period (.) in the blueprint acts as a placeholder for an empty cell.

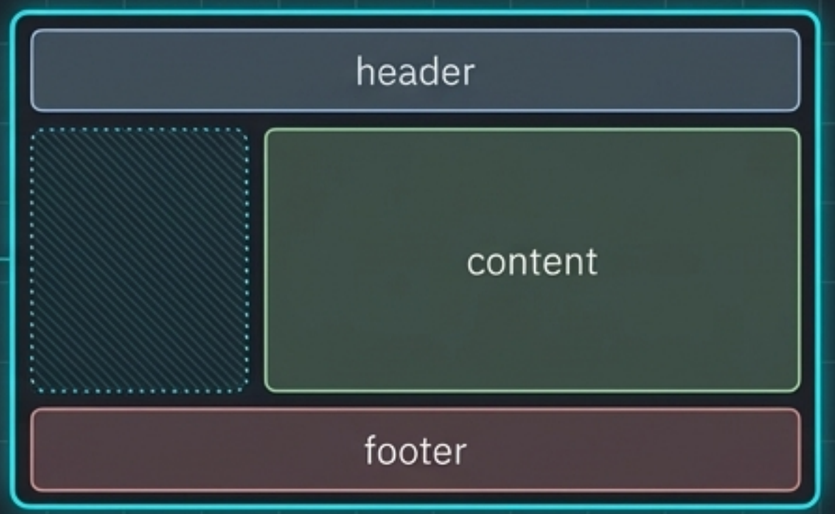
Before

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



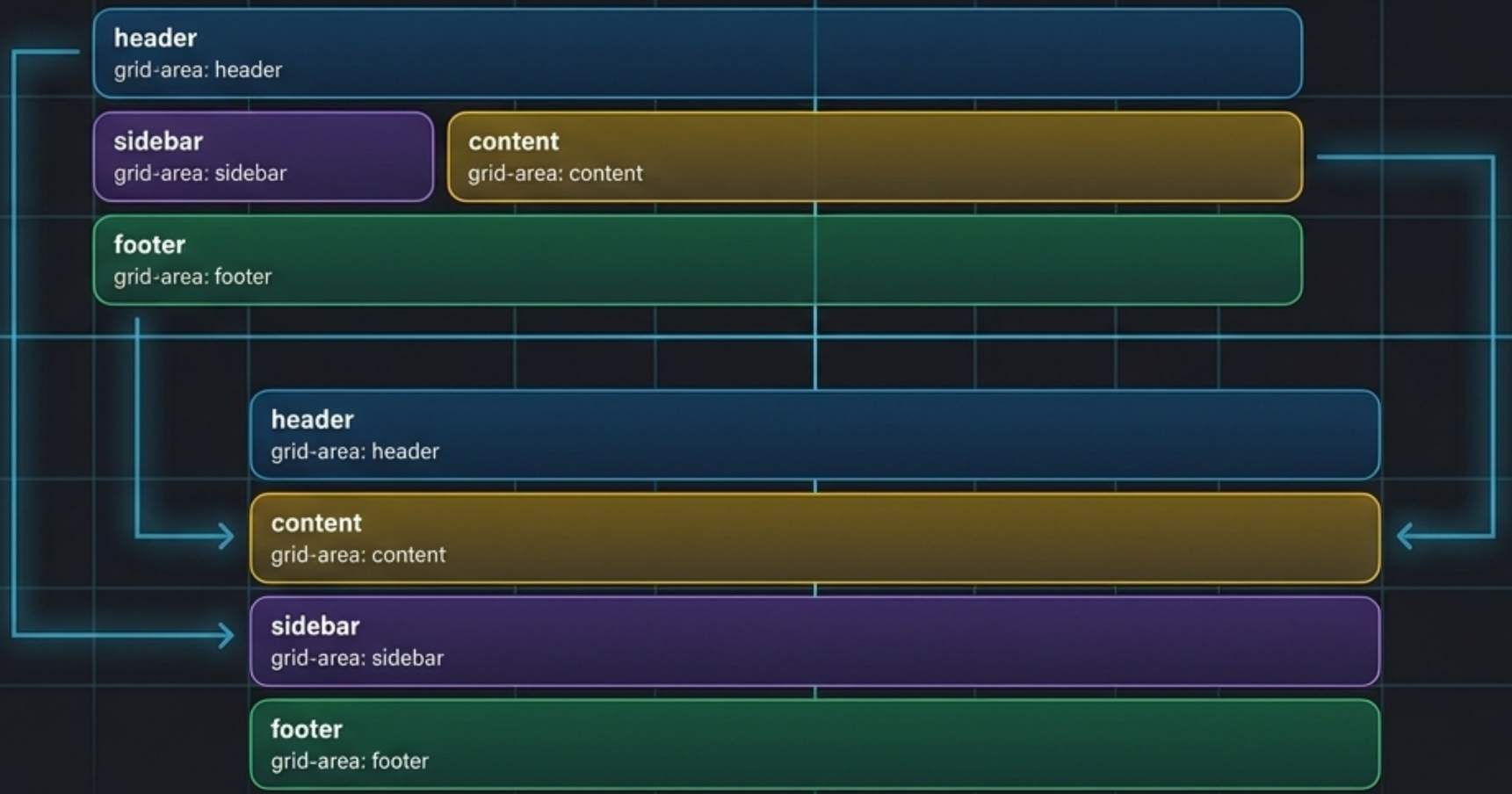
After

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



Swap blueprints. Watch the layout rewire itself.

“Same HTML. Different maps.”



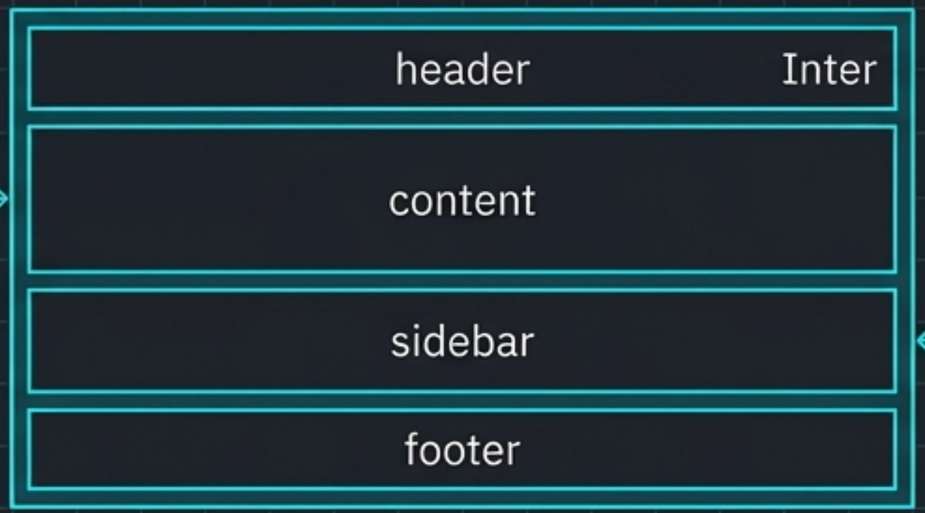
The Code Behind the Change

Layout: classic

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

Layout: hero

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



The Takeaway: Areas are the readable option.

- Great for page layout

- Easy to refactor

- Less fragile than line-number spaghetti

Readable layouts refactor better.

p.s., keep learning!