LECTURE 9

Flexbox and Grid Layout

The two modern CSS layout systems — one-dimensional alignment, and two-dimensional page structure.

CSC336 Web Technologies

OVERVIEW

In This Lecture

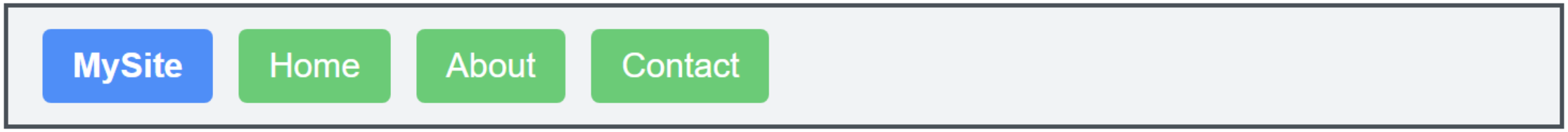
1. Flexbox: container, items, and the two axes
2. flex-direction, justify-content, align-items, flex-wrap, flex-grow/shrink/basis
3. CSS Grid: rows, columns, and the fr unit
4. Grid line placement, spanning multiple cells, and named grid areas
5. Choosing Flexbox vs. Grid — and using both together

Container and Items

```
.nav { display: flex; }
```

One declaration turns every direct child of .nav into a flex item, laid out in a single row by default.

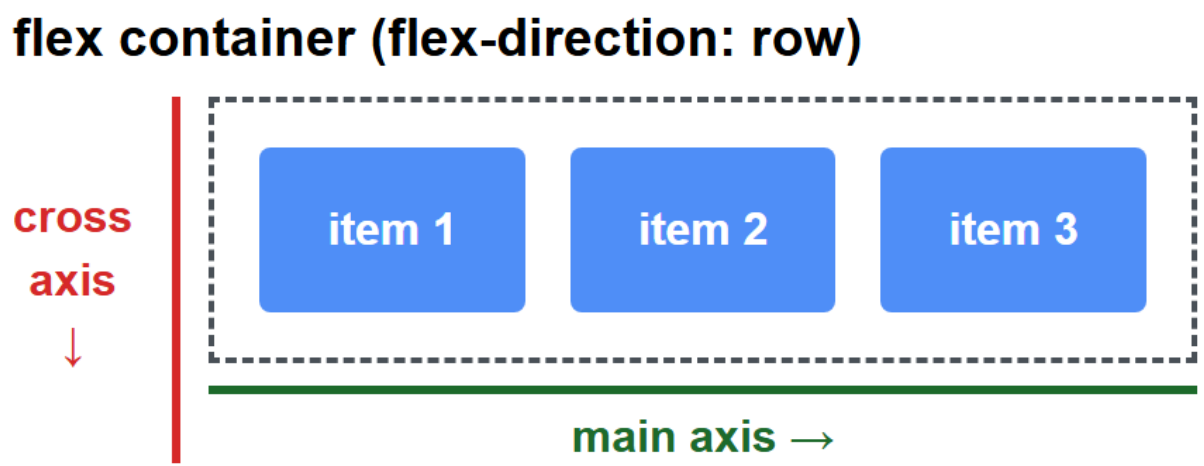
A Flex Nav Bar, Rendered



The Main Axis and the Cross Axis

The main axis runs in the flex direction (row, by default); the cross axis runs perpendicular to it.

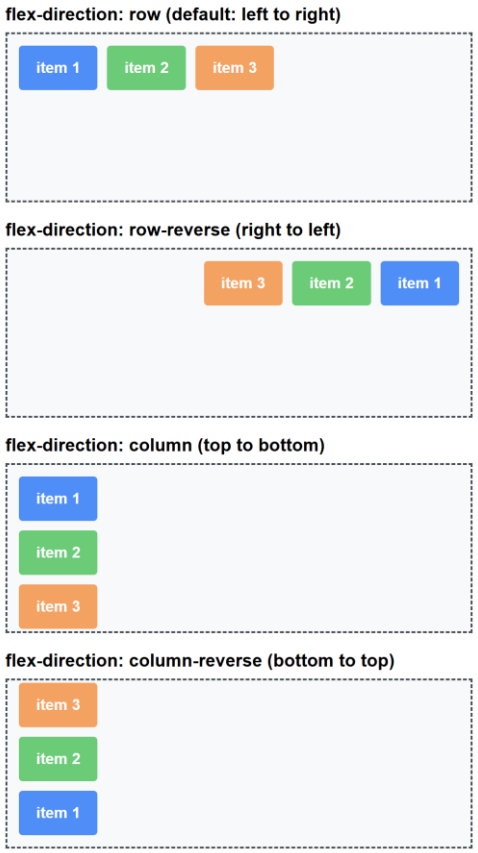
```
.flex-row {
  display: flex; /* main axis: horizontal, left to right */
}
```



flex-direction

```
.container { flex-direction: row | row-reverse | column | column-reverse; }
```

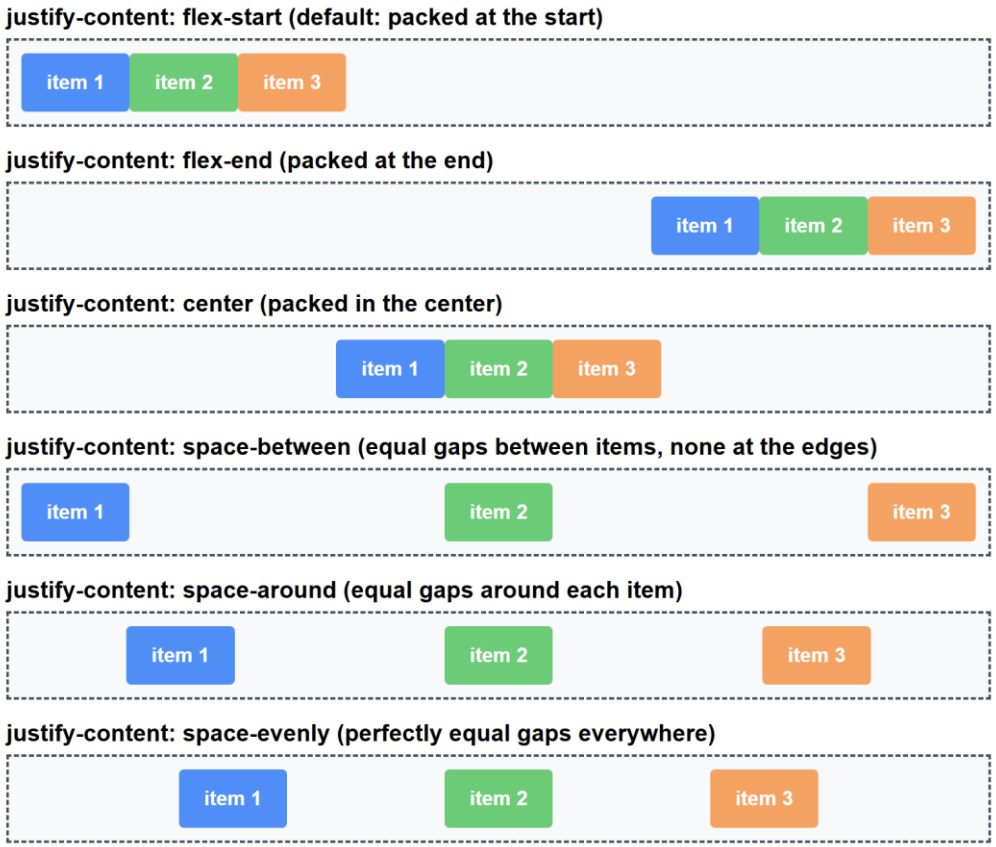
flex-direction, Compared



justify-content (Main Axis)

```
.container { justify-content: flex-start | flex-end | center |
                              space-between | space-around | space-evenly; }
```

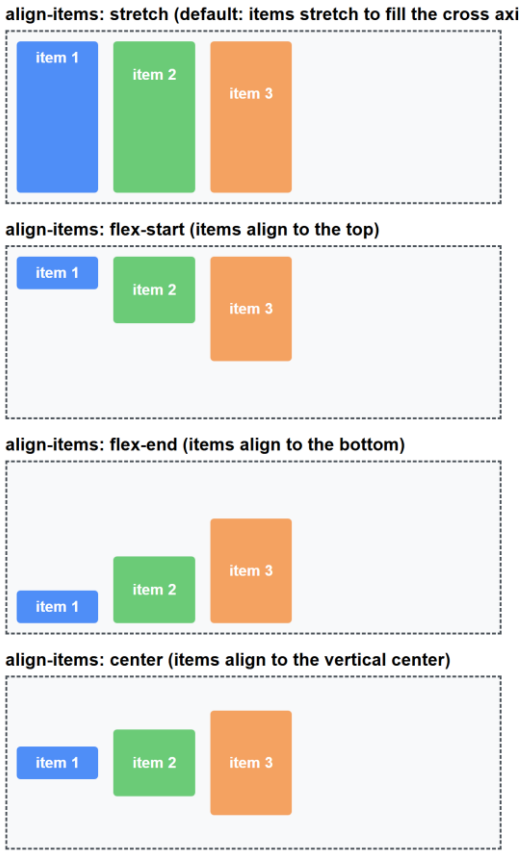
justify-content, Compared



align-items (Cross Axis)

```
.container { align-items: stretch | flex-start | flex-end | center; }
```

align-items, Compared



A FAMOUS TRICK

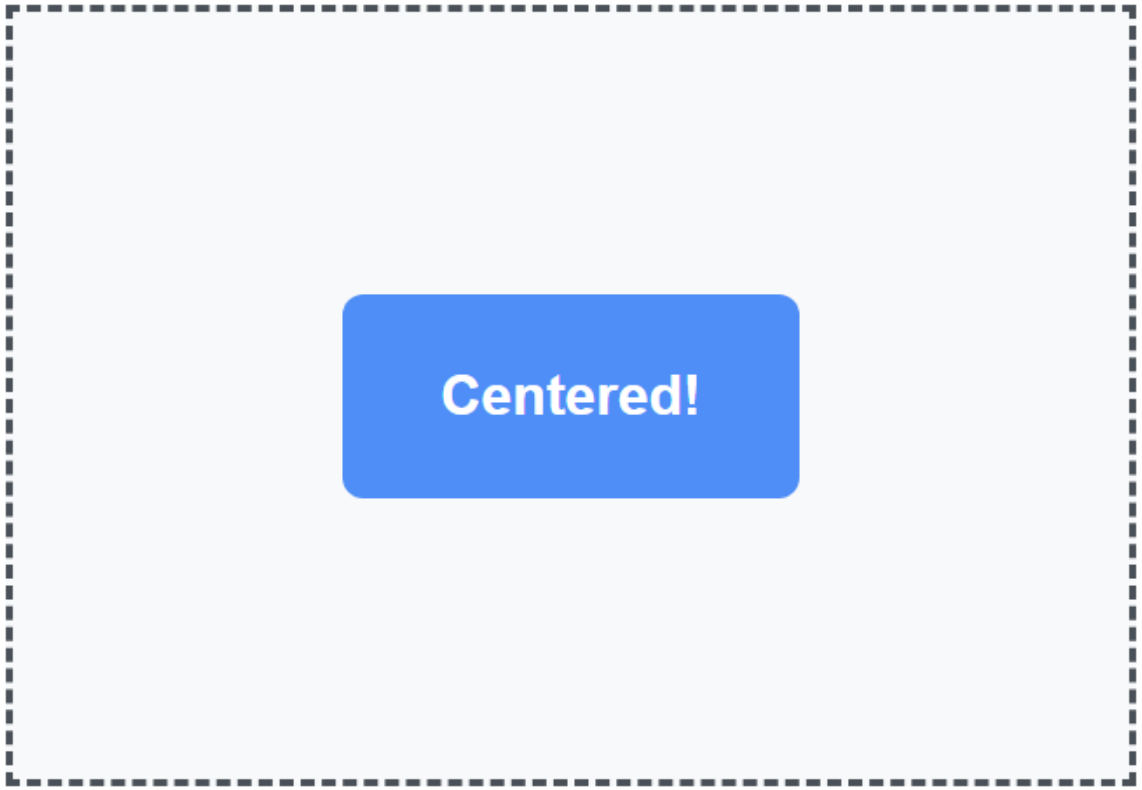
Perfect Centering

```
.container {
  display: flex;
  justify-content: center;
  align-items: center;
}
```

The classic answer to "how do I center a div" — both axes at once.

A FAMOUS TRICK

Centering, Rendered

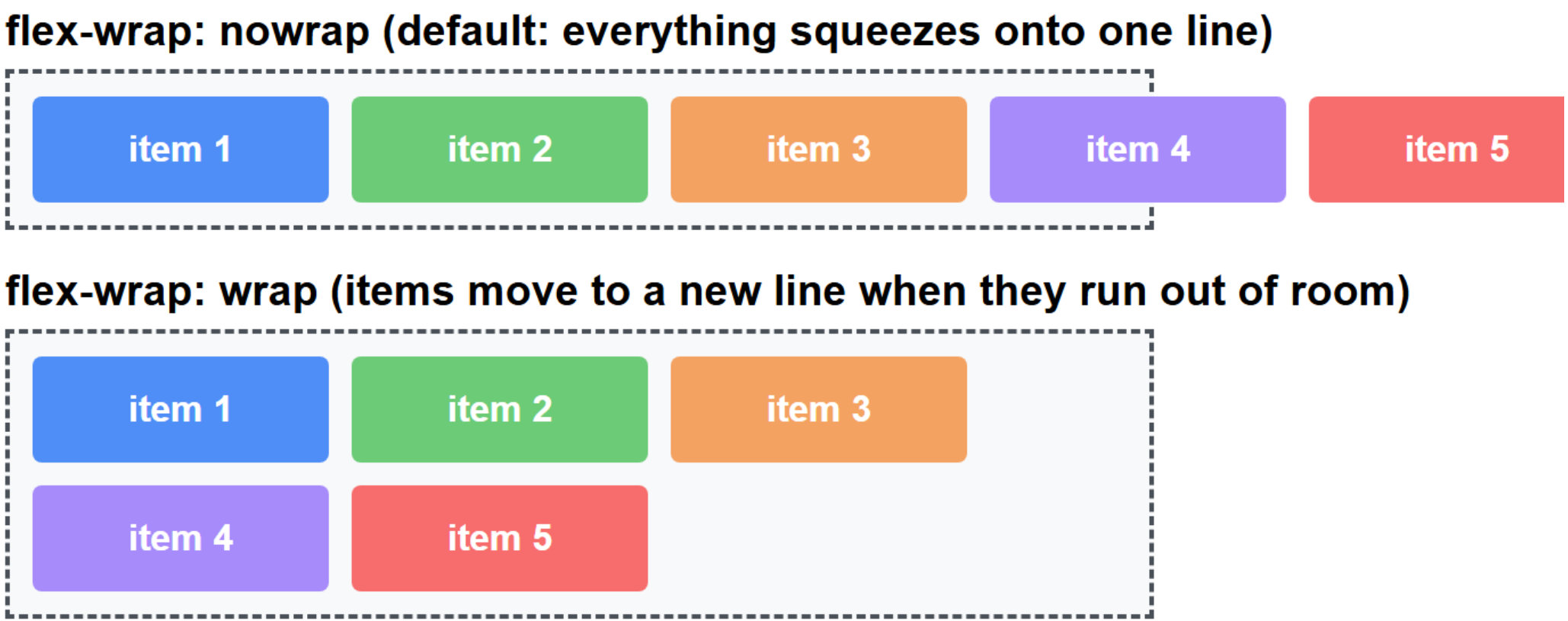


flex-wrap

```
.container { flex-wrap: nowrap | wrap; }
```

nowrap squeezes everything onto one line, even past the container's edge; wrap breaks items onto multiple lines.

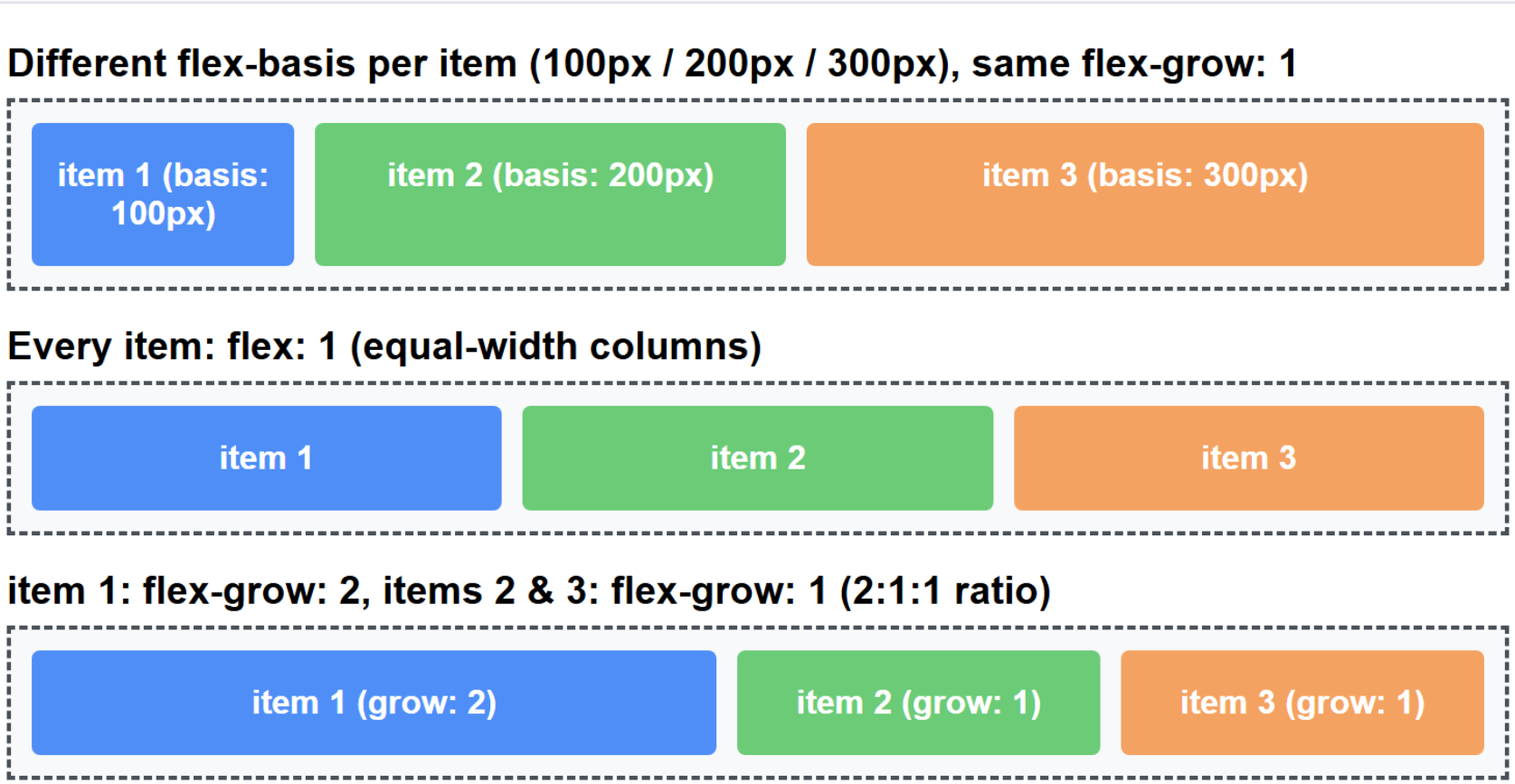
flex-wrap, Compared



flex-grow, flex-shrink, flex-basis

```
.item { flex-basis: 200px; }   /* starting size before growing/shrinking */
.item { flex: 1; }              /* grow to fill equally */
.item { flex-grow: 2; }         /* grow twice as fast as flex-grow: 1 */
```

Growing and Shrinking, Rendered



Rows and Columns

```
.grid {
  display: grid;
  grid-template-columns: 200px 200px 200px;
  grid-template-rows: 150px 150px;
  gap: 10px;
}
```

A Basic Grid, Rendered

The fr Unit

```
grid-template-columns: 1fr 1fr 1fr;      /* three equal columns */
grid-template-columns: 2fr 1fr;          /* left column twice as wide */
grid-template-columns: 250px 1fr;        /* fixed sidebar + flexible main */
```

fr distributes remaining space as a FRACTION — it's Grid's equivalent of flex-grow.

The fr Unit, Compared

grid-template-columns: 1fr 1fr 1fr (three equal-width columns)

grid-template-columns: 2fr 1fr (first column is twice as wide as the second)

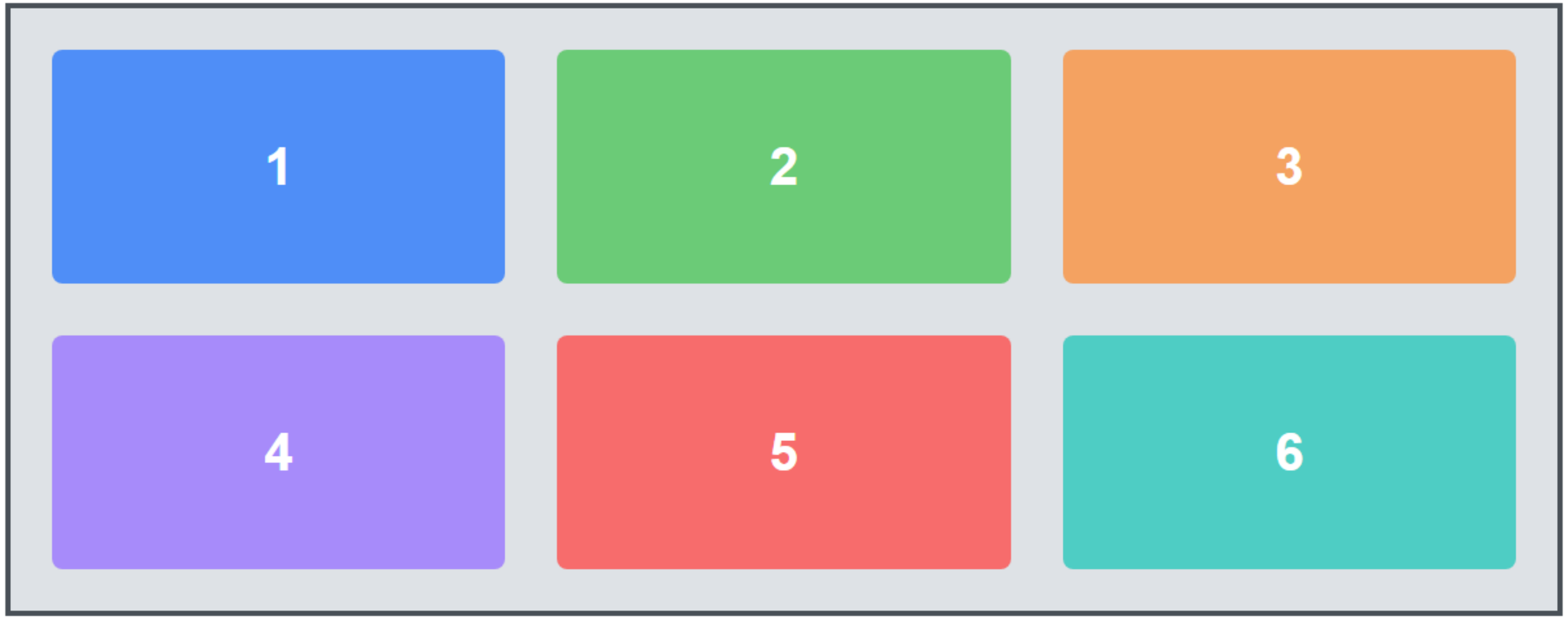
grid-template-columns: 250px 1fr (a fixed sidebar + a flexible main area)

gap

```
.grid {
  display: grid;
  grid-template-columns: repeat(3, 1fr);
  gap: 20px;
}
```

gap replaces the old margin-hack for spacing grid (and flex) items evenly, without extra space at the edges.

gap, Rendered

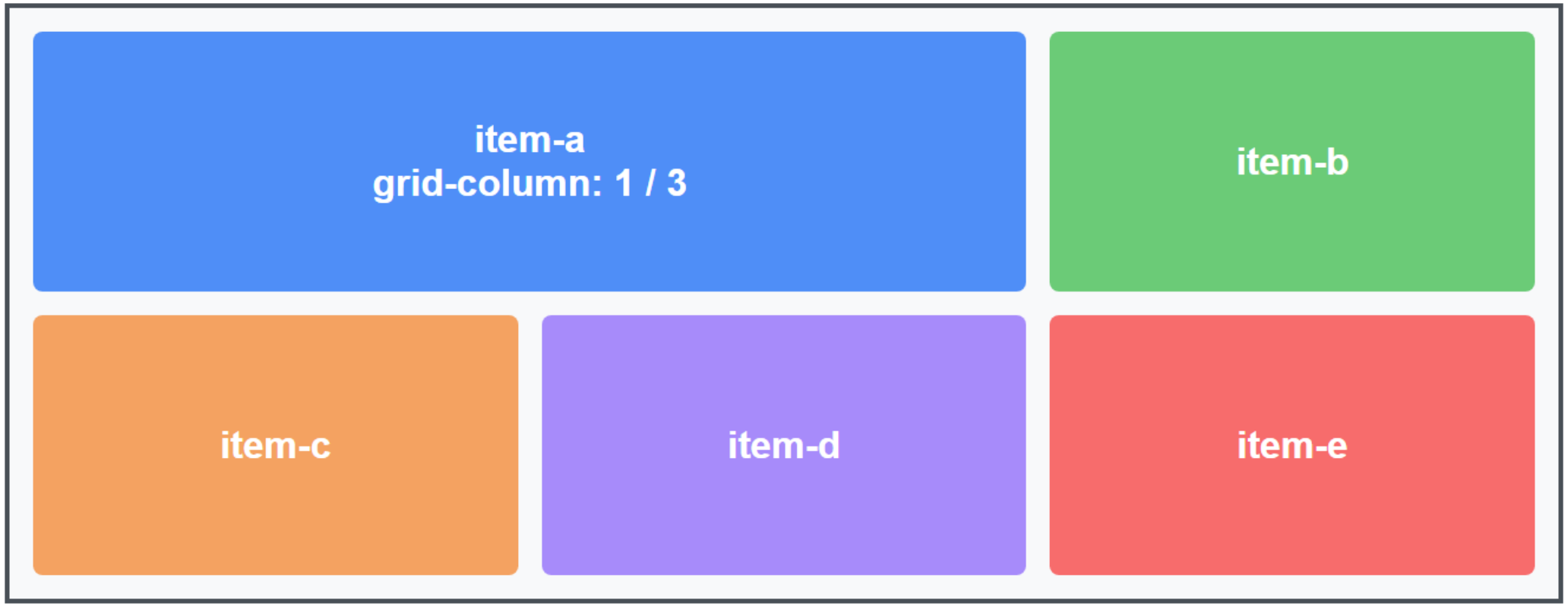


Placing Items on Grid Lines

```
.item-a { grid-column: 1 / 3; grid-row: 1; }
.item-b { grid-column: 3; grid-row: 1; }
```

Grid lines are numbered starting at 1 — an item can be placed at exact, specific cells.

Line Placement, Rendered

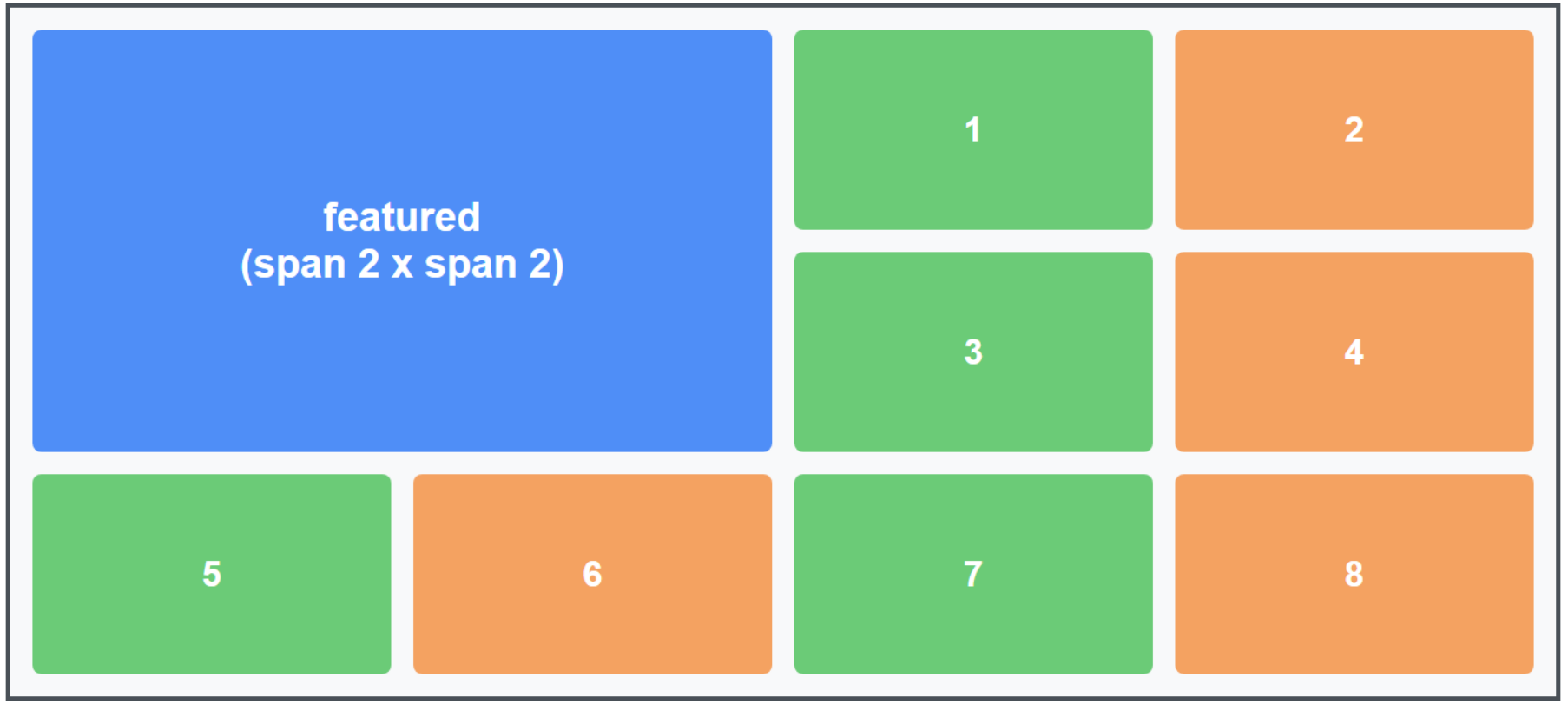


Spanning Multiple Cells

```
.featured {
  grid-column: span 2;
  grid-row: span 2;
}
```

A photo-gallery-style layout: one featured cell occupies a 2x2 block while others fill in around it.

Spanning, Rendered

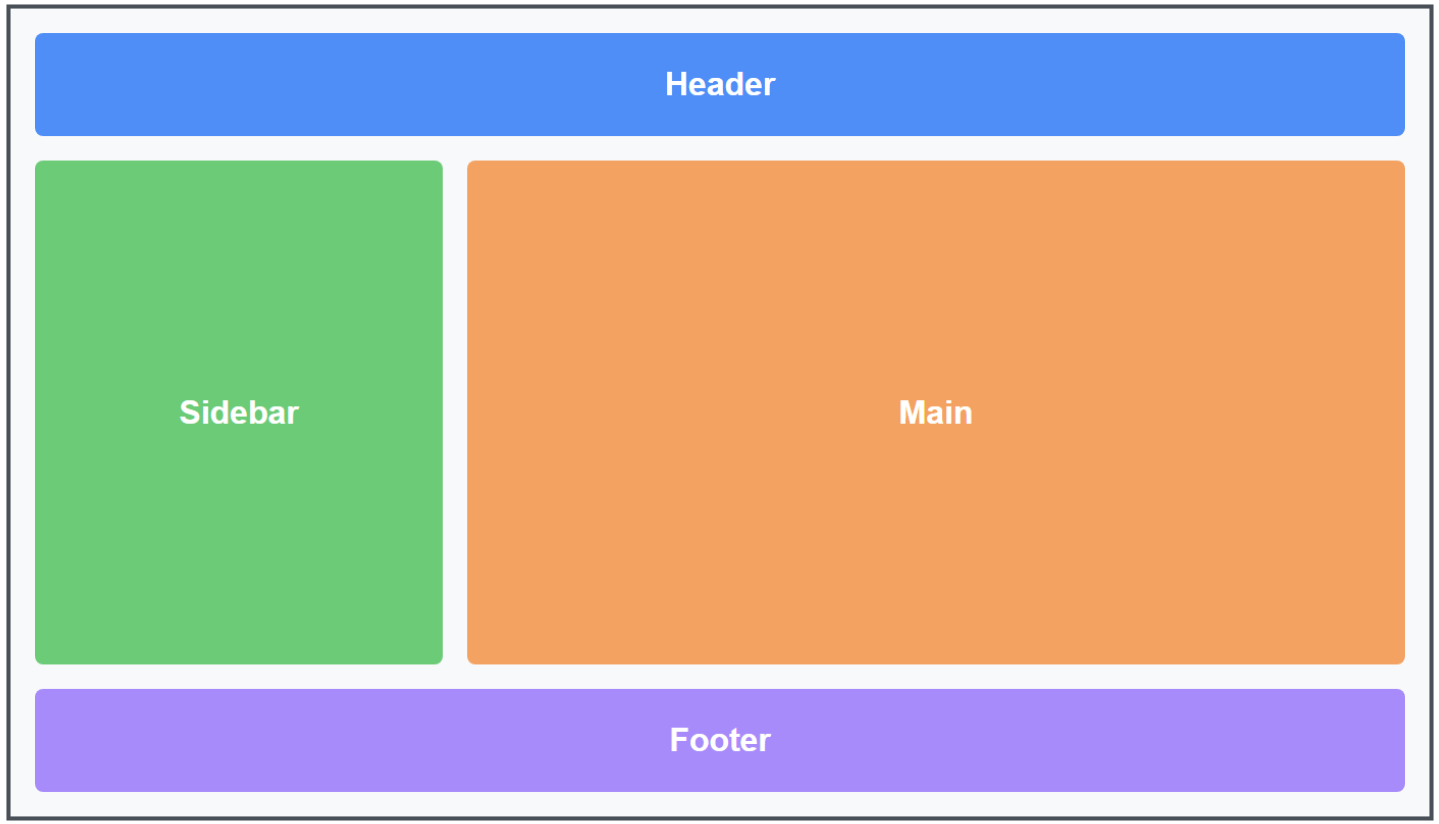


Named Grid Areas

```
.page {
  grid-template-areas:
    "header header"
    "sidebar main"
    "footer footer";
}
.header { grid-area: header; }
```

Naming areas makes a whole-page layout genuinely readable straight from the CSS.

Named Areas, Rendered



DECISION TIME

Choosing Flexbox vs. Grid

	Flexbox	Grid
Dimensions	One-dimensional (a row OR a column)	Two-dimensional (rows AND columns)
Best for	Nav bars, button groups, aligning items	Whole-page layouts, photo galleries
Item placement	Items flow in order	Items placed at exact grid cells

DECISION TIME

The Simple Rule of Thumb

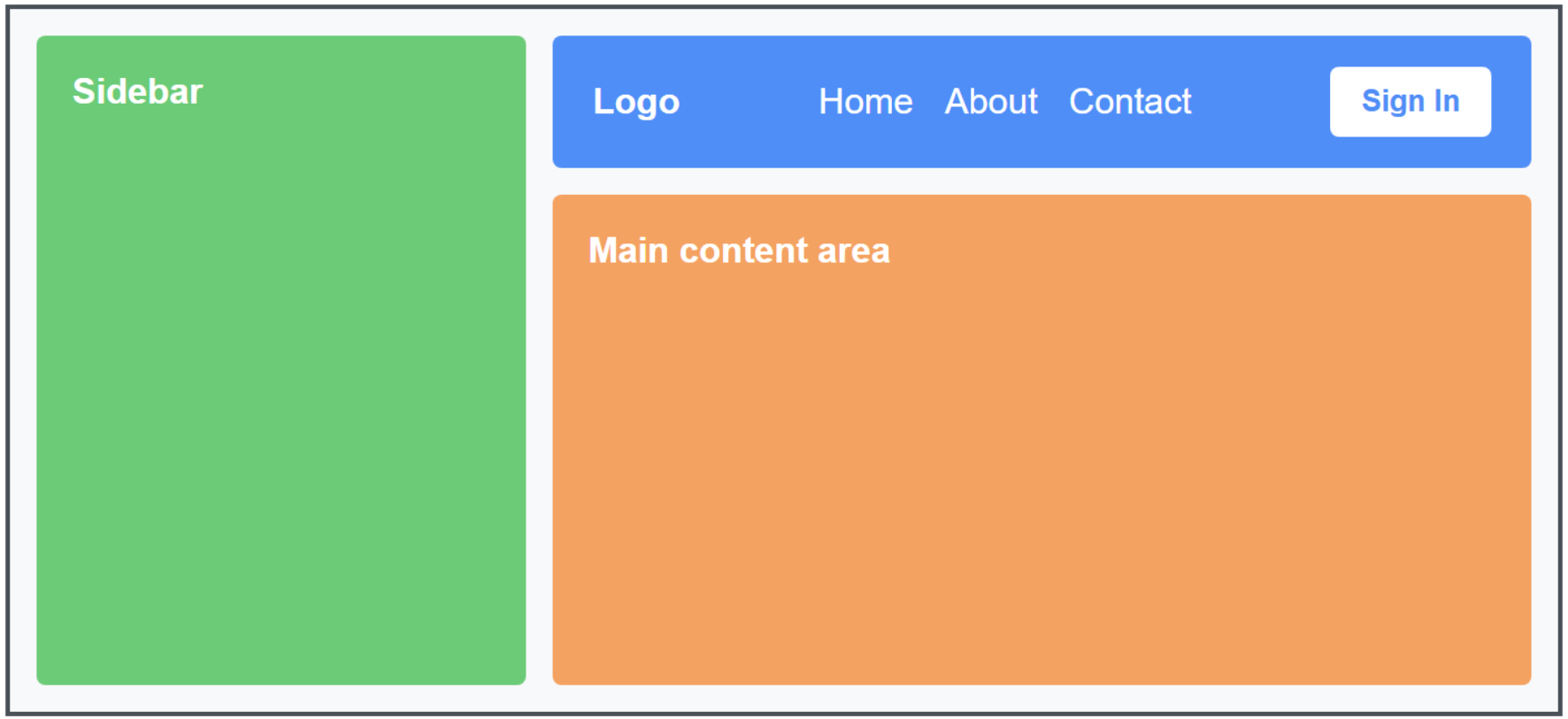
TIP

Arranging things in ONE direction (a row of buttons, a horizontal menu)? Reach for Flexbox. Need to control rows AND columns together (an overall page layout, a gallery)? Reach for Grid. In real projects you'll use both together constantly.

Grid for Skeleton, Flexbox Inside It

```
.page {
  display: grid;
  grid-template-columns: 220px 1fr;
  grid-template-areas: "sidebar main";
}
.header {
  display: flex;
  justify-content: space-between;
  align-items: center;
}
```

Grid + Flexbox, Rendered



KEY TAKEAWAYS

Lecture 9 in Six Points

1. Flexbox lays out items along a main axis and a cross axis, one dimension at a time.
2. justify-content aligns along the main axis; align-items aligns along the cross axis — together they center anything.
3. flex-grow/shrink/basis control how items expand, shrink, and start sized within the available space.
4. CSS Grid lays out rows AND columns together; the fr unit distributes remaining space as flexible fractions.
5. Grid items can be placed on exact lines, span multiple cells, or use named grid-template-areas for a readable layout.
6. Use Flexbox for one-dimensional alignment; use Grid for the overall page skeleton — and combine both in real projects.