# Flexbox & Grid Cheat Sheet

The properties, patterns, and gotchas that cover 95% of real layout work — with copy-paste examples

mirzaa.dev
Free resource — share freely

## FLEXBOX — THE MENTAL MODEL

- Flexbox is **one-dimensional**: it lays items out along a single axis (a row *or* a column)
- The **main axis** follows `flex-direction`; the **cross axis** runs perpendicular to it
- `justify-*` = main axis · `align-*` = cross axis — this one rule decodes every property name
- In `flex-direction: column`, `justify-content` is *vertical* — the axes travel with the direction
- Content-out sizing: items size to their content first, then grow/shrink to fit the container
- Use flex when the *content* should drive the layout; use grid when the *layout* should drive the content

## CONTAINER PROPERTIES

- `display: flex` — children become flex items (`inline-flex` for inline-level)
- `flex-direction: row | row-reverse | column | column-reverse`
- `flex-wrap: wrap` — items flow onto new lines instead of shrinking forever
- `gap: 16px` — space *between* items only; use it instead of margin hacks
- `justify-content: flex-start | center | space-between | space-around | space-evenly`
- `align-items: stretch | center | flex-start | flex-end | baseline`
- `align-content` — spaces the *lines*; only does anything when wrapping produces 2+ lines

## ITEM PROPERTIES

- `flex-grow` — share of leftover space (ratio, not px)
- `flex-shrink` — how eagerly it shrinks below its basis (default 1)
- `flex-basis` — starting size before grow/shrink (beats `width`)
- **Shorthands to memorise:**
- `flex: 1` → `1 1 0%` — equal columns, content size ignored
- `flex: auto` → `1 1 auto` — grows, but content-weighted
- `flex: none` → `0 0 auto` — rigid, sizes to content
- `align-self` — override `align-items` for one item
- `order` — visual only; keyboard/screen-reader order doesn't move (accessibility risk — use sparingly)

## FLEXBOX PATTERNS — COPY, PASTE, DONE

### Perfect centering (both axes)

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

### Navbar with pushed-right actions

```
.nav { display: flex; gap: 24px; }
.nav .actions { margin-left: auto; }
/* auto margins eat the free space */
```

### Media object (avatar + text)

```
.media { display: flex; gap: 12px; }
.media img { flex: none; }
.media .body { min-width: 0; }
```

### Equal-height cards

```
.row { display: flex; gap: 16px; }
.card {
  flex: 1;              /* equal
widths */
  display: flex;
  flex-direction: column;
}
.card .cta { margin-top: auto; }
/* buttons align at the bottom */
```

### Sticky footer (full-height page)

```
body {
  min-height: 100dvh;
  display: flex;
  flex-direction: column;
}
main { flex: 1; }
/* footer sits at the bottom
   even on short pages */
```

### Tag list that wraps

```
.tags {
  display: flex;
  flex-wrap: wrap;
  gap: 8px;
}
```

## FLEXBOX GOTCHAS — THE ONES THAT COST AN AFTERNOON

- **Text won't truncate?** Flex items default to `min-width: auto` — they refuse to shrink below their content. Fix: `min-width: 0` on the item, then `text-overflow: ellipsis` works
- **`width` ignored?** `flex-basis` wins over `width` on flex items — set the basis, not the width
- **Images stretching weirdly?** `align-items: stretch` is the default — give images `align-self: flex-start` or wrap them
- **Last row spreading oddly with `space-between`?** That's wrapping + justification — you want grid's `auto-fit` pattern instead (page 2)

## GRID — THE MENTAL MODEL

- Grid is **two-dimensional**: rows and columns at the same time, defined on the *container*
- **Lines, not cells:** a 3-column grid has 4 numbered lines; items span from line to line
- `1fr` = one share of the *leftover* space (after fixed tracks and gaps are taken out)
- **Implicit tracks:** overflow items create auto-sized rows — control them with `grid-auto-rows`
- Negative numbers count from the end: `grid-column: 1 / -1` = full width, however many columns exist
- Layout-in sizing: the grid defines structure, content fills it — the inverse of flexbox

## CONTAINER PROPERTIES

- `display: grid` · `gap: 16px` (or `row-gap` / `column-gap`)
- `grid-template-columns: 200px 1fr 1fr` — mix fixed and flexible tracks freely
- `repeat(3, 1fr)` — shorthand for repeated tracks
- `minmax(240px, 1fr)` — a track with a floor and a ceiling
- `auto-fit` vs `auto-fill`: both wrap tracks to fit; **auto-fit collapses empty tracks** (items stretch), auto-fill keeps them (items stay small). When unsure: `auto-fit`
- `grid-auto-flow: dense` — backfills gaps left by spanning items

## ITEM PLACEMENT

- `grid-column: 1 / 3` — from line 1 to line 3 (= spans 2 tracks)
- `grid-column: span 2` — span 2 tracks from wherever it lands
- `grid-row: span 2` — same, vertically
- `grid-area: hero` — place into a named template area
- `place-self: center` — shorthand for `align-self` + `justify-self` on one item
- Alignment mirrors flexbox: `justify-*` = inline axis, `align-*` = block axis; `*-items` targets contents, `*-content` targets the track group

## GRID PATTERNS — COPY, PASTE, DONE

### Responsive cards, zero media queries

```
.grid {
  display: grid;
  grid-template-columns:
    repeat(auto-fit, minmax(240px,
1fr));
  gap: 16px;
}
/* 4 cols on desktop, 1 on mobile,
   everything in between for free */
```

### Overlap without position: absolute

```
.stack { display: grid; }
.stack > * { grid-area: 1 / 1; }
/* all children share one cell —
   caption over image, etc. */
```

### Page layout with named areas

```
.layout {
  display: grid;
  grid-template-areas:
    "header header"
    "sidebar main"
    "footer footer";
  grid-template-columns: 240px 1fr;
  min-height: 100dvh;
}
header { grid-area: header; }
/* readable, reorderable per
   breakpoint in one place */
```

### Bento grid (the mirzaa.dev homepage)

```
.bento {
  display: grid;
  grid-template-columns: repeat(4,
1fr);
  grid-auto-rows: 120px;
  gap: 16px;
}
.bento .feature {
  grid-column: span 2;
  grid-row: span 2;
}
```

### Full-bleed section in a capped column

```
.page {
  display: grid;
  grid-template-columns:
    1fr min(65ch, 100%) 1fr;
}
.page > * { grid-column: 2; }
.page .full-bleed { grid-column: 1 /
-1; }
```

## FLEX OR GRID? — 10-SECOND DECISION

- **One row or column of things?** Flexbox — navbars, button groups, tag lists, media objects
- **Rows *and* columns designed together?** Grid — page shells, card grids, dashboards, bento layouts
- **Content should decide the sizes?** Flexbox. **Layout should decide?** Grid
- They nest freely: grid for the page, flex inside each card — this is the standard combination, not a compromise

## GRID GOTCHAS

- `auto-fit` + `minmax` **overflowing on tiny screens?** The 240px floor wins over the viewport — use `minmax(min(240px, 100%), 1fr)`
- `1fr` **rows blowing out?** `fr` can't shrink below content's min size — same fix as flexbox: `min-height: 0` / `min-width: 0` on the item
- **Margins don't collapse** between grid items — spacing is the grid's job; use `gap`
- **Named areas must be rectangular** — no L-shapes; use line-based spans for those