ScopeTreasure Courses

CSS COURSES · LESSON 2.6 · CODE SCOPE SERIES

Flexbox

One-line layouts made easy: the flex container, main and cross axes, justify-content, align-items, gap, wrap — and a real navbar.

Introduction

Putting boxes in a row used to be the hardest beginner problem in CSS — floats, hacks, tears. **Flexbox** ended that era: give a parent `display: flex` and its children line up in a row you fully control — how they spread out, how they align, how they wrap. Flexbox is a **one-dimensional** layout system: it arranges things along **one line** (a row *or* a column). Navbars, button rows, card strips, centring anything — this is the tool you will reach for daily. *(Roman Urdu: Flexbox aik line ka layout hai — cheezon ko seedhi saf mein sajana is ka kaam hai.)*

Key points

- **Two roles:** the **flex container** (the parent with `display: flex`) and the **flex items** (its direct children). You write almost all flex properties on the **container**.
- **Two axes:** the **main axis** runs along the row (left → right by default); the **cross axis** runs across it (top → bottom). `flex-direction: column` swaps them — main axis becomes vertical.
- **`justify-content`** arranges items along the **main** axis: `flex-start` (pack left), `center`, `flex-end`, `space-between` (first and last glued to the edges, space shared between), `space-around` (space around each item).
- **`align-items`** arranges items along the **cross** axis: `flex-start`, `center`, `flex-end`, `stretch` (fill the container's height — the default). One item can override with **`align-self`**.

  (Roman Urdu: Justify = line ki simt mein, align = line ke aar-paar. Bas yehi do lafz yaad rakho.)
- **`gap`** puts clean space *between* items — no margin hacks, no last-child exceptions: `gap: 16px`.
- **`flex-wrap: wrap`** lets items drop to the next line when the row runs out of room — the secret behind card galleries that work on every screen size.
- **The 3-line centre:** `display: flex; justify-content: center; align-items: center;` on a container centres its child horizontally *and* vertically — the most famous three lines in CSS.

Code examples

Example 1 — A real navbar

```
.navbar {
  display: flex;
  justify-content: space-between;
  align-items: center;
  padding: 12px 20px;
  background: #0b0f19;
}
.navbar .links { display: flex; gap: 20px; }
.navbar a { color: #faf9f6; text-decoration: none; }
```

```
<nav class="navbar">
  <a class="logo" href="#">ToolScope</a>
  <div class="links">
    <a href="#">Tools</a><a href="#">Notes</a><a href="#">About</a>
  </div>
</nav>
```

What it does: The nav is a flex container with two items (logo, links group). `space-between` pushes them to opposite ends; `align-items: center` lines them up vertically. The links group is *itself* a flex container — flex inside flex is normal and encouraged — with `gap` doing the spacing.

What you will see: a dark bar, logo left, links right, everything vertically centred — at any screen width, without a single float or margin trick.

Example 2 — Perfect centring in 3 lines

```
.stage {
  height: 300px;
  display: flex;
  justify-content: center;  /* main axis  → horizontal centre */
  align-items: center;      /* cross axis → vertical centre   */
  background: #f2f5f9;
}
```

What it does: Whatever single child `.stage` holds is placed at the exact centre of the box. Swap in `flex-direction: column` and the same two properties still centre it — they follow the axes, not the screen directions.

What you will see: a card floating at the dead centre of the grey stage. Resize the window — it recentres itself. Memorise these three lines; you will type them for years.

Example 3 — A wrapping card gallery

```
.gallery {
  display: flex;
  flex-wrap: wrap;
  gap: 16px;
}
.gallery .card {
  flex: 1 1 220px;   /* grow, shrink, ideal width 220px */
  background: white;
  border: 1px solid #e3e6ec;
  border-radius: 12px;
  padding: 18px;
}
```

What it does: `flex-wrap: wrap` allows new lines. The item rule `flex: 1 1 220px` says: start at 220px wide, **grow** to share leftover space equally, **shrink** if the row is tight. Together they produce a gallery that fits 4 cards on a laptop and 1–2 on a phone — with no media queries at all.

What you will see: six cards flowing into rows that re-form as you resize the window — rows fill evenly, the last row stretches its cards to match. This pattern is behind half the card layouts on the modern web.

Quick recap

- Flex = parent `display: flex`; **justify-content** along the main axis, **align-items** across it.
- `space-between` for navbars, `gap` for clean spacing, `flex-wrap: wrap` + `flex: 1 1 <size>` for self-adjusting galleries.
- Centring anything = container + `justify-content: center` + `align-items: center`.

Common mistakes

1. **Writing justify/align on the items** — those properties belong to the **container**. Items only get `align-self` (an override) and their `flex` sizing. A rule that "does nothing" is usually on the wrong element.
2. **Forgetting which axis is which after `flex-direction: column`** — justify and align swap jobs when the direction turns. If things centre the "wrong way", check the direction first.
3. **Spacing items with margins instead of gap** — margins leave an unwanted lump at the row's end and break `space-between` maths. `gap` spaces *between* items only — use it.

Practice

1. Build the navbar from Example 1, then flip it: logo right and links left, using only `flex-direction` / `justify-content` changes — no HTML edits.

2. Make a gallery of 8 cards with the wrap pattern. Resize the browser slowly and note the widths where rows change from 4 → 3 → 2 → 1 cards.

Practice quiz

1. What does display: flex do?

- Hides the element
- Lays children out in a flexible row or column
- Makes a grid
- Fixes the position

Answer: Lays children out in a flexible row or column

Watch the video lesson: https://youtu.be/WAO5YzKbH4k

Read online: https://scopetreasure.com/notes/css-flexbox · More free tools and notes at https://scopetreasure.com