CURSED SOCIALS · HACK CLUB YSWS · GUIDE PART 2 OF 3

Layouts with Flexbox & Grid (stop stacking everything!)

So you finally made it here. In Part 1 you built your first post card, and it probably looks good — except everything is still stuck in one tall vertical column. Let's fix that.

Right now, if you try to put a profile picture next to a username, or put navigation links across the top of the screen, HTML just stacks them on top of each other like pancakes. That's totally normal — browsers do that by default until you turn on **Flexbox** or **Grid**.

Here's where we are in the 3-part series:

- **Part 1 (done!):** Files, basic HTML tags, colors, fonts, and the Box Model.
- **Part 2 (you are here):** Navbars, side-by-side avatars, photo grids, and making it work on phones.
- **Part 3 (next):** Making like buttons and comments actually work with JS + putting your site online.

1. Flexbox: the magic side-by-side switch

Flexbox sounds intimidating, but 90% of the time it's literally one line of CSS: `display: flex;`. You put it on the **parent box** (the outer container), and every child inside it immediately snaps into a horizontal row.

```
WITHOUT FLEX (default):          WITH display: flex; ON PARENT:
+------------------+             +-----------------------------+
|  [  Box 1  ]     |             | [ Box 1 ] [ Box 2 ] [ Box 3 ] |
|  [  Box 2  ]     |             +-----------------------------+
|  [  Box 3  ]     |
+------------------+
```

> **Biggest beginner trap:** Always put `display: flex;` on the *wrapper* (the parent), NOT on the individual buttons or text inside it. If nothing moved, check which box got the class!

The only 5 Flexbox lines you actually need

Ignore those giant 40-property Flexbox cheat sheets online. Real developers use these five lines for almost everything:

CSS Line	What it actually does
`display: flex;`	Turns on Flexbox and puts everything inside in one horizontal row.
`gap: 12px;`	Adds clean breathing room between each item (way easier than margins!).
`justify-content: space-between;`	Pushes the first item to the far left and the last item to the far right. (Use `center` to center left-to-right.)
`align-items: center;`	Lines everything up vertically in the middle so small icons and tall text don't look crooked.
`flex-wrap: wrap;`	Lets items drop onto a new line if the screen gets too skinny instead of squishing them.

Bonus meme unlock: how to center a div

People joke that centering a box in CSS is impossible. It used to be annoying in 2010 — today it's just three lines on the parent box:

```
.center-me {
  display: flex;
  justify-content: center; /* left-to-right center */
  align-items: center;     /* top-to-bottom center */
}
```

2. Build a real Navbar in 10 lines

Every social app has a top bar: app name on the left, links or buttons on the right. Notice how we wrap the links in their own tiny `<div>` so `space-between` keeps them grouped on the right side:

```
<nav class="navbar">
  <div class="logo">CursedGram</div>
  <div class="nav-links">
    <a href="#">Home</a>
    <a href="#">Profile</a>
    <a href="#">Log out</a>
  </div>
</nav>
```

```
+-----------------------------------------------------------+
|  CursedGram                          Home  Profile  Log out |
+-----------------------------------------------------------+
```

```
.navbar {
  display: flex;
  justify-content: space-between;
  align-items: center;
  background: #ffffff;
  border-bottom: 2px solid #24211f;
  padding: 12px 20px;
  position: sticky; /* glues the navbar to the top when scrolling! */
  top: 0;
}

.nav-links {
  display: flex;
  gap: 16px;
}
```

Wait — look at `position: sticky; top: 0;` right there. Those two lines make your navbar stick to the top of the screen while you scroll down the feed, just like Instagram or Twitter.

3. Upgrade your Post Header (Avatar + Name)

In Part 1, your username sat alone above the caption. Let's put a round profile picture right next to the username and timestamp using the exact same Flexbox trick:

```
<div class="post-header">
  <img class="avatar" src="cat.jpg" alt="profile pic">
  <div>
    <div class="username">@you</div>
    <div class="time">2 mins ago</div>
  </div>
</div>
```

```
.post-header {
  display: flex;
  align-items: center;
  gap: 10px;
  margin-bottom: 12px;
}

.avatar {
  width: 42px;
  height: 42px;
  border-radius: 50%;
  object-fit: cover;
  border: 2px solid #24211f;
}

.username { font-weight: bold; }
.time { font-size: 13px; color: #6b6159; }
```

Notice why `.username` and `.time` are wrapped in an extra `<div>`? Because Flexbox lines up its direct children side by side! Grouping the two text lines together keeps them stacked on top of each other right beside the avatar.

4. CSS Grid (Photo Grids & Sidebars)

Quick rule of thumb so you never get confused between Flexbox and Grid:

- **Use Flexbox** for a single line of stuff (a navbar, a post header, a row of like/comment buttons).
- **Use Grid** when you want neat rows *and* columns (an Instagram profile grid, or a Discord sidebar + chat area).

Recipe A: Instagram-style 3-column photo grid

That `1fr` unit just means "1 equal slice of whatever space is available." So `repeat(3, 1fr)` literally means "give me 3 equal columns":



```
.photo-grid {
  display: grid;
  grid-template-columns: repeat(3, 1fr);
  gap: 8px;
  max-width: 600px;
  margin: 24px auto;
}

.photo-grid img {
  width: 100%;
  aspect-ratio: 1 / 1; /* forces every photo into a perfect square! */
  object-fit: cover;   /* crops neatly instead of stretching faces */
  border-radius: 8px;
}
```

Recipe B: Discord / Twitter sidebar + main feed

Want a skinny menu on the left and a big feed on the right? One grid line does the whole layout:

```
+--------------+-------------------------------------+
|  Sidebar     |  Main Feed                          |
|  (220px)     |  (1fr = takes all leftover space)   |
|  - #general  |  [ Post 1 ]                         |
|  - #memes    |  [ Post 2 ]                         |
+--------------+-------------------------------------+
```

```
.app-layout {
  display: grid;
  grid-template-columns: 220px 1fr;
  gap: 20px;
  max-width: 900px;
  margin: 24px auto;
}
```

5. Making it not explode on phones

Your reviewers and friends might open your link on their phone. If you hard-code `width: 800px;` on a post, phone users have to scroll sideways and everything looks broken. Two quick habits fix 95% of mobile bugs:

1. **Use `max-width` instead of `width`:** Writing `max-width: 500px; width: 100%;` tells the box to stay 500px on a laptop, but shrink nicely to fit a phone screen.
2. **Add one `@media` rule at the bottom of `style.css`:** This tells the browser "hey, if the screen is narrower than 650px, switch my 2-column layout to 1 column."

```
@media (max-width: 650px) {
  .app-layout {
    grid-template-columns: 1fr; /* stack sidebar on top of feed */
  }

  .photo-grid {
    grid-template-columns: repeat(2, 1fr); /* 2 columns instead of 3 */
  }
}
```

> **Pro tip to test mobile on your laptop:** Right-click your page in Chrome, hit **Inspect**, and click the tiny phone/tablet icon at the top-left of DevTools. Boom, instant phone preview.

6. Your Part 2 Upgrade Challenge

Open the `my-social` folder you made in Part 1 and level it up with everything you just learned:

1. Add a sticky `.navbar` at the top of `<body>` with your app name on the left and 3 links on the right.
2. Upgrade your `.post` cards to have a `.post-header` with a round avatar beside the username and timestamp.
3. Add a `.post-actions` row at the bottom of each post using `display: flex; gap: 10px;` so your Like, Comment, and Share buttons sit neatly side by side.
4. Add either a 3-column `.photo-grid` (Instagram profile vibe) OR a `220px 1fr` sidebar (Discord/Twitter vibe).

Wait, why did my layout break? (Quick fixes)

What went wrong	How to fix it in 5 seconds
Items still won't sit side by side	Check that `display: flex;` is on the **parent** container wrapping the items, not on the items themselves.
My navbar has weird white gaps at the edges	Browsers add a sneaky 8px margin to `body` by default. Add `body { margin: 0; }` at the top of `style.css`.
A giant photo broke out of my post card	Add `img { max-width: 100%; display: block; }` so images never overflow their container.
My phone preview looks zoomed out	Make sure `<meta name="viewport" content="width=device-width, initial-scale=1.0">` is inside your `<head>`.

You just finished Part 2!

Your site should officially look like a real social media app now — not a 1995 text page. Drop a screenshot of your layout in **#cursed-socials** on Slack! Up next in **Part 3 (the finale)**: wiring up working like counters, live comment boxes, and dark mode with JavaScript, then dropping your folder onto Netlify to claim your $5-$10 Hack Club grant.