Web Coding Camp — Day 3
Layout: Flexbox & Responsive Design

David Pouliot
Southern Oregon University

Today

So far everything stacks top to bottom. Today: put things *side by side*.

- Why layout used to be hard
- `display: flex` — the modern answer
- Spacing and aligning items
- Building a nav bar
- A row of cards
- Making it work on phones (responsive)

Goal: turn your single column into a real page — a nav bar on top and a row of cards.

Flexbox

The magic line: `display: flex`

Put `display: flex` on a *container*, and its children line up in a row.

```html
<div class="row">
  <div>One</div>
  <div>Two</div>
  <div>Three</div>
</div>
```

```css
.row {
  display: flex;    /* children go side by side */
  gap: 1rem;        /* space between them */
}
```

Spacing and aligning

```css
.row {
  display: flex;
  justify-content: center;  /* left-to-right position */
  align-items: center;      /* top-to-bottom position */
  gap: 1rem;
}
```

- `justify-content` — position along the row.
- `align-items` — position across the row.

Play `flexboxfroggy.com` to master these two as a game.

A Nav Bar

A nav bar is just a flex row of links

```html
<nav>
  <a href="index.html">Home</a>
  <a href="about.html">About</a>
</nav>
```

```css
nav {
  display: flex;
  gap: 1rem;
  justify-content: center;
  background: #6a1b9a;
  padding: 1rem;
}
nav a { color: white; text-decoration: none; }
```

A Row of Cards

Cards that share the space

```css
.cards {
  display: flex;
  gap: 1rem;
  flex-wrap: wrap;       /* drop to next line if too narrow */
}
.card {
  flex: 1;               /* each card grows to fill */
  min-width: 200px;      /* ...but never thinner than this */
  background: white;
  padding: 1rem;
  border-radius: 12px;
}
```

What `flex-wrap` buys you

- Without it, three cards squish into one row no matter how narrow.
- With `flex-wrap: wrap`, cards that don't fit drop to the next line.
- Combined with `min-width`, your cards automatically go from a row (wide screen) to a stack (phone).

Resize your browser window and watch the cards rearrange. That's "responsive" — no extra code needed.

Responsive Design

Tell phones you meant it

One line in your <head> makes phones show the page at real size instead of zoomed out:

```
<meta name="viewport" content="width=device-width, initial-scale=1.0">
```

Without this tag, your site looks tiny on a phone. With it, everything scales to the screen. Add it to every page.

Different rules on small screens

A media query applies CSS only below a certain width.

```css
/* normal rules up here... */

@media (max-width: 500px) {
  nav {
    flex-direction: column;   /* stack the links */
  }
  h1 { font-size: 1.5rem; }
}
```

Everything inside the `@media` block only kicks in on narrow screens.

Project: Multi-Section Site

Give your site a real layout

Turn your single page into a proper site:

- A **nav bar** across the top (flex row of links)
- A **hero** section — big heading and image
- A **row of cards** — your projects, favourites, or sections
- The **viewport** meta tag, so it works on phones

Test it by dragging your browser window narrow. Do the cards stack? Does the text stay readable?

Stretch goals

- Make the nav stick to the top (`position: sticky`)
- Try **CSS Grid** for the cards instead of flexbox
- Add a media query so cards stack on narrow screens
- Build a simple image gallery
- Add a footer with social links

Flying through it? Play `cssgridgarden.com` to learn Grid, the other big layout tool.