CSS Layouts

TJ Dev Club

Carry over your project

Open the same `index.html` you worked on in CSS Beginner. We'll keep building on it. If you don't have it, use the template from last week and save it as `index.html`.

- Make sure your file still loads in the browser
- Keep this same file for all CSS sessions

Milestone

You should still see your page from last week. We'll add a navbar and layout on top.

Some Review

What We Do

Mentoring, Projects, Community

TJ Dev Club meets Wednesdays 8B. Developers of all skill levels are welcome to learn, build, and plug into a supportive developer community.

1.
   Mentoring

   Collaborate with fellow students and club officers through small groups, guided tutorials, and lectures from experts. Mentor others to help them succeed and earn service hours.

2.
   Projects

   Build for competitions or for yourself. Ship at least one project this year. We'll help you win HackTJ.

3.
   Community

   Share your work at frequent competitions, make friends, and stay connected on Discord. Our strong community is like no other.

Some Review


What We Do
Mentoring, Projects, Community
TJ Dev Club meets Wednesdays 8B. Developers of all skill levels are welcome to learn, build, and plug into a supportive developer community.
Mentoring
Collaborate with fellow students and club officers through small groups, guided tutorials, and lectures from experts. Mentor others to help them succeed and earn service hours.
Projects
Build for competitions or for yourself. Ship at least one project this year. We'll help you win HackTJ.
Community
Share your work at frequent competitions, make friends, and stay connected on Discord. Our strong community is like no other.

Some Review

Here’s a example of CSS usage in HTML to hopefully refresh your mind:

```
<style>
  .trebuchet {
    font-family:
      "Trebuchet MS", sans-serif; /* Use Open Sans if it exists, otherwise, use the default "sans-serif" font. */
  }
</style>

<div>This is the default font.</div>

<div class="trebuchet">This is Trebuchet MS.</div>
```

Layouts 101

Before using flexboxes, let's understand how the page stacks by **default**: block elements go from top-to-bottom. These are called **display layouts** and they are managed with the `display` CSS property.

Layouts are also why some elements stack and some don't! Later on in this presentation, we'll explore flexboxes and use them too make a nav bar.

```
<style>
  header, nav, main, footer { padding: 12px; border-radius: 6px; }
  header { background: #e0e7ff; }
  nav { background: #fde68a; }
  main { background: #bbf7d0; min-height: 80px; }
  footer { background: #fecaca; }
  body { font-family: system-ui, -apple-system, Segoe UI, Roboto, Helvetica, Arial, sans-serif; }
</style>

<header>Header</header>
<nav>Navigation</nav>
```

Two columns (no flex yet)

You can place columns side-by-side with `display: inline-block`. You can also use spacing like `gap` and `padding` (which we will learn later) with `display: inline-block`.

```
<style>
  .container { max-width: 900px; margin: 0 auto; }
  .row { font-size: 0; /* remove whitespace between inline-blocks */ }
  .col {
    display: inline-block;
    vertical-align: top;
    width: 50%;
    box-sizing: border-box;
    padding: 12px;
    font-size: 14px; /* restore text size in columns */
  }
```

Centering a box

You can also center a fixed-width element with `margin: auto`, no `display` needed!

```
<style>
  .card {
    width: 260px;
    margin: 24px auto; /* centers horizontally */
    padding: 16px;
    background: #e2e8f0;
    border-radius: 8px;
    text-align: center;
    border: 2px solid #94a3b8;
  }
</style>
```

*Flex*boxes

As the name implies, **flexboxes** allow you to position content on your website horizontally or vertically in a **flex**ible way.

To think of it another way, it’s basically a one-dimensional layout, making it really easy to design a responsive layout (which we’ll learn about later).

Open up your HTML file from last week. If you don’t have one, you can use this template.

*Flex*boxes

```
<style>
  .container {
    display: flex;
    gap: 8px;
  }

  .box {
    width: 60px;
    height: 60px;
    background: #4f46e5;
    color: white;
```

*Flex*boxes

Okay… but what can I even use flexboxes for?

- Navigation bars (at the top of the page)
- Centering things (with margin/padding)
- Arranging items in rows/columns
- Simple responsive layouts (very important for the web)

*Flex*boxes

We're going to style our **very own** navigation bar! Find the `<body>` tag, and make sure you put your cursor between the `<body>` and `</body>`. Then, type in the HTML elements below (don't worry, an officer will explain).

```
<!DOCTYPE html>
<html>
  <!-- content omitted -->

  <body>
    <!-- other elements omitted -->
    <nav class="navigation">
      <div class="item">home</div>
      <div class="item">about me</div>
      <div class="item">i love dev club!</div>
    </nav>
  </body>
</html>
```

display: flex

To use flexbox, we gotta turn it on. Use the `display: flex` property on the container. The `gap` property allows us to evenly space items with a distance.

```
<style>
  .navigation {
    display: flex;
    gap: 20px;
  }
</style>

<nav class="navigation">
  <div class="item">home</div>
  <div class="item">about me</div>
  <div class="item">i love dev club!</div>
```

justify-content

`justify-content` controls tbe spacing between items in a flexbox. Rather than using something like `gap`, we cna use `justify-content` to evenly space all the items (or to do some other cool stuff).

```
<style>
  .navigation {
    display: flex;
    gap: 20px;
    justify-content: space-between; /* try: center, flex-start, flex-end */
  }
</style>

<nav class="navigation">
  <div class="item">home</div>
  <div class="item">about me</div>
```

align-items

Control **VERTICAL** alignment on the cross axis. If you want to have elements at the center, bottom, or top of a flexbox, you should be using this.

```
<style>
  .navigation {
    display: flex;
    gap: 20px;
    align-items: center; /* try: flex-start, flex-end */
    height: 80px; /* gives room to see vertical alignment */
  }
</style>

<nav class="navigation">
  <div class="item">home</div>
```

flex-direction

`flex-direction` allows you to change the main axis direction. If you recall what was said earlier, flexboxes allow you to go in either a **vertical** or **horizontal** direction. This is what you would be using to control that!

```
<style>
  .navigation {
    display: flex;
    gap: 20px;
    flex-direction: column; /* try: row */
    width: 260px;
  }
</style>

<nav class="navigation">
  <div class="item">home</div>
```

Build a simple navbar

Using what we learned, let’s create a navbar with two groups: links on the left and actions on the right.

```
<style>
  .navigation {
    display: flex;
    gap: 20px;
    justify-content: space-between; /* spread left and right groups */
    align-items: center; /* vertically center items */
  }

  .nav-group {
    display: flex; /* lay out items in each group */
    gap: 20px;     /* space items within each group */
```

Milestone

Check with an officer: does your navbar have…

- Left group and right group separated (space-between)
- Items aligned vertically in the center
- Even spacing between items (gap)

If yes, you’re ready to move on!

Box Model (important!)

So earlier we were using `gap` for spacing… but there are many cases where `gap` wouldn't be the best option. Luckily, we have the **box model**! We will be going over the following:

- margin, padding, border all work in UNITS (vh, px, etc.)
- Understanding how spacing works (inside vs outside)
- box-sizing: border-box (life-changing!)

padding vs margin

`padding` adds space INSIDE the element's background; margin adds space OUTSIDE of the element.

```
<style>
  .container {
    background: #f1f5f9;
    padding: 12px;
    border-radius: 8px;
  }

  .box {
    background: #93c5fd;
    border-radius: 6px;
  }
```

Add padding to your navbar

Make the clickable areas larger and easier to tap by adding padding and a background (to make your changes visible of course). You'll also want to set `border-radius` to make your buttons rounded.

```
<style>
  .navigation { display: flex; gap: 20px; }
  .item { padding: 8px 12px; background: #e2e8f0; border-radius: 6px; }
</style>

<nav class="navigation">
  <div class="item">home</div>
  <div class="item">about</div>
  <div class="item">contact</div>
</nav>
```

Add margin to your navbar

Add some `margin` to your navbar to allow for some breathing room, which really improves the look of our site. Add some content below too.

```
<style>
  .navigation { display: flex; gap: 20px; margin: 16px 0; }
  .item { padding: 8px 12px; background: #e2e8f0; border-radius: 6px; }
</style>

<nav class="navigation">
  <div class="item">home</div>
  <div class="item">about</div>
  <div class="item">contact</div>
</nav>
<main>Content below</main>
```

border

`border` just adds a line (or border) around the element, which can be customized with other CSS properties. If you want to learn more about borders and their properties, check out the resources linked at the end.

```
<style>
  .card {
    padding: 12px;
    border: 4px solid #334155;
    background: #e2e8f0;
    border-radius: 8px;
  }
</style>

<div class="card">border: 4px solid #334155</div>
```

Add borders to navbar items

Borders help buttons feel tactile and separated.

```
<style>
  .navigation { display: flex; gap: 20px; }
  .item { padding: 8px 12px; border: 2px solid #334155; border-radius: 6px; }
</style>

<nav class="navigation">
  <div class="item">home</div>
  <div class="item">about</div>
  <div class="item">contact</div>
</nav>
```

Add a border to the navbar

Use a bottom border on the navbar to separate it from the page content.

```
<style>
  .navigation { display: flex; gap: 20px; padding-bottom: 8px; border-bottom: 2px solid #cbd5e1; }
  .item { padding: 8px 12px; border-radius: 6px; }
</style>

<header>Header</header>
<nav class="navigation">
  <div class="item">home</div>
  <div class="item">about</div>
  <div class="item">contact</div>
</nav>
```

How does spacing work?

Let’s compare padding, margin, and border side-by-side.

```
<style>
  .wrap {
    display: flex;
    gap: 8px;
    background: #f8fafc;
    padding: 8px;
    border-radius: 8px;
  }
  .cell { background: #bae6fd; border-radius: 6px; }
  .p8 { padding: 8px; }
  .m8 { margin: 8px; }
```

box-sizing: border-box

Say you want to set a fixed width or height that INCLUDES padding and border. With border-box, the declared width includes padding and border, making layouts **much** easier.

```
<style>
  .row { display: flex; gap: 12px; background: #f1f5f9; padding: 12px; border-radius: 8px; }
  .box { width: 200px; padding: 20px; border: 6px solid #334155; background: #e2e8f0; border-radius: 8px; }
  .content-box { box-sizing: content-box; }
  .border-box { box-sizing: border-box; }
</style>

<div class="row">
  <div class="box content-box">content-box (total > 200px)</div>
  <div class="box border-box">border-box (total = 200px)</div>
</div>
```

Mini-project

Build the wildest website of your dreams. It must contain:

- a navbar
- main content

Use the CSS we learned this meeting and last meeting to decorate and structure your website!

Resources

Stuck? These help a ton:

- MDN Web Docs – CSS: developer.mozilla.org
- W3Schools – CSS: w3schools.com/css
- CSS-Tricks – Flexbox Guide: css-tricks.com
- web.dev – Learn CSS: web.dev/learn/css
- Or just ask an officer for help

We don’t expect you to remember everything from Dev Club meetings. It’s okay to ask for help!

</style>