Extracted from:

Modern CSS with Tailwind, Second Edition

Flexible Styling Without the Fuss

The Pragmatic Bookshelf

Raleigh, North Carolina

The Pragmatic Programmers

Pragmatic exPress

Modern CSS with Tailwind

Second Edition

Flexible Styling Without the Fuss

Covers Tailwind 3.0

Noel Rappin

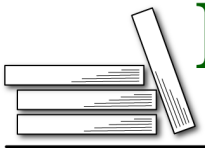
Edited by Katharine Dvorak

Modern CSS with Tailwind, Second Edition

Flexible Styling Without the Fuss

Noel Rappin

The Pragmatic Bookshelf
Raleigh, North Carolina



For our complete catalog of hands-on, practical, and Pragmatic content for software developers, please visit *https://pragprog.com*.

The team that produced this book includes:

CEO: Dave Rankin
COO: Janet Furlow
Managing Editor: Tammy Coron
Development Editor: Katharine Dvorak
Copy Editor: L. Sakhi MacMillan
Layout: Gilson Graphics
Founders: Andy Hunt and Dave Thomas

For sales, volume licensing, and support, please contact *support@pragprog.com*.

For international rights, please contact *rights@pragprog.com*.

ISBN-13: 978-1-68050-940-3
Encoded using the finest acid-free high-entropy binary digits.
Book version: P1.0—May 2022

In the previous chapter, we looked at ways you can use Tailwind to control the display of a single DOM element. In this chapter, we'll look at how Tailwind can manage the layout of multiple elements.

With Tailwind, you can lay out the elements on an entire page and manage common features like navigation, sidebars, and footers. You can also use Tailwind to put together complex groupings of elements within a page, such as cards or hero blocks.

Let's start with some general utilities Tailwind provides to help place elements on a page: the box-to-box relationships.

Containers

Many CSS frameworks use a `container` class as the general top-level container to specify page width. While Tailwind does offer a `container` utility, Tailwind's version does much less than similar classes do in other frameworks. All the `container` utility does in Tailwind is specify the `max-width` of the element based on the width of the browser viewport. For example, any viewport between 640 and 768 pixels wide would be set to a `max-width` of 640 pixels. Once the viewport goes over 768, the `max-width` stays at 768 pixels until the viewport hits 1024 pixels and then jumps again when the viewport reaches 1280 pixels.

The advantage of using a `container` is that it allows you to only worry about those specific widths in your design rather than having to take into account any possible width the viewport might have.

Viewports

In CSS, the viewport is the area of the browser where the user can see content. Usually, the dimension of concern is the width of the viewport because that determines how much content can be placed across the screen without scrolling horizontally. The HTML `meta` tag is used to control the viewport width on mobile screens. By default, mobile browsers often assume a wider display than the actual device (often 980 pixels) and scale the content to fit on screen. That usually looks terrible. You should use the `content="width=device-width,initial-scale=1"` attribute for the browser to use the device size as the viewport rather than scaling the display down from a wider size.

If you're familiar with other frameworks, Tailwind's `container` won't have features you may be expecting. A Tailwind `container` doesn't automatically horizontally center its child elements. To get centering behavior, you pair the container

with mx-auto. A Tailwind container also doesn't introduce a padding or margin to pull its elements away from the browser's border. To get this behavior, you pair the container with an m- or p- utility. So a plausible class list for your top-level element might be class="container mx-auto py-12 px-6".

Floats and Clears

Although a fresh new design will likely use the grid and flexbox tools described in the rest of this chapter to position elements, if you're using Tailwind in a legacy project, you might still need to deal with floats and clearfixes.

In CSS, the float property positions content inside its container. Typically, the float property is used to position a particular element, often an image, to one side or the other of its container, allowing the rest of the container, often text, to stay completely on the other side, rather than mixing the elements together.

Tailwind provides float-left and float-right for positions, and float-none as a reset option.

The CSS clear property forces an element to be placed below any elements it might otherwise overlap with on one or both sides. (Technically, it prevents other elements from floating, which amounts to the same thing.) Tailwind provides utilities to specify a clear behavior on either side, both sides, or no sides: clear-left, clear-right, clear-both, and clear-none.

Position and Z-Index

In CSS, the z-index property is an integer determining how items stack on top of each other along what would be the "z axis" if you ran an axis outward perpendicular to the screen. Tailwind provides the pattern, z-{index}, where the index can be 0, 10, 20, 30, 40, 50, or auto. You can use a negative z-index of those values by using the -z pattern, -z-20, or with an arbitrary value z-[-1].

Tables

The classic way of spacing HTML pages is the table. Unless you're actually displaying tabular data, a CSS grid is now preferable for layout purposes, so Tailwind doesn't provide many specific table utilities.

Tailwind lets you use table-auto to keep the default browser behavior of auto-spacing the columns of a table based on its content. If you want to explicitly specify column widths, you can use table-fixed on the <table> element and then

put an explicit width helper on each column of the table—the fractional helpers are useful for this:

html/page.html

```
<table class="table-fixed border border-collapse">
  <tr>
    <th class="border border-black w-1/6">Small</th>
    <th class="border border-black w-2/6">Medium</th>
    <th class="border border-black w-3/6">Large</th>
  </tr>
</table>
```

Tailwind also lets you merge the borders of adjacent table cells with the help of the border-collapse utility, which is reset with border-separate.

Tailwind offers the odd: or even: modfier to give tables alternating row colors, such as class="odd:bg-white even:bg-grey-300", for example.