Extracted from:

Modern CSS with Tailwind

Flexible Styling Without the Fuss

The Pragmatic Bookshelf

Raleigh, North Carolina

The Pragmatic Programmers

Pragmatic exPress

Modern CSS with Tailwind

Flexible Styling Without the Fuss

Noel Rappin

edited by Katharine Dvorak

Modern CSS with Tailwind

Flexible Styling Without the Fuss

Noel Rappin

The Pragmatic Bookshelf

Raleigh, North Carolina

The team that produced this book includes:

CEO: Dave Rankin
COO: Janet Furlow
Managing Editor: Tammy Coron
Development Editor: Katharine Dvorak
Copy Editor: Corina Lebegioara
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: 978-1-68050-818-5
Encoded using the finest acid-free high-entropy binary digits.
Book version: P1.0—May 2021

Transitions

In CSS, you can specify that one or more properties should gradually transition when they change values, rather than changing instantly. In a full client-side application, you might change values by using JavaScript to modify the CSS classes on an element. In Tailwind, you can use the prefixes to manage some CSS property changes merely in CSS. For example, an element with a class list of "bg-green-500 hover:bg-yellow-500" will change color from green to yellow when the user hovers over it, and the Tailwind transition utilities can make that happen gradually.

In most cases, you'd declare an element to have a class of .transition, which causes the element to use transition effects for the CSS properties, background-color, border-color, box-shadow, color, fill, opacity, stroke, and transform. Often that's all the properties you want to transition, but if you need to transition other properties, you can use .transition-all to place all properties under the transition banner.

If you want to limit the transition to certain properties, Tailwind provides several choices. Typically you would use these because there are changes in other properties that you want to happen instantly.

.transition-color

.transition-opacity

.transition-shadow

.transition-transform

For the transition to actually be visible, you need to specify a duration over which the transition will take place. The default is 0 (but can be changed in the Tailwind configuration), and Tailwind provides the .duration-{milliseconds} family of utilities, where the suffix is one of 75, 100, 150, 200, 300, 500, 700, or 1000, indicating the number of milliseconds the transition should cover.

You can also delay the start of the transition with .delay-{milliseconds} and the same set of numbers, indicating the number of milliseconds before the transition should start.

By default, the transition is applied linearly, meaning the change to the property happens in a series of identically-sized steps. That default is denoted by the Tailwind utility, .ease-linear. If you want the property change to start more slowly, speed up, and then slow down as it gets closer to the end, you can use .ease-in-out. (Or, you can use either .ease-in or .ease-out if you only want the slowdown on one side of the change.) The ease difference is subtle, but especially with motion, it can provide a sense of a change accelerating and then decelerating in a way that can look more natural and engaging.

Transformation

CSS allows you to transform the box of an element in various ways, changing its size, location, rotation, or skew. Tailwind again gives you some reasonable defaults, which when combined with transitions and animation can allow you to build some great effects easily.

All of these transformation utilities need to be combined with `.transform`, which in turn needs to be declared as a class in the same element, as in `"transform scale-100"`.

Changing the Scale

Tailwind lets you change the scale of an element with the `.scale-{percentage}` family, where the suffix is the percentage to scale. The options are 0, 50, 75, 90, 95, 100, 105, 110, 125, and 150, which are, I think, designed to allow for subtle effects like `"transform transition duration-1000 hover:scale-110"` (which would make an element get slightly bigger on hover over the course of a second). Add in `hover:box-shadow-lg`, and it'd seem like the element was getting closer to the user on hover.

If you only want to scale in one direction, you can use `.scale-x-{percentage}` or `.scale-y-{percentage}` with the same set of numbers (`scale-x-95`, `scale-y-125`, and so on).

Rotating

You can rotate an element with `.rotate-{degrees}`, which is a clockwise transformation of a number of degrees. The options are 0, 1, 2, 3, 6, 12, 45, 90, and 180. A counter-clockwise rotation is achieved with `.-rotate-{degrees}` and the same numbers.

Again, the design here is to make it easy for small effects. The rotation is, by default, around an axis in the middle of the element, which Tailwind denotes as `.origin-center`. You can move the origin around by adding the suffixes for the same four directions and four corners that you've seen elsewhere to `.origin-` (for example, `.origin-top`, `.origin-bottom-right`, and so on).

Skew and Translate

For skew, you have `.skew-x-{degrees}`, `.-skew-x-{degrees}`, `.skew-y-{degrees}`, and `.-skew-y-{degrees}`, which take the numerical suffixes 0, 1, 2, 3, 6, or 12, as the number of degrees in the skew.

You can flat out move an element with .translate-x-{size}, .-translate-x-{size}, .translate-y-{size}, or .-translate-y-{size}, each of which takes a numerical suffix. This moves the element along the direction using the same number scale you've seen for padding, margins, and the like, where each number represents 0.25rem. Positive directions are right and down, and negative directions are left and up.

In addition to the number set, you have as suffixes px for a single pixel, full for "move this the exact amount of its size in that dimension," and 1/2 for "move it half the amount of its size in that dimension," as in translate-x-full or translate-y-1/2.