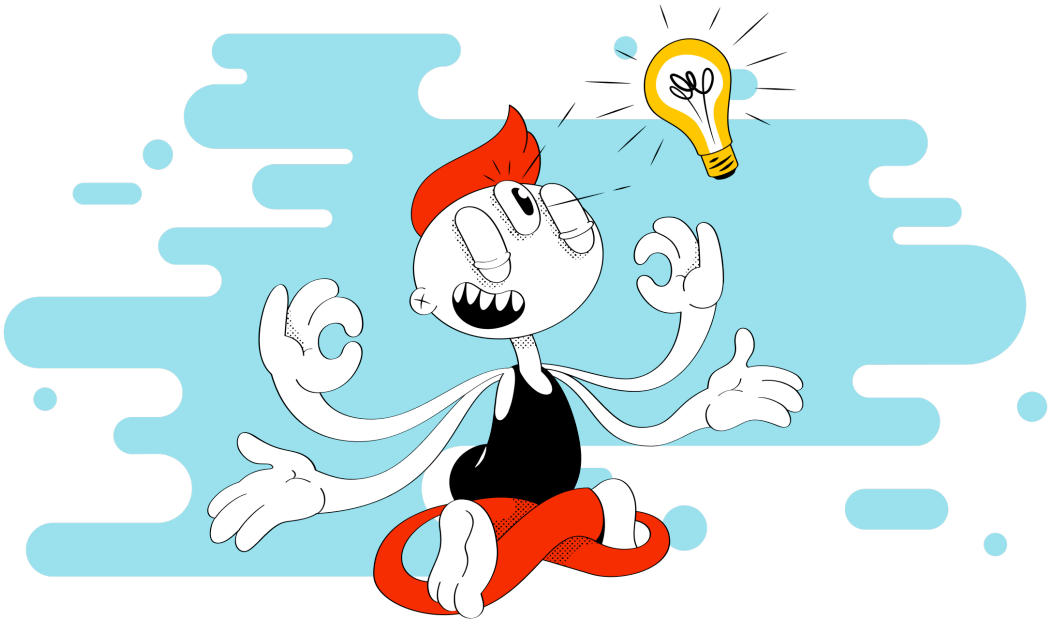
Thoughts on Tailwind CSS

Embracing the Hype: What CSS utility libraries gets right

Written by Seth Corker on Benevolent Bytes blog



What is Tailwind?

Tailwind CSS is a utility-based framework for CSS. If you've been a frontend developer for a while, you may remember some similar frameworks in the past like Tachyons, or may have heard of functional CSS. It's a concept that all of a sudden, has gained a lot of traction in the culmination of a Tailwind. It goes against the common approaches we've been using in CSS for years and isn't just another method for naming conventions. Tailwind is a well-designed framework that sets out a group of utilities for using CSS without necessarily maintaining it as you might be used to.

When building websites with large teams, you need a convention to avoid your stylesheets descending into chaos. Traditionally, we've chosen systems like BEM or OOCSS. These conventions force developers to come up with names that are con-

sistent and help a great deal as a project grows. These naming conventions don't fundamentally change the way you write CSS, just how you name and compose your classes. In more recent years, if you're a React developer, you may have been avoiding these systems entirely by going down the CSS-in-JS route instead. This has the benefit of flexibility and power but has a host of other issues like gatekeeping CSS from people who aren't familiar with JS or React or sacrificing performance. So, how is Tailwind different?

Tailwind CSS differs in a few key areas. The first is the breadth of utilities. The idea of tailwind is to empower developers to work quickly and if you know CSS you most likely guess the names of most of the utilities and be productive after a short session and peek at the docs. Another key area where Tailwind is excellent is documentation, anything you want to do Tailwind will probably support it, show you how and show what the utility classes map to in actual CSS. All this sounds great, but what's the catch?

Why is it controversial?

Tailwind CSS is somewhat controversial, it seems to have an equal number of advocates to opponents. I was on the fence about whether it was worth my time or if it is just another web development fad. After repeated exposure to it, I can now see its place in the ecosystem, and now I even use it for side projects. The core controversy comes from the look of it and how it goes against the idea of CSS. It goes against the 'cascade' part of CSS, it's what makes styling on the web so powerful but equally challenging as a project becomes large. Instead of having multiple classes attached to an element that encapsulates the style, Tailwind proposes each class is almost just like a single property on a rule in CSS.

Making a box and shadow might look like `<div class="shadow-sm bg-white rounded-lg h-18"></div>` where each class performs a very specific function. This is where Tailwind is the most awkward to get used to, in the past you may have done something like `<div class="shadow-box"></div>` and then updated the rule whenever you pleased without touching the HTML. Now, if you want to make all the boxes you've styled a different colour, you're going to be working solely in the HTML to change each class. As with anything, it has tradeoffs. By choosing Tailwind CSS, you're buying into diluted HTML and duplication to gain something else you value more, like a consistent toolset with good defaults that's also extensible. Pick what you value most and see if the downsides are worth it.

Is it the new Bootstrap?

No. Done. I guess you probably want more than that right? Frameworks like Bootstrap have been really valuable in the past and there are plenty of examples of competing frameworks like Foundation or Material Design. As a developer, you choose a different CSS framework/component library to get a different feel. You gain convenience, a framework has a preset library of components that look great and don't require you to touch CSS at all. The tradeoff is that everything looks the same, your Bootstrap website probably looks like everyone else Bootstrap website. If you want to customise anything, you really have to go out of your way to do it.

Tailwind CSS is different because there are no components. There is a predefined set of rules and conventions and utilities but no dropdown or button. You make your own. This is why I think in 5 years time you'll see websites built with Tailwind, but they won't immediately stand out like Bootstrap stands out on the web today. The only caution I have is that there are some things that may become noticeable if you never touch the Tailwind config and customise your theme, colours, and shadows. These are a few of the areas that you'll need to tweak, the colours Tailwind chooses by default are great but in doing so they, as it grows in popularity, they have defined the palette of the web for the foreseeable future.

What I like about Tailwind

Experience

Tooling and documentation makes Tailwind easy to use which is the most valuable when you have limited time. If I wanted to learn the ins and outs of Bootstrap from scratch, I'd be constantly searching for just the right component, the additional class to add to it to give the right feel and the intricate details about how best to use it in the context of a wider design. Tailwind is an abstraction on top of CSS that sits just above it. You need to know CSS to use Tailwind because the naming is similar, and the rest is good defaults and a constrained system of values. Once you know a bit, you can get really far without reading much documentation. When you want to do something more advanced, make customisations or are curious then you can take a look at the docs, and they're wonderful. They tell you what a utility maps to in the actual CSS produced, any variables that influence it and good explanations about how it works. This combined with the editor extensions makes writing tailwind a breeze. Autocomplete works great, and it feels like magic when

create a custom colour in your theme and see it show up with all the expected utility classes in the editor. It's a great developer experience.

Consistency

I love CSS. It's a tool many people struggle with and is much maligned, but I really enjoy writing CSS, I hope I'm not the only one! Over the years though, I've found that CSS isn't usually a big problem but the surrounding conventions can be. When you're working on a website and want a consistent look and feel, you end up deciding on a core set of design tokens. This means colours, spacing, the border thickness, preferred border radius etc. This is where design is essential, you can't make these decisions and hope to have something consistent at the end of it. Often what actually happens, though, is some decisions are made and some aren't. For whatever reason you're tweaking the border colour for just this one button and that's okay. This can add up over time to the point where you have 5 ways of doing box shadows, 3 different greys for borders and 100 different values for spacing between elements. This consistency is difficult to maintain and without a constant watch, it's bound to go awry.

In my experience so far with Tailwind, it seems to tackle this challenge quite well. It's a constrained system. You can't do everything you can with CSS, but that's the point, the set of constraints make it easier to stay consistent. You're spacing will always have the rhythm that Tailwind provides, the box shadow will stay consistent, the colour palette is limited, and your choices will be consistent too. Of course there's room for breaking the rules and going off book, but it's far easier to reign in compared with what you might be accustomed to.

What I don't like

With so much to like, what don't I like? Overall, I think Tailwind is great, but there's a couple of things I still don't like.

Messy HTML

In my ideal workflow, HTML (or JSX) is written first to be semantically correct and organise the hierarchy of information in a component or on a page. CSS is then used to spice things up and make the component look visually appealing. My favourite way of writing CSS is actually scoped to a particular component like Svelte or CSS Modules this usually solves the biggest challenge when working with CSS

in component-based frameworks. Scoping CSS to a component makes it more difficult to break things and has a nice side effect that your names can be concise.

Take the following example:

```
<article class="snippet">
      <header>
            <h1>A great article</h1>
            <span class="chip">New</span>
            <time>20th April 2021</time>
      </header>
      <p class="body">Some text...</p>
 </article>
```

From just looking at this snippet I don't know what it looks like visually, but semantically it's understandable, and the classes don't get in the way. They are sprinkled where needed.

Let's look at how this might go in Tailwind:

```
<article class="p-3 my-2 mx-3 bg-gray-50 border-gray-200
rounded-md shadow-md">
      <header class="flex flex-row items-baseline">
            <h1 class="text-2xl">A great article</h1>
            <span class="bg-red-700 text-white rounded-
md text-sm p-1 h-6 mx-2">New</span>
            <time class="text-gray-700">20th April
2021</time>
      </header>
      <p class="mt-4 mb-2">Some text...</p>
 </article>
```

Here's how it might look in Tailwind, the semantics haven't changed, but the content is now more difficult to pick apart from the elements as Tailwind has polluted the markup with lots of class names.

Tooling required

You can use Tailwind without tools, but you really shouldn't. To achieve a utility class names, each colour in the theme contributes to a set of utilities to generate every permutation you'll need. Every size does the same thing. A class like the set of grey values will end up creating classes like `bg-gray-50` and `text-gray-50` along with the utility class for gradients etc. This can lead to a massive CSS file. The way around this is to use PurgeCSS to remove any unused styles. The challenge is that to reduce the size of the default Tailwind bundle in development, certain features are disabled. You'll notice hover might not transition as expected. It's only enabled for certain properties. Tailwind advertises dark mode, but, it's disabled by default. Everything you add causes your development CSS to grow. This can make development slower and again, relies on tooling which can be restrictive for developers that just want to grab something and go.

Where I think Tailwind shines

There are some great benefits when using it yourself on your projects, but I think it really shines when used with others. Teams can benefit from Tailwind because it acts a foundation of shared understanding. I see setup of the theme at the beginning of the project to set some brand colours, get the shadows to look right and figure out what features will be needed to start with. When you're using it day to day with other team members, Tailwind can be the shared language of styling. You're not arguing over whether to use `px` or `rem`, the benefits of a `rgba(0,128,45, 0.5)` and what fallbacks to use. You'll stop wondering why there are tweaks to move things over an extra pixel. Tailwind will take care of that and design inconsistencies should be handled by using a design system as you normally would.

There is value for teams all working on one project, but there's greater value for teams working on many projects. A developer switching between teams doesn't need to figure out the specifics of how that team does CSS, which naming convention they've used and which classes work best in different circumstances. They have a shared language there too. Agencies working on many client projects could use Tailwind to accelerate their workflow as they move between projects without skipping a beat.

Conclusion

Tailwind CSS is a controversial framework that I backed at a few years ago, just like Tachyons, so what. It's a framework I didn't see myself investing time in even 6 months ago but as time has gone on, I've seen the value in adopting Tailwind. It solves a specific set of problems and the value it brings outweighs the tradeoffs for the projects I work on. If you're on the fence, I suggest giving it a go. There's a bit of a learning curve but once over the first hump, you can grasp it pretty quickly and be productive. If you are going to try it out, I recommend using it with a component-based framework to mitigate the downsides. If you're not using a component-based library, I'd opt for using `@apply` to reduce duplication for key components in your design like buttons, links, headers etc. Overall, whether you should invest time in Tailwind is up to you and your team. There are definite downsides, but it comes down to if you think the value of consistency, developer experience and constrained flexibility is worth it.