netlify create
formerly Stackbit

10 Reasons to Choose Tailwind CSS

Oct 12, 2021 |

altText of the image

We're using Tailwind CSS for our component library. Here are 10 reasons we think it'd be a good fit for your next project.

We might be biased, but we're all in on site builders as one of the modern web's most transformative technologies. But as lumber shortages around the world proved in 2021 ... you can't build much of anything without the necessary materials. To make our site builder even more powerful, we've begun work on our very own component library — *heroes! calls to action! forms!* — which will be added to an updated set of Stackbit themes later this year.

Our component library is open-source so you can customize it to match your project's needs. One of the first things you might notice when looking at the code is: we're using Tailwind CSS.

After evaluating all our options, we felt like building on top of Tailwind CSS would strengthen our library, while improving the developer

experience of using our components with our themes.

We know, we know. This is a polarizing decision. You can find endless articles by developers who **love**, **hate**, and "**meh**" Tailwind CSS. That's why we want to share with you why we've decided to work with Tailwind.

What is Tailwind CSS exactly?

Tailwind is a utility-first CSS framework that is used to rapidly build modern user interfaces without worrying about naming conflicts. Utility class names describe what the classes *do,* like set padding to 4px, rather than where they *go*, like your topnav. The extensive but carefully curated utility classes of Tailwind make it super simple to create beautiful and responsive websites without ever writing a single line of CSS.

Here are the top 10 reasons that Stackbit chose to style its component library with Tailwind CSS:

1. Tailwind is a Time-Saver

Tailwind's utility classes remove the headache of writing and maintaining huge piles of CSS from scratch.

A utility class of Tailwind corresponds to a small set of CSS declarations. For example:

- p-4 means padding: 1rem; (number suffixes in Tailwind are often multiples of 0.25 ***rem***)
- m-4 means margin: 1rem;

- text-lg means font-size: 1.125rem; line-height: 1.75rem;

Putting it all together, this HTML with Tailwind CSS enabled:

Copy

```
<div class="p-4 m-4 text-lg">Hello world</div>
```

Is equivalent to this basic HTML:

Copy

```
<div style="padding: 1rem; margin: 1rem; font-size: 1.125rem; line-he
```

With the help of these utility classes, we don't have to build everything from scratch, and we can create better websites in less time.

2. Utility Classes Follow the Single Responsibility Principle

Tailwind's utility classes are designed to serve one specific purpose. For example:

- w-4 means width: 1rem;
- bg-transparent means background-color: transparent;

Few of the utility classes also encapsulate more than 1-3 CSS declarations at a time.

These classes are almost like shortcuts for coding directly with style attributes. For years, this was generally considered to be frustratingly verbose to the point of being a bad practice, but lately, many

developers have argued that once you accustom yourself to the utility class style, you'll find that it means your HTML elements don't interfere with each other, which simplifies debugging.

For example, when there's no site-wide .card class, and no site-wide .featured class, there's no guessing what's wrong with .

Tailwind CSS includes thousands of such independent utility classes, and you can combine them creatively to build highly customized website components that "just work," no matter where you put them on a page.

3. Documentation is Top-Notch

Tailwind does a great job not only with the simplicity of its framework but also with the simplicity of its documentation.

It's a great place to start learning about how the framework works. You can easily navigate through any topic on their website. Searching width will take you to the width guide, but so will searching w-8, so it's easy to navigate, even for developers who are new to CSS frameworks.

4. A Large and Growing Community

It's always a good idea to judge the popularity of a framework before shifting to it. For frameworks with a solid following, it's easier to find great developers and supportive online communities to help when you're stuck.

Tailwind's immense popularity in the developer community means potential problems can be solved more easily with community support.

5. No More Media Queries!

Goodbye, media queries!

Every utility class in Tailwind can be applied conditionally at different breakpoints, which means we don't have to create separate media query files to make our component responsive.

Tailwind, by default, provides 5 different breakpoints:

- sm: 640px
- md: 768px
- lg: 1024px
- xl: 1280px
- 2xl: 1536px

Let's see this with an example:

Copy

```
<img class="w-16 md:w-32 lg:w-48" src="...">
```

Our image has been given a default width of 4rem, which will be overridden by 8rem and 12rem when the viewport grows to the md and lg breakpoints, respectively.

If the default breakpoints are not enough, we can also create custom breakpoints.

6. It's Trivial to Create Component Styles

Tailwind does not come with reusable component styles like .modal or .navbar, but it lets us create reusable custom classes out of other Tailwind classes with its @apply directive.

Let's create a custom utility class for ourselves to learn how this actually works:

Copy

```
.card {
  @apply w-20 h-20 bg-blue-700;
}
```

Superb! Now we have created our own custom .card class in Tailwind, and we can freely use it as if it were a Tailwind class like any other. *(Including as part of the definition of another class with @apply.)*

7. Purging Unused Classes Minimizes Bundle Size

To make our development experience smooth, Tailwind generates thousands of utility classes, most of which go unused in the production build.

To overcome this problem, Tailwind has a "purge" feature that removes every unused utility class from the production build.

Purging unused classes dramatically reduces the file size of the CSS shipped to our users, which ultimately resulting in a site that loads

faster.

8. Style Opinions are Productive

Like many other frameworks, Tailwind bring opinions with it. One example is the sizing factors used. .m-4 means 1rem and m-5 means 1.25rem. So what about the 0.25rem in between?

We consider these to be productive opinions because they promote consistency in our design. There's very rarely a need to be more fine-grained than the options we're given with Tailwind. But, if we need to be, it's configurable and it's just CSS, so we can always set our own styles.

9. Avoiding the Need for Naming Conventions

Naming classes in CSS is a big headache. (Seriously. Have you ever been in a meeting dedicated solely to discussing class names? We have. And we don't want to do that again.)

Although there are naming convention guidelines, like BEM, Tailwind wins the battle by removing the need to write custom CSS classes in most common scenarios. Often its utility classes are enough to build your design to life.

10. It Feels Like Writing CSS

Tailwind makes it easy to create amazing user interfaces for both junior and senior developers with the help of utility classes. As mentioned in #2, the utilities are mostly just a class reference to a specific style. It quickly feels just like writing CSS, compared to learning a framework like Bootstrap, which includes learning a system of named components.

Still, It's Not Perfect

Tailwind, like any other framework, isn't perfect. The most common complaint is that the long strings of classes can become difficult to read, even in common use cases.

But after evaluating our options, we felt like the positives of Tailwind strongly outweighed the negatives. *We'll take messy (but structurally-sound) markup over messy CSS any day.*

And we're super excited to share the result of using Tailwind with our component library a little later this year. **Don't be a stranger**. We're using Tailwind CSS for our component library. The reception of Tailwind is polarizing and we're on the side of love (well, *most* of us are). This is why we think it'd be a good fit for your next project.

Ashutosh Mishra

Need help?

Talk to an expert

START CREATING

Docs

Changelog

Legacy Changelog

COMPANY

Contact

Terms

License

Privacy