
Jason Arnold Follow
May 6, 2017 · 5 min read

Getting started with SASS

The first questions I asked myself when starting to use SASS were what is it and why should I use it? I try to not change my workflow whenever I hear about a new language or technology because if I'm constantly changing things around then I'm fighting that workflow more than I am building new things.

Having said that, however, I do also think it is a good idea to check out something new from time to time to keep your skills fresh and to make sure you are using the best tools for you.

What is SASS?

First off, it might be a good idea to explain what SASS is and what it does. The browser can only read CSS and those CSS files are the ones linked in the HTML files. So that part doesn't change. What does change is that you no longer write CSS directly. You write SASS and it is then preprocessed into a CSS file. That file looks like just any CSS file, but it is just created on-the-fly from the SASS file.

SASS looks very similar to CSS but does have some differences. There are no curly braces `{}` or semi-colons used so there are less syntactic errors to worry about. Also, SASS uses indentation and nesting so you no longer have descendent selectors that dig several layers deep into a site.

Why SASS?

I've heard about SASS for a while now but hadn't yet taken the time to get to know exactly what it is or how to use it. When learning a new language, I usually start at the language's website to get a feel for it. Sass's website calls it 'CSS with superpowers' and gives this definition:

> *Sass is the most mature, stable, and powerful professional grade CSS extension language in the world.*

Sounds cool. But why should I use it over vanilla CSS?

The reasons given on the site are that SASS adds features to CSS that help to keep it organized and more flexible. One of the great things about CSS is that there is really no method that one must follow when using it. If you know the basics of the language you can write the CSS you need. It gives you a lot of freedom. This can also quickly become a bad thing since your CSS files can get out of control. Even in modestly sized projects, CSS files can get very big, disorganized, and hard to maintain. SASS's features are here to help with these problems.

I just started digging into SASS but two of the features that I'm impressed with so far are variables and nesting.

Variables in SASS are similar to variables in any other programming language. You define one to hold a piece of information you want to use throughout your SASS file. The first thing I think of here is keeping track of colors on a site. If I'm reusing text, background, or border colors, I don't want to have to keep referencing a list to keep track of which one is which. Or If I need to change one of those colors, I then have to find everywhere in the CSS file I've used it. With a SASS variable I set it one place and that is it.

Nesting in SASS is another feature that helps with keeping a file more organized and easier to read. If you have a complex site and need to set a CSS property on a deeply nested element, then you'll probably have

some pretty nasty descendant selectors to get to those elements. Nesting does away with this. Instead of descendent selectors, SASS uses indentation to determine selector hierarchy. This makes for much cleaner, more maintainable code.

Other SASS features include things like partials and imports which allow you to make your CSS more modular and break larger files down to smaller sizes. Also SASS mixins allow you to create CSS declarations that you can use throughout your code.

So, SASS offers quite a few features that vanilla CSS doesn't. So, if this has convinced you to give SASS a try, then let's get started adding it to a project.

Installing and Configuring SASS

You can install SASS through the command line just like most other tools. SASS is installed through a Ruby gem.

```
gem install sass
```

After SASS is installed, you have to tell it where the SASS files are located and then where the CSS files should go when compiled. The syntax for this is pretty simple, you call the `sass` command and then provide an input file and an output file.

```
sass style.sass:style.css
```

Another option for compiling is to add a 'watch' flag so that SASS watches a directory or an individual file and whenever that file changes, the CSS file is automatically compiled.

```
sass --watch style.sass:style.css
```

When you watch files, the terminal will pause until you change the SASS file. When you do you'll see either of these messages depending on if you are using `sass` or `node-sass`

```
// ♥ sass --watch style.sass:style.css
>>> Sass is watching for changes. Press Ctrl-C to stop.
>>> Change detected to: style.sass
      write style.css
      write style.css.map
```

Watch message in sass

SASS in action

Now that you know how to get SASS set up and running the rest is pretty easy. Just create a `.sass` file in your project directory and write your SASS there. When you run the processor (or the file is being watched for changes) the code in that file will be changed to vanilla CSS.

Here is an example I wrote that creates a really basic navbar with a logo on the left and three links along the length of the rest of the page.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <link rel="stylesheet" href="style.css">
    <title>SASS</title>
</head>
<body>
    <div id="navbar">
        <div id="logo"></div>
        <div id="links">
            <ul>
                <li>About</li>
                <li>Blog</li>
                <li>Contact</li>
            </ul>
        </div>
    </div>
</body>
</html>
```

The HTML for the example. Note that the <link> tag is pointing to the usual .css file.

```sass
$logoColor: tomato
$divBorder: black 2px solid

#navbar
    display: flex
    flex-direction: row

    #logo
        background: $logoColor
        height: 100px
        width: 100px
        border-radius: 50%
        border: $divBorder

    #links
        width: 100%

        ul
            display: flex
            flex-direction: row
            justify-content: space-around

            li
                list-style: none
```

SASS code

I created several nested sections for this page so you could see that part of SASS being used. I also included a couple of variables at the top of the file. Here is what the CSS looks like after all of this.

```
#navbar {
  display: flex;
  flex-direction: row; }
  #navbar #logo {
    background: tomato;
    height: 100px;
    width: 100px;
    border-radius: 50%;
    border: black 2px solid; }
  #navbar #links {
    width: 100%; }
    #navbar #links ul {
      display: flex;
      flex-direction: row;
      justify-content: space-around; }
      #navbar #links ul li {
        list-style: none; }
```

I've just started using SASS, but I already see some of the great things that it can provide and how it can help my workflow in future projects. I hope that this post has done the same for you. Please leave any questions or comments below. Thanks!

If you want to see the code for this post, you can find it here.