Do you still need Sass in 2023?

Twin Cities Drupal Camp 2023

Aubrey Sambor

Lead Engineer at Lullabot

Lives in Northampton, MA

25 years of CSS experience!

Find me online

Mastodon
@starshaped@labyrinth.social

Blog
https://star-shaped.org

Drupal.org
https://www.drupal.org/u/starshaped

What will we talk about today?

- An overview of native CSS features that replace Sass functionality
- Using PostCSS to implement up and coming CSS features that aren’t supported yet in all browsers
- Ways to add native CSS to your Sass code to start using modern CSS!
- *Should* you still use Sass in 2023?

What can CSS do in 2023?

Custom properties!

- Available in modern browsers since 2016, can be used in place of Sass variables
    - --my-awesome-variable, to use, var(--my-awesome-variable)
- Since they aren't compiled into CSS like Sass variables are, they can be changed on the fly using Javascript
- Can't be used as a media query variable, but we'll talk about this later!

Codepen

Math in native CSS

- Yes, CSS has the ability to do math!
  - `calc()`: around since 2012
  - Trigonometric functions: new in 2023! Can do cos(), sin(), tan(), and so on.
  - Clock face trigonometric function example

- Sass also has these math functions!
  - `math.cos(100deg)` in Sass is the same as doing `cos(100deg)` in native CSS
  - Trigonometric functions in Sass example
  - calc() also works in Sass

Color functions!

- `color-mix()`: Supported in all major browsers, `color-mix()` can be used in place of Sass's mix() function.
    - Percentage of the mix is reversed between the two methods
    - Can't use a CSS custom property in Sass color functions, bummer
- Let's take a look at how this works in native CSS vs Sass!

Codepen

Nesting!

- Supported in Chrome and Safari ~~but not Firefox 😭~~ AND now, Firefox 117!
- In Safari and Chrome, syntax is almost the same as Sass syntax with one notable exception:
    - A nested element selector needs to be preceded by an & due to limitations in browser parsing engines

Nesting! (continued)

```
article {
  p {
    color: rebeccapurple;
  }
}
```

is invalid in Safari and Chrome. Instead, write

```
article {
  & p {
    color: rebeccapurple;
  }
}
```

Codepen

What if I want other Sass-like functionality on my site?

Enter: PostCSS!

What is PostCSS?

- A NodeJS tool to transpile CSS using various plugins
- Some plugins replicate what is coming from the CSSWG, acting as a polyfill of sorts until all browsers support the feature
- Other plugins add nice to haves to CSS
- Others replicate Sass features or other features that may never be added to the CSS specification

PostCSS plugins with future CSS syntax

- postcss-nesting: follows the CSS Nesting Specification
- postcss-custom-media: defines @custom-media following the Custom Media Specification
- postcss-media-minmax: adds range notation defined by the Media Queries 4 Specification
- postcss-preset-env: collection of plugins all with future CSS syntax!

PostCSS plugins with useful functionality

- cssnano: CSS minifier built on top of PostCSS
- postcss-pxtorem: converts px units to REMs
- postcss-import: transforms import rules by inlining styles

PostCSS plugins that replicate Sass functionality

- postcss-mixins: plugin that replicates mixin functionality in Sass
- postcss-map-get: replicates Sass’s map-get functionality
- postcss-define-function: replicates Sass function… functionality
- postcss-nested: replicates Sass’s version of nesting (differs from the CSS spec!)
- List of PostCSS plugins from the PostCSS website

postcss-nesting vs postcss-nested

- Why are there two different nesting plugins and how are they different?

- `postcss-nested`follows the Sass specification for the & element
  - You don't need to add the & before an element selector like in Sass

- `postcss-nesting`follows the CSSWG specification for the & element
  - As I mentioned earlier, you will need to add a & before any element selectors

- I use `postcss-nesting`because I'd rather stay with the CSSWG recommendations as much as possible!

PostCSS in action!

PostCSS downsides

- You might get used to plugins that don’t follow CSS specifications, and once those specs get added to CSS, you’ll have to update your code to follow the ‘real’ specification.
- Having too many plugins might affect performance
- Setup can be confusing depending on the technology you use
  - I use Eleventy to power my blog and I’m able to build my PostCSS using the eleventy.js file, but you may also use older task runners like gulp or grunt or use an npm/yarn script to parse your configuration file

Okay, this is great. What if I still want to use Sass, though?

You can use Sass AND native CSS together!

Using Sass and native CSS

- You can use custom properties with Sass variables
  - Depending on what you're doing, you can mix the two. You might want to use CSS custom properties for most of your code, but need to switch to Sass to use mixins and maps
- Sass has its own version of calc which works with Sass variables
  - As of dart-sass 1.40.0, you don't need to interpolate your Sass variables to use within calc()! (CodePen uses an older version, however, so my examples still use interpolation)
  - You can use calc() in Sass to get around having to use math.div for the division operator

Codepen

So, SHOULD you use Sass in 2023?

It depends!

Reasons to still use Sass in 2023

- Sass isn’t going away anytime soon, so use it as much as you like!*
- You don’t feel like native CSS has quite caught up to Sass yet
- You make use of complex mixins, functions, and maps
- You still need to support legacy browser versions
- Updating your codebase to remove Sass would be too time consuming
- When you want to!

* as long as you’re using dart-sass, that is!

Wrapping up...

- It’s a great time to be a front end developer with so many new additions to CSS:
    - Custom properties, math functions, color functions, nesting

- PostCSS can enhance native CSS by:
    - Adding CSS that might not be supported by all modern browsers
    - Adding useful functionality such as converting px to rem and CSS minification
    - Adding functionality Sass supports today, such as mixins and functions

- You can use Sass with modern CSS if you want to gradually switch your codebase from one to the other, but…

- You should still use Sass… **if you want to!**

Further reading

- https://gomakethings.com/is-it-time-to-drop-sass/
- https://dev.to/robole/do-not-drop-sass-for-css-1ofm
- https://crinkles.dev/writing/going-back-to-css-only-after-years-of-scss/
- https://chriscoyier.net/2023/07/11/sass-features-in-css/
- https://www.sitepoint.com/postcss-sass-configurable-alternative/
- https://kilianvalkhof.com/2023/css-html/the-gotchas-of-css-nesting/
- https://ryantrimble.com/blog/mixing-colors-with-css/
- https://syntax.fm/show/603/can-vanilla-css-replace-sass-yet

Aubrey Sambor

Lead Engineer at Lullabot

Lives in Northampton, MA

25 years of CSS experience!

Find me online

Mastodon
@starshaped@labyrinth.social

Blog
https://star-shaped.org

Drupal.org
https://www.drupal.org/u/starshaped

Tada 🎉