2015-11-05, Revision B
Blekinge Institute of Technology
Advanced topic in Web development

CSS Preprocessors are cool

Kalle Kihlström

Kalle Kihlström
Web Programming
2013, BTH, Blekinge institute of technology
kalle_kihlstrom@hotmail.com

Abstract

In this article I go through and cover two of the major CSS preprocessors Less and Sass. What features do they offer and which one is actually the best? I'll go through everything from installation and setup to syntax and performance.
I'll also discuss whenever it's worth going over to a CSS preprocessor or continue to use standard CSS. So welcome to the world of CSS preprocessors.

1. Introduction

1.1 Background

CSS (Cascading Style Sheets) is a stylesheet language used to format and style documents written in a markup language. CSS is and has for a long time been a very important technology in web development and is used in almost every website and web application today.

Since the first release of CSS back in 1996 it has evolved a lot and is still doing so. With the addition of the CSS2 and CSS3 specifications we've got new things like media queries, namespaces and additional selectors, but even with these and a lot of other additions, CSS is still lacking some key features and functionality that we are used to from other programming languages.

One possible solution to make up for the limitations in the CSS language is to use something called a preprocessor. A CSS preprocessor converts data written in the preprocessors syntax to ordinary CSS syntax and acts as an extension of the CSS language. Therefor a preprocessor can add additional features and functionality for the developer to use and in the end the preprocessor compiles the code to regular CSS.

1.2 Intention

In this article I plan to go through the features of CSS preprocessors, advantages and disadvantages of using one. I'll cover the two most popular CSS preprocessors, and compare them to each other.

1.3 Method

I'll start by gathering information on CSS preprocessors in general as well as advantages and disadvantages of using one. Then I'm going to select two CSS preprocessors to investigate more in detail. The two most popular preferably.

I'll then research and gather information on each preprocessor and ask the following questions:

- Which limitations of standard CSS does this preprocessor eliminate (added functionality)?
- What does the syntax look like, and how is it to work with?
- How is the performance?
- How is it installed and how is the usability?
- Does it have any limitations?

In the end I'll write a piece about every preprocessor, compare the two preprocessors based on earlier findings and answer the second research question (which preprocessor is the best?) and also answer to the first research question based on my research and discoveries.

2. Results

2.1 Selection of pre-processors

I decided to cover the two most used and most popular[1] CSS pre processors in this article Less[2] and Sass[3]. These two were also the first CSS pre processors that was supported by Codepen.io[4] (a CSS/JavaScript/Markup web based playground for designers). They are also the two CSS preprocessors with the highest star count on GitHub[5] at the time being, with Less sitting close to 13000 stars and Sass at nearly 7000 stars.

2.2 Less

2.2.1 About

Less was first created back in 2009 by Alexis Sellier. Less was originally written in the programming language Ruby but was later translated into JavaScript. The main development has since continued in JavaScript.
Less is today maintained by a core team consisting of seven members who donates the time and effort they afford. Another big part of the continued development of Less is the community who contributes to and supports Less.
Less has since the start been ported to many other languages including Java, .NET, PHP, Python and ruby.[6]

2.2.2 Features

Variables, Less supports variables a basic missing feature of CSS. Variables in Less are constants meaning they can't be over-written. Some values are often repeated many time in a stylesheet, variables make it easier to maintain and control these values. You can assign nearly anything to variables in Less and use them in most contexts such as in selectors, imports, URLs, properties and values.

Mixins, Another addition added in Less is Mixins. Mixins (short for "Mixing in") are defined of a bunch of properties which later can be reused in other rule-sets by calling the name of the mixin. Less mixins does also support incoming arguments which can make them even more reusable.

Nested rules, Another feature added with Less is nested rules. This eliminates the need of repeating the parent(s) element(s) every time when working with nested element structures.

Operations, Less supports mathematical operations such as addition, subtraction. multiplication and division. Less supports operation on numbers, colors and variables and converts units before calculating, if a conversion not is possible Less will ignore it from the result.

Functions (Built in), Less comes with many built in functions for you to use right out of the box, functions for manipulation of colors, doing math, manipulation of strings and much more.

Loops, Less mixins has the ability to call themselves, when combining this with the Less features, pattern matching and guard expressions it's possible to create loop structures.

Comments, Less supports both one line comments ("//") as well as block comments ("/**/").

2.2.3 Syntax

Less syntax looks a lot like the CSS standard syntax with the addition of variables, mixins and other additions to the language..
Below are a few examples of the Less syntax.

Variables and operations

```
@red: #ff0000;
@another-red: @red + #111;

#header {
  color: @another-red;
}
```

Mixins, single line comment and built in function

```
// Border mixin
.bordered(@color) {
  border-left: ridge 3px @color;
  border-right: dashed 2px grey;
}

article {
  color: lighten(red, 25%);
  .bordered(@color);
}
```

2.2.4 Setup & usage

The most common and straightforward way of using less is through Node[7]. The installation process is therefore really easy and compatible with most platforms (Linux, Mac, Windows) and the only dependency is Node.js and Node.js package manager npm[8]. Less can be installed globally with the following command:

```
npm install less -g
```

You can now use the following line in your CLI to compile .less files to .css

```
# compile style.less to style.css
$ lessc bootstrap.less bootstrap.css
```

Less can also be compiled on the client side, through the browser. This is only recommended during development though, since it can slow down the loading time of the page a fair amount.
To get Less compiling in the browser you need to include the less script found on the official Less site [9].

```
# Less files to be compiled
<link rel="stylesheet/less" type="text/css" href="style.less" />

# Include the less compile script
<script src="less.js" type="text/javascript"></script>
```

Another way of using and compiling Less is through a GUI application, There are several of them out there. You can also compile Less through different online compilers.
There are also several online playgrounds that supports less such as Codepen[10] and JsFiddle.[11]

2.3 Sass

2.3.1 About

Sass (short for Syntactically Awesome Stylesheets) was started back in 2007 and has since then been active for almost nine years.[3]
The project was initially started by Hampton Catlin and is an open source project run by a core team with help of contributions. The original implementation of Sass is written in Ruby[12], however just like Less, Sass has been translated to many other languages including an official high performance port for C and C++ called libSass, there are also wrappers for JavaScript, Node, .NET, PHP, perl and many other languages.[13]

2.3.2 Features

Variables, Sass has support for variables, variables in Sass are treated like constants just like in Less. You can assign most things to a variable in Sass including font-stacks, colors, booleans, lists, maps and properties.

Mixins, Sass supports mixins.

Nested rules, Nesting is also supported in Sass which makes the code much easier to maintain and saves a lot of time since you don't have to repeat long selectors when working with nested elements. It's also also possible to nest properties in Sass which is a cool and quite unique addition.

Operations, Mathematical operations such as addition, subtraction. multiplication, division as well as modulus are supported in Sass..

Functions, Sass comes with many handy built-in functions ready which are ready to use, but it does also support creation of own functions.

Loops, Sass comes with direct support for loops, both for-loops and foreach are supported.

Directives & Expressions, Another great addition added in Sass is the support of IF-statements. Sass also supports else and elseif in combination with if-statements.

Comments, Just like Less, Sass supports both one line comments ("//") as well as block comments ("/**/").

2.3.3 Syntax

Unlike Less, Sass offers two ways of writing it's syntax, the original syntax (SASS) which was inherited from Haml[14] and the newer SCSS (Sassy CSS) syntax which is more familiar to the regular CSS syntax. The two syntaxes use different file endings .sass for the old syntax and .scss for the new syntax. Both syntaxes are still supported and the plan is to continue support both of them in the future. The biggest difference between the two is that the new syntax uses semi-colons and brackets just like CSS whereas in the old syntax you can simply skip them and instead just indent the code.[3]

So, a few examples of the syntaxes side by side.

Variables and operations

SCSS

```
$red: #ff0000;
$another-red: $red + #111;

#header {
  color: $another-red;
}
```

SASS

```
$red: #ff0000
$another-red: $red + #111

#header
  color: $another-red
```

Mixins, single line comment and built in function

SCSS

```
// Border mixin
=bordered($color) {
  border-left: ridge 3px
$color;
  border-right: dashed 2px
grey;
}

article {
  color: lighten(red, 25%);
  +bordered(red);
}
```

SASS

```
// Border mixin
=bordered($color)
  border-left: ridge 3px $color
  border-right: dashed 2px grey

article
  color: lighten(red, 25%)
  +bordered(red)
```

Custom function

SCSS

```
@function minus-one($value) {
  @return $value -1;
}
```

SASS

```
@function minus-one($value)
  @return $value -1
```

2.3.4 Setup & Usage

To get started with Sass all you need is Ruby. Newer Macs comes with Ruby pre-installed, while if you are running Linux and Windows you'll need to install it manually, but it's pretty straight forward. When Ruby is installed you run the following line to install Sass:

```
gem install sass
```

Just as Node, you use the package manager in this case gem to install Sass.
When the installation is done you are ready to compile Sass from the CLI with the following line:

```
sass style.scss style.css
```

Sass also supports watching files and folders and will auto compile the any Sass when changes are detected. All you need to do is add the --watch flag as an option:

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

There are also several GUI applications and online compilers which are able to compile Sass to CSS as well as online playgrounds.

3. Discussion

3.1 Personal reflections Less

Personally I've spent more time with Less than any other CSS preprocessor. Why? because it was the first CSS preprocessor I got introduced to, I felt like it met my expectations and demands at the time so I found no real reason to look for other alternatives. Also, when you get used to something and are comfortable in an environment or setting it's often harder to take a step away from it and embrace something new. It's not until now, the past half year or so that I've started looking around and testing other preprocessors.

I feel like Less is easy to take in and the learning curve is great, it's easy to get started and start using the basic functionality but there's also a lot more advanced functionality and features that can be learned over time. The syntax is just like CSS with the addition of the new features introduced with Less, this makes it really easy for anyone with CSS experience to get a hang of it and understand most of the code even though they might have limited experience with Less itself. Less is also really easy and fast to setup and it's easy to setup both client side and server side compilation.

One thing that I miss in Less is the ability to write my own functions, although mixins can substitute the need of functions in many cases. This makes the lack of custom functions more bearable.

3.2 Personal reflections Sass

Lately I've gotten more and more interested in Sass after seeing it more often when browsing sites like CodePen.[10]
I first feared that it would be harder to setup and use than Less but this was not the case. The setup is as easy if not easier than setting up Less since Ruby already comes pre-installed on Macintosh computers. I also like that Sass comes with file and folder watch support through CLI built-in.

When it comes to the syntax, I personally really like the old syntax since it's so simplified, removing the semicolons and brackets makes it both faster, prettier and easier to read in my

opinion. It looks a lot like Haml and is similar to python where you also has the ability to skip brackets and semicolons.
I definitely see why many people prefer the newer SCSS syntax though. It's much closer to standard CSS which is a good thing if you are jumping between regular CSS, Sass and/or other preprocessors in different projects, this will makes the transitions much more fluent, it's definitely also easier for beginners and for people with zero SCSS experience since it's so close to standard CSS.

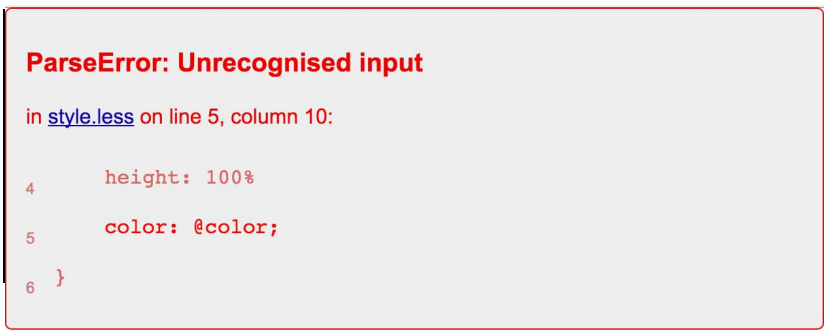
Sass also supports features such as custom functions, for and foreach-loops as well as if, elseif and else statements. These are all very welcomed additions.

3.3 Comparison

Sass and Less are very similar in many ways, especially when looking at the more basic functionality like mixins, variables, operations and nesting. The syntax for these features are almost identical. Although if you chose to go with Sass you also have the opportunity to use the old indented based SASS syntax, additional options are always welcome.

When looking at the performance of the two preprocessors Less beats Sass with quite a large margin when compiling an identical 200KB CSS file. Sass compiled it in 4.9s and in 2.5s with a near to current .sass cache. Less compiles the same file in only 0.5s.[15] You could argue whenever performance is important or not since you generally only compile the files during development. A long compilation time could be quite annoying though when developing, and I feel like as long as the compilation is under or around the 2s mark it's bearable.. 200KB is a quite large stylesheet so take that into consideration when looking at the compilation times.

Poor error messages can be a real pain when developing and debugging. I made a small test scenario to compare the error messages that Less and Sass puts out. As a conclusion I found that the Less error output was much nicer and easier to read while Sass error message is a bit messier and not as easy to read, but still ok.
The first picture below is an error output from Sass and the second picture below is the same error output but outputted by Less.



When looking at more advanced features Sass has more to offer. Sass has built in support for loops and statements which Less is lacking. Most of this can still be achieved in Less but requires you to write additional code and it doesn't look as nice as the built-in features Sass offers.

The support of custom functions in Sass is another advanced feature that Less is lacking.

Both pre-processors are easy to setup and use. Sass depends on Ruby while Less depends on Node.js, both supports Mac, Windows and Linux and both have GUI applications and online compilers for users who find that interesting. They are both ported to many other programming languages and has a wide range of plugins, so no real favour for anyone there. The built-in watch support in Sass is nice, to achieve the same with Less you need a plugin which is easy to install but still an additional step. While Sass has watch support it lacks support of client-side compilation which Less supports through Less.js.

4. Conclusion

You can't really go wrong when choosing between Less and Sass, both are a big step up from CSS and comes with many handy additions. Both are easy to use and setup, both has wide application and plugin support. Both has excellent documentation. Less has prettier Error messages and client-side compilation support from the get go while Sass has watch support. Sass has more features than Less, most of them might not be used regularly but are still nice to have when you need them. On the other side, if you're not a power user you might not even need these features anyway.
All together I feel like Sass is a more competent pre-processor mainly because of its features and more advanced additions it offers over Less.

Less, Sass or another preprocessor is not mandatory to use but I would highly advise everyone working with CSS to use one. The time it takes to setup, learn and use a preprocessor like Less or Sass is just a drop in the ocean compared to the amount of time you can save from using one. No more repetition of browser prefixes or repeating long selectors. No more repetition of the same style-set or color on 20 different places in a file. The ability to write nested selectors is one of my favorite things about CSS preprocessors, It saves a large amount of time and makes the code easier to maintain and read at the same time. It's hard to go back to writing regular CSS after getting used to this.
These are just some reasons why everyone should use a preprocessor like Less or Sass. Another major reason is that it makes it possible to write DRY (Don't repeat Yourself) CSS, which is impossible with just regular CSS.

5. Summary

A CSS preprocessor converts data written in the preprocessors syntax to ordinary CSS syntax and acts as an extension of the CSS language. Therefor a preprocessor can add additional features and functionality for the developer to use and in the end the preprocessor compiles the code to regular CSS. The two largest and most used CSS preprocessors are Less and Sass and they both offer some great additions to CSS. Sass and Less are very similar in many ways, much of the functionality they offer is nearly identical between them. Sass comes with some more advanced functionality such as loops and expressions. In the

end both Sass and Less are mature and feature rich preprocessors and you can't go wrong when choosing one of them, both are great.

5. References

http://www.w3.org/Style/CSS/
http://sass-lang.com/documentation/
http://lesscss.org/features/

[1] https://css-tricks.com/poll-results-popularity-of-css-preprocessors/
[2] http://lesscss.org/#getting-started
[3] http://sass-lang.com/
[4] http://blog.codepen.io/2013/04/08/statistics-on-preprocessor-usage/
[5] https://github.com/showcases/css-preprocessors
[6] http://lesscss.org/about/
[7] https://nodejs.org/en/
[8] https://www.npmjs.com/
[9] http://lesscss.org/#download-options-browser-downloads
[10] http://codepen.io/
[11] https://jsfiddle.net/

[12] https://en.wikipedia.org/wiki/Sass_(stylesheet_language)
[13] http://sass-lang.com/libsass
[14] http://thesassway.com/editorial/sass-vs-scss-which-syntax-is-better
[15] http://www.solitr.com/blog/2014/01/css-preprocessor-benchmark/