**CSS Preprocessors - Sass vs Less**

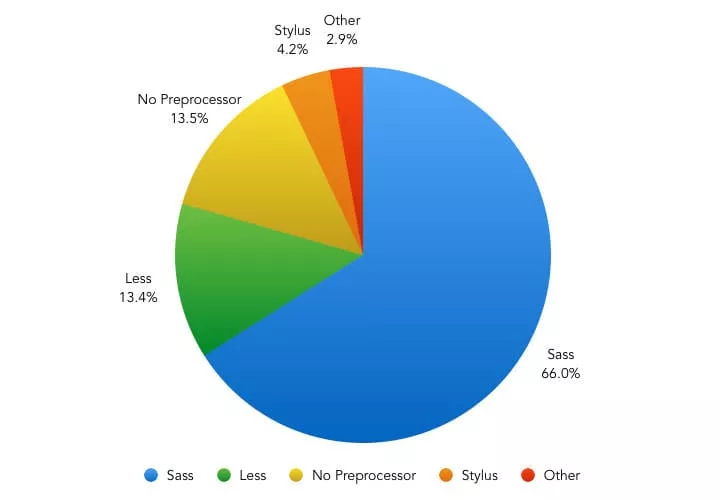


There are three primary CSS preprocessors on the market today, [Sass](http://sass-lang.com/), [Less](http://lesscss.org/), and [Stylus](https://learnboost.github.io/stylus/). In this post, we will be comparing the two preprocessors which seem to be the most widely used among developers, **Sass vs Less**. By switching to a preprocessor can help streamline your development process.

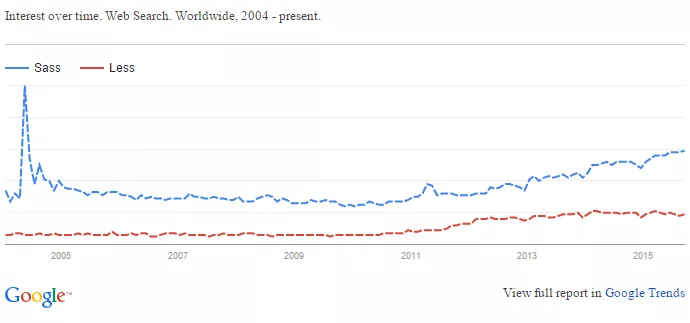
**What is a CSS preprocessor?**[**#**](https://www.keycdn.com/blog/sass-vs-less#what-is-a-css-preprocessor)

Writing CSS can become quite repetitive and little tasks such as having to look up hex color values, closing your tags, etc. can become time-consuming. And so that is where a preprocessor comes into play. A CSS preprocessor is basically a **scripting language that extends CSS** and then **compiles it into regular CSS**.

Ashley Nolan polled developers in a recent case study to see which preprocessors they preferred. 689 people voted for Sass while Less received 140 votes.



And if we take a look at Google Trends for "Sass" (programming language) and "Less" (programming language) it appears there is much more activity with Sass.

**Advantages to using a preprocessor**[**#**](https://www.keycdn.com/blog/sass-vs-less#advantages-to-using-a-preprocessor)

Once you have mastered CSS there are many advantages to using a preprocessor. One of the biggest is probably not having to repeat yourself. In other words, it makes your [CSS Dry](http://vanseodesign.com/css/dry-principles/).

Here are some other advantages.

1. Cleaner code with reusable pieces and variables
2. Saves you time
3. Easier to maintain code with snippets and libraries
4. Calculations and logic
5. More organized and easy to setup

**Sass vs Less[#](https://www.keycdn.com/blog/sass-vs-less" \l "sass-vs-less)**

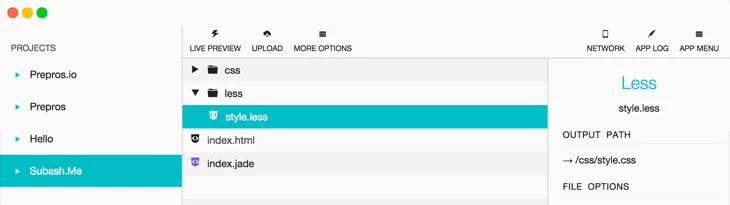
Sass and Less are both very powerful CSS extensions. You can think of them as more of a programming language designed to make CSS more maintainable, themeable, and extendable. Both Sass and Less are backward compatible so you can easily convert your existing CSS files just by renaming the .css file extension to .less or .scss, respectively. Less is JavaScript based and Sass is Ruby based.

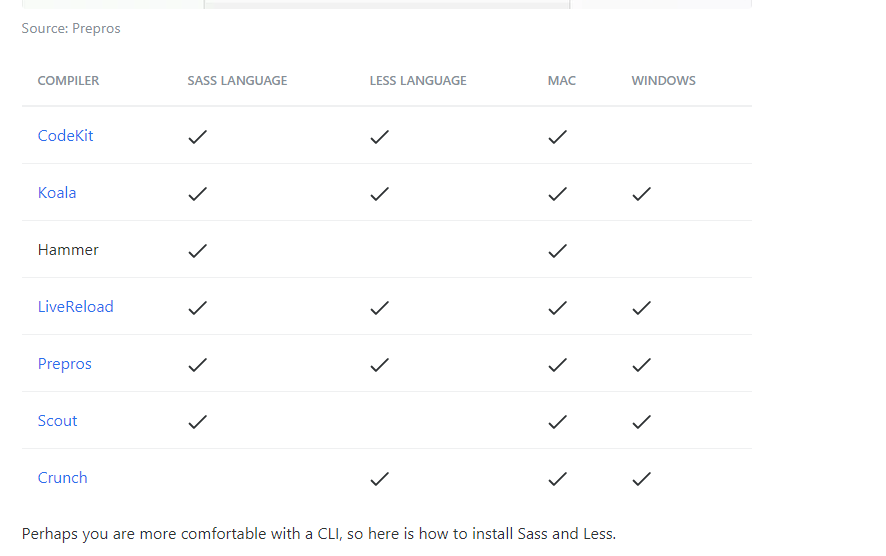
**Installation and setup**[**#**](https://www.keycdn.com/blog/sass-vs-less#installation-and-setup)

There are a couple ways to install and get going with a preprocessor. You can either use a third party application or you can install it from the command line.

**Third party applications**[**#**](https://www.keycdn.com/blog/sass-vs-less#third-party-applications)

If you are just transitioning to preprocessors then we highly recommend using a third party compiler as this will speed things up. Below is a list of some more popular ones, listed in no particular order.



****

**Sass**[**#**](https://www.keycdn.com/blog/sass-vs-less#sass)

To run Sass, **you will need to have Ruby installed**. On newer versions of Linux and OSX, Ruby already comes preinstalled. You can install Sass with the following command. sudo gem install sass

There are also other libraries, such as [libSass](http://sass-lang.com/libsass" \t "_blank) which now bring Sass to NodeJS, without any need for Ruby.

On Windows, you will need to first [install Ruby](http://rubyinstaller.org/). If you are on a Mac, Ruby already comes preinstalled.

1. Open up your terminal or command prompt.
2. Install Sass using the sudo gem install sass command.

**Less**[**#**](https://www.keycdn.com/blog/sass-vs-less#less)

Less is written in JavaScript so you **will need NodeJS** to run it.

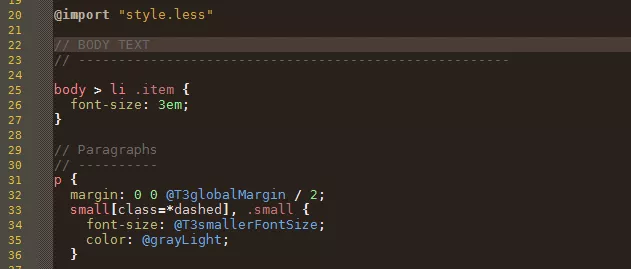
1. On Linux and Mac, you can install it using the npm install -g less command.
2. You can then use the lessc styles.less styles.css command to compile to CSS.

On Windows you will need to install the [NodeJS installer](https://nodejs.org/en/" \t "_blank).

1. Open up your command prompt.
2. Run npm install less.
3. You can then use the lessc styles.less styles.css command to compile to CSS.

**Code editor - Syntax highlighting**[**#**](https://www.keycdn.com/blog/sass-vs-less#code-editor-syntax-highlighting)

After you have installed everything you then will probably want to setup a plugin or extension in your code editor to highlight Sass or Less syntax with the proper colors. Here is a list of popular ones below, in no particular order.



| **CODE EDITOR** | **SASS** | **LESS** |
| --- | --- | --- |
| SublimeText | [Sass Bundle](https://github.com/nathos/sass-textmate-bundle) | [Less-sublime](https://github.com/danro/LESS-sublime) |
| Notepad++ | [Notepad-plus-plus](https://github.com/marvinlabs/notepad-plus-plus-scss-syntax-highlighting) | [Less-for-Notepad-plusplus](https://github.com/azrafe7/LESS-for-Notepad-plusplus) |
| VisualStudio | [SassyStudio](https://marketplace.visualstudio.com/items?itemName=darrenkopp.SassyStudio) | [CSS is Less](https://marketplace.visualstudio.com/items?itemName=NoahRichards.CSSIsLess) |
| TextMate | [SCSS.tmbundle](https://github.com/MarioRicalde/SCSS.tmbundle) | [Less.tmbundle](https://github.com/appden/less.tmbundle) |
| Eclipse | [Eclipse platform](http://colorer.sourceforge.net/eclipsecolorer/) | [Eclipse for Less](http://www.normalesup.org/~simonet/soft/ow/eclipse-less.en.html) |
| Coda | [Coda Sass](https://github.com/keegnotrub/coda-sass-plugin/releases) | [Coda 2 Less](https://github.com/mjvotaw/Coda-2-LESS-Compiler) |

There are also other popular third party frameworks you can use along with the preprocessors such as [Compass](http://compass-style.org/) and [Bourbon](http://bourbon.io/).

**Variables**[**#**](https://www.keycdn.com/blog/sass-vs-less#variables)

With preprocessors, you have the advantage over traditional CSS because you can use variables. You can store things like colors, fonts, or pretty much any value you want to reuse later. See examples below.

**Sass font example**[**#**](https://www.keycdn.com/blog/sass-vs-less#sass-font-example)

$font-stack: Helvetica, sans-serif;

$primary-color: #333;

body {

font: 100% $font-stack;

color: $primary-color;

}

**CSS output**[**#**](https://www.keycdn.com/blog/sass-vs-less#css-output)

body {

font: 100% Helvetica, sans-serif;

color: #333;

}

**Less color example**[**#**](https://www.keycdn.com/blog/sass-vs-less#less-color-example)

@nice-blue: #5B83AD;

@light-blue: @nice-blue + #111;

#header {

color: @light-blue;

}

**CSS output**[**#**](https://www.keycdn.com/blog/sass-vs-less#css-output-1)

#header {

color: #6c94be;

}

**Mixins[#](https://www.keycdn.com/blog/sass-vs-less" \l "mixins)**

Mixins are used to include a bunch of properties or group declarations together.

**Sass border radius example**[**#**](https://www.keycdn.com/blog/sass-vs-less#sass-border-radius-example)

In Sass, you use @mixin:

@mixin border-radius($radius) {

-webkit-border-radius: $radius;

-moz-border-radius: $radius;

-ms-border-radius: $radius;

border-radius: $radius;

}

.box {

@include border-radius(10px);

}

**CSS output**[**#**](https://www.keycdn.com/blog/sass-vs-less#css-output-2)

.box {

-webkit-border-radius: 10px;

-moz-border-radius: 10px;

-ms-border-radius: 10px;

border-radius: 10px;

}

**Less border property example**[**#**](https://www.keycdn.com/blog/sass-vs-less#less-border-property-example)

.bordered {

border-top: dotted 1px black;

border-bottom: solid 2px black;

}

#menu a {

color: #111;

.bordered;

}

.post a {

color: red;

.bordered;

}

**CSS output**[**#**](https://www.keycdn.com/blog/sass-vs-less#css-output-3)

#menu a {

color: #111;

border-top: dotted 1px black;

border-bottom: solid 2px black;

}

.post a {

color: red;

border-top: dotted 1px black;

border-bottom: solid 2px black;

}

**Nesting**[**#**](https://www.keycdn.com/blog/sass-vs-less#nesting)

Nesting is a huge advantage over CSS because it creates a visual hierarchy, similar to what you are used to with HTML. In the examples, you can see how there is less repetition of classes or divs needed since it is now in a cascade approach.

**Sass**[**#**](https://www.keycdn.com/blog/sass-vs-less#sass-1)

nav {

ul {

margin: 0;

padding: 0;

list-style: none;

}

li {

display: inline-block;

}

a {

display: block;

padding: 6px 12px;

text-decoration: none;

}

}

**CSS output**[**#**](https://www.keycdn.com/blog/sass-vs-less#css-output-4)

nav ul {

margin: 0;

padding: 0;

list-style: none;

}

nav li {

display: inline-block;

}

nav a {

display: block;

padding: 6px 12px;

text-decoration: none;

}

**Less**[**#**](https://www.keycdn.com/blog/sass-vs-less#less-1)

#header {

color: black;

.navigation {

font-size: 12px;

}

.logo {

width: 300px;

}

}

**CSS output**[**#**](https://www.keycdn.com/blog/sass-vs-less#css-output-5)

#header {

color: black;

}

#header .navigation {

font-size: 12px;

}

#header .logo {

width: 300px;

}

**Import**[**#**](https://www.keycdn.com/blog/sass-vs-less#import)

The standard CSS @import allows you to split into multiple files. The problem with this is that it creates additional HTTP requests. Sass and Less work a little different. Instead of creating another HTTP request, they combine the files into one. Similar to concatenation.

**Sass**[**#**](https://www.keycdn.com/blog/sass-vs-less#sass-2)

Let says you have a file named \_sheet1.scss. In your base.scss you can include it at the top by using the following: @import 'sheet1'; There is no need to have the extension.

**Less**[**#**](https://www.keycdn.com/blog/sass-vs-less#less-2)

// Variables

@themes: "../../src/themes";

// Usage

@import "@{themes}/tidal-wave.less";

**Extend**[**#**](https://www.keycdn.com/blog/sass-vs-less#extend)

This is another powerful feature. It lets you share properties from one selector to another.

**Sass**[**#**](https://www.keycdn.com/blog/sass-vs-less#sass-3)

In Sass, you use @extend

.message {

border: 1px solid #ccc;

padding: 10px;

color: #333;

}

.success {

@extend .message;

border-color: green;

}

.error {

@extend .message;

border-color: red;

}

.warning {

@extend .message;

border-color: yellow;

}

**CSS output**[**#**](https://www.keycdn.com/blog/sass-vs-less#css-output-6)

.message, .success, .error, .warning {

border: 1px solid #cccccc;

padding: 10px;

color: #333;

}

.success {

border-color: green;

}

.error {

border-color: red;

}

.warning {

border-color: yellow;

}

**Less**[**#**](https://www.keycdn.com/blog/sass-vs-less#less-3)

In Less you use :extend

nav ul {

&:extend(.inline);

background: blue;

}

.inline {

color: red;

}

**CSS output**[**#**](https://www.keycdn.com/blog/sass-vs-less#css-output-7)

nav ul {

background: blue;

}

.inline, nav ul {

color: red;

}

**Operations**[**#**](https://www.keycdn.com/blog/sass-vs-less#operations)

Unlike with CSS, you can do all sorts of operations in both Sass and Less.

**Sass grid calculation example**[**#**](https://www.keycdn.com/blog/sass-vs-less#sass-grid-calculation-example)

.container {

width: 100%;

}

article[role="main"] {

float: left;

width: 600px / 960px \* 100%;

}

aside[role="complimentary"] {

float: right;

width: 300px / 960px \* 100%;

}

**CSS output**[**#**](https://www.keycdn.com/blog/sass-vs-less#css-output-8)

.container {

width: 100%;

}

article[role="main"] {

float: left;

width: 62.5%;

}

aside[role="complimentary"] {

float: right;

width: 31.25%;

}

**Less square root example**[**#**](https://www.keycdn.com/blog/sass-vs-less#less-square-root-example)

sqrt(18.6%)

**CSS output**[**#**](https://www.keycdn.com/blog/sass-vs-less#css-output-9)

4.312771730569565%;

Those are just a few of the benefits you get from using a preprocessor. Most of the examples above were taken from the official [Sass](http://sass-lang.com/guide) and [Less](http://lesscss.org/features/) documentation.

**Active development**[**#**](https://www.keycdn.com/blog/sass-vs-less#active-development)

Both Sass and Less are pretty active on GitHub, you can see some of the recent activity below.

| **GITHUB ACTIVITY** | **SASS** | **LESS** |
| --- | --- | --- |
| # of open issues | [138](https://github.com/sass/sass/issues) | [208](https://github.com/cloudhead/less.js/issues) |
| # of pending pull requests | [11](https://github.com/sass/sass/pulls) | [8](https://github.com/cloudhead/less.js/pulls) |
| # of commits in the last 30 days | [29](https://github.com/sass/sass/commits/) | [12](https://github.com/cloudhead/less.js/commits/master/) |

**Summary**[**#**](https://www.keycdn.com/blog/sass-vs-less#summary)

As you can see CSS preprocessors are very powerful and they can help streamline your development process, especially if you are coming from a programming background. While it appears that Sass is more widely used, there isn't really a preprocessor that is better than the other. It usually comes down to the developer and what they are comfortable with using. Both of them have feature-rich extensions which can help make your CSS more maintainable, themeable, and extendable.

Which CSS preprocessor do you prefer? Let us know below in the comments.

**----------------------------------------------------------------------------------------------------------------------**

**Sass Vs LESS**.

There are 2 CSS preprocessors to choose from:

Less https://lesscss.org/

Sass https://sass-lang.com/

**Sass and LESS** are both very powerful **CSS** extensions. ... Both **Sass and LESS** are backward compatible so you can easily convert your existing **CSS** files just by renaming the .**css** file extension to .**less or** .**scss** , respectively. **LESS** is JavaScript based **and Sass** is Ruby based.

**Cascading style sheets (CSS)**

**Syntactically Awesome Style Sheet (SASS)**

**Sassy CSS (SCSS)**

**Sass vs SCSS**

Sass has 2 syntaxes available:

Sass itself (Syntactically Awesome StyleSheets) in .sass files

SCSS (Sassy CSS) in .scss files, which is something halfway between regular CSS and Sass

The difference between Sass and SCSS is quite subtle.

Remember that:

Sass is the name of the preprocessor

SCSS is easier to learn

all resources on the internet (like The Sass Way) mention Sass, not SCSS

all features are available for both syntaxes

everything in SCSS is available in Sass

We’re actually going to write SCSS but still call it Sass.

#Why SCSS first

We’re gonna use SCSS for a few reasons:

readibility: the syntax is very similar to CSS

learning curve: it only adds a few additional features on top of CSS

compatibility; a CSS file is a valid SCSS file

resources: lots of online articles to read and open source libraries to use

expandibility: it’s easy to go from SCSS to Sass