Syntax

Variables

colors, font stacks, or any CSS value

Sass uses the $ symbol to make something a variable

|  |  |
| --- | --- |
| Sass compiles to: | CSS |
| $font-stack: Helvetica, sans-serif;  $primary-color: #333;  body {  font: 100% $font-stack;  color: $primary-color;  } | body {  font: 100% Helvetica, sans-serif;  color: #333;  } |
| **Nesting** |  |
| nav {  ul {  margin: 0;  padding: 0;  list-style: none;  }  li { display: inline-block; }  a {  display: block;  padding: 6px 12px;  text-decoration: none;  }  } | nav ul {  margin: 0;  padding: 0;  list-style: none;  }  nav li {  display: inline-block;  }  nav a {  display: block;  padding: 6px 12px;  text-decoration: none;  } |
| **Partials** | \_<filename>.scss |
| Usage: | @import |
| Command Line |  |
| sass source/index.scss css/index.css | css/index.css |
| sass --watch input.scss output.css | output.css |
| Watch and Output Directories |  |
| sass --watch app/sass:public/stylesheets | |
| Mixins | |
| @mixin transform($property) {  -webkit-transform: $property;  -ms-transform: $property;  transform: $property;  }  .box { @include transform(rotate(30deg)); } | .box {  -webkit-transform: rotate(30deg);  -ms-transform: rotate(30deg);  transform: rotate(30deg);  } |
| Extend/Inheritance | This is one of the most useful features of Sass. Using @extend lets you share a set of CSS properties from one selector to another. It helps keep your Sass very DRY. In our example we're going to create a simple series of messaging for errors, warnings and successes using another feature which goes hand in hand with extend, placeholder classes. A placeholder class is a special type of class that only prints when it is extended, and can help keep your compiled CSS neat and clean. |
| Placeholder Classes | A placeholder class is a special type of class that only prints when it is extended, and can help keep your compiled CSS neat and clean. |
| /\* This CSS will print because %message-shared is extended. \*/  %message-shared {  border: 1px solid #ccc;  padding: 10px;  color: #333;  }  // This CSS won't print because %equal-heights is never extended.  %equal-heights {  display: flex;  flex-wrap: wrap;  }  .message {  @extend %message-shared;  }  .success {  @extend %message-shared;  border-color: green;  }  .error {  @extend %message-shared;  border-color: red;  }  .warning {  @extend %message-shared;  border-color: yellow;  } | What the above code does is tells .message, .success, .error, and .warning to behave just like %message-shared. That means anywhere that %message-shared shows up, .message, .success, .error, & .warning will too. The magic happens in the generated CSS, where each of these classes will get the same CSS properties as %message-shared. This helps you avoid having to write multiple class names on HTML elements. |
| Operators | Doing math in your CSS is very helpful. Sass has a handful of standard math operators like +, -, \*, /, and %. In our example we're going to do some simple math to calculate widths for an aside & article. |
| .container {  width: 100%;  }  article[role="main"] {  float: left;  width: 600px / 960px \* 100%;  }  aside[role="complementary"] {  float: right;  width: 300px / 960px \* 100%;  } | .container {  width: 100%;  }  article[role="main"] {  float: left;  width: 62.5%;  }  aside[role="complementary"] {  float: right;  width: 31.25%;  } |

--------------Install and CMD watch-------------------

What is Sass?

Sass stands for Syntactically Awesome Stylesheet

Sass is an extension to CSS

Sass is a CSS pre-processor

ranspiling is a term for taking a source code written in one language and transform/translate it into another language.

/\* comment \*/, and in addition it supports inline comments // comment:

If you use Node.js, you can also install Sass using npm by running

npm install -g sass

The most direct way to make this happen is in your terminal. Once Sass is installed, you can compile your Sass to CSS using the sass command. You'll need to tell Sass which file to build from, and where to output CSS to. For example, running sass input.scss output.css from your terminal would take a single Sass file, input.scss, and compile that file to output.css.

You can also watch individual files or directories with the --watch flag. The watch flag tells Sass to watch your source files for changes, and re-compile CSS each time you save your Sass. If you wanted to watch (instead of manually build) your input.scss file, you'd just add the watch flag to your command, like so:

sass --watch input.scss output.css

You can watch and output to directories by using folder paths as your input and output, and separating them with a colon. In this example:

sass --watch app/sass:public/stylesheets

Sass would watch all files in the app/sass folder for