## Less: The dynamic stylesheet language

### LESS extends CSS with dynamic behavior such as variables, mixins, operations and functions.

### LESS runs on both the server-side (with Node.js and Rhino) or client-side (modern browsers only).

<http://lesscss.org/#synopsis>

# (Syntactically Awesome StyleSheets)

**Sass makes CSS fun again**. Sass is an extension of CSS3, adding [nested rules](http://sass-lang.com/#nesting), [variables](http://sass-lang.com/#variables), [mixins](http://sass-lang.com/#mixins), [selector inheritance](http://sass-lang.com/#extend), and [more](http://sass-lang.com/docs/yardoc/file.SASS_REFERENCE.html). It’s translated to well-formatted, standard CSS using the command line tool or a web-framework plugin.

Sass has two syntaxes. The most commonly used syntax is known as “SCSS” (for “Sassy CSS”), and is a superset of CSS3’s syntax. This means that every valid CSS3 stylesheet is valid SCSS as well. SCSS files use the extension .scss.

The second, older syntax is known as [the indented syntax](http://sass-lang.com/docs/yardoc/file.INDENTED_SYNTAX.html) (or just “.sass”). Inspired by [Haml](http://haml-lang.com)’s terseness, it’s intended for people who prefer conciseness over similarity to CSS. Instead of brackets and semicolons, it uses the indentation of lines to specify blocks. Files in the indented syntax use the extension .sass.

While the documentation on this site is mostly in the SCSS syntax, both syntaxes are fully supported – there is no functional difference between them. Use the syntax you prefer.

## Variables

Use the same color all over the place? Need to do some math with height and width and text size? Sass supports variables as well as basic math operations and [many useful functions](http://sass-lang.com/docs/yardoc/Sass/Script/Functions.html).

**.sass**

$blue: #3bbfce

$margin: 16px

.content-navigation

border-color: $blue

color: darken($blue, 9%)

.border

padding: $margin / 2

margin: $margin / 2

border-color: $blue

**.scss**

$blue: #3bbfce;

$margin: 16px;

.content-navigation {

border-color: $blue;

color:

darken($blue, 9%);

}

.border {

padding: $margin / 2;

margin: $margin / 2;

border-color: $blue;

}

**/\* CSS \*/**

.content-navigation {

border-color: #3bbfce;

color: #2b9eab;

}

.border {

padding: 8px;

margin: 8px;

border-color: #3bbfce;

}

## Nesting

Sass avoids repetition by nesting selectors within one another. The same thing works with properties.

**.scss**

table.hl {

margin: 2em 0;

td.ln {

text-align: right;

}

}

li {

font: {

family: serif;

weight: bold;

size: 1.2em;

}

}

**.sass**

table.hl

margin: 2em 0

td.ln

text-align: right

li

font:

family: serif

weight: bold

size: 1.2em

**/\* CSS \*/**

table.hl {

margin: 2em 0;

}

table.hl td.ln {

text-align: right;

}

li {

font-family: serif;

font-weight: bold;

font-size: 1.2em;

}

## Mixins

Even more useful than variables, mixins allow you to re-use whole chunks of CSS, properties or selectors. You can even give them arguments.

**.scss**

@mixin table-base {

th {

text-align: center;

font-weight: bold;

}

td, th {padding: 2px}

}

@mixin left($dist) {

float: left;

margin-left: $dist;

}

#data {

@include left(10px);

@include table-base;

}

**.sass**

@mixin table-base

th

text-align: center

font-weight: bold

td, th

padding: 2px

@mixin left($dist)

float: left

margin-left: $dist

#data

@include left(10px)

@include table-base

**/\* CSS \*/**

#data {

float: left;

margin-left: 10px;

}

#data th {

text-align: center;

font-weight: bold;

}

#data td, #data th {

padding: 2px;

}

## Selector Inheritance

Sass can tell one selector to inherit all the styles of another without duplicating the CSS properties.

**.sass**

.error

border: 1px #f00

background: #fdd

.error.intrusion

font-size: 1.3em

font-weight: bold

.badError

@extend .error

border-width: 3px

**.scss**

.error {

border: 1px #f00;

background: #fdd;

}

.error.intrusion {

font-size: 1.3em;

font-weight: bold;

}

.badError {

@extend .error;

border-width: 3px;

}

**/\* CSS \*/**

.error, .badError {

border: 1px #f00;

background: #fdd;

}

.error.intrusion,

.badError.intrusion {

font-size: 1.3em;

font-weight: bold;

}

.badError {

border-width: 3px;

}

[Sass](http://sass-lang.com/) is an extension of CSS3 which adds nested rules, variables, mixins, selector inheritance, and more. Sass generates well formatted CSS and makes your stylesheets easier to organize and maintain.

**Compass is an open-source *CSS Authoring Framework*.**

In Sass you can use [Compass](http://compass-style.org/), and Compass **will** keep itself updated, and thus the prefix situation is handled for you. Yes you'll have to keep your local preprocessor software updated and compile/push once in a while, but that's trivial and thinking-free.

So what this comes down to is: **Sass has Compass and LESS does not.** But it goes deeper than that. The attempts at creating a real robust project like Compass for LESS haven't succeeded because the LESS language isn't robust enough to do it properly. More on that next.

Winner: Sass