Sass/Less Comparison

In this document I am using Sass's SCSS syntax. You can choose to use the indented syntax in sass, if you prefer it, it has no functional differences from the SCSS syntax.

For Less, I'm using the JavaScript version because this is what they suggest on the website. The ruby version may be different.

Variables

Sass	Less
$color: red;	@color: red;
div {	div {
color: $color;	color: @color;
}	}

Both languages support scoped variable definition within a selector context. However they differ in their behavior:

Sass	Less
$color: black;	@color: black;
.scoped {	.scoped {
$bg: blue;	@bg: blue;
$color: white;	@color: white;
color: $color;	color: @color;
background-color: $bg;	background-color: @bg;
}	}
.unscoped {	.unscoped {
color: $color;	color: @color;
// Would be an error	// Would be an error
// background: $bg;	// background: @bg;
}	}

And their different output:

Sass Output	Less Output
.scoped {	.scoped {
color: white;	color: white;
background-color: blue;	background-color: blue;
}	}
.unscoped { color: white; }	.unscoped { color: black; }

Nested Selectors

Sass and Less have the `&` selector that allows nested selector to refer to the parent scope.

Sass	Less
p {	p {
a {	a {
color: red;	color: red;
&:hover {	&:hover {
color: blue;	color: blue;
}	}
}	}
}	}

Mixins

Sass	Less
@mixin bordered {	.bordered {
border-top: dotted 1px black;	border-top: dotted 1px black;
border-bottom: solid 2px black;	border-bottom: solid 2px black;
}	}
#menu a {	#menu a {
@include bordered;	.bordered;
}	}

Mixins with Arguments / Dynamic Mixins

Sass	Less
@mixin bordered($width: 2px) {	.bordered(@width: 2px) {
border: $width solid black;	border: @width solid black;
}	}
#menu a {	#menu a {
@include bordered(4px);	.bordered(4px);
}	}

Selector Inheritance

Less does not provide selector inheritance.

Sass	Less	CSS Output
.bordered {	N/A	.bordered, #menu a {
border: 1px solid back;		border: 1px solid back; }
}		
#menu a {		
@extend .bordered;		
}		

Colors

Both less and sass provide color math. It was a bad idea in Sass, I'm not sure why less chose to copy it. There's no point in comparing them.

Sass provides a full array of tools for manipulating colors. All color representations (named colors, hex, rgb, rgba, hsl, hsla) are understood as colors and colors can be manipulated.

Sass exposes a long list of color functions not found in CSS:

Accessors:

- `red($color)`

- `green($color)`
- `blue($color)`
- `hue($color)`
- `saturation($color)`
- `lightness($color)`
- `alpha($color)`

Mutators:

- `lighten($color, $amount)`
- `darken($color, $amount)`
- `saturate($color, $amount)`
- `desaturate($color, $amount)`
- `adjust-hue($color, $amount)`
- `opacify($color, $amount)`
- `transparentize($color, $amount)`
- `mix($color1, $color2[, $amount])`
- `grayscale($color)`
- `compliment($color)`

Note: Less.js provides a number of similar color functions.

Numbers

Both Sass and Less support numbers and basic arithmetic. However they differ significantly with respect to how they handle units.

Sass supports unit-based arithmetic, just like you learned in school. Complex units are supported in any intermediate form and will only raise an error if you try to print out the value of a complex unit.

Additionally, Sass has conversion tables so that any comparable units can be combined.

Sass will let you define your own units and will happily print out unknown units into your css. Less will not. Sass does this as a form of future proofing against changes in the w3c specification or in case a browser introduces a non-standard unit.

Sass:

```
1cm * 1em => 1 cm * em
2in * 3in => 6 in * in
(1cm / 1em) * 4em => 4cm
2in + 3cm + 2pc => 3.514in
3in / 2in => 1.5
```

Less:

```
1cm * 1em => Error
2in * 3in => 6in
(1cm / 1em) * 4em => Error
2in + 3cm + 2pc => Error
3in / 2in => 1.5in
```

Conditionals & Control Structures

Less does not provide any conditionals or looping structures. Instead, it provides *mixin guards* and *pattern-matching* which can be used to similar effect.

Sass and Less provides provide boolean types `true` and `false`, the `and`, `or`, and `not` operators as well as `<`, `>`, `<=`, `>=`, `==` operators. There are minor syntax differences between the two (Sass syntax shown here).

Some sass examples:

```
@if lightness($color) > 30% {
  background-color: black;
}
@else {
  background-color: white;
}
```

Looping:

```
@for $i from 1px to 10px {
  .border-#{i} {
    border: $i solid blue;
  }
}
```

A similar example in Less, using mixins:

```
.mixin (@color) when (lightness(@color) > 30%) {
  background-color: black;
}
.mixin (@color) when (lightness(@color) =< 30%) {
  background-color: white;
}
```

Less supports looping via recursion, but not (yet) selector interpolation as shown in the Sass example, so it is of limited use.

Server Side Imports

Both sass and less will import other sass and less files.

Output formatting

Less has three output formats: normal, compressed & yui-compressed. Sass has four: nested, compact, compressed, expanded.

Sass output compression currently beats the google pagespeed css plugin output by a few percent.

Comments

Less and Sass both support C-style (`/* */`) and C++ Style comments (`//`).

Namespaces

Less provides a feature that Sass does not:

```
#bundle () {
  .red { background-color: red }
  .green { background-color: green }
}

.foo {
  #bundle > .red;
}
```

Generates:

```
.foo {
  background-color: red;
}
```

The sass team considered this feature and decided that adding it would create fragility and unexpected interconnectedness.