Sass cheatsheet

Introduction

This is a quick reference to Sass stylesheets.

Sass documentation
(sass-lang.com)

Variables

```
$red: #833;
```

```
body {
  color: $red;
}
```

Nesting

```
.markdown-body {
  a {
    color: blue;
    &:hover {
      color: red;
    }
  }
}
```

to properties

```
text: {
  align: center;          // like text-a
  transform: uppercase;   // like text-t
}
```

Comments

```
/* Block comments */
// Line comments
```

Mixins

```
@mixin heading-font {
  font-family: sans-serif;
  font-weight: bold;
}
```

```
h1 {
  @include heading-font;
}
```

with parameters

```
@mixin font-size($n) {
  font-size: $n * 1.2em;
}
```

```
body {
  @include font-size(2);
}
```

with default values

```
@mixin pad($n: 10px) {
  padding: $n;
}
```

```
body {
  @include pad(15px);
}
```

with a default variable

```
// Set a default value
$default-padding: 10px;
```

```
@mixin pad($n: $default-padding) {
  padding: $n;
}
```

```
body {
  @include pad(15px);
}
```

Extend

```
.button {
  ...
}
```

```
.push-button {
  @extend .button;
}
```

Composing

```
@import './other_sass_file';
@use './other_sass_file';
```

The `@import` rule is discouraged because will get eventually removed from the language. Instead, we should use the `@use` rule.
The `.scss` or `.sass` extension is optional.

Color functions

rgba

```
rgb(100, 120, 140)
rgba(100, 120, 140, .5)
rgba($color, .5)
```

Mixing

```
mix($a, $b, 10%)   // 10% a, 90% b
```

Modifying HSLA

```
darken($color, 5%)
lighten($color, 5%)
```

```
saturate($color, 5%)
desaturate($color, 5%)
grayscale($color)
```

```
adjust-hue($color, 15deg)
complement($color)    // like adjust-hue(_, 180deg)
invert($color)
```

```
fade-in($color, .5)   // aka opacify()
fade-out($color, .5)  // aka transparentize() - halves the opac
rgba($color, .5)      // sets alpha to .5
```

Getting individual values

HSLA

```
hue($color)         // → 0deg..360deg
saturation($color)  // → 0%..100%
lightness($color)   // → 0%..100%
alpha($color)       // → 0..1 (aka opacity())
```

RGB

```
red($color)         // → 0..255
green($color)
blue($color)
```

See: hue(), red()

Adjustments

```
// Changes by fixed amounts
adjust-color($color, $blue: 5)
adjust-color($color, $lightness: -30%)   // like darken(_, 30%)
adjust-color($color, $alpha: -0.4)       // like fade-out(_, .4
adjust-color($color, $hue: 30deg)        // like adjust-hue(_,
```

```
// Changes via percentage
scale-color($color, $lightness: 50%)
```

```
// Changes one property completely
change-color($color, $hue: 180deg)
change-color($color, $blue: 250)
```

Supported: `$red` `$green` `$blue` `$hue` `$saturation` `$lightness` `$alpha`

Other functions

Strings

```
unquote('hello')
quote(hello)
```

```
to-upper-case(hello)
to-lower-case(hello)
```

```
str-length(hello world)
str-slice(hello, 2, 5)      // "ello" - it's 1-based, not 0-bas
str-insert("abcd", "X", 1)  // "Xabcd"
```

Numbers

```
floor(3.5)
ceil(3.5)
round(3.5)
abs(3.5)
```

```
min(1, 2, 3)
max(1, 2, 3)
```

```
percentage(.5)   // 50%
random(3)        // 0..3
```

Units

```
unit(3em)        // 'em'
unitless(100px)  // false
```

Misc

```
variable-exists(red)    // checks for $red
mixin-exists(red-text)  // checks for @mixin red-text
function-exists(redify)
```

```
global-variable-exists(red)
```

```
selector-append('.menu', 'li', 'a')   // .menu li a
selector-nest('.menu', '&:hover li')  // .menu:hover li
selector-extend(...)
selector-parse(...)
selector-replace(...)
selector-unify(...)
```

Feature checks

Feature check

```
feature-exists(global-variable-shadowing)
```

Features

- global-variable-shadowing
- extend-selector-pseudoclass
- units-level-3
- at-error

Loops

For loops

```
@for $i from 1 through 4 {
  .item-#{$i} { left: 20px * $i; }
}
```

Each loops (simple)

```
$menu-items: home about services contact;

@each $item in $menu-items {
  .photo-#{$item} {
    background: url('images/#{$item}.jpg');
  }
}
```

Each loops (nested)

```
$backgrounds: (home, 'home.jpg'), (about, 'about.jpg');

@each $id, $image in $backgrounds {
  .photo-#{$id} {
    background: url($image);
  }
}
```

While loops

```
$i: 6;
@while $i > 0 {
  .item-#{$i} { width: 2em * $i; }
  $i: $i - 2;
}
```

Other features

Conditionals

```
@if $position == 'left' {
   position: absolute;
   left: 0;
}
@else if $position == 'right' {
   position: absolute;
   right: 0;
}
@else {
   position: static;
}
```

Interpolation

```
.#{$klass} { ... }      // Class
call($function-name)    // Functions

@media #{$tablet}
font: #{$size}/#{$line-height}
url("#{$background}.jpg")
```

Lists

```
$list: (a b c);

nth($list, 1)  // starts with 1
length($list)

@each $item in $list { ... }
```

Maps

```
$map: (key1: value1, key2: value2, key3: value3);

map-get($map, key1)
```

See also

- http://sass-lang.com/documentation/Sass/Script/Functions.html
- http://sass-lang.com/documentation/file.SASS_REFERENCE.html#sassscript