Sass Cheatsheet

A comprehensive cheat sheet covering Sass syntax, variables, mixins, functions, and more, designed to help you write cleaner and more maintainable CSS.

Sass Basics

Variables

Defining Variables	Variables in Sass start with a `$` and can store any CSS value.
	`$primary-color: #007bff;`
	`$font-size: 16px;`

Using Variables	Variables can be used throughout your Sass code. `body { font-size: $font-size; color: $primary-color; }`

Scope	Variables have scope. Variables defined inside a block (e.g., a mixin or a selector) are only accessible within that block unless made global with `!global`.
	`@mixin set-color { $local-color: red !global; }`
	`@include set-color;`
	`body { color: $local-color; // Accessible because of !global }`

Nesting

Nesting Selectors	Sass allows you to nest CSS selectors, following the HTML structure. `nav { ul { margin: 0; padding: 0; list-style: none; } li { display: inline-block; } a { display: block; padding: 6px 12px; text-decoration: none; } }`
`&` Operator	The `&` operator refers to the parent selector, useful for pseudo-classes and pseudo-elements. `a { color: blue; &:hover { color: red; } }`
Nesting Properties	You can also nest properties that share the same namespace. `.text { font: { size: 16px; weight: bold; } margin: { top: 10px; bottom: 5px; } }`

Mixins

Defining Mixins	Mixins allow you to define reusable styles. Use `@mixin` to define and `@include` to use. `@mixin border-radius($radius) { -webkit-border-radius: $radius; -moz-border-radius: $radius; border-radius: $radius; }`
Including Mixins	Include mixins in your CSS rules. `.button { @include border-radius(5px); }`

Mixins with Arguments	Mixins can accept arguments.
	`@mixin box-shadow($x, $y, $blur, $color) { -webkit-box-shadow: $x $y $blur $color; -moz-box-shadow: $x $y $blur $color; box-shadow: $x $y $blur $color; }`
	`.box { @include box-shadow(2px, 2px, 5px, rgba(0, 0, 0, 0.3)); }`

Default Arguments	You can also define default values for mixin arguments.
	`@mixin text-style($size: 16px, $color: #000) { font-size: $size; color: $color; }`
	`.title { @include text-style(20px); }`
	`.paragraph { @include text-style; }`

Sass Functions

Color Functions

rgba()	Creates a color with red, green, blue, and alpha values.
	`color: rgba(100, 120, 140, 0.5);`
	`color: rgba($color, 0.5);`

mix()	Mixes two colors together. `color: mix($color1, $color2, 50%); // Mixes 50% of color1 and 50% of color2`

darken(), lighten()	Adjusts the lightness of a color.
	`background-color: darken($primary-color, 10%);`
	`background-color: lighten($primary-color, 10%);`

saturate(), desaturate()	Adjusts the saturation of a color.
	`color: saturate($desaturated-color, 20%);`
	`color: desaturate($saturated-color, 20%);`

grayscale()	Converts a color to grayscale. `color: grayscale($colorful-color);`
adjust-hue()	Adjusts the hue of a color. `color: adjust-hue($color, 30deg);`
complement()	Returns the complementary color. `color: complement($base-color);`

String Functions

quote(), unquote()	Adds or removes quotes from a string.
	`content: quote(string);`
	`font-family: unquote("Open Sans");`

to-upper-case(), to-lower-case()	Converts a string to upper or lower case.
	`text-transform: to-upper-case(lowercase);`
	`text-transform: to-lower-case(UPPERCASE);`

str-length()	Returns the length of a string. `$length: str-length("hello world"); // Returns 11`
str-insert()	Inserts a substring into a string at a specified position. `$new-string: str-insert("abcd", "X", 1); // Returns "Xabcd"`
str-slice()	Extracts a substring from a string. `$substring: str-slice("hello", 2, 4); // Returns "ell"`

Number Functions

percentage()	Converts a number to a percentage. `width: percentage(0.5); // Returns 50%`

round(), ceil(), floor()	Rounds a number to the nearest integer.
	`$rounded: round(3.5); // Returns 4`
	`$ceiled: ceil(3.1); // Returns 4`
	`$floored: floor(3.9); // Returns 3`

abs()	Returns the absolute value of a number. `$absolute: abs(-5); // Returns 5`

min(), max()	Returns the minimum or maximum of a set of numbers.
	`$min: min(1, 2, 3); // Returns 1`
	`$max: max(1, 2, 3); // Returns 3`

random()	Returns a random number. `$random: random(100); // Returns a random number between 1 and 100`

Directives and Control Structures

Import and Use

@import	The `@import` directive includes another Sass file. It's being phased out in favor of `@use`.
	`@import 'variables';`
	`@import 'mixins';`

@use	The `@use` directive loads another Sass file as a module, preventing variable and mixin name collisions.
	`@use 'variables' as vars;`
	`@use 'mixins' as mix;`
	
	`body {`
	`  color: vars.$primary-color;`
	`  @include mix.border-radius(5px);`
	`}`

@forward	The `@forward` directive makes variables and mixins from another Sass file available without explicitly using `@use` in the current file. `@forward 'variables';`

Control Directives

@if, @else if, @else	Conditional statements. `@if $theme == 'dark' { background-color: #000; color: #fff; } @else { background-color: #fff; color: #000; }`
@for	For loops. `@for $i from 1 through 3 { .item-#{$i} { width: 20px * $i; } }`
@each	Each loops for lists and maps. `$sizes: 40px, 50px, 80px; @each $size in $sizes { font-size: $size; } $colors: (light: #fff, dark: #000); @each $key, $color in $colors { .theme-#{$key} { color: $color; } }`
@while	While loops. `$i: 1; @while $i <= 3 { .item-#{$i} { width: 10px * $i; } $i: $i + 1; }`

Other Directives

@extend	Shares the styles of one selector with another. `.error { border: 1px solid red; padding: 10px; } .seriousError { @extend .error; font-weight: bold; }`
@at-root	Causes one or more rules to be emitted at the root of the document. `nav { ul { @at-root { .global-style { color: black; } } } }`
@debug	Prints a SassScript expression to the standard error output stream. Useful for debugging. `@debug 1px + 1px; // Output: 2px`
@warn	Prints a warning to the standard error output stream. `@warn "This mixin is deprecated.";`
@error	Causes Sass to emit a fatal error and halt compilation. `@if $value == null { @error "Value cannot be null."; }`

Data Types and Operators

Data Types

Numbers	Integers and decimals.
	`$number: 10;`
	`$decimal: 3.14;`

Strings	Sequences of characters. `$string: "Hello, Sass!";`

Colors	Color values.
	`$color: #007bff;`
	`$rgba: rgba(0, 0, 0, 0.5);`

Booleans	`true` or `false`. `$is-active: true;`
Null	Represents the absence of a value. `$empty: null;`
Lists	Ordered sequences of values. `$list: 10px, 20px, 30px;`
Maps	Key-value pairs. `$map: (key1: value1, key2: value2);`

Operators

Arithmetic Operators	`+, -, *, /, %` for addition, subtraction, multiplication, division, and modulus.
	`$width: 10px + 5px; // 15px`
	`$height: 20px * 2;  // 40px`

Relational Operators	`>, <, >=, <=` for comparison.
	`@if $width > 10px {`
	`  width: 100%;`
	`}`

Equality Operators	`==, !=` for equality and inequality.
	`@if $theme == 'dark' {`
	`  color: white;`
	`}`

Logical Operators	`and, or, not` for logical operations.
	`@if ($width > 10px) and ($theme == 'dark') {`
	`  font-weight: bold;`
	`}`

String Operators	`+` for string concatenation. `$message: "Hello, " + "Sass!";`