Sass/SCSS Quick Reference

Variables, nesting, mixins, functions, control flow

Syntax

SCSS vs Sass

```
// SCSS (superset of CSS — uses braces)
.nav { display: flex; }

// Sass (indented — no braces or semicolons)
.nav
  display: flex
```

SCSS is the most commonly used syntax

Comparison

SCSS (.scss)	CSS-compatible, braces & semicolons
Sass (.sass)	Indentation-based, no braces
Output	Both compile to standard CSS
Recommended	SCSS (wider adoption, easier migration)

Variables

Defining & Using

```
$primary: #3498db;
$spacing: 16px;
$font-stack: "Helvetica", Arial, sans-serif;

.btn {
  color: $primary;
  padding: $spacing;
  font-family: $font-stack;
}
```

Variable Scope

```
$color: red;        // global
.card {
  $color: blue;     // local to .card
  color: $color;    // blue
}
.other { color: $color; } // red
```

Flags

!default	Set only if not already defined
!global	Promote local var to global scope

Nesting

Selector Nesting

```
.nav {
  ul { list-style: none; }
  li { display: inline-block; }
  a { text-decoration: none;
    &:hover { color: blue; }   // & = parent selector
  }
}
```

Parent Selector (&)

```
.btn {
  &--primary { background: blue; }  // BEM: .btn--primary
  &__icon { margin-right: 4px; }    // BEM: .btn__icon
  .dark & { color: white; }         // .dark .btn
}
```

Property Nesting

```
.box {
  border: { width: 1px; style: solid; color: #ccc; }
  // compiles to: border-width, border-style, border-color
}
```

Mixins

Define & Include

```
@mixin flex-center($direction: row) {
  display: flex;
  justify-content: center;
  align-items: center;
  flex-direction: $direction;
}

.hero { @include flex-center(column); }
```

Content Blocks

```
@mixin responsive($breakpoint) {
  @media (min-width: $breakpoint) { @content; }
}

.sidebar {
  width: 100%;
  @include responsive(768px) { width: 300px; }
}
```

Mixin Features

@mixin name($args)	Define reusable style block
@include name()	Use the mixin
Default params	**$arg: value** for optional params
$args...	Variable arguments (rest params)
@content	Inject caller's content block

Functions

Custom Functions

```
@function rem($px, $base: 16) {
  @return math.div($px, $base) * 1rem;
}
.title { font-size: rem(24); } // 1.5rem
```

Built-in Functions

darken($color, 10%)	Darken a color
lighten($color, 10%)	Lighten a color
mix($c1, $c2, 50%)	Mix two colors
rgba($color, 0.5)	Set alpha channel
math.div($a, $b)	Division (replaces **/**)
math.round($n)	Round number
string.quote($s)	Add quotes to string
if($cond, $t, $f)	Inline conditional

Extends

Extend & Placeholder

```
%flex-center {          // placeholder — not emitted
  display: flex;
  justify-content: center;
  align-items: center;
}

.hero { @extend %flex-center; }
.modal { @extend %flex-center; }
// Both share one CSS rule via selector grouping
```

Extend vs Mixin

@extend	Groups selectors — smaller CSS output
@mixin	Copies declarations — supports arguments
% placeholder	Extend-only (not emitted if unused)
Recommendation	Prefer mixins for parameterized styles

Partials & Import

File Organization

```
// _variables.scss (partial — not compiled alone)
$primary: #3498db;

// main.scss
@use "variables";          // modern: namespaced
.btn { color: variables.$primary; }

@use "variables" as v;     // alias
.btn { color: v.$primary; }
```

Module System

@use 'file'	Load module with namespace
@use 'file' as *	Load without namespace
@use 'file' as alias	Custom namespace
@forward 'file'	Re-export module members
_partial.scss	File not compiled independently

@import is deprecated — use @use and @forward instead

Control Flow

Conditionals

```
@mixin theme($mode) {
  @if $mode == dark {
    background: #333; color: #fff;
  } @else {
    background: #fff; color: #333;
  }
}
```

Loops

```
@for $i from 1 through 4 {
  .col-#{$i} { width: 25% * $i; }
}

@each $name, $color in (primary: blue, danger: red) {
  .text-#{$name} { color: $color; }
}
```

Directives

@if / @else if / @else	Conditional logic
@for $i from a through b	Numeric loop (inclusive)
@for $i from a to b	Numeric loop (exclusive end)
@each $item in $list	Iterate list or map
@while	Loop while condition is true
#{$var}	Interpolation in selectors/props

Maps & Lists

Maps

```
$colors: (primary: #3498db, danger: #e74c3c, success: #2ecc71);

.alert { color: map.get($colors, danger); }

@each $name, $color in $colors {
  .bg-#{$name} { background: $color; }
}
```

Lists

```
$sizes: 8px 16px 24px 32px;
.box { padding: list.nth($sizes, 2); } // 16px
```

Sass/SCSS Quick Reference

Map & List Functions

`map.get($map, $key)`	Get value by key
`map.merge($m1, $m2)`	Merge two maps
`map.keys($map)`	List of all keys
`map.has-key($map, $key)`	Check if key exists
`list.nth($list, $n)`	Get item at index (1-based)
`list.length($list)`	Number of items
`list.append($list, $val)`	Add item to list

Common Patterns

Responsive Breakpoints

```
$breakpoints: (sm: 576px, md: 768px, lg: 992px, xl: 1200px);

@mixin bp($name) {
  @media (min-width: map.get($breakpoints, $name)) {
    @content;
  }
}
.sidebar { width: 100%; @include bp(md) { width: 300px; } }
```

Utility Generator

```
$spaces: (0: 0, 1: 4px, 2: 8px, 3: 16px, 4: 32px);
@each $key, $val in $spaces {
  .mt-#{$key} { margin-top: $val; }
  .mb-#{$key} { margin-bottom: $val; }
  .p-#{$key} { padding: $val; }
}
```

Dark Mode

```
@mixin dark { @media (prefers-color-scheme: dark) { @content; } }
body {
  background: #fff;
  @include dark { background: #1a1a1a; color: #eee; }
}
```