Référence rapide Sass/SCSS

Variables, imbrication, mixins, fonctions, flux de contrôle

Syntaxe

SCSS vs Sass

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

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

SCSS est la syntaxe la plus utilisée

Comparaison

SCSS (.scss)	Compatible CSS, accolades et points-virgules
Sass (.sass)	Basé sur l'indentation, sans accolades
Output	Les deux compilent en CSS standard
Recommended	SCSS (adoption plus large, migration plus facile)

Variables

Définir et utiliser

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

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

Portée des variables

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

Indicateurs

!default	Définir uniquement si pas encore défini
!global	Promouvoir la variable locale en portée globale

Imbrication

Imbrication de sélecteurs

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

Sélecteur parent (&)

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

Imbrication de propriétés

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

Mixins

Définir et inclure

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

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

Blocs de contenu

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

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

Fonctionnalités des mixins

@mixin name($args)	Définir un bloc de styles réutilisable
@include name()	Utiliser le mixin
Default params	**$arg: value** pour les paramètres optionnels
$args...	Arguments variables (rest params)
@content	Injecter le bloc de contenu de l'appelant

Fonctions

Fonctions personnalisées

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

Fonctions intégrées

darken($color, 10%)	Assombrir une couleur
lighten($color, 10%)	Éclaircir une couleur
mix($c1, $c2, 50%)	Mélanger deux couleurs
rgba($color, 0.5)	Définir le canal alpha
math.div($a, $b)	Division (remplace **/**)
math.round($n)	Arrondir un nombre
string.quote($s)	Ajouter des guillemets à une chaîne
if($cond, $t, $f)	Conditionnel en ligne

Extensions

Extend et 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	Regroupe les sélecteurs — sortie CSS plus petite
@mixin	Copie les déclarations — supporte les arguments
% placeholder	Extend uniquement (non émis si inutilisé)
Recommendation	Préférer les mixins pour les styles paramétrés

Partiels et imports

Organisation des fichiers

```
// _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; }
```

Système de modules

@use 'file'	Charger un module avec espace de noms
@use 'file' as *	Charger sans espace de noms
@use 'file' as alias	Espace de noms personnalisé
@forward 'file'	Ré-exporter les membres du module
_partial.scss	Fichier non compilé indépendamment

@import est déprécié — utiliser @use et @forward à la place

Flux de contrôle

Conditionnelles

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

Boucles

```
@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	Logique conditionnelle
@for $i from a through b	Boucle numérique (inclusive)
@for $i from a to b	Boucle numérique (fin exclusive)
@each $item in $list	Itérer une liste ou une map
@while	Boucle tant que la condition est vraie
#{$var}	Interpolation dans les sélecteurs/ propriétés

Maps et listes

Maps

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

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

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

Listes

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

Référence rapide Sass/SCSS

Fonctions de map et liste

`map.get($map, $key)`	Obtenir une valeur par clé
`map.merge($m1, $m2)`	Fusionner deux maps
`map.keys($map)`	Liste de toutes les clés
`map.has-key($map, $key)`	Vérifier si une clé existe
`list.nth($list, $n)`	Obtenir l'élément à l'index (base 1)
`list.length($list)`	Nombre d'éléments
`list.append($list, $val)`	Ajouter un élément à la liste

Motifs courants

Points de rupture responsive

```
$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; } }
```

Générateur d'utilitaires

```
$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; }
}
```

Mode sombre

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