[Nom de la société]

# Préprocesseurs CSS SASS & SCSS

Introduction

Guillaume Rodrigues
[Date]

# Table des matières

# Introduction au SCSS

## Qu'est-ce que SASS ?

Sass est le sigle de Syntatically Awesome Stylesheets. C’est un préprocesseur CSS.

L’objectif d’un préprocesseur CSS est de générer dynamiquement des fichiers CSS pour un site web. Cette génération dynamique permet de simplifier l’écriture du CSS et d’offrir de nombreuses possibilités nouvelles aux développeurs.

Sass permet de créer des variables, des règles encapsulées, des mixins, des fonctions et même d’utiliser des opérations mathématiques. Grâce à cet outil, les développeurs écrivent des fichiers CSS plus lisibles, plus performants et plus facilement maintenable.

De plus la transition est très simple : Sass est parfaitement compatible avec le code et la syntaxe CSS, ce qui permet à quiconque à l’habitude du CSS classique d’adopter Sass à son rythme.

Chaque fichier CSS étant aussi un fichier Sass, il vous suffit de déplacer le contenu de vos fichiers .css dans des fichiers Sass et tout devrait se comporter comme avant !

En revanche, il sera nécessaire de compiler nos fichiers sass ou scss. Cela signifie qu’un compileur va parcourir toutes les instructions des différents fichiers afin de créer la feuille de styles (css) finale.

**Syntaxes possibles** :

- **SASS** : fichiers « .sass ». Les instructions ne sont pas délimitées par des accolades mais par l'identation.

```
.div
        margin: 10px
        font-size: 14px
        border-radius: 5px
        border: 1px solid black
```

- **SCSS** : fichiers « .scss ». La syntaxe du CSS est conservée (accolades pour séparer les jeux de règles, et « ; » pour séparer les instructions.

```
.div {
        margin: 10px;
        font-size: 14px;
        border-radius: 5px;
        border: 1px solid black;
}
```

## Qu'est-ce que SCSS ?

Comme vu juste au-dessus, SCSS (Sassy CSS) est simplement une syntaxe de SASS, un langage de préprocesseur CSS qui permet d'écrire des styles de manière plus élégante et efficace.

Il faut savoir que les compileurs sass et scss sont compatibles entre eux. Donc au final, vous pourrez choisir le format que vous préférez, les fonctionnalités étant les mêmes.

## Syntaxe de base

### 1. Variables

Les variables permettent de stocker des valeurs réutilisables.

En SCSS, elles sont définies avec un $.

```scss
$primary-color: #3498db;

body {
  color: $primary-color;
}
```

### 2. Imbrication/Nesting

SCSS permet d'imbriquer les règles CSS, ce qui facilite la gestion des sélecteurs descendants.

```scss
nav {
    background: #333;

    ul {
        list-style: none;

        li {
            display: inline-block;

            a {
                color: white;
                text-decoration: none;

                &:hover {
                    color: red;
                }
            }
        }
    }
}
```

Ce qui, après compilation, sera converti en css :

```
nav {
  background: #333;
}
nav ul {
  list-style: none;
}
nav ul li {
  display: inline-block;
}
nav ul li a {
  color: white;
  text-decoration: none;
}
nav ul li a:hover {
  color: red;
}
```

### 3. Partials et importation

Les partials sont des fichiers SCSS partiels qui peuvent être importés dans d'autres fichiers SCSS. Un fichier partial commence par un underscore : « _ ».

L'utilisation de ces partials est extrêmement utilise pour améliorer la maintenabilité et réutilisabilité du code.

Ainsi, on peut imaginer un « partial » qui stocke toutes nos variables, un partial qui gère le design de la navbar (et qui sera embarqué dans chaque page utilisant une navbar).

- Fichier « partial » : _variables.scss

```
$primary-color: #3498db; //Défintion d'une variable
```

- Fichier main.scss

```
@import 'variables';

body {
    color: $primary-color; //Utilisation de la variable dans la color du
body
}
```

- Résultat : styles.css

```
body {
  color: #3498db;
}
```

**4. Mixins**

Les mixins sont des blocs de code réutilisables que vous pouvez inclure dans d'autres règles SCSS.

```
@mixin border-radius($radius) {
    -webkit-border-radius: $radius;
    -moz-border-radius: $radius;
    border-radius: $radius;
}

.box {
    @include border-radius(10px);
}
```

**5. Héritage**

L'héritage permet à un sélecteur de réutiliser les propriétés d'un autre sélecteur.

```
%message-shared {
    border: 1px solid #ccc;
    padding: 10px;
    color: #333;
}

.message {
    @extend %message-shared;
}

.success {
    @extend %message-shared;
    border-color: green;
}

.error {
    @extend %message-shared;
    border-color: red;
}
```

### 6. Opérateurs

SCSS supporte l'utilisation d'opérateurs mathématiques comme +, -, *, /, et %.

```
.container {
    width: 100% - 20px;
    margin: 10px 5px;
}
```

## Compilation de SCSS

Les fichiers SCSS vont devoir être compilés (transformés en .css) pour pouvoir être liés à nos pages web. Au cours de la compilation, il sera aussi possible de demander à « minifier » le fichier CSS final afin de réduire son poids au maximum.

Les compilateurs sont compatibles avec SASS et SCSS.

Pour pouvoir installer les compilateurs, nous allons avoir besoin de « npm », l'outil de dépendances utilisé par NodeJS.

Dans un premier temps donc, installons NodeJS : https://nodejs.org/fr

### Outils de compilation

1. **Node-sass**

   Node-sass est un des compilateurs les plus populaires. Vous pouvez l'installer via npm.

   ```
   npm install -g node-sass
   ```

   Puis, pour compiler un fichier SCSS :

   ```
   node-sass style.scss style.css
   ```

2. **Dart-sass**

   Dart-sass est le compilateur officiel et recommandé par l'équipe de SASS.

   ```
   npm install -g sass
   ```

   Pour compiler un fichier SCSS :

   ```
   sass style.scss style.css
   ```

3. **VSCode Extensions**

   Utilisez l'extension Live Sass Compiler pour compiler SCSS en temps réel.

   - Allez dans l'onglet des extensions de VSCode.
   - Recherchez Live Sass Compiler et installez-le.
   - Une fois installé, vous pouvez cliquer sur Watch Sass en bas de VSCode pour compiler automatiquement vos fichiers SCSS.
4. **PHPStorm**

   PHPStorm a un support intégré pour la compilation de SASS/SCSS. Vous pouvez configurer cela via les File Watchers.

   - Allez dans File > Settings > Tools > File Watchers.
   - Cliquez sur le + pour ajouter un nouveau File Watcher.
   - Sélectionnez SCSS dans la liste.
   - Configurez les détails nécessaires comme l'emplacement du compilateur SCSS.

## Utilisation d'outils de développement

Pour automatiser la compilation et l'importation, vous pouvez utiliser des outils de développement comme Gulp ou Webpack.

Par exemple, on peut lancer Gulp pour qu'il surveille les modifications de nos fichiers scss et pour qu'il rebuild les css à la volée. C'est très pratique lorsqu'on est en phase de développement.

### Utilisation de Gulp

1. **Installation de Gulp et des plugins nécessaires** :

```
npm install gulp gulp-sass –save-dev
```

2. **Configuration de Gulp** (fichier gulpfile.js) :

```
const gulp = require('gulp');
const sass = require('gulp-sass')(require('sass'));

gulp.task('sass', function () {
    return gulp.src('styles.scss')
        .pipe(sass().on('error', sass.logError))
        .pipe(gulp.dest('./'));
});

gulp.task('watch', function () {
    gulp.watch('styles.scss', gulp.series('sass'));
});

gulp.task('default', gulp.series('sass', 'watch'));
```

3. **Lancer Gulp** :

```
gulp
```

Cela permet de surveiller automatiquement les modifications de votre fichier SCSS et de le compiler en CSS.