

## Préprocesseur SASS



**Auteur :**

Yoann Depriester

**Date création :**

16-05-2023

**Relu, validé & visé par :**

☒ Jérôme CHRETIENNE  
☒ Sophie POULAKOS  
☒ Mathieu PARIS

**Date révision :**

16-05-2023

## Préprocesseur SASS

### Table des matières

Qu'est-ce que SASS ? .....	3
Préprocesseur.....	3
Installer SASS en ligne de commande .....	4
Lancer le Preprocessing.....	4
Surveiller un fichier .SCSS individuellement.....	4
Surveiller un répertoire contenant plusieurs fichiers .SCSS .....	4
Utiliser SASS grâce à l'extension Live Sass Compiler .....	5
Premiers Pas avec SASS .....	6
Les Variables .....	7
Le Nesting .....	8
Le Nesting : gagner en efficacité grâce au BEM .....	9
BLOC .....	9
ELEMENT .....	10
MODIFIER .....	10
Combiner BEM et Nesting grâce au & pour un maximum d'efficacité.....	11
Les Partials et les Modules .....	14
Gestion des Partials : le système de fichiers 7-1 .....	15
Les Mixins .....	16
Les Extends .....	17

#### Auteur :

Yoann Depriester

#### Date création :

16-05-2023

#### Relu, validé & visé par :

☒ Jérôme CHRETIENNE  
☒ Sophie POULAKOS  
☒ Mathieu PARIS

#### Date révision :

16-05-2023



Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.

## Préprocesseur SASS

### Qu'est-ce que SASS ?

Sass (Syntactically Awesome Style Sheets), soit « des feuilles de styles syntaxiquement géniales ».

C'est un langage de script préprocesseur compilé en CSS, créé par Natalie Weizenbaum et Chris Eppstein en 2006.

Sass a très vite gagné en popularité en facilitant et optimisant la création des feuilles de style CSS. Même si par la suite le langage CSS a connu des évolutions qui ont rattrapé Sass (comme l'utilisation de variables que l'on retrouve désormais en CSS), Sass est toujours populaire, on le retrouve notamment au cœur de certains frameworks CSS très utilisés (Bootstrap, Materialize, Bulma, etc...)

Sass est un préprocesseur CSS. (Il en existe d'autres comme Less ou encore PostCss)

### Préprocesseur

Un Préprocesseur est un programme qui procède à des transformations sur un code source, avant l'étape de traduction proprement dite (compilation ou interprétation).

Sass va apporter plus de fonctionnalités à CSS, notamment, l'utilisation de variables, l'import de fichier, l'imbrication des sélecteurs, etc... Tout cela va nous permettre d'écrire du Code CSS plus intuitivement, et plus structuré.

Nous allons donc écrire du code CSS (avec des fonctionnalités en plus) dans des fichiers scss ou sass, les navigateurs ne pourront pas interpréter directement ce type de fichier, c'est pourquoi notre code en scss sera recompilé en CSS classique pour les navigateurs.

Site Officielle : <https://sass-lang.com/>

#### Auteur :

Yoann Depriester

#### Date création :

16-05-2023

#### Relu, validé & visé par :

☒ Jérôme CHRETIENNE  
☒ Sophie POULAKOS  
☒ Mathieu PARIS

#### Date révision :

16-05-2023



Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.

## Préprocesseur SASS

### Installer SASS en ligne de commande

- 1) Installer Node.js
- 2) Ouvrir un terminal et entrer la commande :

```
npm install sass -g
```

- 3) Pour s'assurer que Sass est bien installé, entrer dans le terminal la commande :

```
sass --version
```

### Lancer le Preprocessing

Une fois SASS correctement installé, il est possible de le lancer dans le terminal pour qu'il compile automatiquement à chaque sauvegarde un fichier .SCSS en un fichier .CSS. Pour cela, il faut soit demander à SASS de surveiller un fichier .SCSS individuellement, ou de surveiller un répertoire pouvant contenir plusieurs fichiers.

#### Surveiller un fichier .SCSS individuellement

- 1) Ouvrir un terminal et s'assurer d'être dans le répertoire du projet
- 2) Entrer la commande :

```
sass --watch repertoire_du_fichier/fichier.scss repertoire_ou_placer_le_css/fichier.css
```

#### Exemple :

Si, dans votre répertoire /projet, notre fichier main.scss se trouve dans le répertoire /SCSS, et que l'on souhaite qu'il soit compilé en un fichier style.css se trouvant dans un répertoire /CSS, nous devons entrer :

```
sass --watch SCSS/main.scss CSS/style.css
```

#### Surveiller un répertoire contenant plusieurs fichiers .SCSS

- 1) Ouvrir un terminal et s'assurer d'être dans le répertoire du projet
- 2) Entrer la commande :

```
sass --watch repertoire_des_fichiers_SCSS :repertoire_des_fichiers_CSS
```

#### Exemple :

Si, dans votre répertoire /projet, nos fichiers main.scss et gallery.scss se trouvent dans le répertoire /SCSS, et que l'on souhaite qu'ils soient compilés dans le répertoire /CSS, nous devons entrer :

```
sass --watch SCSS :CSS
```

#### Auteur :

Yoann Depriester

#### Relu, validé & visé par :

☒ Jérôme CHRETIENNE  
☒ Sophie POULAKOS  
☒ Mathieu PARIS

#### Date création :

16-05-2023

#### Date révision :

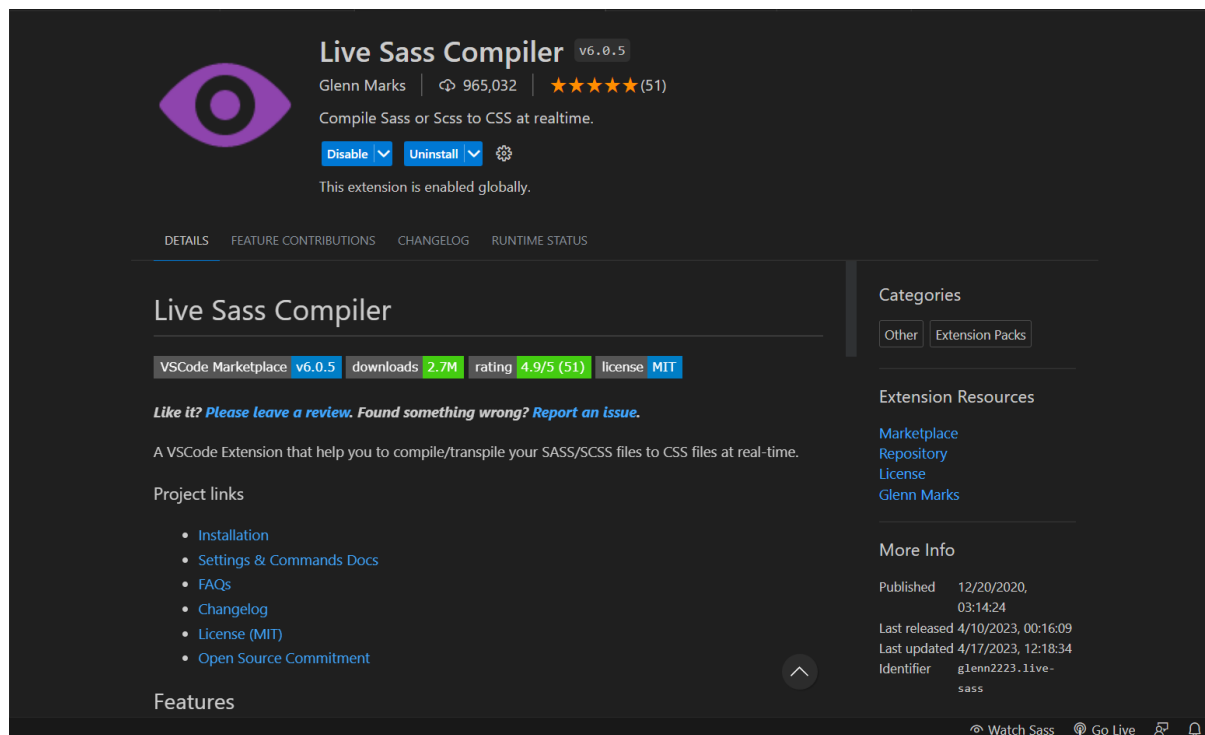
16-05-2023



Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.

## Préprocesseur SASS

Utiliser SASS grâce à l'extension Live Sass Compiler



The screenshot shows the VS Code Marketplace page for the 'Live Sass Compiler' extension by Glenn Marks. The extension is version 6.0.5, has 965,032 downloads, and a 4.9/5 star rating from 51 reviews. It is described as a tool to compile Sass or Scss to CSS in real-time. The page includes tabs for Details, Feature Contributions, Changelog, and Runtime Status. The 'Details' tab is active, showing the extension's name, version, downloads, rating, and license (MIT). It also lists project links such as Installation, Settings & Commands Docs, FAQs, Changelog, License (MIT), and Open Source Commitment. On the right, there are sections for Categories (Other, Extension Packs), Extension Resources (Marketplace, Repository, License, Glenn Marks), and More Info (Published date, Last released, Last updated, Identifier).

Live Sass Compiler est une extension pour VS Code. Par défaut, il surveille les fichiers .SCSS et les compile automatiquement à chaque sauvegarde dans le même répertoire.

Ainsi, si nous avons un fichier main.scss dans un répertoire /SCSS, Live Sass Compiler le compilera en un fichier main.css dans le répertoire /SCSS.

### Auteur :

Yoann Depriester

### Relu, validé & visé par :

☒ Jérôme CHRETIENNE  
☒ Sophie POULAKOS  
☒ Mathieu PARIS

### Date création :

16-05-2023

### Date révision :

16-05-2023



Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.

## Préprocesseur SASS

### Premiers Pas avec SASS

La première chose à savoir avec SASS, c'est qu'en plus d'avoir une syntaxe similaire à celle des langages de programmation (grâce à l'usage de variable, boucle, fonction, ...), il supporte très bien l'écriture de code CSS.

Dans l'exemple ci-dessous, nous définissons des règles pour l'ensemble de nos balises, importons des polices d'écritures nécessaires pour le design de notre site, et on règle la taille de police sur la balise html pour utiliser plus tard l'unité **rem** en toute quiétude.

```
/* ***** */
    RESET CSS
/* ***** */
* {
    margin: 0;
    padding : 0;
    text-decoration: none;
    list-style: none;
}

/* ***** */
    IMPORT DE FONTS
/* ***** */

@font-face {
    font-family: AbrilFatface;
    src: url(./Font/AbrilFatface-Regular.ttf);
}
@font-face {
    font-family: MinionItalic;
    src: url(./Font/MinionPro-It.otf);
}

/* ***** */
    FONT SIZE DU ROOT
/* ***** */
html {
    font-size: 16pt;
}
```

#### Auteur :

Yoann Depriester

#### Date création :

16-05-2023

#### Relu, validé & visé par :

☒ Jérôme CHRETIENNE  
☒ Sophie POULAKOS  
☒ Mathieu PARIS

#### Date révision :

16-05-2023



Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.

## Préprocesseur SASS

### Les Variables

Un apport important de SASS est l'utilisation de variables. Les variables sont des conteneurs permettant de stocker des valeurs, pour les exploiter plus tard à différents endroits du code.

#### AVANTAGE :

- Eviter d'avoir les même valeurs qui se répètent partout dans le CSS
- Facilité de mise à jour des valeurs CSS en n'ayant besoin de les modifier qu'à un seul et unique endroit

Définir une variable utilise la syntaxe suivante :

*\$nomVariable : valeur\_de\_la\_variable ;*

Puis, pour l'utiliser, on l'appelle simplement par son nom :

*color : \$nomVariable ;*

#### Exemple :

```
/******  
    VARIABLES  
******/  
//COULEURS  
$primary-color: wheat;  
$secondary-color: orange;  
$tertiary-color : #333;  
  
//FONT-FAMILY  
$font-title : AbrilFatface;  
$font-legend : MinionItalic;  
$font-regular : helvetica, verdana, arial, sans-serif;  
  
//FONT-SIZE  
$size-big-title : 3rem;  
$size-second-title : 2rem;  
$size-legend : 1.5rem;  
$size-regular : 1rem;  
  
h1 {  
    color : $secondary-color;  
    font-family: $font-title;  
    font-size: $size-big-title;  
}
```

#### Auteur :

Yoann Depriester

#### Relu, validé & visé par :

☒ Jérôme CHRETIENNE  
☒ Sophie POULAKOS  
☒ Mathieu PARIS

#### Date création :

16-05-2023

#### Date révision :

16-05-2023



Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.

## Préprocesseur SASS

### Le Nesting

Le cœur de la syntaxe SASS est le Nesting. Le principe du Nesting est d'imbriquer les sélecteur CSS les uns dans les autres. C'est-à-dire qu'un sélecteur, ainsi que ses règles CSS, se verra être déclaré entre les accolades d'un premier sélecteur. Une fois compilé, ils formeront une combinaison de sélecteur. Cela permet de gagner en clarté grâce à une hiérarchie visuelle similaire à celle du HTML.

#### Exemple :

*Ici un sélecteur **li** est imbriqué dans un sélecteur **ul***

```
ul {  
  display: flex;  
  li {  
    color : orange;  
    font-family: helvetica, verdana, arial, sans-serif;  
  }  
}
```

*Un fois compilé en CSS, nous voyons la combinaison **ul li** apparaître.*

```
ul {  
  display: flex;  
}  
  
ul li {  
  color : orange;  
  font-family: helvetica, verdana, arial, sans-serif;  
}
```

#### AVANTAGE :

- La hiérarchie de notre html est conservée au sein du SCSS
- Le code est donc plus propre
- Le code est plus facile à maintenir, car les endroits à modifier sont plus facilement accessible, en évitant de chercher une aiguille dans une botte de foin
- Cela fonctionne avec les combinateurs tels que +, ~ ou bien >

#### Auteur :

Yoann Depriester

#### Date création :

16-05-2023

#### Relu, validé & visé par :

☒ Jérôme CHRETIENNE  
☒ Sophie POULAKOS  
☒ Mathieu PARIS

#### Date révision :

16-05-2023



Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.



## Préprocesseur SASS

### Le Nesting : gagner en efficacité grâce au BEM

Le BEM est tout simplement une manière efficace et conventionnelle de nommer les class d'un élément HTML. Il signifie tout simplement Bloc – Element – Modifier.

Cette convention de nommage a pour particularité, outre d'être clair sur le rôle de chaque élément HTML, de rendre encore plus efficace le Nesting de SASS.

#### BLOC

Dans la convention de nommage BEM, les blocs sont les éléments macroscopiques constituant une page d'un site. Ils sont facilement identifiables par leur fonction propre, et sont donc autonomes et indépendants du reste (mais il est possible d'avoir un Bloc dans un autre Bloc, comme un Menu de Navigation au sein d'un Header).

#### Exemple :

- Le Header, une Card, une Barre de Recherche, le Menu de Navigation sont des Blocs, car ces éléments se suffisent à eux même pour fonctionner et avoir du sens.
- Un titre ou une image ne sont pas des Blocs, car leur fonction et leur sens dépend de leur contexte d'utilisation.

Une fois un Bloc identifié, on peut le nommer (généralement via une class). Le plus simple est de lui octroyer un nom décrivant sa fonction.

#### Exemple :

Ici nous avons identifié 2 Blocs au sein de notre header, notre Menu de Navigation primaire qu'on a nommé **.primaryNav**, ainsi qu'une Liste de Réseau Sociaux qu'on a nommé **.socialNetwork**

```
<header>
  <nav class="primaryNav">
    <a href="">Accueil</a>
    <a href="">Blog</a>
    <a href="">Contact</a>
  </nav>

  <ul class="socialNetwork">
    <li><a href=""></a></li>
    <li><a href=""></a></li>
    <li><a href=""></a></li>
  </ul>
</header>
```

#### Auteur :

Yoann Depriester

#### Date création :

16-05-2023

#### Relu, validé & visé par :

☒ Jérôme CHRETIENNE  
☒ Sophie POULAKOS  
☒ Mathieu PARIS

#### Date révision :

16-05-2023



Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.

## Préprocesseur SASS

### ELEMENT

Un Element est une pièce constituante d'un Bloc. Seuls, ils n'auraient pas beaucoup de sens, mais montés ensemble, ils donnent corps au Bloc en lui donnant une fonction.

*Exemple : Un titre, une image de miniature, un court texte et un lien permettent de donner forme à une Card qui présente un article sur la page d'accueil d'un blog.*

Pour nommer un Element, on doit indiquer le nom du bloc dont il fait parti, suivi d'un double underscore (certains développeurs utilisent uniquement un underscore simple), puis un nom décrivant l'Element. Le tout étant attaché ensemble évidemment.

Exemple :

*Ici nous avons nommé les liens de notre Menu Navigation et les éléments de Liste de nos Réseaux Sociaux en reprenant le nom de leur Bloc suivi de `__link`*

```
<header>
  <nav class="primaryNav">
    <a class="primaryNav__link" href="">Accueil</a>
    <a class="primaryNav__link" href="">Blog</a>
    <a class="primaryNav__link" href="">Contact</a>
  </nav>

  <ul class="socialNetwork">
    <li class="socialNetwork__link"><a href=""></a></li>
    <li class="socialNetwork__link"><a href=""></a></li>
    <li class="socialNetwork__link"><a href=""></a></li>
  </ul>
</header>
```

### MODIFIER

Imaginons que nous ayons un ensemble de Blocs ou d'Elements similaire (Card, Titre d'article). Tous ont une apparence identique, mais nous voudrions inclure un peu de variation, par exemple pour faire ressortir une Card ou un Titre du reste. Nous voulons donc donner à ce Bloc ou à cet Element unique des règles CSS qui lui sont propre. Pour cela, nous allons avoir besoin d'un Modifier.

*Exemple : Au sein de notre Menu de Navigation primaire, nous souhaitons que mettre en surbrillance le lien correspondant à la page en train d'être visitée.*

#### Auteur :

Yoann Depriester

#### Date création :

16-05-2023

#### Relu, validé & visé par :

☒ Jérôme CHRETIENNE  
☒ Sophie POULAKOS  
☒ Mathieu PARIS

#### Date révision :

16-05-2023



Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.

## Préprocesseur SASS

Pour nommer un Modifier, on reprend le nom du Bloc ou de l'Element, auquel on accole un double tirets (encore une fois, certains développeurs n'utilise qu'un seul tiret) et un nom décrivant le Modifier.

Exemple :

Ici nous avons donné un Modifier au premier lien de notre Menu de Navigation primaire, en accord avec la convention BEM. On a repris le nom de l'Element auquel on a rajouté **--active**

```
<header>
  <nav class="primaryNav">
    <a class="primaryNav__link primaryNav__link--active"
href="">Accueil</a>
    <a class="primaryNav__link" href="">Blog</a>
    <a class="primaryNav__link" href="">Contact</a>
  </nav>

  <ul class="socialNetwork">
    <li class="socialNetwork__link"><a href=""></a></li>
    <li class="socialNetwork__link"><a href=""></a></li>
    <li class="socialNetwork__link"><a href=""></a></li>
  </ul>
</header>
```

### Combiner BEM et Nesting grâce au & pour un maximum d'efficacité

Lorsque nous avons vu le Nesting au tout début, nous avons vu que l'imbrication des sélecteurs étaient compilée en CSS en laissant un espace entre, ce qui permet de créer les combinaisons de sélecteur. Cependant il existe une autre syntaxe de Nesting, bien plus efficace, utilisant le symbole &

Ce symbole & va permettre de coller directement tous les caractères qui le suivent, juste derrière le sélecteur dans lequel ils sont imbriqués, sans laisser d'espace.

#### Auteur :

Yoann Depriester

#### Date création :

16-05-2023

#### Relu, validé & visé par :

☑ Jérôme CHRETIENNE  
 ☑ Sophie POULAKOS  
 ☑ Mathieu PARIS

#### Date révision :

16-05-2023



Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.

## Préprocesseur SASS

Exemple : Ici nous appliquons des règles CSS à notre Menu de Navigation en tirant parti du Nesting et de la convention de langage BEM.

```
.primaryNav{
  display: flex;
  & a:visited {
    text-decoration: none;
  }
  &:hover {
    background-color: wheat;
  }
  &__link {
    color : #333;
    &:hover {
      color : orange;
    }
    &--active{
      color : orange;
    }
  }
}
```

### Auteur :

Yoann Depriester

### Date création :

16-05-2023

### Relu, validé & visé par :

☒ Jérôme CHRETIENNE  
☒ Sophie POULAKOS  
☒ Mathieu PARIS

### Date révision :

16-05-2023



Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.

## Préprocesseur SASS

Une fois compilé, le SCSS donnera le CSS ci-dessous :

```
.primaryNav {  
  display: flex;  
}  
.primaryNav a:visited {  
  text-decoration: none;  
}  
.primaryNav:hover {  
  background-color: wheat;  
}  
.primaryNav__link {  
  color: #333;  
}  
.primaryNav__link:hover {  
  color: orange;  
}  
.primaryNav__link--active {  
  color: orange;  
}
```

Nous voyons bien comment le symbole **&** a reconstitué nos combinaisons de sélecteur (en conservant l'espace lorsqu'on en mettait un directement à la suite), les noms de nos classes (en collant directement les morceaux de nom entre eux), ou même en permettant de manière similaire l'utilisation de pseudo-classe ou pseudo-élément.

### Auteur :

Yoann Depriester

### Date création :

16-05-2023

### Relu, validé & visé par :

☒ Jérôme CHRETIENNE  
☒ Sophie POULAKOS  
☒ Mathieu PARIS

### Date révision :

16-05-2023



Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.

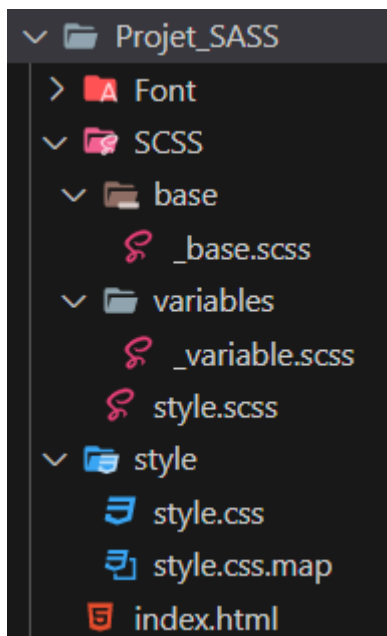
## Préprocesseur SASS

### Les Partials et les Modules

Il n'est pas nécessaire d'écrire l'entièreté de notre code SCSS dans un seul fichier. Il est même préférable de découper notre code en plusieurs fichiers thématiques afin de gagner en maintenabilité. SASS nous permet de faire une telle chose grâce aux Partials.

Un Partial est un fichier SCSS dont le nom commence par un underscore. Lors de la compilation en fichier CSS, un Partial ne sera pas compilé de manière automatique. Pour l'être, il doit être importé en tant que Module dans un fichier SCSS normal. Pour ce faire, on utilise l'instruction **@import**, suivi du chemin relatif vers le Partial, sans inclure l'underscore de son nom (très important).

*Exemple : Ici nous avons déplacé une partie de notre SCSS dans un fichier **\_base.scss** au sein du répertoire **/base**, et nos variables dans un fichier **\_variable.scss** au sein du répertoire **/variables**.*



Pour les importer en tant que module, on utilise **@import**, suivi du chemin menant à chaque Partial

```
@import 'base/base';
@import 'variables/variable';

h1 {
  color: $secondary-color;
  font-family: $font-title;
  font-size: $size-big-title;
}
```

#### Auteur :

Yoann Depriester

#### Relu, validé & visé par :

☑ Jérôme CHRETIENNE  
☑ Sophie POULAKOS  
☑ Mathieu PARIS

#### Date création :

16-05-2023

#### Date révision :

16-05-2023



Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.

## Préprocesseur SASS

### Gestion des Partials : le système de fichiers 7-1

Afin de mieux s'y retrouver lorsque l'on découpe notre SCSS en Partials, une bonne pratique est de s'appuyer sur une structure conventionnelle de fichiers.

La structure que nous allons voir se nomme le **Système de Fichiers 7-1**. Le 7 correspond aux 7 répertoires thématiques pour découper le code, et le 1 correspond au fichier SCSS principal qui va les importer pour les compiler. Ces répertoires sont les suivant :

1. Base : fondation du site comme les polices de caractères et les normes appliquées sur tout le site comme le box-sizing.
2. Utils : variables, fonctions, mixins et extensions.
3. Layout (Mise en Page) : les blocs BEM qui contiennent ce qui est réutilisable, comme un header pour les mises en pages de grande taille ou un footer. Il s'agit de blocs globaux.
4. Components : les blocs BEM plus indépendants comme les boutons. Il s'agit de blocs particuliers, plus petit que les Layouts mais qui leur donnent corps par leur réunion.
5. Pages : blocs de code spécifiques à une page non réutilisés ailleurs.
6. Thèmes : code thématique, comme un style customisé pour Noël ou pour l'été.
7. Vendors (Tiers) : feuille de style externe comme Bootstrap ou JQuery UI. C'est tout le CSS venant de l'extérieur.

Cependant, afin d'éviter des erreurs de compilation, il est préférable d'importer les Partials dans l'ordre suivant :

1. Utils :
  - a. Variables
  - b. Fonctions
  - c. Mixins
  - d. Extensions
2. Feuilles de style de tiers (Vendors)
3. Base
4. Composants
5. Layout
6. Pages
7. Thèmes

#### Auteur :

Yoann Depriester

#### Relu, validé & visé par :

☒ Jérôme CHRETIENNE  
☒ Sophie POULAKOS  
☒ Mathieu PARIS

#### Date création :

16-05-2023

#### Date révision :

16-05-2023



## Préprocesseur SASS

### Les Mixins

Les mixins permettent de stocker du code CSS, un peu comme le font les variables avec des valeurs.

On déclare une mixin de la façon suivant : **@mixin nom-de-la-mixin (\$argument : valeur-par-défaut-de-l'argument) { règlesCSS }**.

**\$argument** est une variable n'appartenant qu'à la mixin (elle n'est pas obligée d'être définie si elle n'apparaît pas dans le code CSS : **@mixin taille-texte {font-size : 1.5rem ;}**).

Exemple :

```
/* *****  
    MIXINS  
***** */  
@mixin cardArticle($theme: wheat){  
    background-color: $theme;  
    color: #333;  
    border: 3px solid #333;  
}
```

On appelle une mixin de la façon suivante : **@include nom-de-la-mixin ;**. L'\$argument prendra alors la valeur par défaut. Il est cependant possible de préciser une autre valeur, ou même d'appeler une variable (pour en exploiter la valeur). De plus, plusieurs arguments peuvent être définis au sein d'une mixin, chaque déclaration étant séparée par une virgule.

Exemple :

```
.card {  
    @include cardArticle;  
    @include cardArticle($theme: white);  
    @include cardArticle($theme: $primary-color);  
}
```

Le premier @include sera compilé de la manière suivante :

```
.card {  
    background-color: wheat;  
    color: #333;  
}
```

### AVANTAGES :

Les mixins évitent de répéter du code lors de son écriture, mais si elles sont appliquées à plusieurs sélecteurs, le code CSS compilé montrera des répétitions de code, chose qu'on souhaite éviter selon les bonnes pratiques du DRY (Don't Repeat Yourself).

#### Auteur :

Yoann Depriester

#### Date création :

16-05-2023

#### Relu, validé & visé par :

☒ Jérôme CHRETIENNE  
☒ Sophie POULAKOS  
☒ Mathieu PARIS

#### Date révision :

16-05-2023



Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.



## Préprocesseur SASS

### Les Extends

Les extensions de SASS sont similaires aux mixins, mais au lieu de dupliquer un ensemble de propriété CSS, elles dupliquent des sélecteurs.

On déclare une extension de la façon suivante : **%nomExtension {règlesCSS}**.

Exemple : Ici nous définissons une Extension à utiliser sur tous nos titres courants

```
/*  
*****  
EXTENDS  
*****  
*/  
%regularTitle {  
    font-size: $size-second-title;  
    font-family: $font-title;  
    color: orange;  
}
```

Pour les utiliser, on appelle l'Extension voulu dans chaque sélecteur visé grâce à la syntaxe suivante : **@extend %nomExtension**. Les sélecteurs visés seront alors regroupés lors de la compilation pour n'obtenir qu'un seul bloc de règle CSS.

Exemple :

```
.card__title{  
    @extend %regularTitle;  
}  
  
.primaryNav__link {  
    @extend %regularTitle;  
}
```

Ces appels de l'Extension **%regularTitle** sera compilé de la manière suivante :

```
.primaryNav__link, .card__title {  
    font-size: 2rem;  
    font-family: AbrilFatface;  
    color: orange;  
}
```

### AVANTAGE :

Tout comme les Mixins, les Extends contribuent fortement à l'écriture d'un code lisible qui suit le principe du Don't Repeat Yourself.

#### Auteur :

Yoann Depriester

#### Date création :

16-05-2023

#### Relu, validé & visé par :

☒ Jérôme CHRETIENNE  
☒ Sophie POULAKOS  
☒ Mathieu PARIS

#### Date révision :

16-05-2023



Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.