

# SASS

# Le CSS , pas si facile :

Le développement d'une application web passe nécessairement par l'écriture des feuilles de style.

Le CSS a beaucoup évolué depuis sa création et sa version actuelle (CSS 3) permet de réaliser des prouesses. Malgré tout, ce langage possède des défauts que tout développeur ne manquera pas de remarquer :

- l'absence d'indentation, les répétitions ou la ressemblance des différents éléments le rendent rapidement difficile à lire
- le niveau de spécificité des sélecteurs peut devenir un casse-tête
- difficile de prévoir tous les effets de bords, débbuger un .css est très coûteux en temps et en patience

# *Sass* c'est quoi ?

**Sass** (Syntactically awesome stylesheets) est un préprocesseur qui traduit le langage de script scss (**Sassy CSS**) en CSS.

Il agit en surcouche de CSS et y ajoute de nombreuses fonctionnalités permettant de faciliter et d'accélérer la création des feuilles de style pour application web.



## ENCAPSULATION

- Indentation
- Regroupement des éléments
- Spécificité gérée localement

## MODULES

- Fichiers plus lisibles
- Informations mieux ordonnées

## FUNCTIONS

- Paramètres en entrée
- Valeur en sortie
- Structures de contrôle
- Fonctions incluses

## AJOUTS

## HERITAGE

- Moins de répétitions
- Favorise la cohérence

## VARIABLES

- Syntaxe simplifiée
- Déclaration sur place

# Installation

## Méthode 1 :

Télécharger le package depuis GitHub puis l'ajouter au PATH de votre système d'exploitation.

github

site internet

## Méthode 2 :

Depuis une console de commande :

```
npm install -g sass
```

puis l'ajouter au PATH de votre système d'exploitation.

<https://github.com/sass/dart-sass/releases/tag/1.69.5>

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

# Utilisation

Depuis un terminal de commande :

```
C:\Users\ [ ] \Desktop\cours\react_course\frontend> sass --watch ./style/scss/index.scss:./style/css/index.css
```

Chemin actuel

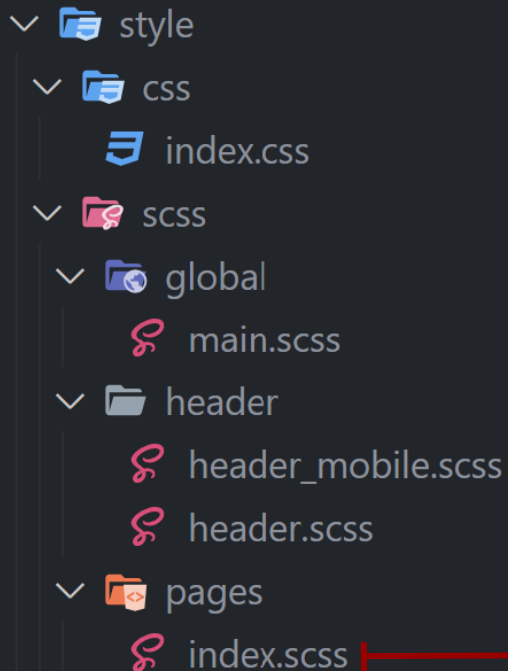
Commande Sass

Chemin du fichier  
à compiler

Chemin du fichier  
compilé

**flag watch:**  
sass recompile le code à  
chaque modification,  
annulé avec **ctrl + c**  
optionnel

# Modularité




On peut regrouper tous les fichiers `.scss` dans un même fichier global qui sera le seul à être compilé en `.css`



```
@import "../global/main.scss";  
@import "../header/header.scss";  
@import "../header/header_mobile.scss";
```

# Variables

```
$bg-black :  #333;  
$fs-section-title: 2rem;
```

Déclaration

```
.about--container {  
  background: $bg-black;  
}
```

```
h1,h2 {  
  font-size: $fs-section-title;  
}
```

Utilisation



# Encapsulation


```
.about--container {  
  background: ■ #e1e1e1;  
  
  p {  
    font-size: 1.2rem;  
    color: □ #333;  
  
    &:hover {  
      color: ■ crimson;  
    }  
  }  
}
```



```
.about--container {  
  background: ■ #e1e1e1;  
}  
  
.about--container p {  
  font-size: 1.2rem;  
  color: □ #333;  
}  
  
.about--container p:hover {  
  color: ■ crimson;  
}
```

# Héritage

```
%text--lg {  
  font-size: 2rem;  
  font-weight: bold;  
}  
  
%section--title {  
  @extend %text--lg;  
  color: #333;  
  text-align: center;  
}  
  
.about--title--wrapper {  
  @extend %section--title;  
  border: 2px outset rgba(0, 0, 0, 0.5);  
  padding: 0 12px;  
}
```



si une classe n'est jamais utilisée, elle ne sera pas compilée dans le fichier .css ciblé

cette classe hérite de %text--lg

cette classe utilise les règles attribuées à %section--title et %text--lg  
l'héritage est transmis sur plusieurs générations



# Fonctions

- \$width et \$color sont les paramètres en entrée de la fonction
- @include <nom\_de\_la\_fonction> pour récupérer sa valeur en sortie
- si on ne précise pas de paramètre d'entrée, alors on obtiendra la valeur de sortie avec les paramètres par défaut

```
@mixin outsetBorder($width: 2px, $color: black #000) {  
    border: $width outset $color;  
    border-radius: 4px;  
}  
  
.someBox {  
    @include outsetBorder;  
}  
  
.someOtherBox {  
    @include outsetBorder($width: 4px, $color: black #333)  
}
```


# Structures de contrôle

SCSS


```
.container {  
  @if($theme == "dark") {  
    background:  #333;  
  }  
  @else {  
    background:  #eeeeee;  
  }  
}
```



CSS

```
.container {  
  background:  #333;  
}
```



```
.container {  
  background:  #eeeeee;  
}
```

# Fonctions intégrées

Il existe de très nombreuses fonctions intégrées à Sass.

- manipulation algébriques
- manipulation des couleurs, de l'opacité, ...
- manipulation des chaînes de caractère
- itérations sur une liste d'éléments

```
@use 'sass:math';

$nombre: 10.4;

.title {
  font-size: math.floor($nombre)px; // => 10
}
```

# Pour conclure

**Sass** est accessible et simple d'utilisation pour les **débutants**.

Dès la première utilisation, **modularité**, **encapsulation** et l'utilisation des **variables** permettent de gagner en clarté et rapidité.

Les autres aspects du **préprocesseur** peuvent être appris au fur et à mesure sans être bloquants.

Cheat sheet

<https://github.com/code4mk/sass-cheatsheet/blob/master/pdf/sass-cheatsheet-latest.pdf>

<https://devhints.io/sass>