

Présentation de Sass

Syntactically Awesome Style Sheet

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



Sass c'est quoi ?



Sass est une **extension** de **CSS**

Sass est un **préprocesseur** **CSS**

Sass est entièrement compatible avec toutes les versions de **CSS**

Sass réduit la répétition des **CSS** et fait donc gagner du temps

Sass est gratuit à télécharger et à utiliser

Pourquoi utiliser Sass ?



Les feuilles de style CSS deviennent de plus en plus volumineuses, complexes et difficiles à gérer. C'est là qu'un **préprocesseur CSS** peut vous aider.

Sass vous permet d'utiliser des fonctionnalités qui n'existent pas dans **CSS** :

- (les variables)
- les structures de contrôle usuelles, les boucles
- les règles imbriquées
- les importations
- les mixins
- l'héritage
- les fonctions intégrées et d'autres éléments.

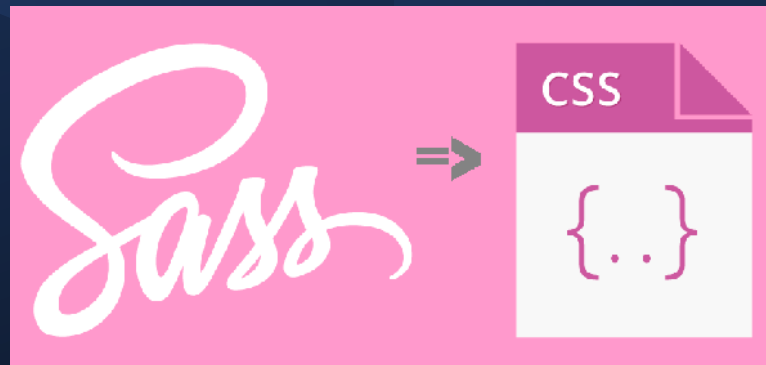
Comment fonctionne Sass ?



Un navigateur **ne comprend pas** le code **SASS**. Par conséquent, vous aurez besoin d'un préprocesseur Sass pour **convertir** le code **Sass** en **CSS** standard.

Ce processus est appelé **transpilation**.

Un **transpileur** convertit un fichier **SASS** (**.scss**) en un fichier **CSS** (**.css**)



Installer Sass sous Windows



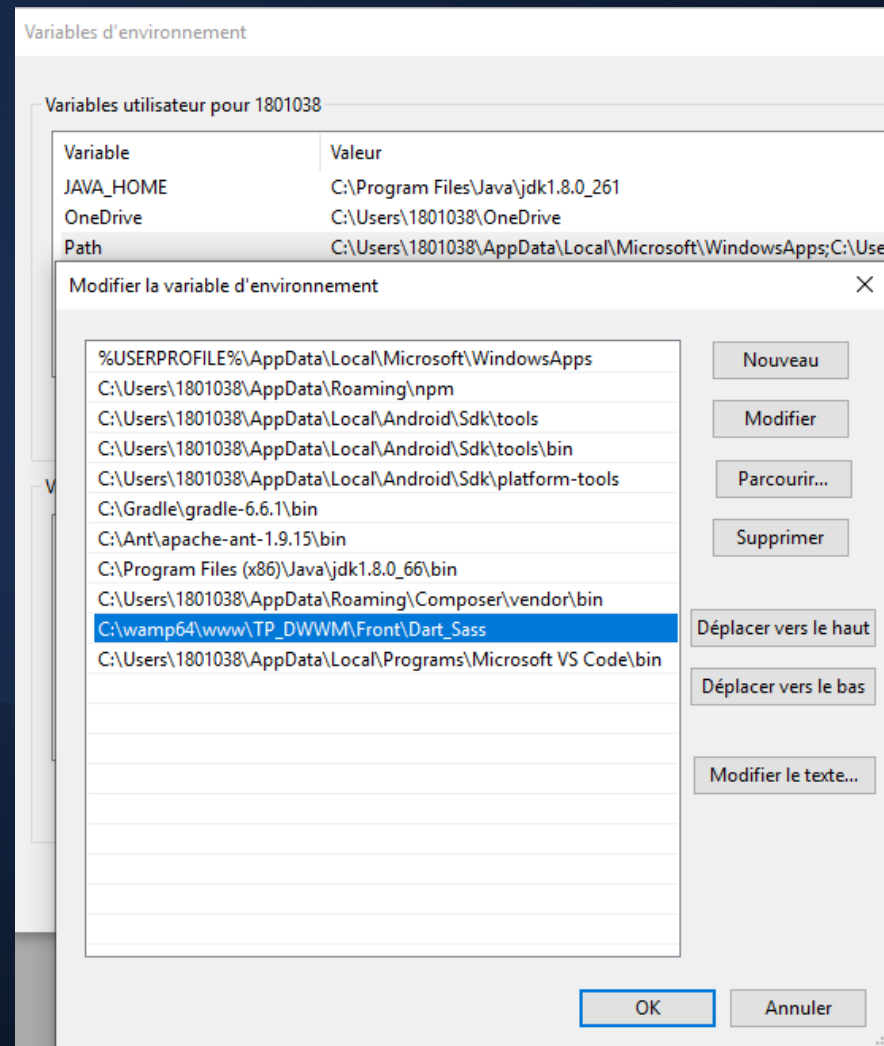
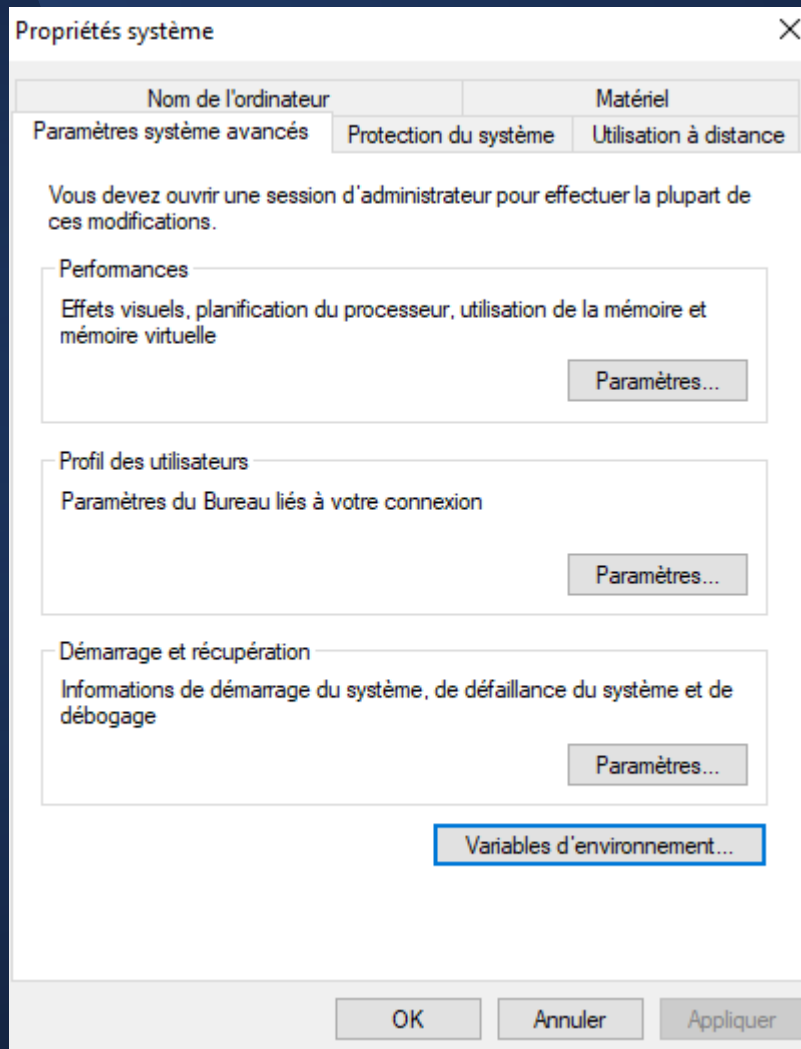
Rendez-vous à l'URL suivante pour récupérer la **version autonome** de **Sass** :

<https://github.com/sass/dart-sass/releases/tag/1.52.3>

Lorsque vous avez récupéré l'archive ([dart-sass-1.52.3-windows-x64.zip](#)), vous pouvez copier le dossier **dart-sass** sur votre ordinateur, il contient un dossier src et un fichier BATCH sass.bat.

Afin de pouvoir bénéficier de la transpilation sur votre système Windows à partir de la ligne de commande, vous allez devoir ajouter **dart-sass** au **PATH** de votre système ...

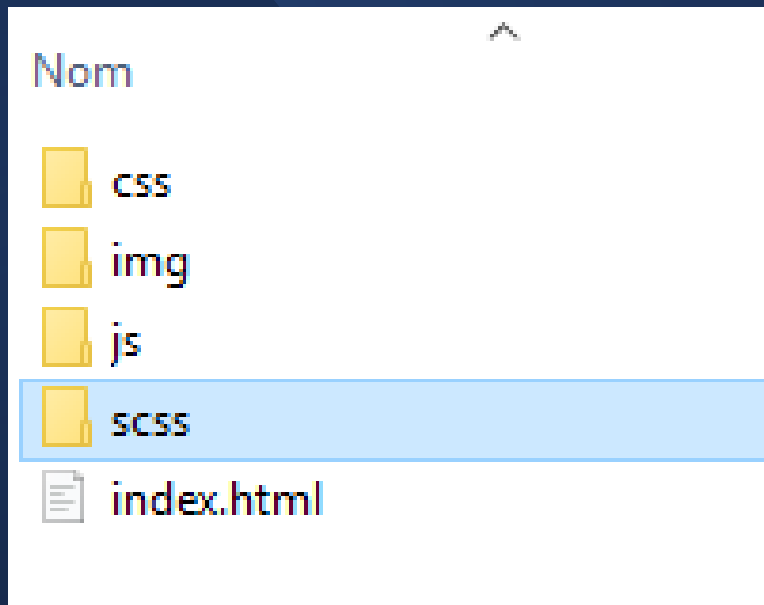
Installer Sass sous Windows ...



Workflow avec Sass



Désormais, vous pouvez créer un dossier **SASS** qui contiendra vos divers projets Web. Vous pouvez adopter la **structure classique pour votre Projet1** :



Vous pourrez créer un dossier **scss** qui contiendra l'ensemble de vos fichiers **Sass** (.scss)

Le dossier **css** contiendra vos fichiers **CSS** (.css) après **transpilation** des fichiers **Sass**.

Workflow avec Sass ...



Depuis le Terminal de votre **IDE VSC** vous pouvez désormais saisir la ligne de commande suivante :

```
SORTIE  TERMINAL  JUPYTER  CONSOLE DE DÉBOGAGE  PROBLÈMES
PS C:\wamp64\www\TP_DWWM\Front\SASS\Projet1> sass scss/style.scss css/style.css --watch
```

Vous devez pointer sur le répertoire de votre **Projet1** et l'instruction **sass** sera reconnue grâce à votre travail sur le **PATH**.

La suite de l'instruction s'interprète ainsi : « Surveillez le fichier **style.scss** du dossier **scss** et après **transpilation** le fichier **style.css** sera créé dans le dossier **css**. »

L'apport de **--watch** à l'instruction permet de générer le fichier **style.css** dès qu'une **sauvegarde** sera demandée sur le fichier **style.scss**.

Remarque : Vous pouvez aussi surveiller le dossier **scss** au profit du dossier **css**
En utilisant l'instruction : **sass scss : css --watch**

Les Variables avec Sass



Les **variables** sont un moyen de stocker des **valeurs** que vous pouvez réutiliser ultérieurement.

Avec **Sass**, vous pouvez stocker des **informations** dans des **variables**, telles que :

- Chaîne de caractères
- Nombre
- Couleur
- Booléen
- Liste
- Nul

Sass utilise le symbole **\$**, suivi d'un nom, pour déclarer les **variables** :

```
$nomvariable: value;
```

Les Variables avec Sass ...



```
style.scss x
Front > SASS > Projet1 > scss > style.scss > ...
1  $font-stack: Helvetica, sans-serif;
2  $primary-color: #333;
3
4
5  body {
6
7      font: 100% $font-stack;
8      background-color: $primary-color;
9      font-family: sans-serif;
10 }
```

```
style.scss # style.css x
Front > SASS > Projet1 > css > # style.css > ...
1  body {
2      font: 100% Helvetica, sans-serif;
3      background-color: #333;
4      font-family: sans-serif;
5  }
```

La **portée d'une variable** n'est disponible qu'au niveau de l'imbrication ou elle est définie. Si on veut étendre sa portée depuis une imbrication il faut utiliser le **commutateur !global**.

```
h1 {
    $myColor: green !global;
}
```

Les règles imbriquées avec Sass



Sass vous permet d'**imbriquer** des **sélecteurs** CSS de la même manière que ce que vous faites déjà en HTML.

```
$font-color: ■ rgb(223, 215, 215);
```

```
.table {  
  width: 100%;  
  
  td {  
    border: 2px solid ■ red;  
    color: $font-color;  
  }  
}
```

```
.table {  
  width: 100%;  
}  
.table td {  
  border: 2px solid ■ red;  
  color: ■ rgb(223, 215, 215);  
}
```

Conseil : Il n'est pas conseillé de dépasser plus de **2 niveaux d'imbrication** !!!

Les règles imbriquées avec Sass ...



Avec Sass, vous pouvez utiliser le symbole **&** afin de **référencer** le **sélecteur parent** et éviter ainsi la répétition de ce sélecteur.

```
.bouton{
  text-decoration: none;
  border-radius: 3px;
  padding: 10px 18px;
  background-color: ■rgb(34, 199, 28);
  color: ■#fff;

  &.rechercher{
    background-color: ■crimson;

    &:hover{
      background-color: ■rgb(128, 12, 35);
      color: ■dodgerblue;
    }
  }
}
```

```
.bouton {
  text-decoration: none;
  border-radius: 3px;
  padding: 10px 18px;
  background-color: ■rgb(34, 199, 28);
  color: ■#fff;
}

.bouton.rechercher {
  background-color: ■crimson;
}

.bouton.rechercher:hover {
  background-color: ■rgb(128, 12, 35);
  color: ■dodgerblue;
}
```

```
<a href="#" class="bouton">Envoyer</a>
<a href="#" class="bouton rechercher">Rechercher</a>
```

Envoyer

Rechercher

Envoyer

Rechercher

Importation de fichiers Sass



Une façon d'écrire du code **DRY** (**D**on't **R**epeat **Y**ourself) consiste à conserver le code associé dans des **fichiers séparés**.

Tout comme **CSS**, **Sass** prend également en charge la directive **@import**. Cette directive permet **d'inclure** un fichier dans un autre.

La directive **@import** permet **d'inclure le fichier dans le CSS résultant**, ainsi aucun appel **HTTP** supplémentaire n'est requis lors de l'exécution. (à la différence de ce qui se passe avec cette même directive dans le cas de fichiers CSS : **@import url('monfichier.css')**)

Vous pouvez importer autant de fichiers que nécessaire dans le fichier principal résultant, la bonne pratique c'est de définir dans des fichiers **scss** distincts les **couleurs**, les **fonts**, ... Comme vous ne voulez pas que ces fichiers **scss** soient **transpilés**, vous devez utiliser la notation partielle « **Scss partielle** » :

_nomfichier.scss et ensuite faire référence à ce fichier dans votre **scss** résultant grâce à la directive **@import** :

```
@import 'monfichier';
```

Importation de fichiers Sass ...



Exemple d'application :

style.scss X _typo.scss _couleurs.scss

Front > SASS > Projet1 > scss > style.scss > ...

```
1 @import "typo";
2 @import "couleurs";
3
4 h1{
5   font-size: 3em;
6   font-family: "myJournalFont";
7   color: $font-color;
8   letter-spacing: 2px;
9 }
10
```

style.scss _typo.scss X _couleurs.scss # style.css

Front > SASS > Projet1 > scss > _typo.scss > ...

```
1 @font-face {
2   font-family: "myJournalFont";
3   src: url("Fascinate-Regular.ttf") format('truetype');
4 }
```

style.scss _typo.scss _couleurs.scss X

Front > SASS > Projet1 > scss > _couleurs.scss > ...

```
1 $font-stack: Helvetica, sans-serif;
2 $primary-color: #333;
3 $font-color: rgb(223, 104, 217);
4
```

Le Journal

Les Mixins avec Sass



La directive **@mixin** permet de créer du **code CSS réutilisable** sur l'ensemble du site Web.

La directive **@include** permet de « *consommer* » ce **mixin**.

1 - Les Mixins sans argument

```
@mixin important-text {  
  color: ■ rgb(243, 237, 237);  
  font-size: 25px;  
  border: 1px solid ■ rgb(142, 142, 151);  
}  
  
.danger {  
  @include important-text;  
  background-color: ■ rgb(236, 102, 24);  
  text-decoration: none;  
  
  &:hover{  
    background-color: ■ rgb(236, 11, 11);  
  }  
}
```

```
<a href="#" class="danger" >Survolez-moi</a>
```



Survolez-moi



Survolez-moi

Les Mixins avec Sass ...



2 - Les Mixins avec arguments

Vous pouvez utiliser des **valeurs par défaut** pour vos **arguments**.

Vous pouvez constituer votre propre **bibliothèque** de **mixins** et vous pourrez les utiliser dans votre projet Web en utilisant la directive **@import** déjà rencontrée.

@import 'mixins'

```
@mixin title_style($color, $background: #eee, $fontsize){  
  
  margin: 0 0 20px 0;  
  font-family: "myJournalFont";  
  font-size: $fontsize;  
  font-weight: bold;  
  text-transform: uppercase;  
  color: $color;  
  background: $background;  
}  
  
h2{  
  @include title_style(rgb(34, 30, 29), "", 25px)  
}  
  
h3{  
  @include title_style(rgb(84, 88, 84), "", 15px)  
}
```

JE SUIS UN TITRE ASSEZ IMPORTANT !!!

JE SUIS UN TITRE DE MOINDRE IMPORTANCE !!!

L'Héritage avec Sass



La directive Sass **@extend** permet de réaliser **l'héritage** au sein de vos sélecteurs. En effet, elle vous permet de **partager** un certain nombre de propriétés **d'un sélecteur à un autre** :

```
.button-basic {  
  border: none;  
  padding: 15px 30px;  
  text-align: center;  
  text-decoration: none;  
  font-size: 16px;  
  cursor: pointer;  
}  
  
.button-report {  
  @extend .button-basic;  
  background-color: red;  
  color: white;  
}  
  
.button-submit {  
  @extend .button-basic;  
  background-color: green;  
  color: white;  
}
```

```
<a href="#" class="button-report">Annuler</a>  
<a href="#" class="button-submit">Envoyer</a>
```



Annuler

Envoyer

Après chaque **transpilation**, contrôlez bien le **CSS** généré !!!

L'Héritage avec Sass ...



Si la **classe** que vous **étendez** n'existe que pour être **héritée** par d'autres sélecteurs, vous pouvez utiliser les **placeholders** avec la directive **@extend**. Il suffit de **préfixer** le nom de la classe que vous voulez **étendre** par le symbole **%**

```
%button-basic{  
  border: none;  
  padding: 15px 30px;  
  text-align: center;  
  text-decoration: none;  
  font-size: 16px;  
  cursor: pointer;  
}
```

```
.button-report {  
  @extend %button-basic;  
  background-color: red;  
  color: white;  
}
```

```
.button-submit {  
  @extend %button-basic;  
  background-color: green;  
  color: white;  
}
```

```
<a href="#" class="button-report">Annuler</a>  
<a href="#" class="button-submit">Envoyer</a>
```




Après cette **transpilation**, contrôlez bien le **CSS** généré et ce en quoi il diffère du code **CSS** précédent !!!

Les Fonctions avec Sass



Sass propose tout un set de **fonctions** utilisables qui vous permettront de gérer les **chaînes de caractères**, les **numériques**, les **couleurs**, les **listes** ... Vous pouvez aussi créer **vos propres fonctions** et les utiliser.

Je vous propose de découvrir ici les fonctions **lighten()** et **darken()** qui sont très pratiques :

```
$link : rgb(88, 211, 31);
a{
  color : $link;
  font-weight: bold;
  font-size: 30px;
  &:hover,
  &:focus {
    color : lighten($color: $link, $amount: 30%);
  }
  &:active{
    color : darken($color: $link, $amount : 8%);
  }
}
```

Je vous laisse découvrir le rendu par vous-même.

Expérimentez toutes ces nouvelles fonctionnalités offertes par **Sass** !!!

Après chaque **transpilation**, **contrôlez systématiquement** le **CSS** obtenu !!!

Allez plus loin avec Sass ...



Pour aller plus loin avec les **Fonctions** proposées par **Sass** suivez le lien :

<https://sass-lang.com/documentation/at-rules/function>

Il existe plusieurs **Frameworks Sass** mais le plus célèbre d'entre eux reste **COMPASS**, suivez le lien :

<http://compass-style.org/>



CRÉDITS
OEUVRE COLLECTIVE DE L'AFPA
Sous le pilotage de la DIIP
et du centre sectoriel Tertiaire

EQUIPE DE CONCEPTION
M. Restoueix Sacha (Formateur)

Date de mise à jour : 26/06/2022
Date de dépôt légal : 2022

© AFPA 2022

Reproduction interdite

Article L 122-4 du code de la propriété intellectuelle.

« Toute représentation ou reproduction intégrale ou partielle faite sans le consentement de l'auteur ou de ses ayants droits ou ayants cause est illicite. Il en est de même pour la traduction, l'adaptation ou la reproduction par un art ou un procédé quelconques ».