



Titres professionnels Développeur Web et Web Mobile & Concepteur Développeur d’Applications



Le SASS

Formateur : Hugo VIROT

SOMMAIRE

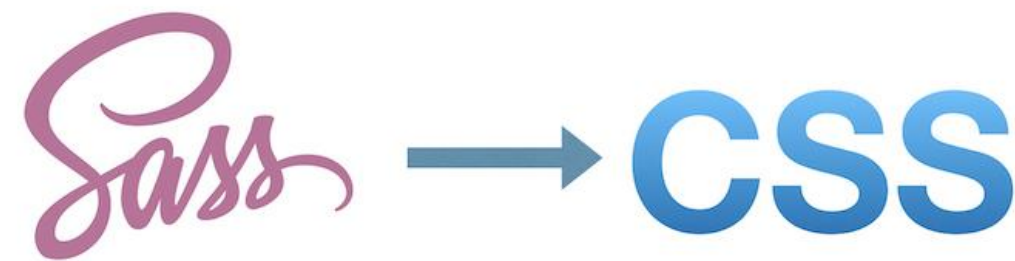
- 1. Présentation 2
- 2. Syntaxe 3
- 3. Mise en place 4
- 4. Les concepts principaux du SASS 6
- 5. Infos diverses 11



1. Présentation

SASS est un autre type de CSS, beaucoup plus flexible et dynamique. Il va révolutionner votre façon de travailler en CSS !

- **SASS** signifie **Syntactically Awesome Style Sheets** (“Feuilles de style syntaxiquement fantastiques”).
- Apparu en 2006, c’est un langage de script préprocesseur, qui est transformé en CSS « classique » par un compilateur.
- On obtient alors un **fichier CSS lisible par un navigateur** (qui ne comprend pas le SASS). **Seul ce fichier sera transféré sur le serveur lors de la mise en production** (et pas les fichiers SASS).



.sass

```
$mainColor: #94b0da;

.title {
  color: $mainColor;
}

.card {
  background-color: $mainColor;
}
```

.CSS

```
.title {
  color: #94b0da;
}

.card {
  background-color: #94b0da;
}
```

- **SASS** permet de faire énormément de choses qu’on ne peut pas faire en CSS, et qui s’apparentent habituellement aux langages de programmation :
 - ✓ Utiliser des **variables** et des **fonctions** (il est possible de faire des variables en CSS, mais de façon moins intuitive)
 - ✓ Réutiliser des morceaux de code (appelés “**mixins**”)
 - ✓ Structurer le code
 - ✓ Etc.



2. Syntaxe

- **SASS** utilise une syntaxe indentée particulière, peu intuitive. Elle se présente sans points virgules et sans accolades, ce qui peut déstabiliser.
- Pour simplifier l'utilisation de SASS, une nouvelle syntaxe a été créée : **SCSS** (Sassy Cascading Style Sheets : feuilles de style en cascade "à la sauce SASS"). C'est celle-ci que nous utiliserons.

| SASS | SCSS | CSS |
|--|---|--|
| <pre> \$color: red \$color2: lime a color: \$color &:hover color: \$color2 </pre> | <pre> \$color: #f00; \$color2: #0f0; a { color: \$color; &:hover { color: \$color2; } } </pre> | <pre> a { color: red; } a:hover { color: lime; } </pre> |

Exemple avec les 3 syntaxes

- Sur cet exemple, on constate que la syntaxe centrale **SCSS** est plus proche du CSS classique, et donc plus intuitive.
- C'est le **SCSS** que nous utiliserons. Nous créerons donc uniquement des fichiers en **.scss** (jamais en **.sass**).
- Les fichiers **SASS** utilisent l'extension **.sass**, les fichiers **SCSS** sont en **.scss**.

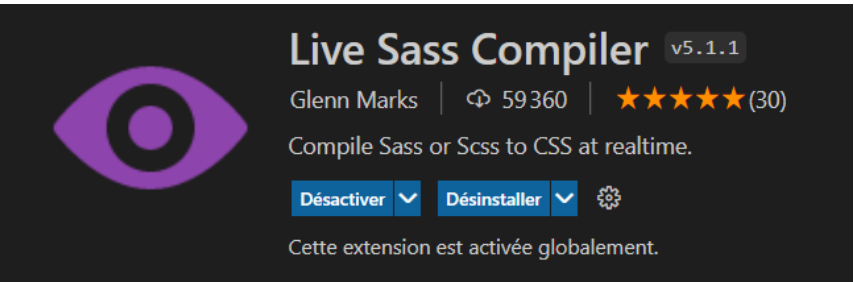


3. Mise en place

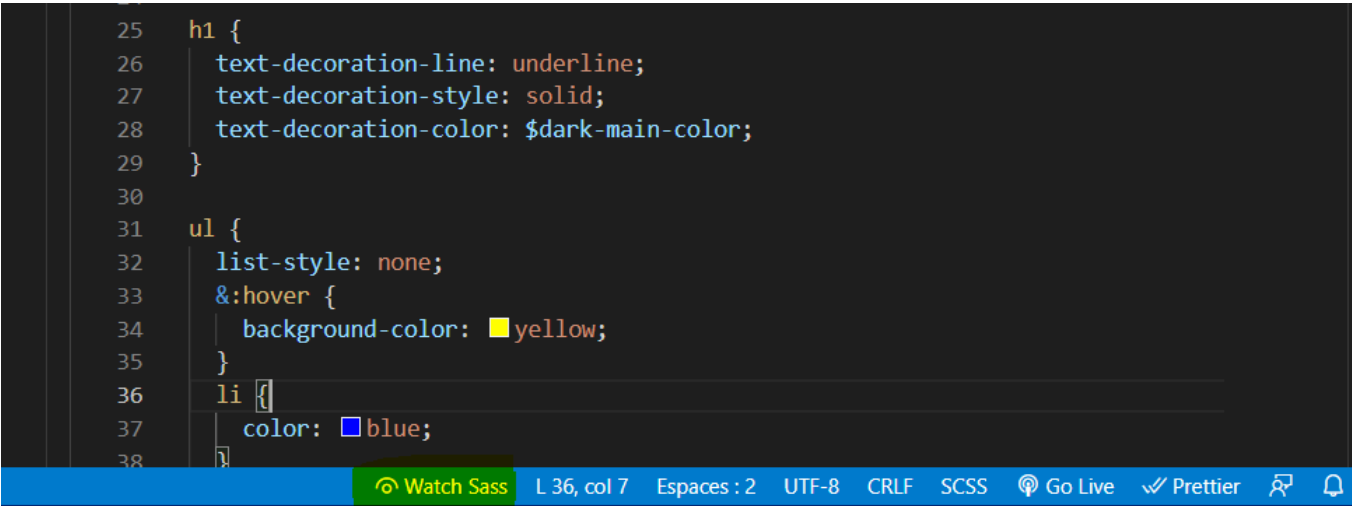
Projet simple HTML/CSS

- Créez les fichiers suivants :
 - **index.html**
 - Votre feuille CSS classique **style.css** (ajoutez le lien dans les balises head comme d’habitude)
 - Un **style.scss** (fichier principal de votre SASS)
- Pour utiliser **SASS**, vous aurez donc besoin d’un **compilateur**, qui va **transformer vos fichiers SCSS en une seule feuille de style CSS**.
- Pour cela, il vous faut **installer le plug-in Live Sass Compiler** dans VS Code.

```
<> index.html
# style.css
# style.css.map
🔗 style.scss
```



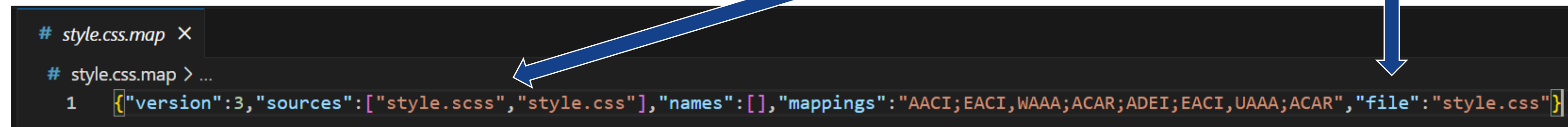
- Une fois installé, **cliquez sur “Watch SASS”** en bas de votre écran.





```
Generated:
c:\Users\hugov\OneDrive\Documents\Arinfo\Cours CDA\SASS\test sass\style.css.map
c:\Users\hugov\OneDrive\Documents\Arinfo\Cours CDA\SASS\test sass\style.css
-----
Watching...
```

- Live Sass Compiler vient de créer automatiquement le fichier **style.css.map**, qui définit les **fichiers d'entrée** (appelés « **sources** ») et celui de **sortie** (appelé « **file** »).



```
# style.css.map ✕
# style.css.map > ...
1 [{"version":3,"sources":["style.scss","style.css"],"names":[],"mappings":"AACI;EACI,WAAA;ACAR;ADEI;EACI,UAAA;ACAR","file":"style.css"}]
```

- **Vous pouvez maintenant écrire du SASS. Il sera compilé automatiquement.** Pensez bien à relancer le plugin à chaque nouvelle session de travail dans VS Code.

Projet Laravel

- Suivez la procédure expliquée sur ce [site](#)
- SASS est inclus automatiquement si vous avez installé le package d'authentification Laravel UI.

Projet Vue JS

- Suivez la procédure expliquée sur ce [site](#)

4. Les concepts principaux du SASS

Les variables

- Le principe est le même que dans vos langages de programmation favoris.
- En SASS, elles permettent par exemple de **changer facilement les couleurs de l'ensemble du site**, en créant 2 **variables** pour stocker les 2 couleurs principales (comme ci-dessous).
- Pour **déclarer une variable**, on procède ainsi : **\$ + nom variable : valeur**.

```
$mainColor: #005183;
```

- Ensuite, **on associe la variable** (ici, la couleur principale) **à l'élément** (ou aux éléments) souhaité(s).
Simple et efficace !

```
p {
  color: $mainColor;
}
```



Lorem ipsum dolor sit amet consectetur adipisicing elit. Officiis temporibus

```
$mainColor: #005183;
$secondColor: #e6d40f;

h1 {
  color: $mainColor
}

h2 {
  color: $secondColor
}
```



Test h1

Lorem ipsum dolor sit amet, consectetur itaque asperiores

test h2

Lorem ipsum dolor sit amet, consectetur itaque asperiores

Test h2

Lorem ipsum dolor sit amet, consectetur itaque asperiores

- Cela évite par exemple de faire un « rechercher et remplacer » pour changer votre couleur principale, utilisée de nombreuses fois dans votre feuille CSS (ou pire, de remplacer une à une toutes les occurrences à la main...). Au lieu de cela, vous modifiez uniquement la valeur de la variable !



➤ Si vous avez de nombreuses variables, pour plus de lisibilité, vous pouvez **les placer dans un fichier spécial** :

✓ créez le fichier **_variables.scss***

```
_variables.scss
index.html
style.css
style.css.map
style.scss
```

✓ à l'intérieur, **créez toutes les variables nécessaires**

```
_variables.scss X index.html
_variables.scss > ...
1 $mainColor: #005183;
2 $secondColor: #e6d40f;
```

✓ **importez le fichier dans style.scss** comme ceci :

```
style.scss X _variables.scss index.html
style.scss > ...
12 @import url('./_variables.scss');
13
14 h1 {
15     color: $mainColor
16 }
17
18 h2 {
19     color: $secondColor
20 }
21
```

Ensuite, il n'y a plus qu'à utiliser vos variables.

Vous obtenez le même résultat que précédemment, avec un code mieux organisé.

* **l'underscore est très important car il permet d'indiquer à SASS qu'un fichier ne doit pas être compilé.** Ce fichier a uniquement vocation à être **importé dans un autre**.
On appelle ce type de fichiers des **feuilles partielles (partials en anglais)**.

L'imbrication

- Elle permet **d'organiser votre style** et ainsi de le rendre **beaucoup plus lisible**.
- Principe : **disposer vos sélecteurs SCSS par rapport à leur position hiérarchique dans le HTML**.
- On obtient un **résultat étagé, beaucoup plus clair**.
- Exemple : **imbrication** appliquée sur le style d'une navbar

```
16 <body>
17
18   <header>
19     <nav>
20       <div id="logo">
21         
22       </div>
23       <ul>
24         <li class="rose">home</li>
25         <li>portfolio</li>
26         <li>about</li>
27         <li>blog</li>
28         <li>contact</li>
29       </ul>
30     </nav>
31   </header>
```

Code HTML de la navbar

```
style.scss 1 X
style.scss > ...
1 @import "nav";
2
```

Import style navbar dans fichier de style principal

```
_nav.scss X
_nav.scss > header
1 header {
2   background-color: black;
3
4   nav {
5     display: flex;
6     align-items: center;
7     width: 80%;
8     margin: auto;
9
10    #logo {
11      width: 20%;
12    }
13
14    ul {
15      display: flex;
16      justify-content: space-around;
17      width: 80%;
18      list-style: none;
19
20      li {
21        color: white;
22      }
23    }
24  }
25 }
```

Code SCSS de la navbar (très lisible)

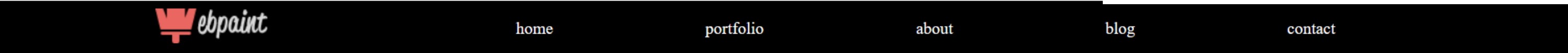


```
_nav.scss # style.css X
# style.css > ...
1 header {
2   background-color: black;
3 }
4 header nav {
5   display: flex;
6   align-items: center;
7   width: 80%;
8   margin: auto;
9 }
10 header nav #logo {
11   width: 20%;
12 }
13 header nav ul {
14   display: flex;
15   justify-content: space-around;
16   width: 80%;
17   list-style: none;
18 }
19 header nav ul li {
20   color: white;
21 }
```

CSS généré (proche de celui que vous auriez écrit sans encapsulation !)

```
> images
_nav.scss
index.html
style.css
style.css.map
style.scss 1
```

Fichiers



résultat

Les mixins

- Un **mixin** est un ensemble de styles à appliquer à un élément. Exemple : les différentes règles CSS que vous appliquez à un bouton.
- Plutôt que de répéter ces règles à chaque bouton, on va les grouper dans un **mixin** pour pouvoir les réutiliser facilement.
- Vous pouvez faire soit un **fichier regroupant tous vos mixins** (appelé par exemple `_mixins.scss`), soit un **fichier par mixin** (conseillé s'ils sont nombreux, car plus clair).

- Pour mettre en place un **mixin** (par exemple celui contenant les règles de notre bouton) :

- ✓ on crée d'abord le **fichier** qui va le contenir : ici, appelons-le `_button.scss`
- ✓ on y **déclare** notre **mixin** contenant les **règles** de notre bouton
- ✓ on **importe** le fichier dans `style.scss`
- ✓ on **utilise** le **mixin** avec la syntaxe `@include`

Résultat :

Lorem ipsum

```
style.scss X
1 @import 'button';
2
3
4 button {
5     @include button
6 }
```

```
_button.scss X
1 @mixin button {
2     font: {
3         family: verdana, courier;
4         size: 1.25rem;
5         weight: bold
6     }
7     text: {
8         decoration: none;
9         align: center;
10    }
11    background-color: darkblue;
12    color: white;
13    padding: 1rem;
14    width: 12.5rem;
15    border: none;
16    border-radius: 15px;
17    display: block;
18    margin: 0.5rem auto 0;
19 }
```

- Autre exemple de **mixin** : un paragraphe

```

_paragraph.scss x
_paragraph.scss > paragraph
1  @mixin paragraph {
2      font: {
3          family: 'Comic sans MS', 'courier';
4          size: 1em;
5          weight: 'light';
6          style: italic
7      }
8      line-height: 1.5em;
9      color: #005bce;
10     margin: auto 20vw
11 }

```

```

style.scss x
style.scss > ...
1  @import 'paragraph';
2
3  p {
4      @include paragraph
5  }

```

Lorem ipsum dolor sit amet, consectetur adipisicing elit. Ducimus aspernatur impedit eos. Harum quam obcaecati commodi vitae, provident rem dolor illo modi, cupiditate, iusto est accusantium consequuntur itaque asperiores delectus.

Lorem ipsum dolor sit amet, consectetur adipisicing elit. Ducimus aspernatur impedit eos. Harum quam obcaecati commodi vitae, provident rem dolor illo modi, cupiditate, iusto est accusantium consequuntur itaque asperiores delectus.

- Pour une seule utilisation d'un groupe de règles CSS dans votre projet, il n'y a pas forcément besoin de créer un **mixin**. **A partir de deux, cela devient intéressant.**
- La création d'un **mixin** permet aussi de **le réutiliser plus tard** dans un autre projet. Il suffit de copier-coller le **mixin** dans le nouveau projet, puis de l'importer et de l'utiliser dans styles.scss.

5. Infos diverses

- **SASS propose d'autres mécanismes intéressants**, notamment les fonctions, l'héritage... Je vous invite à les découvrir si vous le souhaitez, en consultant la **documentation officielle** : <https://sass-lang.com/>
- **Les media queries sont compatibles avec l'imbrication**. Il est conseillé de les utiliser comme ceci, en réécrivant **@media** dans le sélecteur de chaque élément concerné, et en précisant les règles qui vont s'appliquer sur l'élément.

Fichier style.scss

```
style.scss
1  @import 'variables';
2
3  body {
4    #section1 {
5      h1 {
6        color : $mainColor;
7        font-size: 50px;
8
9        @media screen and (max-width: 768px) {
10          color: $secondColor;
11          font-size: 100px;
12          text-align: center;
13        }
14      }
15    }
16  }
```



Aperçu avec 769 px de large (règles de base)



Aperçu avec 768 px de large (les règles de la media query s'appliquent)

- Autrefois utilisé dans des versions (maintenant obsolètes) écrites en Ruby (Ruby Sass, implémentation originale), puis en C++ (LibSass), **SASS est maintenant en Dart**, ce qui lui assure des performances bien supérieures lors de la compilation.
- Vous pouvez aussi **installer SASS directement** (sans passer par le plug-in), ce qui permet de l'utiliser ensuite en ligne de commande. On peut par exemple passer par NPM :

```
npm install -g sass
```

, mais il y a d'autres possibilités. Pour plus d'informations, consultez : <https://sass-lang.com/install/> et <https://sass-lang.com/guide/>