Simplon Boulogne sur Mer

Part - 1

Sass, c'est quoi ?

Sass (**S**yntactically **A**wesome **S**tyle**s**heets) est un **langage de génération de feuilles de style CSS** créé par Hampton Catlin et Nathalie Weizenbaum. Il ajoute de nouvelles règles comme les variables, les mixins, l'héritage et plus encore.

Sass a un Framework appelé Compass qui permet de gagner du temps, et gère automatiquement quand c'est nécessaire, les problématiques cross browser.

Pourquoi utiliser Sass ?

- Pour mieux organiser les feuilles de styles.
- Pour échapper a la duplication de le code.
- Pour mettre en place une optique DRY
- Etc...

Installation

Step 1 : On installe **Ruby**

Step 2 : On rentre la ligne de commande pour installer Sass

```
$ gem install sass
```

Step 3 : Vérifier si Sass est bien installé en utilisant

```
sass -v
```

Comment utiliser Sass

Sass possède deux syntaxe : **SASS et SCSS.**

SASS

Utilise l'indentation des lignes pour spécifier des blocs au lieu des accolades.
C’est la syntaxe la plus ancienne.

SCSS

Cette syntaxe a un formalisme proche du CSS3 et est de plus en plus adoptée par les développeurs web.

SASS

```
$color: red
$color2: lime

a
    color: $color
    span
        color: $color2
```

SCSS

```
$color: #f00;
$color2: #0f0;

a {
    color: $color;
    span {
        color: $color2;
    }
}
```

CSS

```
a {
    color: red;
}

a span {
    color: lime;
}
```

Compiler votre fichier Sass à Css

Quand vous faite une sauvegarde dans votre fichier Sass ou Scss, exécutez la commande ci-dessous pour **compiler votre fichier en css :**

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

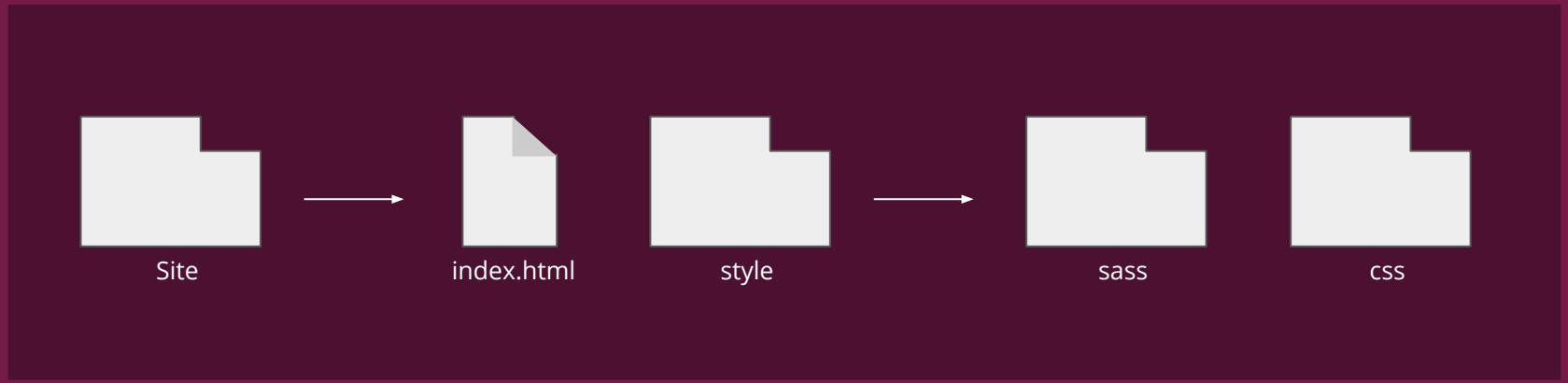
Compiler votre fichier Sass à Css

Afin d'éviter de toujours devoir rentrer la ligne de commande à chaque sauvegarde du fichier style, nous pouvons demander à Sass de "surveiller" notre fichier sass et de modifier automatiquement le style.css à chaque sauvegarde.

```
sass --watch style.scss:style.css
```

Architecture

Pour garder un dossier propre, il est conseillé de séparer les fichiers SASS et CSS dans différents dossiers.

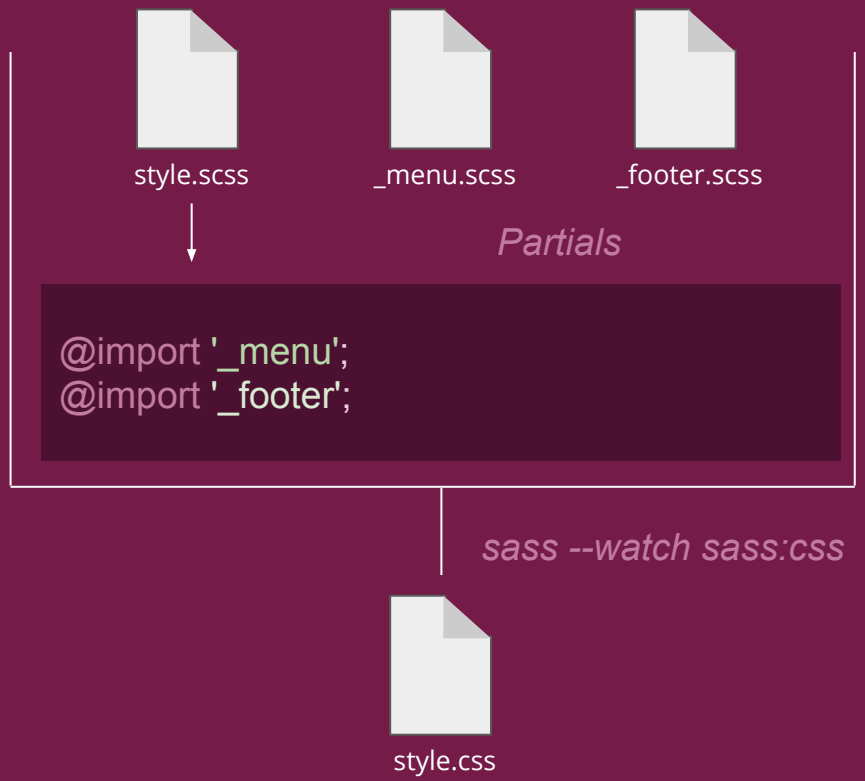


Architecture - Tips

Vous pouvez également “surveiller” votre fichier SASS pour ne pas avoir à rentrer la même ligne de commande à chaque sauvegarde !

```
sass --watch sass:css
```

Importations



Si vous le souhaitez, vous pouvez créer plusieurs fichiers sass et les compiler en un seul fichier css. Pour cela :

Step 1 : Créer vos différents fichiers sass

Step 2 : Dans votre style.scss, rajouter la ligne ‘@import’ suivit du nom de votre fichier entre guillemets

Step 3 : Compilez et c’est bon !

Commentaires

En Sass, les commentaires uni-lignes ne sont pas retranscrit en CSS. Mais les commentaires multi-lignes fonctionnent parfaitement.

style.scss

```
// Un commentaire uni-ligne

/* Un commentaire
multi-ligne
!!!
*/
```


style.css

```
/* Un commentaire
multi-ligne
!!!
*/
```

Variables

style.scss

```
$content: “Cacahuete”;
$case: uppercase;

h1 {
        text-transform: $case;
}

p {
        content: $content;
}
```

style.css

```
h1 {  text-transform: uppercase; }
p {  content: “Cacahuete”; }
```

Variables

style.scss

```scss
$turquoise: #1abc9c;
$pumpkin: #d35400;

h1 {
  color: $pumpkin;
}

p {
  span {
        color: $turquoise;
  }
}
```

style.css

```css
h1 { color: #d35400; }
p span { color: #1abc9c; }
```

Imbrications

style.scss

```scss
$color: #f00;
$color2: #0f0;

a {
    color: $color;
    span {
        color: $color2;
        &:hover{
            color: $color;
        }
    }
}
```

style.css

```css
a {
    color: #f00;
}

a span {
    color: #0f0;
}

a span:hover {
    color: #f00;
}
```

Imbrications

style.scss

```scss
h1{
        font:{
        family: arial;
        size: 18px;
        weight: bold;
        }
}
```

style.css

```css
h1 {
  font-family: arial;
  font-size: 18px;
  font-weight: bold;
}
```

Héritages

style.scss

```scss
.message {
  border: 1px solid #ccc;
  padding: 10px;
  color: #333;
}

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

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

.warning {
  @extend .message;
  border-color: yellow;
}
```

style.css

```css
.message, .success, .error, .warning {
  border: 1px solid #ccc;
  padding: 10px;
  color: #333;
}

.success {
  border-color: green;
}

.error {
  border-color: red;
}

.warning {
  border-color: yellow;
}
```

Opérations

style.scss

```scss
$largeur: 200px;

.rectangleA {
        width: $largeur + 200px;
}

.rectangleB {
        width: $largeur + 800;
}

.rectableC {
        width: $largeur - 5px;
}

.rectangleD {
        width: $largeur * 4;
}

.rectangleE {
        width: $largeur / 2;
}
```

style.css

```css
.rectangleA {
        width: 400px;
}

.rectangleB {
        width: 1000px;
}

.rectableC {
        width: 195px;
}

.rectangleD {
        width: 800px;
}

.rectangleE {
        width: 100px;
}
```

Concaténation

Comme en JS, vous pouvez concaténer en Sass

```
p:before{
        content: "ap" + "p";                    // “app”
}

a:before{
        content: "ap" + p;                      // “app”
}

p{
        font-family: sans- + "serif";           // sans-serif
}
```

Division - Tips

Essayez ce bout de code en Sass ... :)

```
span{
        color: red;
        margin-left: 10px / 10px;
}
```

Division - Tips

Pour pouvoir faire une division en Sass vous devez remplir l'une de ces conditions :

- Une opérande/ Les opérandes sont des valeurs dans des variables
- Les opérandes se trouvent entre parenthèse
- Le symbole / est utilisé dans une expression mathématique

```
$large : 200px
p
        width: $large / 2
```

```
.rectangle
        height : (100px / 2)
```

```
p
        margin-left : 5px * 8px / 2px
```

Opérations sur les couleurs !

#010401 + #020507

01 + 02 = 03

04 + 05 = 09

01 + 07 = 08

=

#030908

Fonctions

Mot clé qui sert à définir une fonction en Sass.

Nom de la fonction. Ce nom doit respecter les mêmes règles que pour les variables.

```
@function nomFonction( parametres ){

    //Operations effectuees par la fonction

}
```

Indiquent le début et la fin de la fonction.

[Entrée] Entre parenthèses, les paramètres sont les valeurs avec lesquelles la fonction va travailler.

Functions - Calculer une hauteur

style.scss

```scss
@function calculHauteur($largeur){
        $hauteur: $largeur * 2;
        @return $hauteur;
}

div {
        width: 250px;
        height: calculHauteur(250px);
}
```

style.css

```css
div {
  width: 250px;
  height: 500px;
}
```

On crée une fonction **calculHauteur (nom de la fonction)** qui vas nous permettre de calculer la hauteur (ty captain obvious) en fonction d'une **largeur donnée (paramètre).**

Functions - Calculer une hauteur

style.scss

```scss
@function calculHauteur($largeur){
        @return $largeur * 2;
}

div {
        width: 250px;
        height: calculHauteur(250px);
}
```

style.css

```css
div {
  width: 250px;
  height: 500px;
}
```

Peut-être également saisi comme ceci :)

Functions - Tips

```scss
@function RGBlight($red, $green, $blue){
  $newRed: $red + 20;
  $newGreen: $green + 20;
  $newBlue: $blue + 20;
  $lightenColor: rgb($newRed, $newGreen, $newBlue);
  @return $lightenColor;
}

span{
  background-color: rgb(59, 152, 175);
        &:hover{
        background-color: RGBlight(59, 152, 175);
        }
}
```

```scss
span {
  background-color: #3b98af;
}

span:hover {
        background-color: #4facc3;
}
```

Il y a certains types de fonctions que vous n'avez pas besoin de créer car ...

Là par exemple, on veut rendre une couleur RGB plus clair.

Functions - Tips

```
$color1 : rgb(59, 152, 175);
$amount : 40;
$color2 : lighten($color1, $amount);
```

Sass à déjà créer une fonction pour avoir le résultat voulu ! (Et qui fait le travail encore mieux !)

+ De fonctions SASS ?

http://sass-lang.com/documentation

RTFM* <3

Read the fxcking Manual

Mixins

style.scss

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

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

style.css

```css
.box {
  -webkit-border-radius: 10px;
  -moz-border-radius: 10px;
  -ms-border-radius: 10px;
  border-radius: 10px;
}
```

Un mixin est une directive qui vous permet de **définir des règles CSS multiples.**

Mixins

style.scss

```scss
@mixin center() {
        display: block;
        margin-left: auto;
        margin-right: auto;
}

.container {
        @include center();
}

.image-cover {
        @include center;
}
```

style.css

```css
.container {
  display: block;
  margin-left: auto;
  margin-right: auto;
}

.image-cover {
  display: block;
  margin-left: auto;
  margin-right: auto;
}
```

Les Ressources - App

- CodeKit (Payant)
- Compass.app (Payant, Open Source)
- Ghostlab (Payant)
- Hammer (Payant)
- Koala (Open Source)
- LiveReload (Payant, Open Source)
- Prepros (Payant)
- Scout (Open Source)

Les Ressources

- http://sass-lang.com/
- https://la-cascade.io/se-lancer-dans-sass/
- http://thesassway.com/
- https://sass-guidelin.es/fr/#propos-de-sass