Installation et utilisation de Sass

SASS signifie Syntactically Awesome StyleSheets.

Installation

Il faut un logiciel pour générer notre fichier CSS à partir de notre fichier SASS. Nous pouvons utiliser Ruby ou Node.js. Ici nous utiliserons ce dernier.

Node.js

Pour installer Node.js, il faut le télécharger sur son site https://nodejs.org/.

Une fois l'installation réalisée, nous pouvons vérifier la présence de Node.js en tapant dans la console :

```
$ node -v
```

Node-sass

On installe ensuite le module node-sass avec la commande suivante dans le terminal :

```
$ npm install -g node-sass
```

Premier pas

Mise en place

Pour les exemples donnés dans ce document, nous utilisons la page html suivante :

```
<!DOCTYPE html>
<html lang="fr">
<head>
  <meta charset="UTF-8">
  <title>Utilisation de sass</title>
  <link rel="stylesheet" href="styles.css">
</head>
<body>
  <header>
    <h1>Présentation de Sass</h1>
    <p>Un préprocesseur CSS puissant</p>
  </header>
  <section>
    <h2>En résumé</h2>
    <p>Sass (Syntactically Awesome Stylesheets) est un langage de génération dynamique de feuilles de style initialement d
    <img src="logo-sass.png" alt="Logo de Sass">
    <h2>Présentation</h2>
    <p>Sass est un métalangage de feuilles de style en cascade (CSS). C'est un langage de script qui est interprété en CSS
  </section>
  <aside>
    <h3>En plus</h3>
    <p>Sass peut être étoffé avec Compass (pratique pour les préfixes des différents navigateurs).</p>
  </aside>
  <footer>
    <ul>
```

```
      <li><a href="http://sass-lang.com/" title="Site officiel Sass">Page officielle de Sass</a></li>
      <li><a href="https://github.com/sass/sass">Ruby - Sass</a></li>
      <li><a href="https://github.com/sass/node-sass">Node Sass</a></li>
    </ul>
  </footer>
</body>
</html>
```

Nous créons à côté de cette page le fichier styles.scss.

Génération du fichier css

Principe de base

Avec le terminal, nous nous plaçons dans le dossier où se trouve le fichier .scss.
À chaque fois que nous modifions ce document , nous exécutons la commande suivante dans le terminal :

```
$ node-sass styles.scss styles.css
```

Automatisation

Nous pouvons automatiser la génération du css à chaque enregistrement du fichier scss de la façon suivante :

```
$ node-sass --watch styles.scss styles.css
```

Options

Quelques options utiles :

- minification

```
$ node-sass --compressed styles.min.scss styles.css
```

- nombre d'indentations :

```
$ node-sass --5 styles.scss styles.css
```

- aides de la commande :

```
$ node-sass --help
```

- génération du fichier map (pour voir dans l'inspecteur du navigateur la référence vers le fichier scss) :

```
$ node-sass --source-map true
```

Premier usage

Nous écrivons du css dans notre fichier .scss et générons le fichier .css correspondant.

Si vous écrivez le code suivant dans votre fichier SCSS :

```
body{
  color: grey;
  margin: 0;
}
```

Si vous exécutez la commande de génération du fichier CSS (cf. supra), vous obtenez un fichier CSS contenant :

```
body{
  color: grey;
  margin: 0;
}
```

Écrire avec Sass - niveau 1

Il existe plusieurs principes d'écriture qui aident à réaliser plus simplement son fichier css :

- l'imbrication des règles
- Le rappel du parent : &
- l'imbrication des propriétés
- les variables

Imbrication des règles

On peut imbriquer une règle dans une autre de la façon suivante :

SCSS

```
header{
  margin-bottom: 10px;
  p{
    margin: 5px 0 0;
  }
}
```

Résultat en CSS

```
header{
  margin-bottom: 10px;
}
  header p{
    margin: 5px 0 0;
  }
```

Rappel du parent : &

On peut également rappeler un parent dans une règle imbriquée.

SCSS

```
a{
  text-decoration: none; color: #CD679A;
  }
  &:hover{
    color: #601137;
  }
}
```

Résultat en CSS

```
a{
  text-decoration: none; color: #CD679A;
}
  a:hover{
    color: #601137;
  }
```

Imbrication des propriétés

SCSS

```
li a{
  border:{
    bottom-width: 2px;
    bottom-color: #601137;
    bottom-style: solid;
  }
}
```

Résultat en CSS

```
li a{
  border-bottom-width: 2px;
  border-bottom-color: #601137;
  border-bottom-style: solid;
}
```

Variables

L'usage des variables permet tenir à jour plus simplement ses styles.

SCSS

```
$colorLink : #CD679A;
$colorLinkHover: #601137;
a{
  text-decoration: none; color: $colorLink;
  }
  &:hover{
    color: $colorLinkHover;
  }
}
```

Résultat en CSS

```
a{
  text-decoration: none; color: #CD679A;
}
  a:hover{
    color: #601137;
  }
```

Écrire avec Sass - niveau 2

@import

Afin de conserver des documents CSS les plus courts possibles, il est bon de séparer l'ensemble de ses déclarations de styles en plusieurs documents SCSS. La déclaration @import permet d'importer un fichier scss dans un autre.
Par exemple, pour le fichier _variables.scss, on utilise la commande :

```
@import 'variables';
```

Vous remarquez qu'il n'est pas nécessaire d'écrire le _ ni l'extension du fichier. La présence du _ est une convention pour manifester que ce fichier est importé par un autre.

Pour utiliser au mieux cette possibilité d'importation, vous veillerez à créer un fichier styles.scss qui ne contiendra que des déclarations @import. Chaque fichier appelé contiendra les styles pour une partie de la page (en-tête, section, pied-de-page …) ou certaines pages (contact, portfolio …).

Opérations

On peut faire en SASS les opérations classiques y compris le modulo.

Interpolations

Le contenu d'une variable peut être interpolé (interprété) en utilisant les accolades.

SCSS

```
$target : li;

nav #{$target}{
  list-style-type: none;
}
```

Résultat en CSS

```
nav li{
  list-style-type: none;
}
```

Aller plus loin : règles et directives

Afin de simplifier davantage nos déclarations, il est utile d'utiliser du code écrit dans notre fichier à plusieurs reprises sans avoir besoin de le réécrire à chaque fois.

@mixin - sans variable

SCSS

```
@mixin border-box{
  -moz-border-radius: 10px; -webkit-border-radius: 10px; border-radius: 10px;
}

footer{
  @include border-box;
}
```

Résultat en CSS

```
footer{
  -moz-border-radius: 10px; -webkit-border-radius: 10px; border-radius: 10px;
}
```

@mixin - avec variable

Le principe de base est le même que dans l'exemple précédent ; mais nous pouvons, en plus, utiliser des variables. Nous sommes donc en présence de fonctions pour générer du css en fonction de nos besoins.

SCSS

```
@mixin border-box($dimension){
  -moz-border-radius: $dimension; -webkit-border-radius: $dimension; border-radius: $dimension;
}

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

Résultat en CSS

```
footer{
  -moz-border-radius: 10px; -webkit-border-radius: 10px; border-radius: 10px;
}
```

@extend

L'utilisation de placeholder nous permet de réutiliser une déclaration de class dans une autre partie de notre code de la façon suivante :

SCSS

```
.bordureBase{
  border: #601137 solid;
}

header{
  @extend .bordureBase;
  border-width: 2px;
}
```

CSS

```
.bordureBase{
  border: #601137 solid;
}

header{
  border: #601137 solid 2px;
}
```