Introduction à SASS

Différence entre SASS et SCSS

https://grafikart.fr/tutoriels/differences-sass-scss-329

Organisation des dossiers

On fait comme on veut.

Là, les fichiers sources sont désormais dans le dossier scss (les fichiers du dossier src ne sont plus utilisés)

Ils ont tous été renommés : _nom.scss

1. extension : scss
2. prefixe : _

Ils contiennent donc le code css d'origine : pas de soucis, SASS c'est du CSS. Vous pouvez donc les modifier en SASS à votre guise.

css/styles.css devient le résultat de la compilation

Le fichier styles.css initial a été renommé style-import.css : il n'est donc plus utilisé. Pour rappel, l'importation en CSS implique l'envoi de tous les fichiers via http (pas bon !)

Le nouveau fichier styles.css est le résultat de la compilation SASS (il est re-créé automatiquement à chaque compilation)

Le fichier scss correspondant se trouve dans le dossier scss/styles.scss.

css/scss/styles.scss est le fichier qui sera compilé

Ce fichier n'a pas de prefix : _

Il fait la même chose : importation (en syntaxe SASS) des sources mais des sources scss : il fait donc partie des sources.

C'est lui qu'il faut compiler avec la commande shell suivante (en partant du dossier scss):

sass styles.scss ../styles.css

La différence est que tout le code se trouvera dans le fichier styles.css généré, le navigateur ne chargera donc que ce fichier.

A quoi sert le prefix _

Lorsque on utilise une tâche qui compile automatiquement à chaque enregistrement d'un fichier source, seul les fichiers qui ne portent pas de préfixe _ sont compilés.

Dit autrement, la tâche consiste à surveiller un changement dans le dossier scss. Si changement (modification d'un fichier), tous les fichiers scss sont compilés, sauf ceux précédés par le préfixe _

Si un fichier n'a pas de préfixe _ ,il est compilé : on se retrouve alors avec le fichier css correspondant :

- css/scss/variables.scss est compilé en css/variables.css
- css/scss/_variables.scss est ignoré

Etant donné que le fichier styles.scss importe le code des autres fichiers, lui seul a besoin d'être compilé.

Syntaxe de base

Documentation : https://sass-lang.com/guide Tutoriel vidéo : https://grafikart.fr/formations/sass-preprocesseur

- Importation et commentaire (voir styles.scss)
- Déclarer des variables (voir _variables.scss)
- Chemins de fichier déclarés dans le CSS (voir _fonts.scss)

Seul le fichier _banner.scss a été modifié pour vous donner un exemple (en dehors des modifications citées ci-dessus)

Vous y trouverez :

Le principe d'imbrication

Sélecteur SASS

```
.site-header {
    text-align : center;

    &-nav-link {
        color : white;
    }
}
```

Code css généré

.site-header { text-align : center; }

.site-header-nav-link { color : white; }

le caractère **&** reprend le sélecteur de la règle parente Là, on voit l'intérêt du nommage de classe !

Dans le cas suivant, on cible les pseudos classes de nav-link : plus rapide à écrire !

```
.site-nav-link {

    color : white;

    &:hover,
    &:focus,
```

```
    &:active {
        text-decoration : none;
    }
}
```

qui génére

```
.site-nav-link {
     color : white;
}

.site-nav-link:hover,
.site-nav-link:focus,
.site-nav-link:active {
    text-decoration : none;
}
```

Précautions

Indentation du code

Il va sans dire que l'indentation du code est primordiale pour éviter la catastrophe si une fermeture saute par mégarde.

Attention aux sélecteurs d'enfants à éviter

On serait tenté de faire :

```
header {
    nav {
        a {
            text-decoration : none;
        }
    }
}
```

Dans ce cas, on génère :

```
header nav a {
    text-decoration : none;
}
```

Rappel : il est préférable de cibler les éléments directement, avec une classe plutôt qu'un identifiant trop spécifique

@extend et placeholder

@extend permet de reproduire une règle sur plusieurs éléments

```
.btn {
    display : inline-block;
}

.nav-link {
    @extend .btn;
}
```

produira

```
.btn,
.nav-link {
    display : inline-block;
}
```

Avec un placeholder : sélecteur % (n'existe pas en CSS)

```
%btn {
    display : inline-block;
}

.nav-link {
    @extend %btn;
}
```

ne produira que :

```
.nav-link {
   display : inline-block;
}
```

Voir : @extend

@mixins : créer des fonctions

@mixins

Fonctions de couleurs

Obtenir une transparence, foncer ou éclaircir une couleur, désaturer, etc.

sass::color