PROGRAMMATION MULTIMEDIA (S4)

Langage SASS

IUT Laval
Le Mans Université

Aous Karoui
aous.karoui@univ-lemans.fr

Organisation

- Définition
- Installation
- Mise en pratique
- Écouteurs
- Variables
- Règles d’imbrication
- L’opérateur &
- L’importation
- Les partials
- L’héritage
- Les fonctions
- Les mixins

Langage SASS - Définition

- Un langage dynamique pour la création / gestion automatique de fichiers *css,* appelé aussi préprocesseur

- Propose les deux syntaxes sass et scss
- Sass: historiquement plus ancien, utilise l'indentation au lieu des accolades pour spécifier des blocs
- Scss: plus récent, a un formalisme plus proche du css3

- SASS s'occupe de transformer en *css* valide le code écrit en *Sass* ou en *Scss*

Langage SASS — Exemples de syntaxe

- sass

```
.links
    width: 200px
    height: 50px
    background-color: gray
    border: 1px solid red
    border-radius: 5px

.header
    position: relative
    top: 0
    left: 0
    width: 100%
    height: 70px
    background-color: gray
```

- scss

```
.links{
    width: 200px;
    height: 50px;
    background-color: gray;
    border: 1px solid red;
    border-radius: 5px;
}

.header{
    position: relative;
    top: 0;
    left: 0;
    width: 100%;
    height: 70px;
    background-color: gray;
}
```

Langage SASS - Installation

- Pour installer *SASS*, il faut avoir *Ruby* au préalable

- Installation Ruby
- https://rubyinstaller.org/downloads/
- Dans RubyInstallers, choisir la version la plus récente
- Après téléchargement, exécuter et installer

Langage SASS - Installation

- Lancer l'invite de commande (la console)
- Exécuter la commande:
- gem install sass
- Patienter jusqu'à avoir l'écran suivant:

```
Fetching: sass-3.4.19.gem (100%)
Successfully installed sass-3.4.19
Parsing documentation for sass-3.4.19
Installing ri documentation for sass-3.4.19
Done installing documentation for sass after 34 seconds
1 gem installed
```

- Pour vérifier que l'installation s'est bien passée
- Taper sass –v dans la console

Langage SASS — Mise en pratique

- Ensuite, créez un fichier nommé « fichier_initial.scss"
- Placez le fichier créé dans un répertoire nommé "sass"
- Dans votre fichier, placer le code suivant

```
body{
    color: white;
    background-color: black;
}
```

- Accédez à votre répertoire depuis la ligne de commande
- Lancer la génération de feuille de styles *css* via la commande suivante

```
sass fichier_initial.scss fichier_genere.css
```

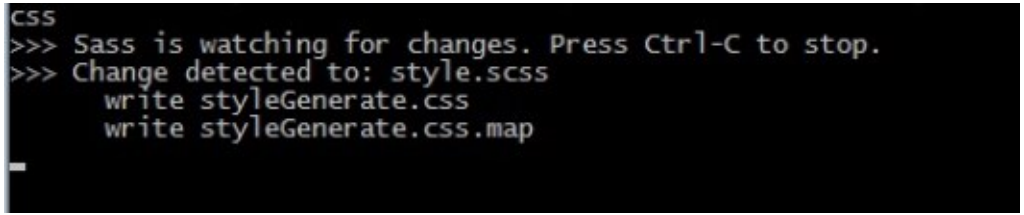
- Retrouvez le fichier créé par SASS dans votre répertoire

Langage SASS – Écouteur sur fichier

- Pour mettre à jour, il est possible de retaper la commande précédente à chaque fois ou alors (il vaut mieux):
- Placer un écouteur sur votre fichier .scss

```
sass --watch fichier_initial.scss fichier_genere.css
```

- Sauvegarder les modifications sur votre fichier .scss et observez les modifications sur le fichier généré



Langage SASS – Écouteur sur répertoire

- Créer deux répertoires: "*scss*" et "*css*"
- Maintenant, au lieu de demander à Sass d'écouter un fichier .scss en particulier, vous lui demanderez d'écouter tous les fichiers se trouvant dans le dossier scss, et de générer les feuilles de styles qui sont équivalentes à chacun de ces fichiers dans le dossier regroupant vos feuilles de styles (css)
- En vous plaçant dans le dossier racine utilisez la commande suivante

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

Langage SASS – Les variables

- Voici les différents types de variables prise en compte par le langage SASS

Types de données	Exemples
Nombres	1, 2, 3, 4, ...
Chaines de caractères	"Zoomer", "Surligner", bold, left
Couleurs	red, rgba(255, 0, 0, 0.5), #04a3ff
Des listes de valeurs séparées par des espaces ou des virgules	10px 5px 2px, Helvetica, sans-serif, Arial

- Ainsi le code suivant en *scss* sera transformé en *css* comme suit:

```
$contenu: "Ceci est une phrase"

.before{
    content: $contenu;
    text-align: center;
}
```

```
.before{
    content: "Ceci est une phrase";
    text-align: center;
}
```

Langage SASS — Les règles d'imbrication

- SASS permet également d'imbriquer les balises les unes dans les autres
- Pour cet exemple:

```
<div id="menu">
  <div id="sous-menu">
  </div>
</div>
```

```
#menu{
    width: 100%;
    height: 70px;
    position: relative;
    top: 0;
    left: 0;

    #sous-menu{
        width: 700px;
        height: 70px;
        line-height: 70px;
        margin: 0 auto;
    }
}
```

scss

```
#menu{
    width: 100%;
    height: 70px;
    position: relative;
    top: 0;
    left: 0;
}

#menu #sous-menu{
    width: 700px;
    height: 70px;
    line-height: 70px;
    margin: 0 auto;
}
```

css

Langage SASS – Imbrication des propriétés

- Consiste à écrire une seule fois l'espace de nom et imbriquer à l'intérieur les propriétés (pour éviter les répétitions)

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

scss

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

css

Langage SASS — L'opérateur &

- C'est l'équivalent du « *this* » en *java*.
- Il permet d'utiliser la règle d'imbrication pour faire référence au sélecteur dont il est l'enfant.
- Par exemple si vous êtes à l'intérieur du sélecteur p, l'opérateur & fait référence à ce p.
- Ainsi, écrire p{&} revient à écrire p{p}.

- Voici un exemple d'utilisation :

```scss
#main {
  color: black;
  .link {
    font-weight: bold;

    &:active { display: none; }
  }
}
```

scss

```css
#main {
  color: black;
}

#main .link {
  font-weight: bold;
}

#main .link:active {
  display: none;
}
```

css

Langage SASS — L'opérateur &

- Cas pratique:
- Vous souhaitez changez l'apparence d'une balise <a> au survol de la souris
- Trouvez le code scss qui vous permet d'avoir le css suivant

```
a {
  font-weight: bold;
  text-decoration: none;
}

a:hover {
    text-decoration: underline;
 }
```

Langage SASS – L'importation

- Permet d'importer du code à partir d'autres fichiers
- Consiste à créer des fichiers appelés « briques réutilisables »
- Il suffit d'écrire

  @import nom-fichier-scss (sans ajouter l'extension du fichier)

- Pour importer depuis un répertoire

  @import 'sous-dossier'/fichier_à_importer (sans ajouter l'extension du fichier)

- Cas pratique:
- Pour votre portfolio, créer une page « menu.scss » que vous importez dans les trois pages de votre site web: Accueil, Expériences, Contact

Langage SASS — Les partials

- C’est les mêmes fichiers à importer sauf que ces derniers ne génèrent aucune feuille de style.
- Pour créer un partial, il suffit de commencer le nom du fichier par un *underscore* « _ »
- Exemple: ‘_menu.scss‘

- Vous pouvez appliquer à votre exemple de menu

Langage SASS – L'héritage

- Parfois, deux éléments HTML peuvent avoir à peu prés le même style. Dans ce cas, vous aurez besoin de réutiliser le style de l'un dans l'autre. Par exemple, vous avez deux divs affichant des messages, l'une ayant la classe *error* , et l'autre la classe *success* .

Ces deux divs ont :

- La même largeur
- La même hauteur
- La même police

```scss
.error{
    width: 500px;
    height: 30px;
    font-family: arail;

    color: red;
}

.succes{

    //La classe success herite de la classe error
    @extend .error;
}
```

scss

```css
.error, .succes {

  width: 500px;
  height: 30px;
  font-family: arail;

  color: red;

}
```

css

- Maintenant, adaptez votre code pour que le message d'erreur soit de couleur rouge, tandis que le message de succès soit de couleur verte

Langage SASS — Les fonctions

- La déclaration d'une fonction se fait presque comme en Php.
- Il y a tout de même le symbole '@' qui se rajoute juste avant les mots-clés
- Voici un exemple

```scss
//On defini notre fonction ici
@function calculHauteur($largeur){

  @return $largeur * 2;

}

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

SCSS

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

CSS

Langage SASS — Les mixins

- Les *mixins* permettent de définir des groupes de styles *css* réutilisables. Par exemple, si vous avez un même style que vous voulez appliquer à plusieurs éléments de vos pages web, vous pouvez les définir dans un *mixin*.
- Imaginez que vous avez plusieurs éléments auxquels vous voulez appliquer les styles suivants :
- Border-radius : 50%;
- Width : 50px;
- Heigth : 50px;
- Font-family : arial;
- Font-weight : bold;
- Vous pouvez en faire un mixin.
- Pour définir un mixin, il suffit de faire comme suit :

```
@mixin nomMixin{

    //Les differents styles

}
```

- Pour utiliser les styles définis dans un mixin, il suffit de faire : @include suivi du nom que vous avez donné au mixin.

```
@include nomMixin
```