

SS

est un métalangage qui vient se greffer par dessus les **CSS** pour en améliorer la lisibilité et faciliter la maintenance.

permet :

- d'utiliser des **variables**,
- d'utiliser des **mixins** pour éviter les répétitions,
- de **compiler** différents fichiers,
- ce qui permet de **modulariser** les CSS,
- d'**imbriquer** les sélecteurs pour représenter leur hiérarchie,
- de régler le niveau de **compression** des fichiers css,
- etc...



chiers Sass ont une extension **.sass** ou **.scss**.

ormément aux pratiques les plus répandues actuellement,
allons privilégier l'extension **.scss**.

aller Sass

charger la version autonome de dart-sass qui corresponds à votre système d'opérations

<https://github.com/sass/dart-sass/releases/>

er le dossier dart-sass à la racine de votre projet Web

er dans le terminal :

```
Affiche la version de Sass  
/dart-sass/sass --version  
y 2.4.2p198 (2017-09-14 revision 59899) [x86_64-darwin16]
```

apez pas le signe \$; celui-ci ne sert qu'à indiquer le début de la commande.
s avoir tapé la commande, appuyez sur **Enter** pour exécuter la commande.

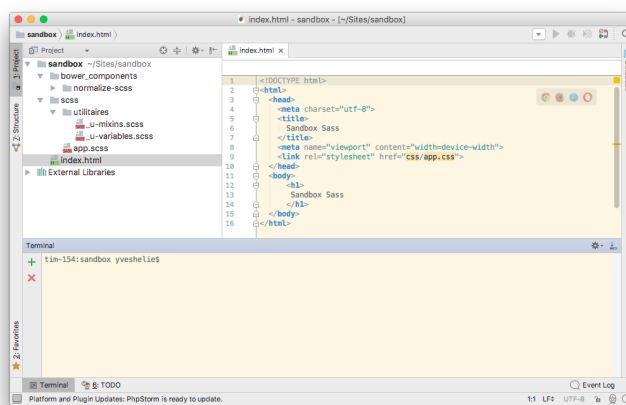
développer nos feuilles de styles, nous utilisons principalement :
un **éditeur de code** pour écrire nos styles en **Sass**;
le **Terminal** pour passer les commandes de compilation à Sass;
Chrome (**un navigateur**) pour afficher le résultat de la page.



pour simplifier, nous utiliserons **PHPStorm** qui possède un terminal intégré.
Glissez le dossier **sandbox** sur l'icône de **PHPStorm** qui est dans le "dock".

Il est important que bien glisser le dossier **sandbox** et non pas un dossier de niveau supérieur (ex. c03i_sass).

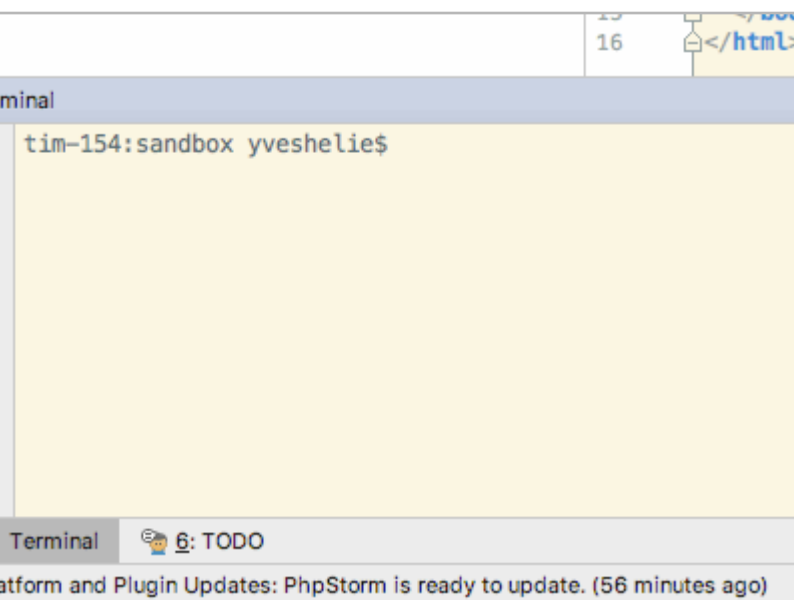
Arborescence



Éditeur de

Terminal

ez sur le bouton **Terminal** au bas, à gauche de l'éditeur :



l'éditeur par défaut, vous vous retrouverez dans le dossier **sandbox**. Si ce n'est pas le cas, naviguez vers le dossier **sandbox** ici :

Naviguer vers le dossier du “sandbox”
dans Sites/sandbox/

L'interface offerte dans **PhpStorm** est l'équivalent de l'application **Terminal** offerte par MacOS.

Compiler les styles Sass

z dans le terminal :

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

ez **app.scss** dans l'éditeur de code et tapez :

```
{  
  color : red;
```

ervez :

- la réponse de Sass dans le **terminal**;
- l'apparition du dossier **css** et à l'intérieur, du fichier **app.css**;
- le fichier **app.css** dans l'éditeur de code;
- l'apparition d'un dossier **.sass-cache**.

Leurs paramètres peuvent être utilisés lors de l'utilisation de la commande `sass`.

Syntaxe complète est :

```
s --watch input:output --no-cache --style [style]
```

ss	Nom de la commande.
watch	Indique à la commande qu'elle doit s'exécuter en mode "surveillance". Le paramètre est facultatif; s'il est absent, la commande ne s'exécutera qu'une seule fois.
out:output	Détermine la provenance des fichiers Sass et la destination des fichiers CSS. Il est possible de cibler tant les dossiers que les fichiers. (<code>scss:css</code> <i>ou</i> <code>scss/app.scss:css/app.css</code>).
no-cache	Pour ne pas créer de fichiers temporaires (.sass-cache). Équivalent du paramètre <code>-C</code> .
style [style]	Modifie la lisibilité et/ou la compression du fichier de sortie. Équivalent du paramètre <code>-t</code> . Quatre options sont possibles : <code>nest</code> (par défaut), <code>expanded</code> , <code>compact</code> <i>ou</i> <code>compressed</code> .

Pour arrêter la surveillance des fichiers, utilisez `ctrl-c` dans le terminal.

Pour en savoir plus sur les commandes Sass, affichez l'aide grâce au paramètre `-h` (`help`).

Compression du fichier de sortie

Pendant le développement, on utilise le mode `expanded` car il offre une lisibilité maximale.

Pour la mise en production, `compressed` est à privilégier puisqu'il permet de produire des fichiers plus légers en supprimant les tabulations et les sauts de ligne inutiles.

À l'échelle du niveau des **commentaires**, deux types peuvent être utilisés :

Les commentaires **bloc** sont des commentaires "généraux"

Les commentaires en **ligne** s'adressent aux développeurs

Sass original	En mode <code>expanded</code>	En mode <code>compressed</code>
<pre>/** Commentaire bloc */ // Commentaire en ligne article { h1 { color: red; } }</pre>	<pre>/** Commentaire bloc */ article h1 { color: red; }</pre>	<pre>article h1{color:red}</pre>

ichage des règles avec Sass

age des sélecteurs

peut écrire des sélecteurs contextuels (ou descendants) en représentant l'arborescence par d
lades.

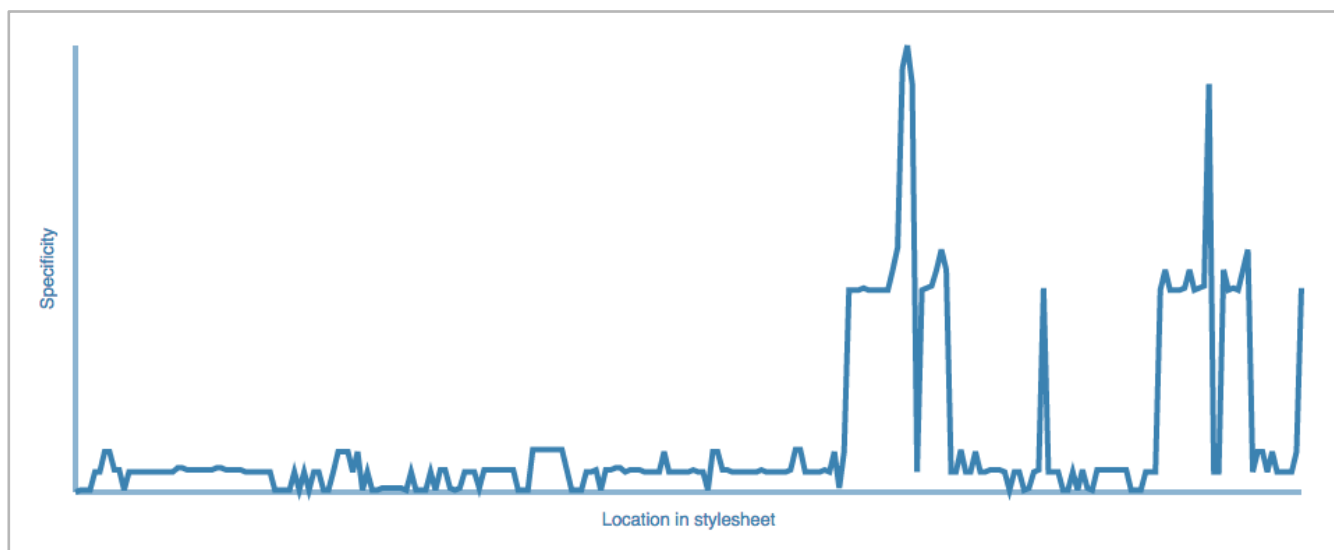
CSS	Sass
<pre>/* { width: 80%; /* ul { list-style-type: none; /* li { float: left; /* li a { font-weight: bold;</pre>	<pre>nav { width: 80%; ul { list-style-type: none; } li { float: left; a { font-weight: bold; } } }</pre>

chage des sélecteurs est pratique, mais à utiliser avec beaucoup de modération!

s une approche **BEM** ou **OOCSS**, on cherche à rendre les aspects de présentation les plus indépendants possibles du DOM (de l'arbre HTML). Il faut donc éviter autant que possible les sélecteurs contextuels.

us, les sélecteurs contextuels complexifient la maintenance, introduisent des problèmes de spécificité et alourdissent les fichiers CSS.

évaluez votre usage de la spécificité, testez votre CSS avec ce générateur de courbe :
<http://jonassebastianohlsson.com/specificity-graph/>



néorie, il n'y a pas de limites au nombre de niveaux de nichage qu'on peut effectuer.

Pratique, une limite (maximum!) de **quatre niveaux** de contextualisation est suggérée par l'éco-
lass; viser **trois niveaux** serait encore mieux!

À éviter	À privilégier
<pre>main { nav { ul { li { a { [...] } } } } }</pre>	<pre>main { nav { ul { [...] } li { a { [...] } } } }</pre>

Source : The Inception Rule (<http://thesassway.com/beginner/the-inception-rule>)

age des propriétés

vous pouvez aussi nicher les propriétés plutôt que de répéter les premiers mots :

Sass	CSS
<pre>fakeshadow { border: { style: solid; left: { width: 4px; color: #888; } right: { width: 2px; color: #ccc; } } }</pre>	<pre>.fakeshadow { border-style: solid; border-left-width: 4px; border-left-color: #888; border-right-width: 2px; border-right-color: #ccc; }</pre>

age des pseudo-classes avec '&'

possible de nicher des sélecteurs de pseudo-classes telles que `:hover` en utilisant l'esperlu
e `&` fait référence au sélecteur parent.

Sass	CSS
<pre>color: #ce4dd6; &:hover { color: #ffb3ff; } &:visited { color: #c458cb; }</pre>	<pre>a { color: #ce4dd6; } a:hover { color: #ffb3ff; } a:visited { color: #c458cb; }</pre>

Page de la syntaxe BEM

perluette permet aussi de nicher les noms de classes formés avec la nomenclature BEM.

Sass	CSS
<pre>block { background-color: #2fd1af; height: 10vh; &__element { background-color: #d17d71; &--modificateur { margin: 15px; } } }</pre>	<pre>.block { background-color: #2fd1af; height: 10vh; } .block__element { background-color: #d17d71; } .block__element--modificateur { margin: 15px; }</pre>

age des requêtes médias

eu de séparer de leur contexte les modifications à apporter à un module ou à un élément, Sa
permet de définir chaque changement à même le contexte de l'élément.

ode sera donc beaucoup plus facile à comprendre et à maintenir.

Sass	CSS
<pre>font-size: 32px; @media(min-width: 640px){ font-size: 48px; }</pre>	<pre>h1 { font-size: 32px; } @media (min-width: 640px) { h1 { font-size: 48px; } }</pre>

vrai que cette approche multiplie les requêtes `media` dans la compilation CSS, mais il a été
ontré que les impacts sur la performance étaient insignifiants.

oir des variables

nous permet l'utilisation du symbole \$ pour créer une variable.

```
couleurDominante: navy; // couleur nommée
couleurSecondaire: #000080; // couleur hexadécimale
chaine: " avec chaine ajoutée"; // chaîne
baseFontSize: 10px; // valeur numérique
bordureMince: 1px solid $couleurDominante; // valeurs multiples
paddingNormal: 15px 10px 20px 10px; // valeurs multiples

h2 {
color: $couleurDominante;

paragraphe {
font-size: $baseFontSize;
border: $bordureMince;
padding: $paddingNormal;
:after {
content: $chaine;
```

mixins

mixins permettent de définir un groupe de règles qui sont communes à plusieurs sélecteurs.

qu'un petit groupe de règles CSS est régulièrement utilisé, il est sage d'en faire un mixin. Ainsi, pour apporter un changement, il ne suffit que de le faire à un seul endroit!

Création d'un mixin

un mixin est déclaré avec le symbole `@` suivi du mot-clé `mixin` puis du nom subjectivement choisi pour décrire le groupe de règles.

```
@mixin traitsCommuns {  
  border-radius: 10px;  
  border: 1px solid green;  
  padding: 10px;  
}
```


ation d'un mixin

l'utiliser, il suffit d'utiliser le symbole @, suivi du mot-clé `include`, puis du nom du `mixin`.

Sass	CSS
<pre><code>mixin traitsCommuns { border-radius: 10px; border: 1px solid green; padding: 10px; header { color: #274D87; @include traitsCommuns; footer { color: #3264AF; @include traitsCommuns;</code></pre>	<pre><code>header { color: #274D87; border-radius: 10px; border: 1px solid green; padding: 10px; } footer { color: #3264AF; border-radius: 10px; border: 1px solid green; padding: 10px; }</code></pre>

mixins permettent de stocker des fragments de code (snippets) réutilisables tels que le clearfi

```
ixin clearfix {  
:after {  
  content: " ";  
  display: table;  
  clear: both;
```

plus d'exemples, voir le fichier _utilitaires.scss.

bibliothèques de mixins Sass sont également créées par des développeurs afin de faciliter le
il d'intégration.

Compass : <http://compass-style.org>

Bourbon : <http://bourbon.io>

Susy : <http://susy.oddbird.net> ← obsolète maintenant!

fonctions

us de permettre des opérations mathématiques, Sass offre des fonctions mathématiques et
ation de base.

de détails sur les fonctions de Sass : <http://sass-lang.com/documentation/Sass/Script/Functions>.

ctions mathématiques

Sass	CSS
<pre>largeurMaximale: 1000px; .item1 { width: \$largeurMaximale/2; .item2 { width: percentage(4/12);</pre>	<pre>.item1 { width: 500px; } .item2 { width: 33.3333333333%; }</pre>

Fonctions de coloration

Fonctions de coloration facilitent la tâche des intégrateurs en permettant de décliner facilement une palette de couleurs.

peuvent servir, par exemple, à créer facilement des effets de survol :

Sass	CSS
<pre>couleurPrincipale : red; color: \$couleurPrincipale; a: hover { // Rendre la couleur 10% plus foncée color: darken(\$couleurPrincipale, 10%); }</pre>	<pre>a { color: red; } a: hover { color: #cc0000; }</pre>

Quelques exemples de fonctions de coloration : lighten, darken, grayscale, saturate, desaturate, mix, fadeout, fade, spin, mix...

Pour en savoir plus sur les fonctions de coloration, consultez ce guide : <http://jackiebalzer.com/co>

ctions personnalisées

permet également la déclaration de nos propres fonctions utilitaires. Jumelées à la création mixins, le tout devient un outil assez puissant afin de nous faciliter la tâche d'intégration!

ction de calcul rem

Sass	CSS
<pre>function calculateRem(\$size) { \$remSize: \$size / 10px; return #{ \$remSize }rem; mixin fontSize(\$size) { font-size: \$size; font-size: calculateRem(\$size); } @include fontSize(15px);</pre>	<pre>h5 { font-size: 15px; font-size: 1.5rem; } /* Résultat compilé : * - Solution de repli * - Valeur en rem */</pre>

ction de calcul de grille fluide

Sass

Déclaration de la fonction fluidize
 $\text{valeurCible} / \text{valeurContexte} * 100 = \text{valeurCible en \%}$

```
function fluidize($target, $context){  
  @return ($target / $context) * 100%;  
}
```

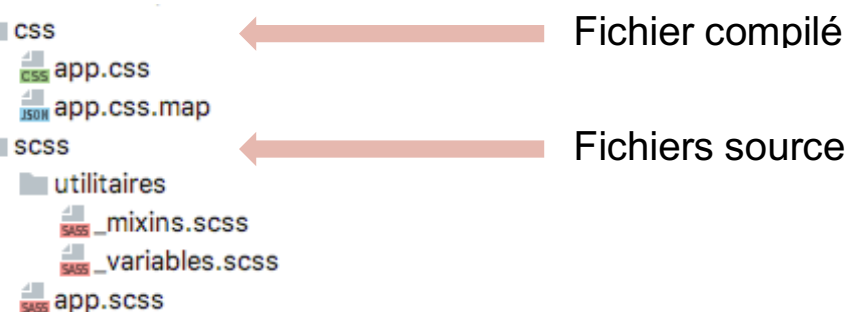
```
.sidebar{  
  width: fluidize(350px, 1000px);  
}
```

CSS

```
.sidebar {  
  width: 35%;  
}
```

ularisation des styles et importation par Sass

es grands avantages de Sass est de faciliter la modularisation des CSS, car il nous permet
porter différents fichiers Sass et de compiler le tout dans un seul fichier de sortie. Cette
ionnalité nous permettra donc de morceler nos règles CSS en différents fichiers.



chier principal, **styles.css** ou **app.css**, contiendra une table des matières en début de docum
expliquer la structure de ses sections. Chaque section est chapeautée par son intitulé qui do
spondre à ce qui est annoncé dans la table des matières.

t être rigoureux dans la création des entêtes de section car le but de la table des matières es
voir naviguer rapidement à la section en faisant une sélection puis en utilisant la fonction FIND

que section contiendra un groupe de règles ou une instruction `@import 'section'`.

Exemple d'un fichier app.scss qui importera les fragments :

```
@author Prénom Nom - courriel
-----
Table des matières
-----

Bibliothèques (dans le dossier bower_components)
  Normalize
  Utilitaires
  Variables (attention à l'ordre d'importation!)
  Mixins
  Sandbox

Bibliothèques */
  Normalize **/
import '../bower_components/normalize-css/normalize';

Utilitaires */
  Variables **/
import 'utilitaires/_variables';
  Mixins **/
import 'utilitaires/_mixins';

Sandbox **/
]
```


Processus de compilation des fichiers Sass

À la fin du moment de la compilation, Sass compilera automatiquement dans le fichier maître **app.css** les fichiers **app.scss** et les fichiers **utilitaires** **utilitaires.scss** importés. Il créera également des fichiers **utilitaires.css** pour chacun des fichiers trouvés dans les fichiers **utilitaires** **utilitaires.scss**.

Importation de fichiers Sass

Pour importer un fichier dans Sass, utilisez l'instruction `@import`.

```
@import "utilitaires/_variables.scss";  
@import "utilitaires/_mixins"; // L'extension .scss est facultative
```

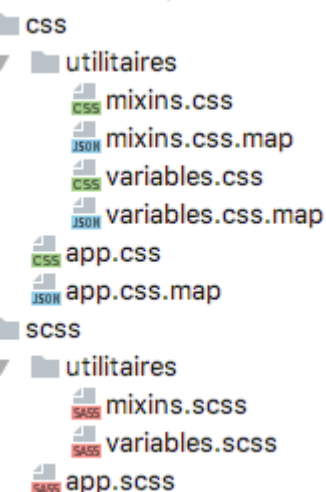
Remarque : l'instruction `@import` utilisée par Sass n'a pas besoin de contenir l'extension `.scss`, et vous pouvez importer plusieurs fichiers à la fois.

```
@import "utilitaires/_variables", "utilitaires/_mixins";
```

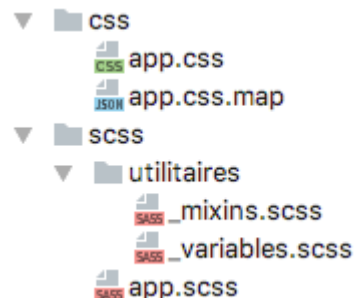
ers fragments (partials)

éviter de créer des fichiers .css inutiles pour chacun des fragments (partials), il faut utiliser u
enclature prévue par Sass. Il suffit d'ajouter un souligné (*underscore*) devant le nom de fichier
exemple, **variables.scss** ou **mixins.scss**.

Résultat sans soulignés



Résultat avec soulignés



de l'importation, il n'est pas nécessaire de mettre le souligné devant le nom du fichier.

```
import "utilitaires/variables", "utilitaires/mixins";
```


exes

Watchers de PHPStorm

-vous remarqué cette petite question qui apparaît dans le haut des fichiers .scss dans Storm?



us cliquez sur le lien File Watcher dans la question, cela ouvrira la fenêtre de paramètres :
Storm > préférences > Tools > File Watchers

s cette fenêtre, utilisez le bouton  pour ajouter un watcher vous permettra de vous passer c
nal puisque PhpStorm lancera lui-même la commande `sass --watch`.

ependant, il est important de paramétrer le *watcher* autrement, car par défaut les fichiers CSS v
éer au même niveau que le fichier .scss plutôt que dans le répertoire distinct /css.

s les champs de saisie *Arguments* et *Output paths to refresh*,
nt le paramètre *\$FileNameWithoutExtension*, ajouter le lien relatif "../css/".

