

# Sass

# Utilisation d'un préprocesseur CSS.

# Intégration web

## Enjeux

- Mettre en page et réaliser le style d'un site,
- Ne pas perdre de temps dans la rédaction du code,
- Écrire un code lisible et facile à maintenir.

## Bonnes pratiques

- Structurer et commenter son code,
- Utiliser des class,
- Gérer la mise en page puis le style.



- Modification de certains valeurs récurrentes malaisée (couleurs par exemple),
- Le document CSS comprend vite 1000 lignes ou plus,
- Beaucoup de répétition d'écriture ou utilisation de nombreuses class sur une même balise.

# Préprocesseur

## Principes

- Génération d'un fichier CSS à partir d'un ou plusieurs fichiers sources,
- Langage semblable au CSS avec des ajouts offrant de nouvelles fonctionnalités,
- Utilisation d'un moteur pour la génération,
- Principaux : Sass, Less et Stylus.

## Éléments

- Imbrication des règles,
- Utilisation de variables,
- Présence d'opérateurs mathématiques,
- Possibilité d'appeler les propriétés d'une class sur la ou les balise(s) ciblée(s),
- Création de fonctions pour automatiser l'écriture.

# Sass – Mise en place

#### Installation

- Windows : installation de l'interpréteur Ruby (déjà installé sur Macintosh).
- Pour tout le monde, installation de Sass :

```
$ gem install sass
```

## Commande pour la compilation du CSS

Commande de base : sass <fichier scss> <fichier css>

```
$ sass styles.css
```

Automatisation: sass --watch <fichier scss>:<fichier css>

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

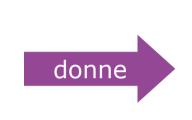
Compression du CSS: sass –style compressed <fichier scss>:<fichier css>

```
$ sass --style compressed styles.scss:styles.css
```

#### **Imbrication**

On indique les propriétés CSS d'un élément HTML à l'intérieur des déclarations du parent.

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



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

## Rappel du parent

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

Dans une imbrication, un parent peut être cité dans une règle imbriquée à l'aide du signe &.

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

#### **Variables**

Le nom des variables débute toujours par \$.

## @import

Afin de conserver des documents de travail 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.

Conventionnellement, on nomme tous les fichiers appelés selon le schéma :

```
_<nom du fichier>.scss
```

On les importe de la façon suivante :

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

# **Opérations**

```
aside{ SCSS width: (100% / 3); }
```

Tous les opérateurs, y compris le modulo, sont disponibles.



```
aside {
    width: 33.3333%;
}
```

#### Fonctions

L'usage de fonction (mixin) permet d'utiliser le même code sans avoir besoin de l'écrire à plusieurs reprises.

```
@mixin border-box{
   -moz-border-radius: 10px;
   -webkit-border-radius: 10px;
   border-radius: 10px;
   border: 2px solid teal;
}
footer{
   @include border-box;
}
```



```
footer{
    -moz-border-radius: 10px;
    -webkit-border-radius: 10px;
    border-radius: 10px;
    border: 2px solid teal;
}
```

## Fonctions (suite)

Comme dans tout langage informatique, on peut utiliser des arguments.

```
@mixin round-border($dimension){
   -moz-border-radius: $dimension;
   -webkit-border-radius: $dimension;
   border-radius: $dimension;
}
footer{
   @include round-border(10px);
}
```

```
donne
```

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

### @extend

Le placeholder @extend permet de réutiliser une déclaration de class dans une autre partie du code.

```
.bordureBase{
   border: #601137 solid;
}
header{
   @extend .bordureBase;
   border-width: 2px;
}
```



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