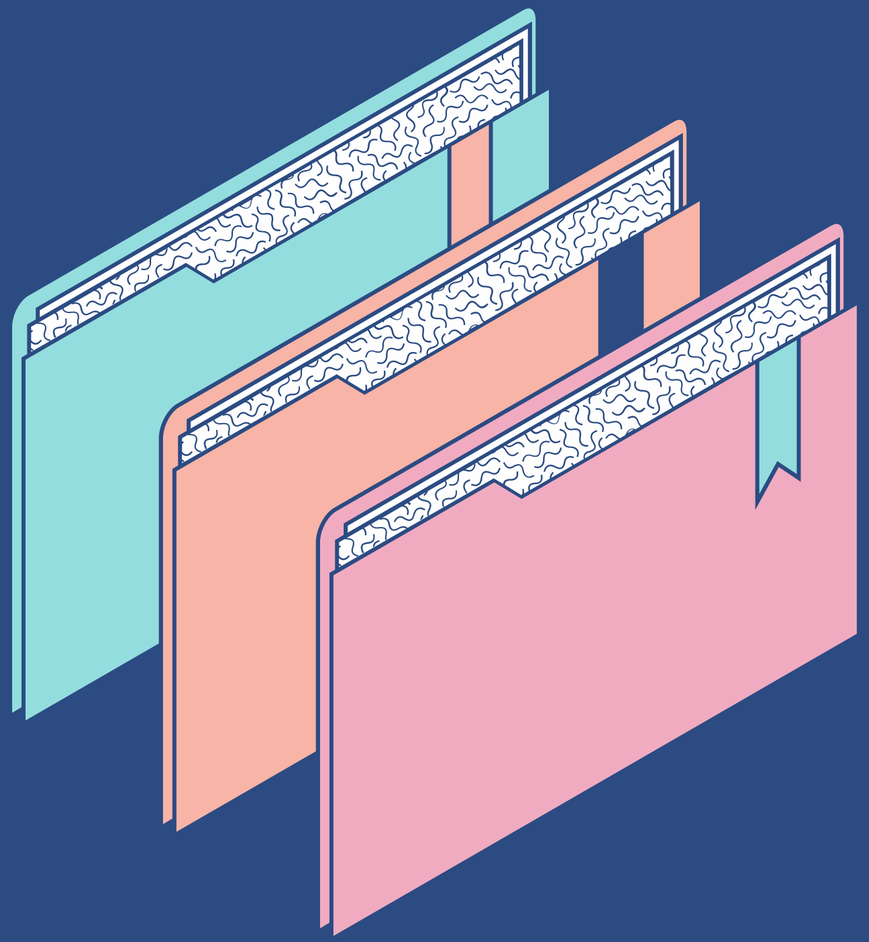
Préprocesseur CSS

Sass

SOMMAIRE



Qu'est ce que SASS ?

- SASS est un "préprocesseur" CSS : Le Sass sera toujours compilé en Css
- C'est un langage outil permettant la simplification, la structuration et la maintenabilité du code CSS
- Le SASS permet d'étendre les possibilités du CSS en ajoutant des variables, fonctions et logique conditionnelle et récursive
- La structure et la syntaxe du SASS est très proche du CSS

Qu'est ce que SASS ?

- Le SASS possède deux syntaxes différentes représentées par les extensions : ".sass" et ".scss"
- Le ".sass", appelé syntaxe indentée est la syntaxe origine, mais est **obsolète aujourd'hui** de part sa complexité d'usage et sa faible compatibilité.
- Le ".scss", qui est une syntaxe plus moderne, et compatible avec l'ensemble des projets CSS.

Qu'est ce que SASS ?

- Le SASS NE peut PAS être interprété par les navigateurs.

Il sera donc nécessaire de compiler nos fichiers .scss en fichiers .css, qui pourront ensuite être lu par les navigateurs.

Installation de SASS

- Il existe de multiples façon de compiler vos fichiers sass en fichiers css.

La façon la plus brute est d'utiliser directement le compilateur sass via lignes de commandes.
De nombreuses applications existent, en fonction de votre environnement, pour simplifier ces opérations.

https://sass-lang.com/install

Installation de SASS : Lignes de commandes

- La façon la plus simple d'installer sass en lignes de commandes est d'utiliser npm.
Npm nécessite l'installation de NodeJs au préalable pour fonctionner

- Il suffit d'exécuter la commande "npm install -g sass" pour installer sass globalement sur votre ordinateur
- Ensuite, vous pouvez compiler votre scss à l'aide de la commande sass :
sass source/stylesheets/index.scss build/stylesheets/index.css

https://sass-lang.com/install

Installation de SASS : Applications

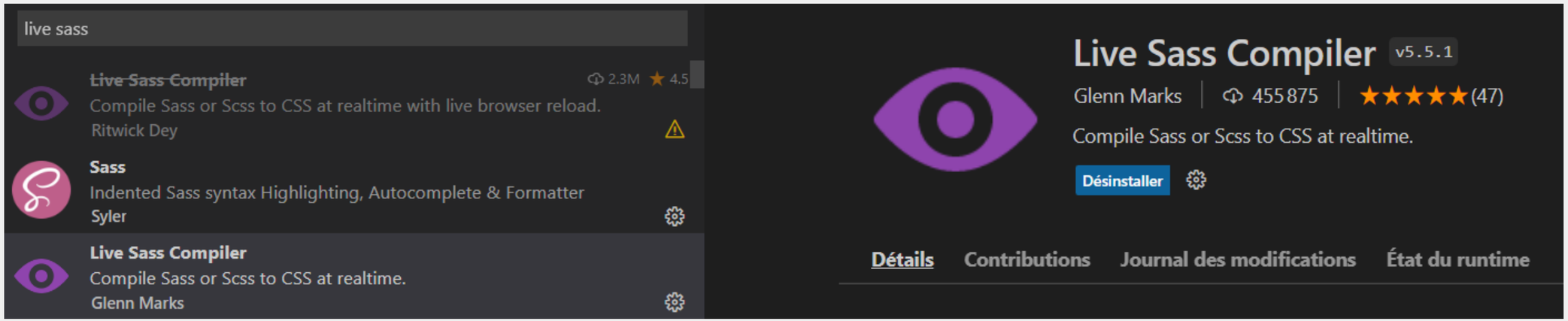
Il existe un grand nombre d'applications permettant de compiler vos fichiers sass.
Des applications existent sur tous les OS, mais la majorité de celles-ci sont payantes:

- CodeKit (Paid) Mac
- Hammer (Paid) Mac
- LiveReload (Paid, Open Source) Mac Windows
- Prepros (Paid) Mac Windows Linux
- Scout-App (Free, Open Source) Windows Linux Mac

https://sass-lang.com/install

Installation de SASS : VisualStudio

Vsc possède une multitude d'extension, dont certaines permettent la compilation de scss en css



Installation de SASS

EXERCICE D'APPLICATION 1 : 15min

- Créer un nouveau projet
- Créez un nouveau dossier "style" dans votre projet, et ajoutez y un fichiers "style.scss"
- Cliquez sur l'option "Watch Sass" dans le footer de vsc

  Vous devriez voir apparaitre deux nouveaux fichiers "style.css" et "style.css.map"
- *Ajoutez le code suivant au sein de votre fichier "style.scss" :*

  Observez le résultat sur "style.css"

```scss
$padding-dimensions: 1rem 2rem 3rem 4rem;
.block {
  padding: $padding-dimensions;
}
```

Découverte des Variables

Pourquoi utiliser des variables ?

L'utilisation de variables permet d'améliorer la clarté et la maintenabilité du code.
Les variables permettent de mutualiser les éléments récurrents.

En sass, les variables se déclarent à l'aide du symbole "$"

```
$main-color: #005183;
```

utilisation des variables

index.html | styles.scss | styles.css

```scss
$main-color: #005183;
$secondary-color: #CCAA11;

body, h1, h2, p, span, a, li{
    color: $main-color;
}
h1{
    text-decoration-line: underline;
    text-decoration-style: solid;
    text-decoration-color:  $secondary-color;
}
```

Définition du thème par variable

Personnalisation des variables

Il est possible de définir une valeur par défaut pour nos variables.
Ces valeurs ne s'appliqueront uniquement si aucune autre valeur n'a été définie pour notre variable.
Cela s'effectue à l'aide du drapeau "!default"

```scss
$syntax-01: 2rem 4rem 6rem 8rem;
$syntax-01: 1rem 2rem 3rem 4rem !default;
.syntax-01 {
    padding: $syntax-01;
}
```

Portée des variables

La portée des variables définit l'espace dans lequel les variables seront accessibles.
Une portée locale sera accessible uniquement dans la propriété en question, une portée globale sera accessible dans l'ensemble de votre fichier.

```scss
$syntax-01: 2em; /* Variable globale */

.syntax-02{
    $syntax-02 :4em; /* Variable locale */
    $syntax-03 :6em !global; /* Variable globale */
}
```

Exercice variables

EXERCICE D'APPLICATION 2 : 30min

- Reprenez le projet créé précédemment
- Ecrivez le doctype du fichiers index.html, et liez le fichier style.css
- Ajoutez du contenu html (titres, paragraphes ...)
- Ajoutez des variables scss pour gérer les marges et padding de vos éléments

Découverte des fonctions

Pourquoi utiliser des fonctions ?

- Une fonction est un bloc de code permettant d'effectuer une suite logique d'instructions.
- Elles servent à améliorer la lisibilité, la maintenance et le temps de développement
- Il existe deux grandes catégories de fonctions, les fonctions **prédéfinies** et les fonctions **personnalisées**

Utilisation des fonctions prédéfinies

- Une fonction prédéfinie est une fonction propre au langage, qui a déjà été codée, et qu'il vous suffit d'utiliser.

- Les fonctions prédéfinies servent à effectuer des tâches génériques.

- Pour les utiliser, il suffit de les appeler par leur nom, et de leur fournir les éventuels paramètres nécessaires.

Utilisez toujours les fonction prédéfinies lorsque vous en avez l'opportunité !

La fonction random()

- La fonction random(X) permet de retourner un nombre aléatoire entre :
  0 et X si X<1
  1 et X si X>1

```
$random-color: rgb(random(255), random(255), random(255));
```

La fonction round()

- La fonction round(X) permet d'arrondir un nombre décimal à l'entier supérieur.
Elle s'utilise majoritairement à la suite d'un calcul pour obtenir une valeur entière.

```
$largeur: 1440;

body, h1, h2, p, span, a, li{
    color: $random-color;
    margin: 0 round($largeur/100)+px;
}
```

Création de fonctions

- Une fonction se construit à l'aide de la règle @function, et de sa définition (nom, arguments et retour)

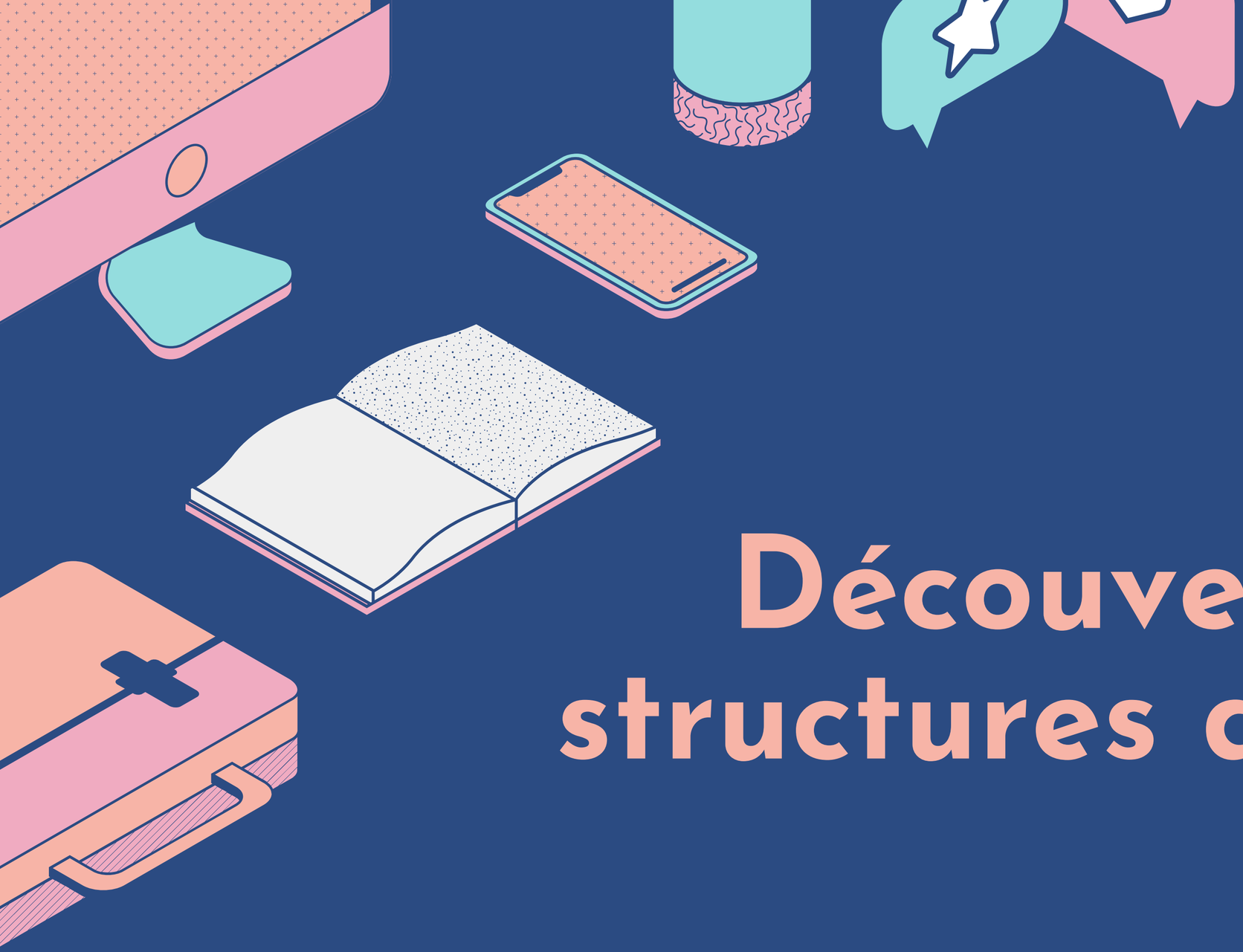
```
@function fois2($nb){
    $resultat: $nb * 2;
    @return $resultat;
}

body, h1, h2, p, span, li{
    color: $main-color;
    margin: 0 fois2(10)+px;
}
```

Exercice fonctions

EXERCICE D'APPLICATION 3 : 30min

- Reprenez le projet créé précédemment
- Ajoutez une variable $width avec une taille par défaut fixée (par exemple 1200px)
- Ecrivez 2 méthodes, permettant de recalculer respectivement les padding et les marges de vos éléments en fonction de la taille de votre fenêtre :
  - marges = width / 100
  - padding = width / 50



Découverte des structures de contrôle

Pourquoi utiliser des structures de contrôle ?

- Une structure de contrôle permet d'exécuter une instruction si une condition est vérifiée, et de la répéter un certain nombre de fois.

- Elles permettent, elles aussi ,de rendre votre code plus lisible, plus maintenable et plus optimisé.

Les structures de contrôle

- Sass met à disposition 4 règles permettant de mettre en place des structures de contrôle :

  - **@if et @else** permettent d'exécuter un bloc de code SI (if) une condition est remplie ou d'exécuter un autre code SINON (else) ;
  - **@each** évalue un bloc pour chaque élément d'une liste
  - **@for** évalue un bloc un nombre de fois précisé lors de la création de la règle (le nombre d'évaluation est connu à l'avance) ;
  - **@while** évalue un bloc jusqu'à ce qu'une certaine condition de sortie soit remplie (le nombre d'évaluation n'est pas connu à l'avance).

@if / @else

```
@if($light-theme){
    color: $dark;
    background-color: $light;
}@else{
    color: $light;
    background-color: $dark;
}
```

@while


```scss
$width: 1000;
$margin: 0;

@while($margin < $width/50){
    $margin: $margin + 1; //Ajoute 1 à la valeur précédente de $margin
}
```

Attention : Votre boucle while doit TOUJOURS se finir.

@each

SCSS Sass

```scss
$sizes: 40px, 50px, 80px;

@each $size in $sizes {
  .icon-#{$size} {
    font-size: $size;
    height: $size;
    width: $size;
  }
}
```

CSS

```css
.icon-40px {
  font-size: 40px;
  height: 40px;
  width: 40px;
}

.icon-50px {
  font-size: 50px;
  height: 50px;
  width: 50px;
}

.icon-80px {
  font-size: 80px;
  height: 80px;
  width: 80px;
}
```

@for

```scss
$base-color: #036;

@for $i from 1 through 3 {
  ul:nth-child(3n + #{$i}) {
    background-color: lighten($base-color, $i * 5%);
  }
}
```

Exercice structures de contrôle

EXERCICE D'APPLICATION 4 : 30min

- Reprenez le projet créé précédemment
- Remplacer vos 2 méthodes permettant de redimensionner vos padding et vos margin par une méthode unique resize()

Indice : la méthode resize devra prendre en paramètre le type de donnée à redimensionner

Vous devez utiliser des conditions au sein de votre méthode

Découverte des opérateurs

Opérateurs de concaténation

Ces opérateurs permettent de fusionner plusieurs chaînes de caractères :

- \+ : Retourne une chaine qui contient les deux expressions de départ concaténées
- \- : Retourne une chaine contenant les deux expressions de départ concaténées et séparées par "-"
- / : Retourne une chaine contenant les deux expressions de départ concaténées et séparées par "/"

Opérateurs de concaténation

```php
$str1 : "12";
$str2 : "34";

$str3 : $str1+$str2; /* 1234 */
$str4 : $str1-$str2; /* 12-34 */
$str5 : $str1/$str2; /* 12/34 */
```

Opérateurs arithmétiques

- Ces opérateurs permettent d'effectuer des calculs entre nombres à l'intérieur de la méthode calc() :

  - \+ : Addition
  - \- : Soustraction
  - * : Multiplication
  - / : Division
  - % : Modulo (reste d'une division euclidienne)

Opérateurs arithmétiques

```scss
$nb3 : calc($nb1+$nb2); /* 46 */
$nb4 : calc($nb1-$nb2); /* -22 */
$nb5 : calc($nb1*$nb2); /* 408 */
$nb6 : calc($nb1/$nb2); /* 0.35... */
$nb7 : calc($nb1%$nb2); /* 0 */
```

Opérateurs de comparaison

- Ces opérateurs permettent de comparer des valeurs entre elles :
  - == : Permet de tester l'égalité sur les valeurs (renvoie true si les valeurs sont égales)
  - != : Permet de tester la différence des valeurs (renvoie true si les valeurs sont différentes)
  - < : Permet de tester si une valeur est strictement inférieure à une autre
  - > : Permet de tester si une valeur est strictement supérieure à une autre
  - <= : Permet de tester si une valeur est inférieure ou égale à une autre
  - >= : Permet de tester si une valeur est supérieure ou égale à une autre
  -

Opérateurs de comparaison

```scss
$nb1 : 12;
$nb2 : 34;

@if($nb1 == $nb2){
    $str: "equals";
}
@else if($nb1 != $nb2) {
    $str: "differents";
}
@else if($nb1 < $nb2) {
    $str: "inferieur strict";
}
@else if($nb1 > $nb2) {
    $str: "superieur strict";
}
@else if($nb1 <= $nb2) {
    $str: "inferieur ou égal";
}
@else if($nb1 >= $nb2) {
    $str: "superieur ou égal";
}
```

Opérateurs logiques

- Ces opérateurs permettent de créer des conditions plus complexes.
- Il existe 3 types d'opérateurs logiques :
  - and : renvoie ture si chaque comparaison renvoie true
  - or : renvoie true si l'une des comparaison renvoie true
  - not : inverse le résultat logique d'une comparaison

Opérateurs logiques

```
@if(($nb1 < 10) and ($nb1<$nb2)){ /* Return false */

}

@if(($nb1 < 10) or ($nb1<$nb2)){ /* Return true */

}

@if(not($nb1 < 10) and ($nb1<$nb2)){ /* Return true */

}
```

Imbrication de sélecteurs

Imbrication de sélecteurs

- Afin de gagner en lisibilité, sass permet l'imbrication des sélecteurs de manière analogue aux balises html :

```scss
ul{
    list-style: none;
    li{
        color: aqua;
    }
}
```

SCSS

```css
ul {
  list-style: none;
}
ul li {
  color: aqua;
}
```

CSS

Imbrication de sélecteurs

- Il existe un sélecteur parent "&", permettant de faire référence au sélecteur exterieur

```scss
ul{
    list-style: none;
    &:hover{
        color: antiquewhite;
    }
    li{
        color: aqua;
    }
}
```

SCSS

```css
ul {
  list-style: none;
}
ul:hover {
  color: antiquewhite;
}
ul li {
  color: aqua;
}
```

CSS