Initiation à SASS

A installer dans VS Code

Easy Sass

It's about time someone created a standalone extension to compile SASS/SCSS files in Visual Studio Code. Don't you think? Automatically compiles SASS/SCSS files to .css and





Site de SASS

Sass: Syntactically Awesome Style Sheets

Syntatically Awesome Style Sheets

https://sass-lang.com/

Ce que nous allons faire



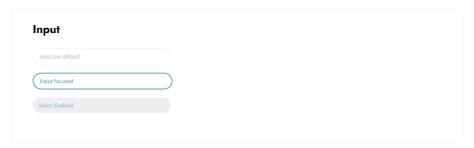
Gestion des tailles d'icônes



Gestion des titres



Gestion des composants



Cet exercice nous a permis d'aborder :

La nested syntaxe, les partials, diférentes @rules (<code>@extends</code> , <code>@mixins</code>), les media queries, les variables, les boucles et l'utilisation de quelques fonctions bien sympathiques !

Ce que nous allons utiliser

Variables

Variables

Sass variables are simple: you assign a value to a name that begins with \$, and then you can refer to that name instead of the value itself. But despite their simplicity, they're one of the most useful tools Sass brings to the table.



https://sass-lang.com/documentation/variables

Nested syntaxe

Sass Basics

Before you can use Sass, you need to set it up on your project. If you want to just browse here, go ahead, but we recommend you go install Sass first. Go here if you want to learn how to get everything setup.



figure | https://sass-lang.com/guide#topic-3

Partials

Sass Basics

Before you can use Sass, you need to set it up on your project. If you want to just browse here, go ahead, but we recommend you go install Sass first. Go here if you want to learn how to get everything



https://sass-lang.com/guide#topic-4

Mixins

@mixin and @include

Mixins allow you to define styles that can be re-used throughout your stylesheet. They make it easy to avoid using non-semantic classes like .float-left, and to distribute collections of styles in libraries. Mixins are defined using the @mixin at-rule, which is written @mixin { ... } or @mixin name() { ...



https://sass-lang.com/documentation/at-rules/mixin

Extend (héritage)

@extend

There are often cases when designing a page when one class should have all the styles of another class, as well as its own specific styles. For example, the BEM methodology encourages modifier classes that go on the same elements as block or element classes.



file | https://sass-lang.com/documentation/at-rules/extend

Boucle for

@for

The @for rule, written @for from to { ... } or @for from through { ... }, counts up or down from one number (the result of the first expression) to another (the result of the second) and evaluates a block for each number in between.



for https://sass-lang.com/documentation/at-rules/control/for

Boucle foreach

@each

The @each rule makes it easy to emit styles or evaluate code for each element of a list or each pair in a map. It's great for repetitive styles that only have a few variations between them. It's usually written @each in $\{ \dots \}$, where the expression returns a list.



fig. https://sass-lang.com/documentation/at-rules/control/each