


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


 <https://marketplace.visualstudio.com/items?itemName=pook.easysass>



Site de SASS

Sass: Syntactically Awesome Style Sheets

Syntactically Awesome Style Sheets

 <https://sass-lang.com/>

Ce que nous allons faire

INITIATION À SASS : DESIGN SYSTEM

Gestion des tailles d'icônes

 .icon-16px

 .icon-24px

 .icon-32px

 .icon-48px

 .icon-64px

Gestion des titres

Ceci est un h1 Jost - bold - 35px

Ceci est un h2 Jost - bold - 31px

Ceci est un h3 Jost - bold - 27px

Ceci est un h4 Jost - bold - 23px

Ceci est un h5 Jost - bold - 20px

Ceci est un h6 Jost - bold - 16px

Gestion des composants

Input

Input par défaut

Input focused

Input disabled

Cet exercice nous a permis d'aborder :


La nested syntaxe, les partials, différentes @rules (`@extends` , `@mixins`), 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.

 <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 setup.

 <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.

 <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.

 <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 { ... }, where the expression returns a list.

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