

SASS, CSS with superpowers

Sass est le langage d'extension CSS professionnel le plus mature, le plus stable et le plus puissant au monde.

SASS BASICS

SASS est un préprocesseur CSS qui permet de créer des feuilles de style dynamiques en utilisant des variables, des fonctions et des mixins.


[Documentation Officielle](#)

Preprocessor

Les feuilles de style CSS peuvent être amusantes, mais lorsqu'elles deviennent de plus en plus volumineuses.

Elles sont souvent difficiles à comprendre et difficiles à maintenir.

C'est là qu'un préprocesseur peut être utile.

Sass possède des fonctionnalités qui n'existent pas encore en CSS, comme l'imbrication ( depuis 2024, CSS a la capacité d'imbrication), les mixins, l'héritage et d'autres trucs astucieux qui vous aident à écrire des CSS robustes et faciles à maintenir. Une fois que vous aurez commencé à bricoler avec Sass, il prendra votre fichier Sass prétraité et l'enregistrera comme un fichier CSS normal que vous pourrez utiliser dans votre site web.

La façon la plus directe d'y parvenir est de le faire dans votre terminal.

Une fois Sass installé, vous pouvez compiler votre Sass en CSS en utilisant la commande `sass`. Vous devrez indiquer à Sass le fichier à partir duquel il doit être compilé, ainsi que l'emplacement de sortie du CSS.

Par exemple, l'exécution de la commande `sass input.scss output.css` depuis votre terminal prend un seul fichier Sass, `input.scss`, et le compilerait en `output.css`.

Vous pouvez également surveiller des fichiers ou des répertoires individuels à l'aide de l'option `--watch`.

L'option `watch` indique à Sass de surveiller les modifications apportées à vos fichiers sources et de recompiler CSS à chaque fois que vous sauvegardez votre Sass. Si vous voulez surveiller (au lieu de compiler manuellement) votre fichier `input.scss`, il vous suffit d'ajouter l'option `watch` à votre commande, comme suit :

```
sass --watch input.scss output.css
```

Vous pouvez regarder et sortir des répertoires en utilisant des chemins de dossiers comme entrée et sortie, et en les séparant par deux points. Dans cet exemple :

```
sass --watch app/sass:public/stylesheets
```

Sass surveille tous les fichiers du dossier app/sass pour détecter les modifications et compile les feuilles de style CSS dans le dossier public/stylesheets.

💡 Sass a deux syntaxes !

La syntaxe SCSS (.scss) est la plus utilisée. Il s'agit d'un surensemble de CSS, ce qui signifie que tous les CSS valides sont également des SCSS valides.

La syntaxe indentée (.sass) est plus inhabituelle : elle utilise l'indentation plutôt que les accolades pour imbriquer les instructions, et les nouvelles lignes plutôt que les points-virgules pour les séparer. Tous nos exemples sont disponibles dans les deux syntaxes

Variables

Les variables sont un moyen de stocker des informations que vous souhaitez réutiliser dans votre feuille de style. Vous pouvez stocker des couleurs, des piles de polices ou toute autre valeur CSS que vous pensez vouloir réutiliser.

Sass utilise le symbole **\$** pour déclarer une variable.

En voici un exemple :

```
$font-stack: Helvetica, sans-serif;  
$primary-color: #333;  
  
body {  
  font: 100% $font-stack;  
  color: $primary-color;  
}
```

Nesting

Lorsque vous écrivez du HTML, vous avez probablement remarqué qu'il possède une hiérarchie visuelle et imbriquée claire. Sass vous permettra d'imbriquer vos sélecteurs CSS de manière à suivre la même hiérarchie visuelle que votre HTML. Sachez que des règles *trop imbriquées se traduiront par un CSS surqualifié* qui pourrait s'avérer difficile à maintenir et qui est généralement considéré comme une mauvaise pratique.

Dans cette optique, voici un exemple de styles typiques pour la navigation d'un site :

En Sass

```
nav {  
  ul {  
    margin: 0;  
    padding: 0;  
    list-style: none;  
  }  
  
  li { display: inline-block; }  
  
  a {  
    display: block;  
    padding: 6px 12px;  
    text-decoration: none;  
  }  
}
```

En CSS

```
nav ul {  
  margin: 0;  
  padding: 0;  
  list-style: none;  
}  
nav li {  
  display: inline-block;  
}  
nav a {  
  display: block;  
  padding: 6px 12px;  
  text-decoration: none;  
}
```

Vous remarquerez que les sélecteurs `ul`, `li` et `a` sont imbriqués dans le sélecteur `nav`. C'est un excellent moyen d'organiser votre CSS et de le rendre plus lisible.

Partials

Vous pouvez créer des fichiers Sass partiels qui contiennent de petits extraits de CSS que vous pouvez inclure dans d'autres fichiers Sass.

C'est un excellent moyen de modulariser votre CSS et d'en faciliter la maintenance.

Un **partial** est un fichier Sass nommé avec *un trait de soulignement*. Vous pouvez le nommer quelque chose comme `_partial.scss`. Le trait de soulignement indique à Sass que le fichier n'est qu'un fichier partiel et qu'il ne doit pas être généré dans un fichier CSS. Les fichiers partiels de Sass sont utilisés avec la règle `@use` équivalent de `@import` en CSS.

Modules

Vous n'êtes pas obligé d'écrire tout votre Sass dans un seul fichier.

Vous pouvez le diviser comme vous le souhaitez avec la règle `@use`.

Cette règle charge un autre fichier Sass en tant que module, ce qui signifie que vous pouvez faire **référence à ses variables, mixins et fonctions** dans votre fichier Sass avec un namespace basé sur le nom du fichier.

L'utilisation d'un fichier inclura également le CSS qu'il génère dans votre sortie compilée !

```
// _base.scss
$font-stack: Helvetica, sans-serif;
$primary-color: #333;

body {
  font: 100% $font-stack;
  color: $primary-color;
}

// styles.scss
@use 'base';

.inverse {
  background-color: base.$primary-color;
  color: white;
}

body {
  font: 100% Helvetica, sans-serif;
  color: #333;
}

.inverse {
  background-color: #333;
  color: white;
}
```

Mixins

Certaines choses en CSS sont un peu fastidieuses à écrire, en particulier avec CSS3/4 et les nombreux **préfixes de fournisseurs** qui existent.

Un mixin vous permet de créer des **groupes de déclarations CSS** que vous souhaitez réutiliser dans votre site.

Cela permet de garder votre Sass très *DRY*.

Vous pouvez même passer des valeurs pour rendre votre mixin plus flexible.

Voici un exemple pour le thème officiel du site SASS :

[playground sass](#)

Pour créer un mixin, il faut utiliser la directive *@mixin* et lui donner un nom.

Nous avons nommé notre mixin theme.

Nous utilisons également la variable \$theme à l'intérieur des parenthèses afin de pouvoir passer un thème de notre choix.

Après avoir créé votre mixin, vous pouvez l'utiliser comme déclaration CSS en commençant par *@include* suivi du nom du mixin.

Inheritance

L'utilisation de *@extend* vous permet de partager un ensemble de propriétés CSS d'un sélecteur à un autre.

Dans notre exemple, nous allons créer une série simple de messages pour les erreurs, les avertissements et les succès en utilisant une autre fonctionnalité qui va de pair avec extend, les classes placeholder. Une classe d'espace réservé est un type spécial de classe qui ne s'imprime que lorsqu'elle est étendue, et qui peut aider à garder votre CSS compilé propre et net.

exemple:

[Extend / Inheritance](#)

Le code ci-dessus indique à **.message**, **.success**, **.error** et **.warning** de se comporter comme %message-shared. Cela signifie que partout où %message-shared apparaît, .message, .success, .error et **.warning** apparaissent également.

La magie s'opère dans le CSS généré, où chacune de ces classes aura les mêmes propriétés CSS que **%message-shared**, ce qui permet d'éviter d'avoir à écrire plusieurs noms de classes sur les éléments HTML.

Cela vous évite d'avoir à écrire plusieurs noms de classes sur les éléments HTML. Vous pouvez étendre la plupart des sélecteurs CSS simples en plus des classes à espaces réservés (placeholder) dans Sass, mais l'utilisation d'espaces réservés est le moyen le plus simple de vous assurer que vous n'étendez pas une classe qui est imbriquée ailleurs dans vos styles, ce qui peut entraîner des sélecteurs non intentionnels dans votre CSS.

Notez que le CSS dans **%equal-heights** n'est pas généré, parce que **%equal-heights** n'est jamais étendu.

Operators

Il est très utile d'effectuer des calculs mathématiques dans votre CSS.

Sass dispose d'une poignée d'opérateurs mathématiques standard comme `+`, `-`, `*`, `math.div()` et `%`.

Dans notre exemple, nous allons effectuer quelques calculs mathématiques simples pour calculer les largeurs d'un html `article` et d'une `aside`.

exemple

Nous avons créé une grille fluide très simple, basée sur 960px.

Les opérations de Sass nous permettent de prendre des valeurs en pixels et de les convertir en pourcentages sans trop de difficultés.

author: [Benoit Lepage](#)

traduction de la documentation officielle : [Sass](#)