Preprocesseurs

SCSS

Introduction à SCSS (Sass)

SCSS, ou Sassy CSS, est une syntaxe de Sass (Syntactically Awesome Stylesheets), un préprocesseur CSS qui permet d'écrire des feuilles de style de manière plus efficace et maintenable. SCSS est conçu pour être une surcouche de CSS, ce qui signifie que tout code CSS valide est également un code SCSS valide.

Différence entre Sass et SCSS

Sass utilise une syntaxe sans accolades ni points-virgules, ce qui peut sembler plus propre pour certains développeurs

SCSS utilise une syntaxe similaire à CSS, avec des accolades et des points-virgules, ce qui facilite l'adoption pour ceux qui connaissent déjà bien CSS

Différence entre Sass et SCSS

SAAS

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

SCSS

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

Origines

SCSS a été créé pour améliorer la gestion et l'écriture des feuilles de style CSS, en ajoutant des fonctionnalités qui ne sont pas disponibles en CSS natif, comme les variables, les mixins, l'imbrication de règles, et bien d'autres.

Utilité de SCSS

SCSS offre plusieurs avantages significatifs par rapport au CSS traditionnel, ce qui le rend extrêmement utile pour les développeurs web.

Avantages de SCSS

Modularité : Permet de diviser les fichiers CSS en plusieurs morceaux plus gérables, appelés partiels, qui peuvent être importés dans un fichier principal.

Réutilisation du code : Grâce aux variables, mixins et fonctions, SCSS permet de réutiliser le code et d'éviter les répétitions. Par exemple, une couleur ou une taille de police peut être définie une fois dans une variable et utilisée partout dans le projet.

Cas d'utilisation

SCSS est utilisé dans de nombreux projets web pour gérer les styles CSS de manière plus efficace. Voici quelques exemples concrets :

- **Sites web complexes** : Pour les sites avec de nombreux composants et styles, SCSS aide à organiser et à gérer les styles de manière modulaire.
- **Applications web** : Dans les applications web où les styles peuvent changer fréquemment, SCSS permet une maintenance et une mise à jour plus faciles.
- **Frameworks CSS** : De nombreux frameworks CSS, comme Bootstrap, sont construits en utilisant SCSS pour offrir une personnalisation et une extensibilité faciles.

Installation et Utilisation de SCSS

Pré-requis pour l'installation

Avant de pouvoir utiliser SCSS, vous devez vous assurer d'avoir les outils nécessaires installés sur votre ordinateur :

- **Node.js et npm** : Node.js est un environnement d'exécution JavaScript, et npm (Node Package Manager) est le gestionnaire de paquets qui vient avec Node.js. Vous pouvez les télécharger et les installer à partir du site officiel Node.js.
- **Un éditeur de texte** : Utilisez un éditeur de texte moderne comme Visual Studio Code, Sublime Text, ou Atom. Ces éditeurs offrent des fonctionnalités qui facilitent l'écriture de SCSS, comme la coloration syntaxique et l'autocomplétion.

Processus d'installation

Une fois les pré-requis installés, vous pouvez passer à l'installation de Sass via npm.

- Ouvrez votre terminal ou invite de commande.
- Exécutez la commande suivante pour installer Sass globalement sur votre système :

```
npm install -g sass
```

Vérification de l'installation

Après l'installation, vérifiez que Sass est correctement installé en exécutant la commande suivante :

```
sass --version
```

Cette commande devrait afficher la version de Sass installée.

Configuration initiale

Création d'un fichier SCSS

Dans votre éditeur de texte, créez un nouveau fichier avec l'extension .scss. Par exemple, style.scss

Ajoutez quelques styles simples pour tester

exemple slide suivant ==>

Configuration initiale

```scss
$primary-color: #333;

body {
  font-family: Arial, sans-serif;
  color: $primary-color;
}

h1 {
  color: lighten($primary-color, 20%);
}
```

Compilation du fichier SCSS

Pour compiler votre fichier SCSS en CSS, utilisez la commande suivante dans le terminal

```
sass style.scss style.css
```

Cette commande prend le fichier style.scss en entrée et génère un fichier style.css.

Compilation et Utilisation de SCSS

L'une des principales tâches lors de l'utilisation de SCSS est la compilation des fichiers SCSS en CSS. Voici comment procéder.

La compilation manuelle se fait en utilisant la commande sass

```
sass input.scss output.css
```

```
sass style.scss style.css
```

Cette commande convertit le fichier style.scss en style.css

Surveillance des fichiers

Pour faciliter le développement, vous pouvez utiliser la fonctionnalité de surveillance des fichiers. Cela permet de compiler automatiquement votre fichier SCSS chaque fois qu'il est modifié.

```
sass --watch style.scss:style.css
```

Avec cette commande, chaque modification apportée à style.scss sera automatiquement reflétée dans style.css.

Exemple pratique

Création d'un fichier SCSS simple :

- Créez un fichier nommé example.scss avec le contenu suivant :

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

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

.container {
  width: 80%;
  margin: 0 auto;
}

nav {
  background-color: darken($primary-color, 10%);
  ul {
    list-style: none;
```

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

.container {
  width: 80%;
  margin: 0 auto;
}

nav {
  background-color: darken($primary-color, 10%);
  ul {
    list-style: none;
    padding: 0;
  }
  li {
    display: inline-block;
    margin-right: 10px;
  }
  a {
    color: lighten($primary-color, 20%);
    text-decoration: none;
    &:hover {
      text-decoration: underline;
    }
  }
}
```

Compilation et vérification

Compilez ce fichier en utilisant la commande sass

```
sass example.scss example.css
```

Ouvrez example.css pour vérifier le CSS généré.
Activez la surveillance pour le fichier example.scss

```
sass --watch example.scss:example.css
```

Modifiez le fichier example.scss et observez comment example.css est automatiquement mis à jour.

Principes Fondamentaux de SCSS

Variables

Les variables en SCSS permettent de stocker des valeurs réutilisables telles que des couleurs, des tailles de police, des marges, etc. Elles facilitent la maintenance et la modification des styles.

Définition et Utilité

Les variables sont définies avec le symbole $ suivi du nom de la variable. Elles permettent de centraliser les valeurs qui peuvent être utilisées à plusieurs endroits dans le fichier SCSS, ce qui simplifie les mises à jour.

Principes Fondamentaux de SCSS

Syntaxe

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

body {
  color: $primary-color;
  font-family: $font-stack;
}
```

Principes Fondamentaux de SCSS

Exemple pratique

Thèmes de couleur

```scss
$background-color: #f0f0f0;
$text-color: #333;

body {
  background-color: $background-color;
  color: $text-color;
}
```

Tailles de police

```scss
$base-font-size: 16px;
$heading-font-size: 2em;

body {
  font-size: $base-font-size;
}

h1 {
  font-size: $heading-font-size;
}
```

Principes Fondamentaux de SCSS

Imbrication

L'imbrication permet d'organiser le code CSS de manière hiérarchique, reflétant la structure HTML. Cela rend le code plus lisible et maintenable.

Définition et Utilité

L'imbrication en SCSS permet de regrouper des styles associés sous un même parent, ce qui rend le code plus propre et plus facile à lire.

Principes Fondamentaux de SCSS

Syntaxe

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

Principes Fondamentaux de SCSS

Exemple pratique

Navigation

```scss
nav {
  background-color: #333;
  ul {
    list-style: none;
    margin: 0;
    padding: 0;
  }
  li {
    display: inline;
  }
  a {
    color: white;
    text-decoration: none;
    padding: 10px;
    &:hover {
      background-color: #444;
    }
  }
}
```

Formulaires

```scss
form {
  margin: 20px 0;
  label {
    display: block;
    margin-bottom: 5px;
  }
  input, textarea {
    width: 100%;
    padding: 8px;
    margin-bottom: 10px;
    border: 1px solid #ccc;
  }
}
```

Principes Fondamentaux de SCSS

Mixins

Les mixins permettent de créer des blocs de styles réutilisables qui peuvent être inclus dans d'autres règles de style.

Définition et Utilité

Les mixins sont utilisés pour encapsuler des styles CSS réutilisables. Ils permettent de passer des arguments pour rendre les styles plus dynamiques.

Principes Fondamentaux de SCSS

Syntaxe

```scss
@mixin border-radius($radius) {
  -webkit-border-radius: $radius;
     -moz-border-radius: $radius;
          border-radius: $radius;
}

.box { @include border-radius(10px); }
```

Principes Fondamentaux de SCSS

Exemple pratique

Ajout de préfixes

```scss
@mixin transform($property) {
  -webkit-transform: $property;
  -ms-transform: $property;
  transform: $property;
}

.rotate {
  @include transform(rotate(45deg));
}
```

Réutilisation de styles

```scss
@mixin button($color, $background) {
  color: $color;
  background-color: $background;
  padding: 10px 20px;
  border: none;
  border-radius: 5px;
  &:hover {
    background-color: darken($background, 10%);
  }
}

.btn-primary {
  @include button(white, blue);
}

.btn-secondary {
  @include button(black, gray);
}
```

Principes Fondamentaux de SCSS

Héritage

L'héritage permet de partager des styles entre des sélecteurs en utilisant l'extension @extend.

Définition et Utilité

L'héritage via @extend permet de partager un ensemble de styles entre plusieurs sélecteurs sans avoir à les répéter. Cela améliore la maintenabilité du code en réduisant la duplication.

Principes Fondamentaux de SCSS

Syntaxe

```scss
%message-shared {
  border: 1px solid #ccc;
  padding: 10px;
  color: #333;
}

.success { @extend %message-shared; border-color: green; }
.error { @extend %message-shared; border-color: red; }
```

Principes Fondamentaux de SCSS

Exemple pratique

Messages d'alerte

```scss
%alert {
  padding: 20px;
  border-radius: 5px;
  margin-bottom: 20px;
}

.alert-success {
  @extend %alert;
  background-color: #dff0d8;
  border-color: #d6e9c6;
  color: #3c763d;
}

.alert-danger {
  @extend %alert;
  background-color: #f2dede;
  border-color: #ebccd1;
  color: #a94442;
}
```

Boutons

```scss
%button-base {
  display: inline-block;
  padding: 10px 20px;
  text-align: center;
  border-radius: 5px;
  text-decoration: none;
}

.btn-primary {
  @extend %button-base;
  background-color: blue;
  color: white;
}

.btn-secondary {
  @extend %button-base;
  background-color: gray;
  color: black;
}
```

Principes Fondamentaux de SCSS

Fonctions

Les fonctions en SCSS permettent de créer des fonctions personnalisées pour calculer des valeurs dynamiques.

Définition et Utilité

Les fonctions permettent de créer des calculs dynamiques et de retourner des valeurs qui peuvent être utilisées dans d'autres styles. Elles améliorent la flexibilité et la réutilisabilité du code.

Principes Fondamentaux de SCSS

Syntaxe

```scss
@function calculate-rem($size) {
  @return $size / 16px * 1rem;
}

.container {
  font-size: calculate-rem(32px);
}
```

Principes Fondamentaux de SCSS

Exemple pratique

Conversion d'unités

```scss
@function em($pixels, $base-font-size: 16px) {
  @return $pixels / $base-font-size * 1em;
}

.container {
  font-size: em(18px);
}
```

Couleurs dynamiques

```scss
@function shade-color($color, $percent) {
  @return mix(black, $color, $percent);
}

.box {
  background-color: shade-color(#ff0000, 20%);
}
```

L’architecture 7 - 1

L'architecture 7-1 est une approche bien connue pour organiser les fichiers SCSS de manière efficace et maintenable. Cette méthode divise les styles en différents types de fichiers pour une meilleure organisation et une gestion simplifiée.

L'architecture 7-1 utilise une structure de dossiers bien définie pour organiser les fichiers SCSS. Voici à quoi cela ressemble généralement :

- base/ : Contient les styles de base qui s'appliquent à l'ensemble du projet, tels que les réinitialisations CSS, les variables globales et les mixins de base.
- components/ : Contient les styles spécifiques aux composants réutilisables de l'interface utilisateur, tels que les boutons, les formulaires, les cartes, etc.
- layout/ : Contient les styles de mise en page globale de l'application, tels que les grilles, les en-têtes, les pieds de page, etc.
- pages/ : Contient les styles spécifiques aux pages individuelles de l'application.
- themes/ : Contient les styles spécifiques à différents thèmes de l'application, tels que les styles sombres et clairs.
- abstracts/ : Contient les styles abstraits, tels que les variables, les mixins et les fonctions.
- vendors/ : Contient les styles provenant de bibliothèques tierces ou de plugins.

Avantages de l'architecture 7-1

- **Organisation claire** : La structure de dossier bien définie rend les fichiers SCSS faciles à trouver et à gérer.
- **Réutilisation du code** : La séparation des styles en différents fichiers encourage la réutilisation des styles et réduit la duplication.
- **Maintenance simplifiée** : En divisant les styles en modules distincts, les mises à jour et les modifications deviennent plus simples et moins risquées.
- **Extensibilité** : La structure modulaire facilite l'ajout de nouvelles fonctionnalités ou la modification de l'apparence sans affecter le reste du code.

Annexe architecture

```
styles/
│
├── base/
│   ├── _reset.scss
│   ├── _variables.scss
│   └── ...
│
├── components/
│   ├── _buttons.scss
│   ├── _forms.scss
│   └── ...
│
├── layout/
│   ├── _grid.scss
│   ├── _header.scss
│   ├── _footer.scss
│   └── ...
│
├── pages/
│   ├── _home.scss
│   ├── _about.scss
│   └── ...
│
├── themes/
│   ├── _dark-theme.scss
│   ├── _light-theme.scss
│   └── ...
│
├── abstracts/
│   ├── _variables.scss
│   ├── _mixins.scss
│   ├── _functions.scss
│   └── ...
│
└── vendors/
    ├── _bootstrap.scss
    ├── _normalize.scss
    └── ...
```