Sass

Pierre Bretéché

Dawan

pbreteche@dawan.fr

March 30, 2021

Overview

Introduction

Principe

Préprocesseur pour feuilles de style
Offre des fonctionnalités supplémentaires telles:

- variables
- mixins
- imbrication
- héritage

Philosophie

Arrivé en 2006 pour s'intégrer dans Rails
Applique le principe du «codage sec» : "DRY" (Don't Repeat Yourself)
Initialement, syntaxe similaire à Haml (indented syntax) pour Sass,
Puis SCSS (Sassy CSS) avec accolades et point-virgules;

Installation

Aujourd'hui, le plus souvent en tant que module node:

```
$ npm install -g sass
```

Historiquement en tant que paquet Ruby:

```
$ gem install sass
```

Mais aussi, directement avec le gestionnaire de paquet du SE:

```
$ apt|yum|brew|choco|whatever install sass
```

Compilation

Un fichier, une fois

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

Un fichier, re-compilé automatiquement

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

Un répertoire, re-compilé automatiquement:

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

Syntaxe de base

Variables

Permet de factoriser des valeurs, et les réutiliser. Utile quand:

- une même valeur est utilisée **volontairement** plusieurs fois (et non par coïncidence)
- cette valeur peut être amenée à changer à l'avenir

Variables

En Sass

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

body
  font: 100% $font-stack
  color: $primary-color
```

En SCSS (syntaxe utilisée pour la suite de la formation)

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

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

Imbrication

```
nav {
  ul {
    margin: 0;
    padding: 0;
    list-style: none;
  }

  li { display: inline-block; }

  a {
    display: block;
    padding: 6px 12px;
    text-decoration: none;
  }
}
```

Scoping

Les variables ont pour portée le groupe dans lequel elles sont déclarées.

```
$color: red;
h1 {
        color: $color
}

p {
        $color: blue;
        color: $color;
}
```

Référence parent

Possibilité de faire référence au parent

```
a {
  font-weight: bold;
  text-decoration: none;
  &:hover { text-decoration: underline; }
  body.firefox & { font-weight: normal; }
}
```

```
#main {
  color: black;
  a {
    font-weight: bold;
    &:hover { color: red; } /* #main a:hover */
  }
}
```

Commentaire

```
/*
multi-ligne: preserve
*/

// mono-ligne: supprime
```

Type de données

nombre	1.2, 43, -21
dimension	4px, 0.7em, 12ch
chaîne	"hello", 'bonjour', une chaîne avec espace
booléen	true, false
null	null
liste	valeurs, séparées, avec, virgules
mapping	(clé1: val1, clé2: val2)
référence de fonction	

Chaîne

Sass reconnaît plusieurs syntaxe: avec ou sans quotes
Il est recommandé de travailler en utf-8. En tout premier du fichier main.scss:

```
@charset 'utf-8';
```

Bonne pratique:
utiliser les 'single-quotes' pour les chaînes de caractères contenu, url, etc.
attention ! ne pas en mettre pour les identifiants comme red, cyan, sans-serif, etc.

Fonctions

Multitude de fonctions pour calculer des valeurs sur des dimensions, des couleurs (saturation, opacité, luminosité, ...)
`https://sass-lang.com/documentation/Sass/Script/Functions.html`

Interpolation

Possibilité d'utiliser les variables dans les sélecteurs et nom de propriété:

```
$name: foo;
$attr: border;
p.#{$name} {
  #{$attr}-color: blue;
}
```

Partials

Permet d'éclater le style sur plusieurs fichiers plus petits et plus simple
Préfixer le nom de fichier avec un "_": _reset.scss
Importer avec la directive @import. Nom de fichier sans le préfixe _, ni l'extension .scss:

```
@import 'reset';
```

Architecture

Patron 7.1:
7 dossiers - 1 fichier

- base/
- components/
- layout/
- pages/
- theme/
- abstracts/
- vendors/
- main.scss

Dossier

base/	style des éléments de base, ex: reset, typography
layout/	système de grille, disposition ex:navigation, grid, header, footer, sidebar, forms
components/	vision plus locale que le layout, penser «widget» ex: button, carousel, cover, dropdown
pages/	en cas de style spécifique à certaines pages ex: home, contact
theme/	ex: theme, admin
abstracts/	helpers ex: variables, functions, mixins, placeholders
vendors/	style externe ex: bootstrap, jquery-ui

Syntaxe avancée

Mixins

Permet de grouper une liste de propriétés à réutiliser ensemble
Utile par exemple pour les préfixes constructeur.

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

.box { @include transform(rotate(30deg)); }
```

Arguments

Possibilité d'avoir des valeurs par défaut

```
@mixin sexy-border($color, $width: 1in) {
  border: {
    color: $color;
    width: $width;
    style: dashed;
  }
}
p { @include sexy-border(blue); }
h1 { @include sexy-border(blue, 2in); }
```

Héritage

Permet de réutiliser les règles associées à un autre sélecteur => factorisation!
possibilité de faire un "faux" sélecteur avec le préfixe "%"

```
/* This CSS will print because %message-shared is extended.
%message-shared {
  border: 1px solid #ccc;
  padding: 10px;
  color: #333;
}

// This CSS won't print because %equal-heights is never exter
%equal-heights {
  display: flex;
  flex-wrap: wrap;
}

.message {
  @extend %message-shared;
}
```

Héritage

```
%message-shared {
  border: 1px solid #ccc;
  padding: 10px;
  color: #333;
}
.message {
  @extend %message-shared;
}
.success {
  @extend %message-shared;
  border-color: green;
}
.error {
  @extend %message-shared;
  border-color: red;
}
.warning {
  @extend %message-shared;
  border-color: yellow;
}
```

Fonction

Comme en programmation! :)

```
$grid-width: 40px;
$gutter-width: 10px;

@function grid-width($n) {
  @return $n * $grid-width + ($n - 1) * $gutter-width;
}

#sidebar { width: grid-width(5); }
```

Structure de contrôle

Un peu comme en programmation aussi!
@if @else if @else

```
$type: monster;
p {
  @if $type == ocean {
    color: blue;
  } @else if $type == matador {
    color: red;
  } @else if $type == monster {
    color: green;
  } @else {
    color: black;
  }
}
```

Structure de contrôle

@for @each

```
@for $i from 1 through 3 {
  .item-#{$i} { width: 2em * $i; }
}
```

```
@each $animal in puma, sea-slug, egret, salamander {
  .#{$animal}-icon {
    background-image: url('/images/#{$animal}.png');
  }
}
```

Affectation multiple

```
@each $animal, $color, $cursor in (puma, black, default),
                                  (sea-slug, blue, pointer),
                                  (egret, white, move) {
  .#{$animal}-icon {
    background-image: url('/images/#{$animal}.png');
    border: 2px solid $color;
    cursor: $cursor;
  }
}
```

Tant que

```
$i: 6;
@while $i > 0 {
  .item-#{$i} { width: 2em * $i; }
  $i: $i - 2;
}
```