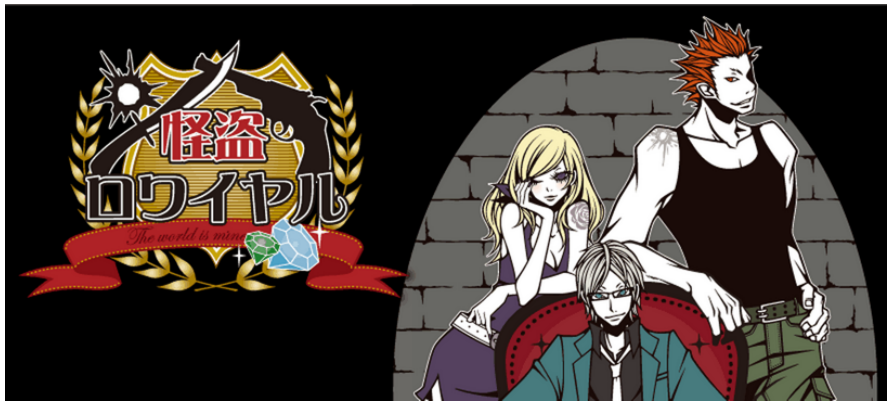
ハンズオンで実践！
Sass基礎知識

August 11, 2015

Toru Enokido
UX Design Gr.
Design Strategies Office.
DeNA Co., Ltd.

自己紹介

榎戸 徹 @55enokky



2014年 1月～10月
怪盗ロワイヤルフロントエンドを担当

KenCoM

2014年 10月～
KenCoMフロントエンドを担当

本日のアジェンダ

- Sassの概要
- Sassの基本操作
- DeNAフロントエンドの特徴

Sassの概要

Sassとは？

Sass(Syntactically Awesome Stylesheets)はCSSのメタ言語。
(セレクタの入れ子や変数といった)CSSにはない複雑な機能を使えるようにした言語。

Sass記法とSCSS記法があるがDeNAでは主SCSS記法を使っています。
既存のCSSのファイル拡張子を「.scss」にするだけでそのままSass化することができます。

Sass

```
.container
  .box-1
    width: 100px;
    height: 100px;
    background-color: #ccc;
```

SCSS

```
.container {
  .box-1 {
    width: 100px;
    height: 100px;
    background-color: #ccc;
  }
}
```

Sassのいいところ

- 記述の簡略化
- 同じ値を使いまわすことができる
- 四則演算ができる
- コードの再利用が可能
- 条件分岐などのプログラム的な処理ができる
- CSSファイルを圧縮できる
- 便利なフレームワークが使える

他のCSSメタ言語

- LESS
- Stylus
- TASS
- PCSS
- CSS Crush

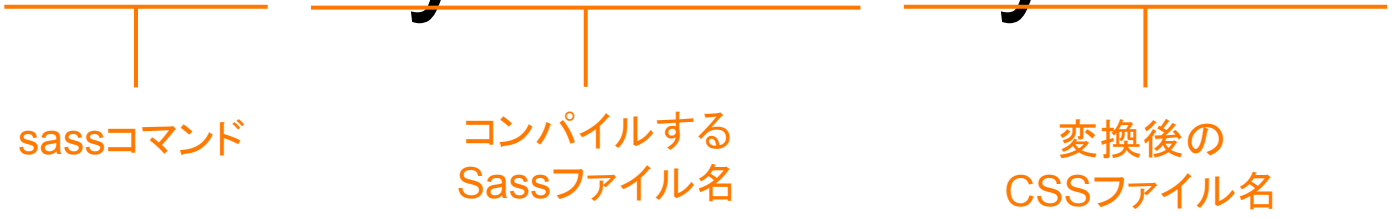
Sassのコンパイル

Sassファイルはそのままではブラウザから読み込むことができないため、
CSSファイルにコンパイル(変換)しなければなりません。

```
$ sass style.scss:style.css
```

Sassのコンパイル

```
$ sass style.scss:style.css
```



アウトプットスタイルの指定

コンパイルするSassファイルはStyleオプションを指定することでアウトプットスタイルを変更することができます。
Styleオプションは「--style」と書き、オプションのあとに半角スペースを空けてスタイルを指定します。

$ sass style.scss:style.css --style アウトプットスタイル

アウトプットスタイルには4種類指定できる。

nested
　Sassの階層をインデントで残すためネストされているような見た目になります。(デフォルトのスタイル)

expanded
　ルールセットとプロパティで改行した可視性の高いスタイル。

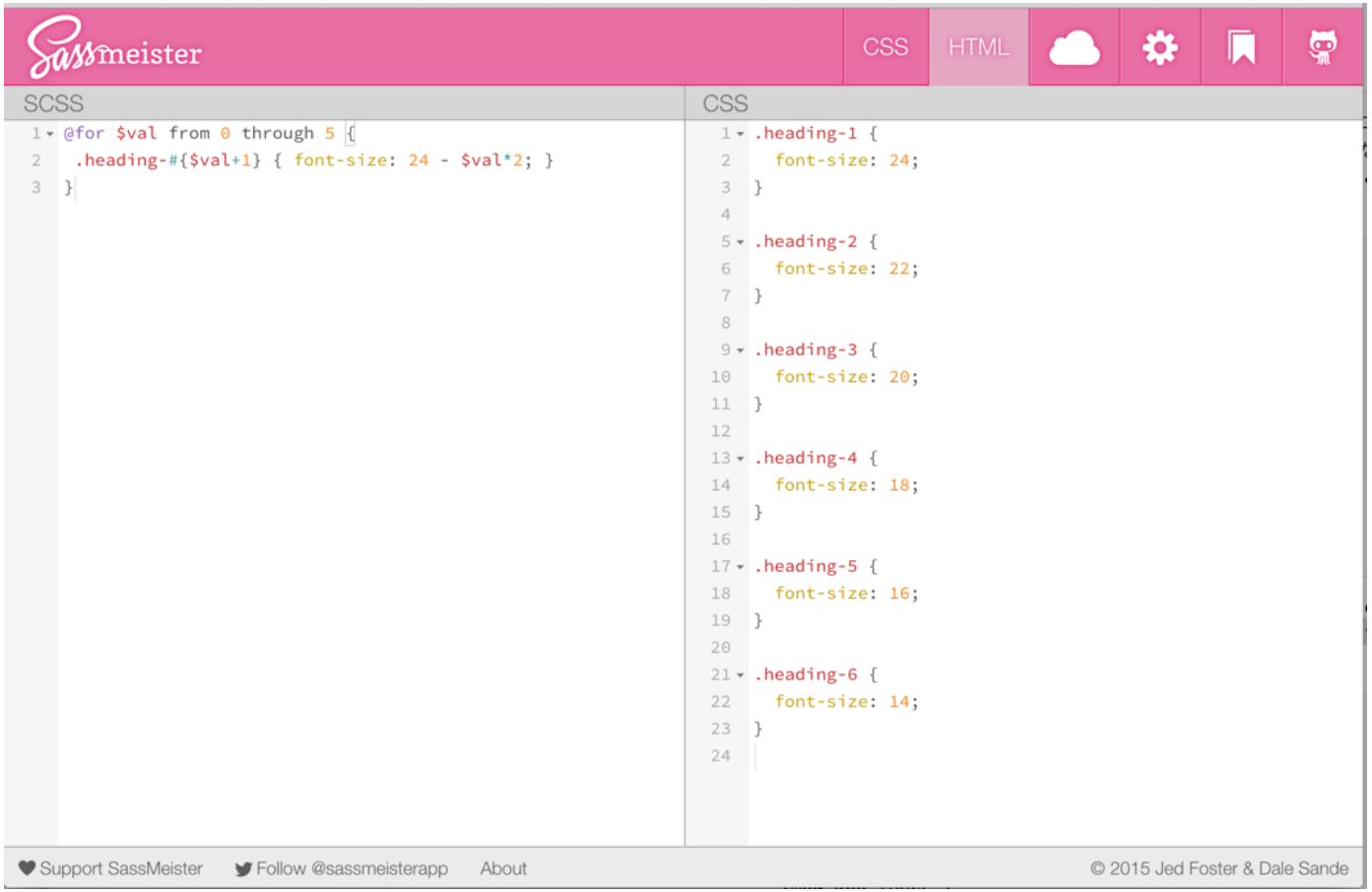
compact
　セレクタとプロパティがシングルラインになったスタイル。

compressed
　サイズ軽量化を優先し、インデントや改行をすべて取り除いて圧縮されたスタイル。

Sassがうまくインストールできなかった人は

オンラインでSassをリアルタイムにコンパイルしてくれるサービスを使ってみましょう。

http://sassmeister.com/

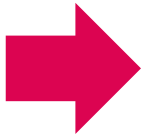


Sassの基本操作

ネスト

HTMLの構造にあわせてスタイルを入れ子で書くことができます。

```
.main {
  width: 600px;
  p {
    margin: 0 0 1em;
    em {
      color: #f00;
    }
  }
  small {
    font-size: small;
  }
}
```



```
.main {
  width: 600px;
}
.main p {
  margin: 0 0 1em;
}
.main p em {
  color: #f00;
}
.main small {
  font-size: small;
}
```

変数

変数にあらかじめ値を代入しておくことで、任意の場所で値を参照することができます。

```scss
$value: 15px;
$red: #EF3C52;

.item-1 {
  margin-left: $value;
  padding: $value;
  background-color: $red;
}

.item-2 {
  font-size: $value;
  margin-right: $value;
  color: $red;
}
```

```css
.item-1 {
  margin-left: 15px;
  padding: 15px;
  background-color: #EF3C52;
}

.item-2 {
  font-size: 15px;
  margin-right: 15px;
  color: #EF3C52;
}
```

変数

変数にあらかじめ値を代入しておくことで、任意の場所で値を参照することができます。

```
$imgPath: "../common/images/";

.box-1 {
  background-image: url("#{$imgPath}bg.jpg");
  background-repeat: no-repeat;
  background-size: cover;
}
```

```
.box-1 {
  background-image: url("../common/images/bg.jpg");
  background-repeat: no-repeat;
  background-size: cover;
}
```

親セレクタを参照する&(アンパサンド)

&(アンパサンド)を使うと親セレクタが参照できます。

```
.btn {
  background-color: #fff;
  &:hover {
    background-color: #F69156;
  }
  &:disabled {
    background-color: #bbb;
  }
  &.is-active {
    background-color: #EF3C52;
  }
}
```

```
.btn {
  background-color: #fff;
}
.btn:hover {
  background-color: #F69156;
}
.btn:disabled {
  background-color: #bbb;
}
.btn.is-active {
  background-color: #EF3C52;
}
```

演算

プロパティの値を四則演算(+-*/%)することができます。

+ :加算
- :減算
* :乗算
/ : 除算
% :剰余算(割った余り)

```
.item {
  $value: 15px;
  margin-left: $value * 2;
  padding: $value - 5px;
}
```

```
.item {
  margin-left: 30px;
  padding: 10px;
}
```

@for

@for は指定した値から終了の値まで、1つづつ値を増やしながら繰り返し処理が行われます。
@for には 2つの構文があります。

```
@for $変数名 from 開始の値 through 終了の値 {
  …(スタイルなど)...
}

@for $変数名 from 開始の値 to 終了の値 {
  …(スタイルなど)...
}
```

through と to では終了の値が異なります。
through: 指定した値を**含んで**繰り返し
to: 指定した値を**含まず**に繰り返し

@mixin

@mixin はスタイルの集まりを定義しておき、他の場所から呼び出して使う機能。

```
@mixin ミックスイン名($引数) {
  …(スタイルなど)…
}
```

■ mixin の呼び出し方

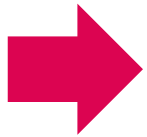
```
@include ミックスイン名;
```

スタイルの継承

@extend をつかうと指定したセレクタのスタイルを継承することができます。

```
.item-1 {
  $value: 15px;
  margin-left: 15px;
  padding: 15px;
  background-color: #f0f;
}

// .item で定義したスタイルを継承する
.item-2 {
  @extend .item-1;
}
```



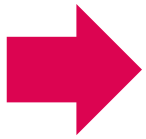
```
.item-1, .item-2 {
  margin-left: 15px;
  padding: 15px;
  background-color: #f0f;
}
```

プレースホルダーセレクター

@extend はセレクタを継承する機能なのでコンパイル後のCSSには必ず継承元のセレクタも生成されるが、@extend 専用のプレースホルダーセレクタを用いることでセレクタを生成させないようにできます。

```
%item-base {
  $value: 15px;
  margin-left: 15px;
  padding: 15px;
  background-color: #f0f;
}

// %item-base で定義したスタイルを継承する
.item {
  @extend %item-base;
  color: blue;
  font-size: 14px;
}
```



```
.item {
  margin-left: 15px;
  padding: 15px;
  background-color: #f0f;
}

.item {
  color: blue;
  font-size: 14px;
}
```

便利なSassフレームワーク

- Compass
- Bourbon
- Bootstrap
  - LESS でつくられたフレームワークだが Sass 化されたものが GitHub で公開されている
- Susy

DeNAフロントエンドの特徴

DeNAフロントエンドの特徴

- 自動化進んでます
  - Grunt、Gulp、Asset Pipeline etc...
- ブラウザは幅広くは対応していない(IE9～・Android 4.0～)※プロジェクトによる
- SGではCSS3を多用
- 導入しているフロントエンド技術もプロジェクトによってさまざま
  - みんな新しいもの(技術)好きなので気になった技術はどんどん取り入れてwikiなどを活用している
  - 勉強会も積極的
- コードレビューしっかりしてます