

Sass/Compass講習会

2014/2/4



本講習の目的

- Sass/Compassの基本を理解する
- 開発環境を設定して扱えるようになる
- 実案件で活用する
- 協力会社ともやりとりできるようになる

本講習で使う資料・ファイル

資料一式はこちらからダウンロードしてください。

http://webnaut.jp/download/markup_140204/sass.zip

▼本スライド資料(/~/sass.pdf)

出力して手元に控えつつ、ソースをコピーして使う事もあると思うのでPDFは開いておいてください。

▼ファイルセット(/~/fileset/)

設定のテストや実習で使用するファイルです。

Sassのコンパイル時の問題(後述)がありますので、日本語が含まれないディレクトリ(デスクトップなど)にコピーしてください。

目次

1. Sassとは／機能紹介
2. Sassを触ってみよう
3. Sass開発環境の設定
4. [Work1]ネストとインポート※
5. Compassとは／機能紹介
6. Compassの設定
7. [Work2]Compassを使ったCSS Sprite実践※
8. [Work3]納品時の作業※
9. 書籍、参考サイト紹介
10. [Work4]追加課題※

※実習パート

Sassとは／機能紹介

Sassとは

CSSをより便利に効率的に書けるように
パワーアップさせた言語

■特長

- CSSをプログラマ的に書ける(入れ子構造、変数、演算など)
- 別途CSSファイルをコンパイル(変換)する作業が必要
- Compassなど便利なフレームワークが使える

公式サイト:

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



■読み方、表記について

Sass : 「サス」 です。

※ドットインストールでは「サーズ」と発音しています。
どちらが正しいかは不明ですが、
世間では「サス」が一般的のようです。

<http://yomikata.org/word/sass>

ちなみに、

CSSは全部大文字 ／ Sassは大文字+小文字

※他 : HTML, JavaScript, LESS, Stylus, PHP, CGI

■他のCSSメタ言語との比較

LESS



- Bootstrapで採用
- クライアントサイド (JavaScript) でCSS生成が可能
- Sassよりも機能が少ない

Stylus



- 記述の簡素化 (波括弧 {}, コロン, セミコロン, カンマを省略可能)
- CSSハックをそのまま使用可能
- Node.jsベースのフレームワークでも使われている

■それでもSassを使う理由

- ユーザー数が多い
- 大手企業でも導入が進んでいる（LINE株式会社、株式会社DeNA、クックパッド株式会社など）
- ツールが豊富、日本語の情報も多く導入のハードルが下がっている。
- 便利なフレームワーク（Compassなど）が使える



事実上、CSSメタ言語のスタンダードと言って良い。

■ 記法[SASS][SCSS]の違い

```
ul {  
  margin: 0 0 1em;  
}  
ul li {  
  margin-bottom: 5px;  
}
```

CSS

SCSSが一般的なので
こちらを使用



```
ul  
  margin: 0 0 1em;  
li  
  margin-bottom: 5px;
```

SASS

```
ul {  
  margin: 0 0 1em;  
  li {  
    margin-bottom: 5px;  
  }  
}
```

SCSS

- ・記述が簡素
- ・インデントや改行の制限あり

- ・CSSと互換性がある
- ・インデントや改行の制限なし


Sassの機能紹介

1.ネスト(Nesting)

CSSをHTML構造に合わせて入れ子で記述

```
.oya {  
  color: #000;  
  .kodomo {  
    color: #111;  
    .mago {  
      color: #222;  
    }  
  }  
}
```

SCSS

 入れ子

```
.oya {  
  color: #000;  
}  
.oya .kodomo {  
  color: #111;  
}  
.oya .kodomo .mago {  
  color: #222;  
}
```

CSS

 親のセレクトタも適用

■POINT

- HTMLのツリー構造が把握でき、セレクトタが特定しやすい
- セレクトタの追加、変更が容易（作業効率／メンテナンス性の向上）

※階層を深くしすぎると可読性が落ちる

参照：<http://book.scss.jp/code/c3/01.html>

■合わせて覚えてたい！

親セクタの参照 “&” (アンパサンド)

<http://book.scss.jp/code/c3/02.html>

```
p {  
  a {  
    font-size: 16px;  
  }  
  a:hover {  
    color: #666666;  
  }  
}
```

SCSS

```
p {  
  a {  
    font-size: 16px;  
    &:hover {  
      color: #666666;  
    }  
  }  
}
```

SCSS

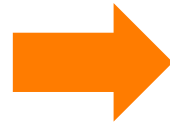
 &:親セクタの指定

例えばこんな使い方

例：ボタンパーツの状態変化を表すスタイル

```
a.btn {  
  color: #fff;  
  &.is-over {  
    color: #666;  
  }  
  &.is-success {  
    color: #333;  
  }  
  &.is-info {  
    color: #f00;  
  }  
}
```

SCSS



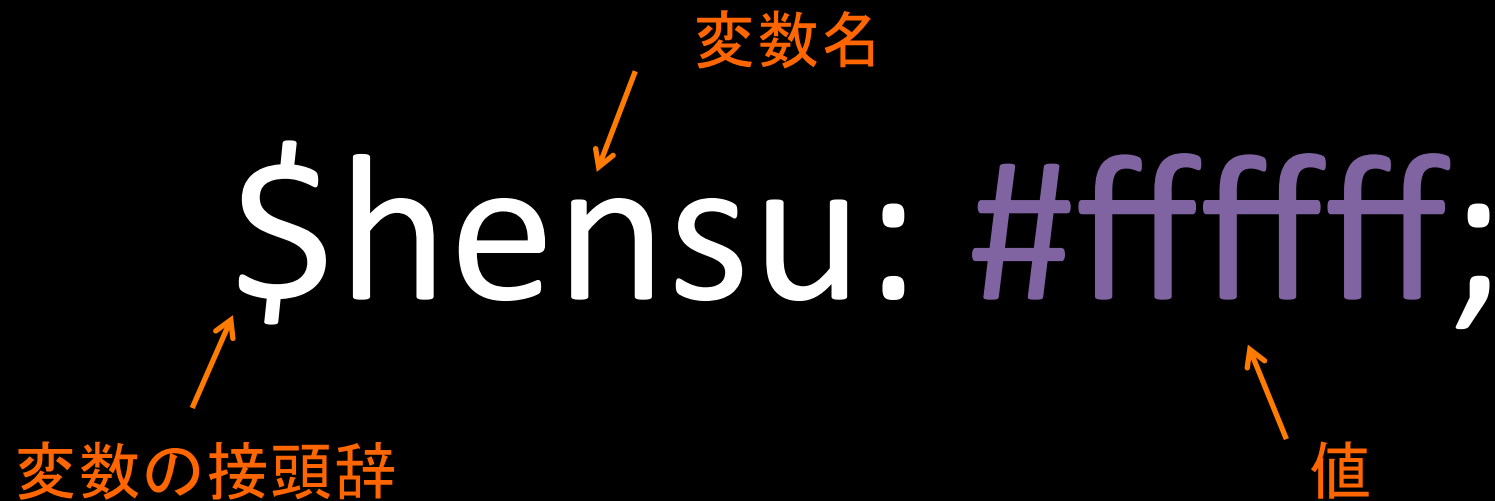
```
a.btn {  
  color: #fff;  
}  
a.btn.is-over {  
  color: #666;  
}  
a.btn.is-success {  
  color: #333;  
}  
a.btn.is-info {  
  color: #f00;  
}
```

CSS

→スタイルを構造的に把握できる

2.変数 (Variables)

- 変数とは、**データに付けるラベル**のようなもの
- Sassで扱えるデータ型: 数値、文字列、真偽(true,false)、リスト ※変数に変数を入れる事も可能
- 四則演算(+-* /)



The diagram shows a Sass variable declaration `$hensu: #ffffff;` on a black background. Three orange arrows point to specific parts of the code: one to the dollar sign (\$), one to the variable name 'hensu', and one to the value '#ffffff'. Each arrow is accompanied by a label in orange text: '変数の接頭辞' (Variable prefix) for the dollar sign, '変数名' (Variable name) for 'hensu', and '値' (Value) for '#ffffff'.

変数の接頭辞

変数名

値

SCSS

SCSS

```
$main-color: #000000;  
$space: 10px;  
.global-header {  
  background-color: $main-color;  
}  
.module {  
  border-color: $main-color;  
  margin: $space;  
  padding: $space / 2;  
}
```



CSS

```
.global-header {  
  background-color: #000000;  
}  
.module {  
  border-color: #000000;  
  margin: 10px;  
  padding: 5px;  
}
```

■ POINT

- 色や数値など覚えにくい値を都度コピーする必要がなくなる
- 変数を一元管理して使用すればスタイルに一貫性があり修正に強いサイトができる

参照 : <http://book.scss.jp/code/c3/05.html>

3.ミックスイン(Mixin)

スタイルの集まりを定義して、複数の箇所で呼び出せる

ミックスインの定義

ミックスイン名

引数(任意)

SCSS
(定義側)

```
@mixin kadomaru($value) {  
  -webkit-border-radius: $value;  
  -moz-border-radius: $value;  
  -ms-border-radius: $value;  
  -o-border-radius: $value;  
  border-radius: $value;  
}
```

スタイルの集まり

```
.box1 {  
  @include kadomaru(4px);  
}  
.box2 {  
  @include kadomaru(10px);  
}
```

SCSS
(呼び出し側)

@include:
ミックスインの呼び出し



```
.box1 {  
  -webkit-border-radius: 4px;  
  -moz-border-radius: 4px;  
  -ms-border-radius: 4px;  
  -o-border-radius: 4px;  
  border-radius: 4px;  
}  
.box2 {  
  -webkit-border-radius: 10px;  
  -moz-border-radius: 10px;  
  -ms-border-radius: 10px;  
  -o-border-radius: 10px;  
  border-radius: 10px;  
}
```

CSS

■ POINT

- Sassの花形機能
- 引数を駆使すれば様々な応用が可能
- 変数と同様、使う時は一元管理

参照 : <http://book.scss.jp/code/c4/02.html>

他にも押さえておきたい基本機能

- Sass で使えるコメント

<http://book.scss.jp/code/c3/04.html>

- Sass の @import

<http://book.scss.jp/code/c3/07.html>

- スタイルの継承ができる @extend

<http://book.scss.jp/code/c4/01.html>

- 条件分岐や繰り返し処理

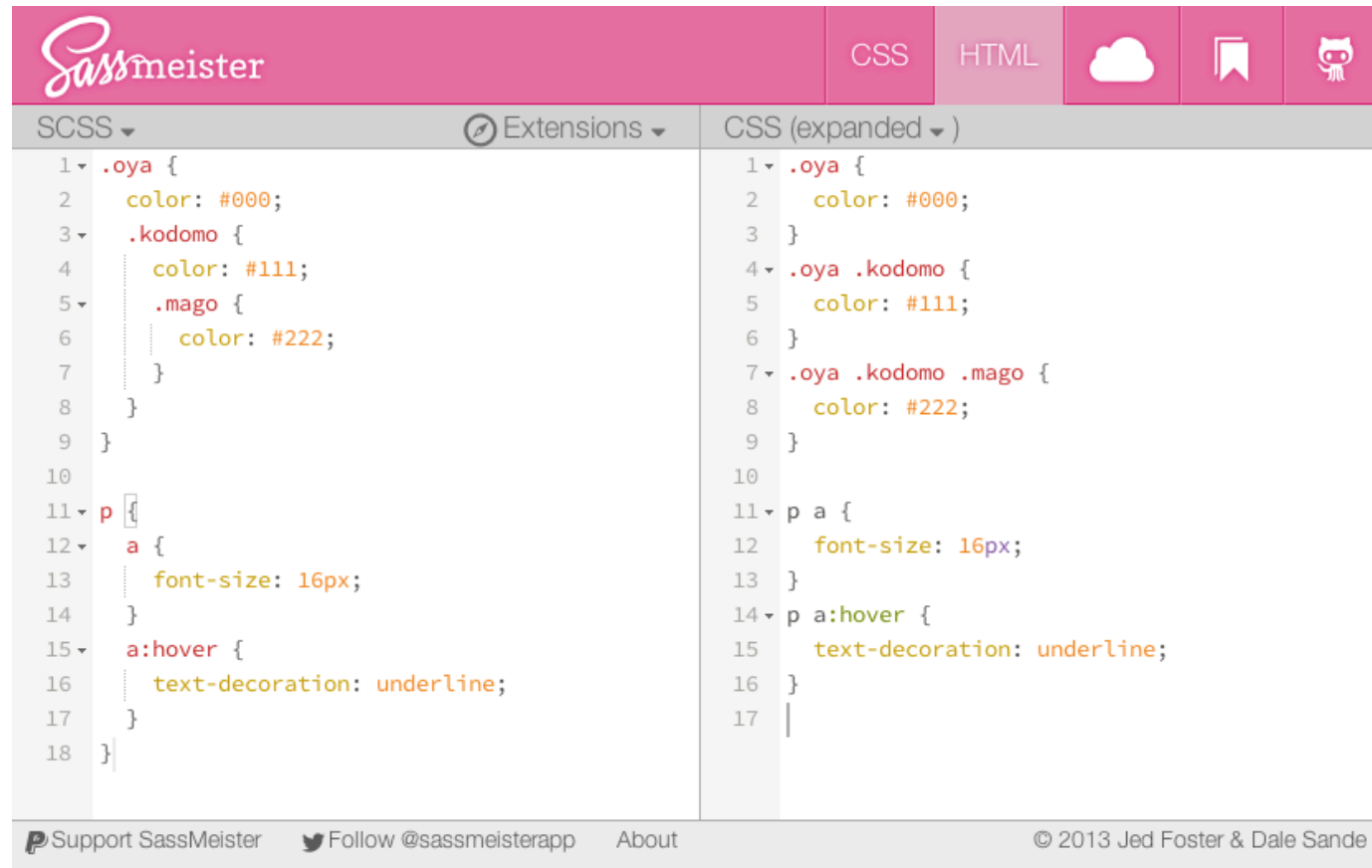
<http://book.scss.jp/code/c4/03.html>

- 自作関数を定義する @function

<http://book.scss.jp/code/c4/05.html>

Sassを使ってみよう

SassをリアルタイムでCSSに変換できるWebサービス



<http://sassmeister.com/>

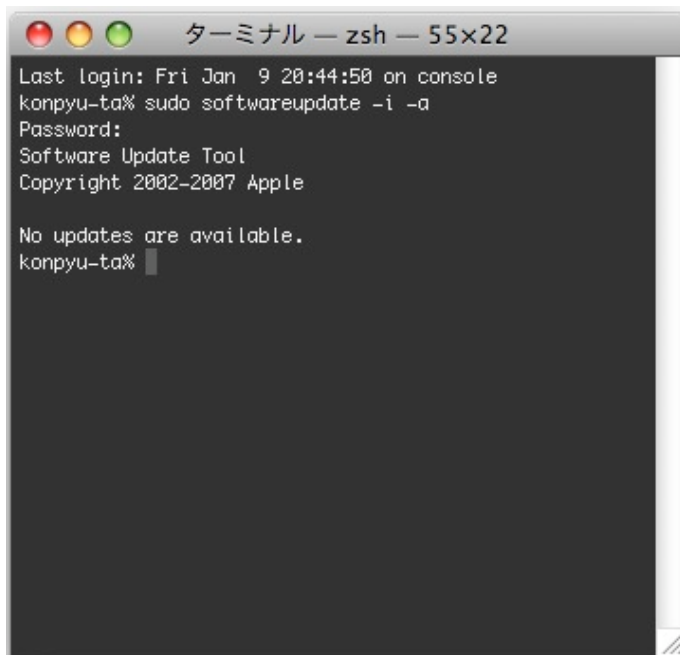
2〜3min 触ってみましょう。

Sass開発環境の設定

SassのファイルをCSSにコンパイルする方法

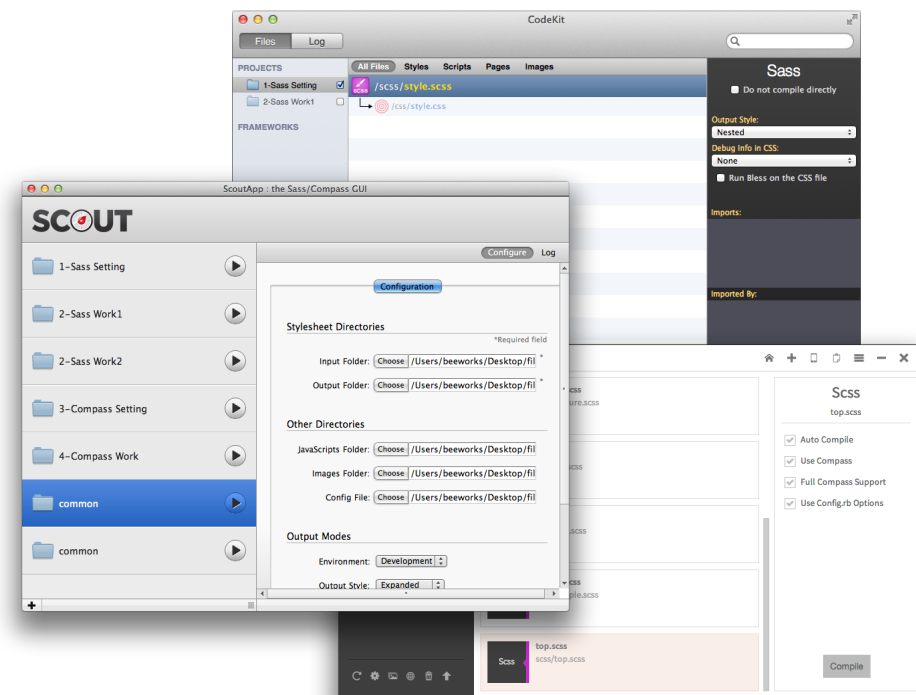
■CUIツール

コマンドプロンプト・ターミナル等



■GUIツール

Scout, CodeKit, Prepros等



使いやすいこちらを使用

[Work1]ネストとインポート

Work1-1.ルールのネスト

/fileset/2-Sass Work1/を使用

ここでは、

- 1.まずはCSSで普通に書く
 - 2.その後Sassの記法で書いてCSSにコンパイルする
- を比較してCSSとの作業の違いを理解して頂きます。

Work1-1.ルールのネスト

まずはこの2つのファイルの中身を見てみましょう。

/fileset/2-Sass Work1/before/index.html

/fileset/2-Sass Work1/before/css/style.css

Work1-1.ルールのネスト

ここで課題。

下記HTMLの変更をCSSに反映してください。

変更箇所：

- <section>を<article>タグに変更
- リストのスタイルを5つ(.list1～.list5)に変更

※HTML内にコメントで変更箇所を記載しています。

変更後のHTML：

/fileset/2-Sass Work1/after/index.html

作業用CSS(beforeと同じもの)：

/fileset/2-Sass Work1/after/css/style.css

Work1-1.ルールのネスト

制限時間: 2min

Work1-1.ルールのネスト

解答:

**/fileset/2-Sass Work1/after/
answer.css**

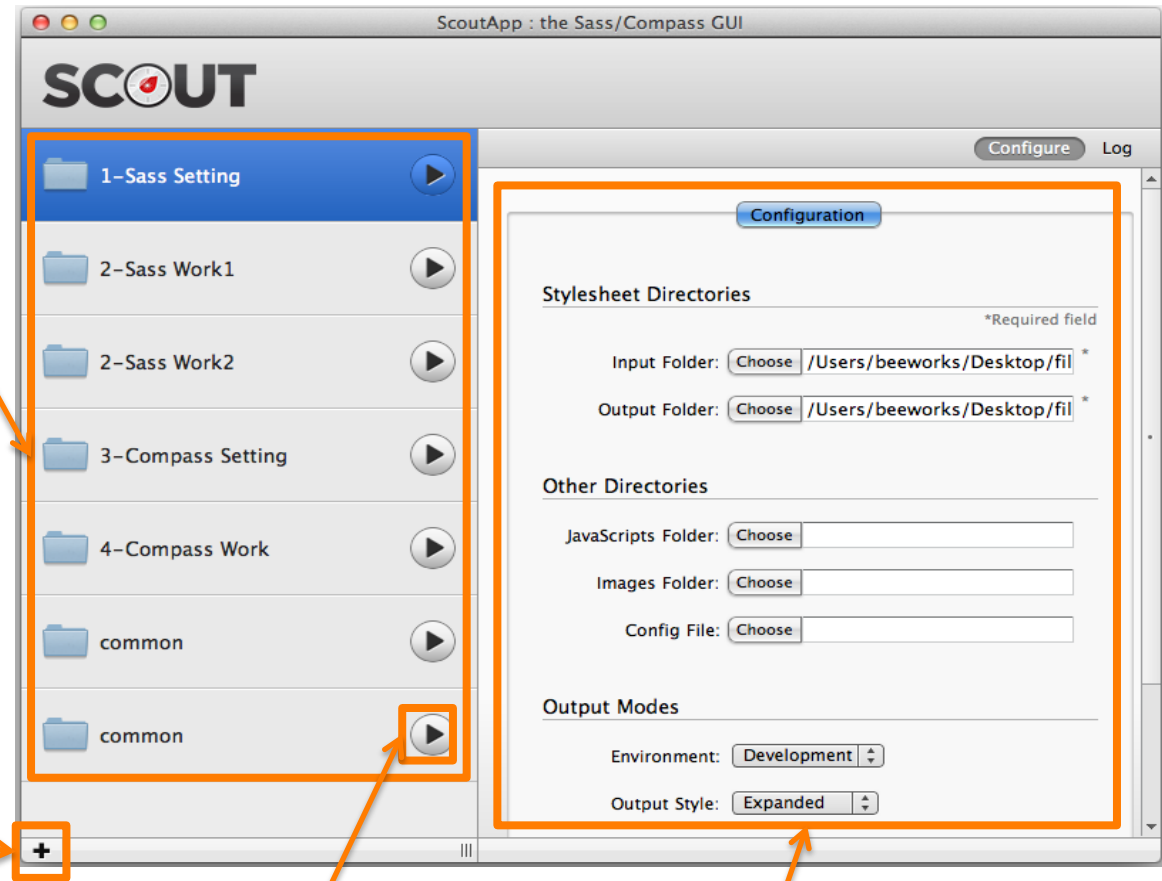
Work1-1.ルールのネスト

では、次にこの工程をSassでやってみましょう。

■コンパイラの設定 - Scoutの構成説明

プロジェクト
(1サイト単位)

プロジェクト
追加



実行

設定／ログ表示

■コンパイラの設定 - プロジェクトの追加

③ 実行

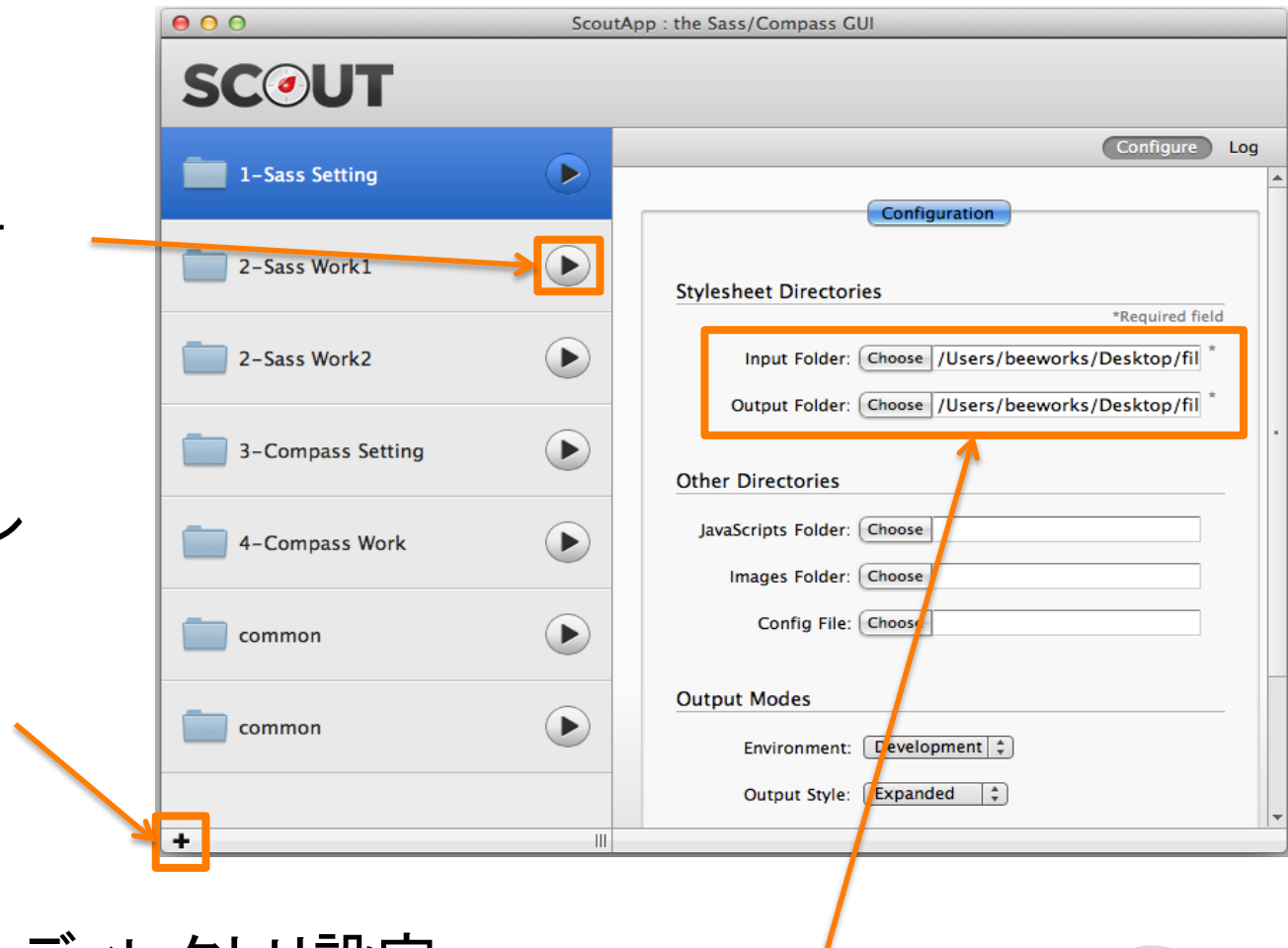
① プロジェクトのフォルダを追加

/fileset/2-Sass Work1/

② Stylesheet ディレクトリ設定

Input Folder: /fileset/2-Sass Work1/after/scss/

Output Folder: /fileset/2-Sass Work1/after/css/



■コンパイラの設定 – 変換テスト

次にこのSassファイルをエディタで開いて、
そのまま保存してみましょう

/fileset/2-Sass Work1/after/scss/style.scss

ScoutがSassファイルの更新を検知してCSSを生成します。
以下のCSSが上書きされれば準備OK。

/fileset/2-Sass Work1/after/css/style.css

```
>>> Change detected at 11:10:49 to: style.scss
create style.css

Dear developers making use of FSSM in your projects, be
taking place in the new shared guard/listen project.
Please
let us know if you need help transitioning! ^_^b - Travis
Tilley >>> Compass is polling for changes. Press Ctrl-C to
Stop.
```

ログ表示

■Sassファイルの更新

それでは先ほどと同じ修正作業をやってみましょう。

作業用Sassファイル:

/fileset/2-Sass Work1/after/scss/style.scss

※先ほどのanswer.cssと同じスタイルになれば完了です。

こまめに保存して書き出されたcssを確認してみましょう。

※cssはSassで上書きされるので、確認用を開いたらすぐに閉じること。

■ Sassファイルの更新

制限時間: 5min

解答：

**`/fileset/2-Sass Work1/after-answer/
scss/style.scss`**


```
#main {  
  article {  
    margin-bottom: 50px;  
    h1 {  
      font-size: 140%;  
    }  
    p, ul {  
      margin-bottom: 1.5em;  
    }  
    ul {  
      li.list1 {  
        font-size: 110%;  
      }  
      li.list2 {  
        font-size: 120%;  
      }  
      li.list3 {  
        font-size: 130%;  
      }  
      li.list4 {  
        font-size: 140%;  
      }  
      li.list5 {  
        font-size: 150%;  
      }  
    }  
    p.notes {  
      color: red;  
    }  
  }  
}
```

解説:

 が更新した箇所

Work1-2.インポート

CSSでのインポート:

```
@import url("main.css");  
@import url(main.css);  
@import "main.css";  
@import 'main.css';
```

Sassでのインポート:

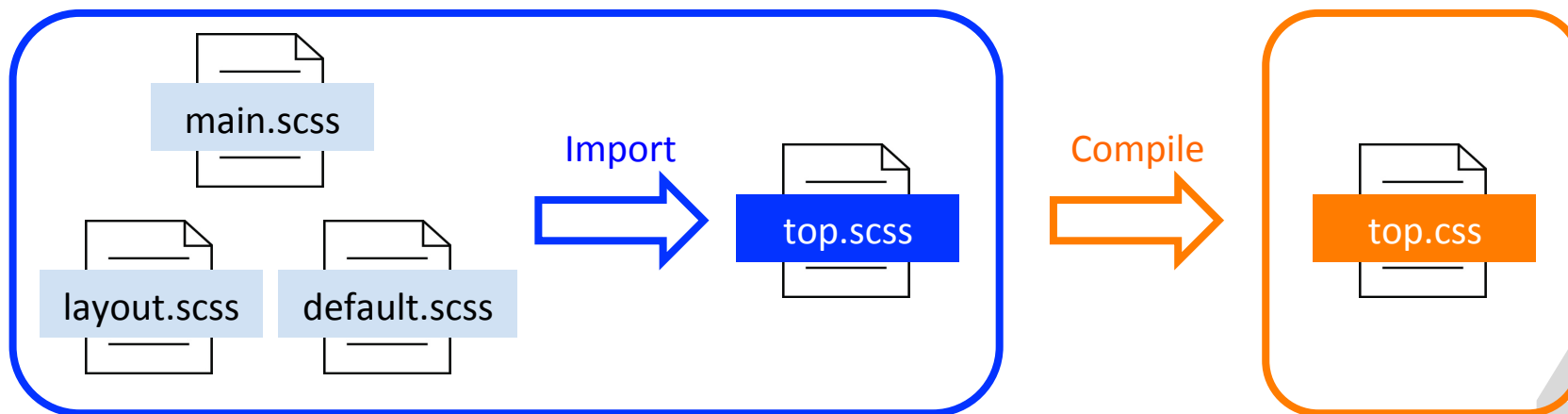
```
@import url("main.scss");  
@import url(main.scss);  
@import "main.scss";  
@import 'main.scss';
```

基本的な指定方法は同じ

Work1-2.インポート

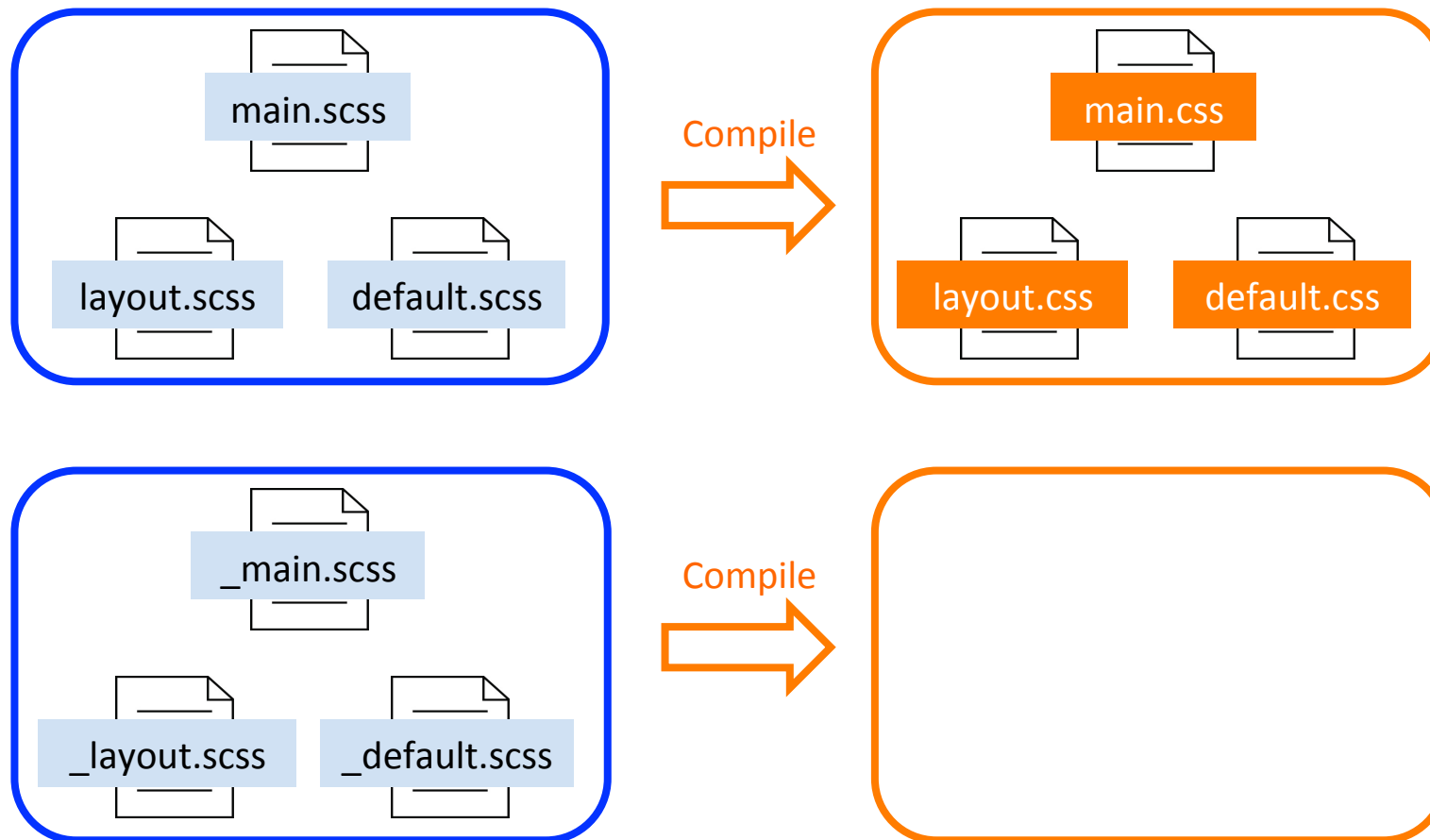
■ Sassのインポートを使うメリット:

- ・HTTPリクエスト数を減らし、表示高速化
→ユーザー側メリット
- ・Sassファイルを分割する事で管理しやすくなる
→制作側メリット



Work1-2.インポート

■ パーシャル: ファイル名の先頭に”_”(アンダースコア)があると、コンパイル後にCSSファイルは生成されない



Work1-2.インポート

では実際にインポートをやってみましょう。

/fileset/2-Sass Work2/を使います。

/fileset/2-Sass Work2/scss/_common.scss
/fileset/2-Sass Work2/scss/_mixin.scss
/fileset/2-Sass Work2/scss/_reset.scss
/fileset/2-Sass Work2/scss/_setting.scss の4ファイルを
/fileset/2-Sass Work2/scss/style.scss にまとめる作業です。

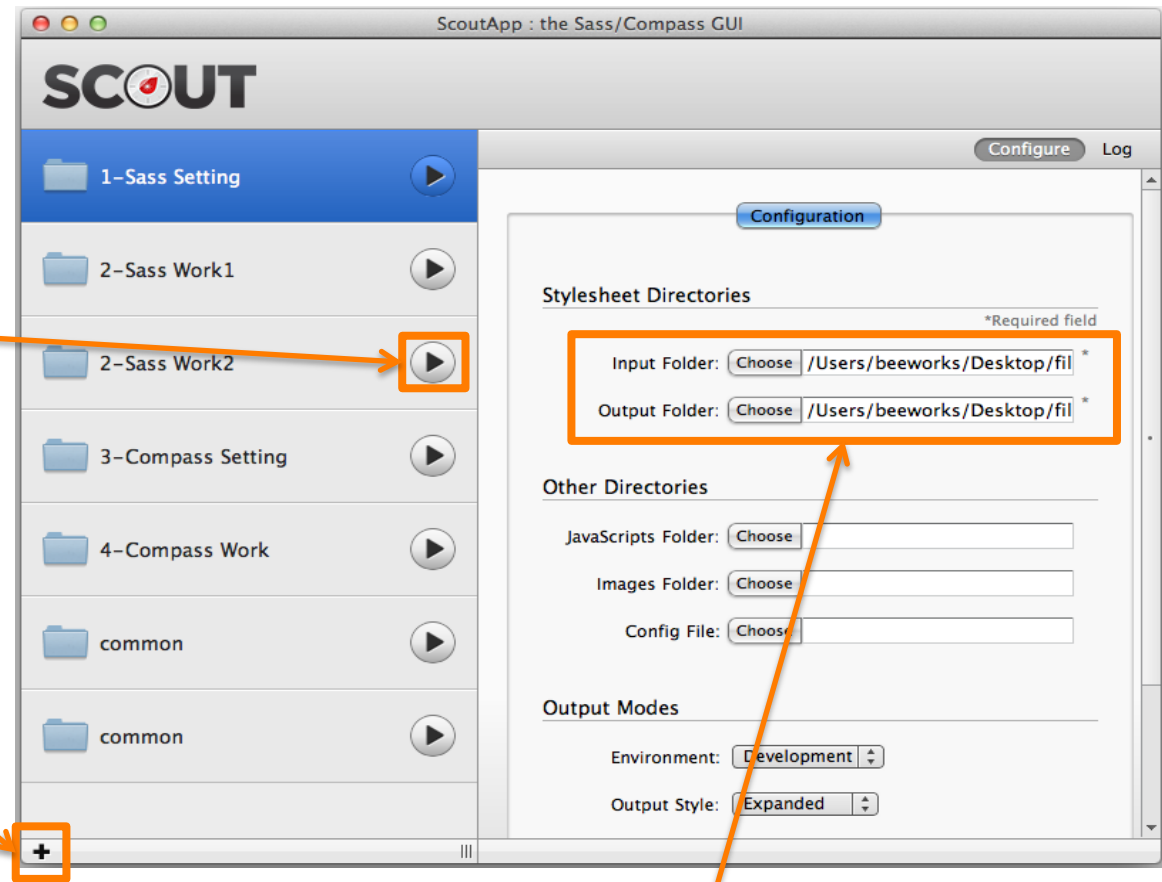
※style.scssにはすでにインポートの記述まで入れてありますので、ここではコンパイルだけをやることとします。

Work1-2.インポート

Work1-1と同様に、
Scoutの設定から

③実行

①プロジェクトの追加
/fileset/2-Sass Work2/



②Stylesheet ディレクトリ設定

Input Folder: /fileset/2-Sass Work2/after/scss/

Output Folder: /fileset/2-Sass Work2/after/css/

Work1-2.インポート

コンパイル後、下記のファイルが生成されればOKです。
style.cssを開いてみると各Sassファイルの記述がインポート
されていると思います。

/fileset/2-Sass Work2/css/style.css

Work1-2.インポート

このように、Sassを使う際は、モジュール単位や用途毎にファイルを分ける事で、複数人での作業やバージョンの管理等も行いやすくなります。

その他の機能はこちらで確認ください。

- Sassの @import

<http://book.scss.jp/code/c3/07.html>

ここまでのまとめ

- CSSで面倒に感じていた作業や管理が楽になる
- コンパイル環境はGUIツールで簡単に設定
- 全部を無理に使わなくても使いたい機能(ネストだけ、インポートだけ)で使ってもOK

Compassとは／機能紹介

50

Compass (コンパス) とは

CSS(Sass)の機能を拡張して
さらに**パワーアップ**させたもの

Compassの主な機能

- 便利なミックスインが標準搭載

CSS3モジュール(角丸、グラデ、Opacity)、ユーティリティ(リンクカラー、clearfix、min-height/widthなど)、タイポグラフィなど

- 独自の関数

画像関連(パス、サイズ取得)、色調整など

- スプライト画像などの画像処理
- プラグインでさらに拡張

公式サイト: <http://compass-style.org/>

■ミックスイン例

```
@mixin kadomaru($value) {  
  -webkit-border-radius: $value;  
  -moz-border-radius: $value;  
  -ms-border-radius: $value;  
  -o-border-radius: $value;  
  border-radius: $value;  
}  
.box {  
  @include kadomaru(4px);  
}
```

SCSS

```
.box {  
  -webkit-border-radius: 4px;  
  -moz-border-radius: 4px;  
  -ms-border-radius: 4px;  
  -o-border-radius: 4px;  
  border-radius: 4px;  
}
```

CSS

```
.box {  
  @include border-radius(4px);  
}
```

CompassのMixin

Compass

■ Compassを使うには...？

Compassで扱うファイルはconfig.rbというRubyの設定ファイル1つだけ。

あとはSass内で「Compassを使います」と宣言をするだけで使えます。

```
@import "compass";
```

SCSS

Compassの設定

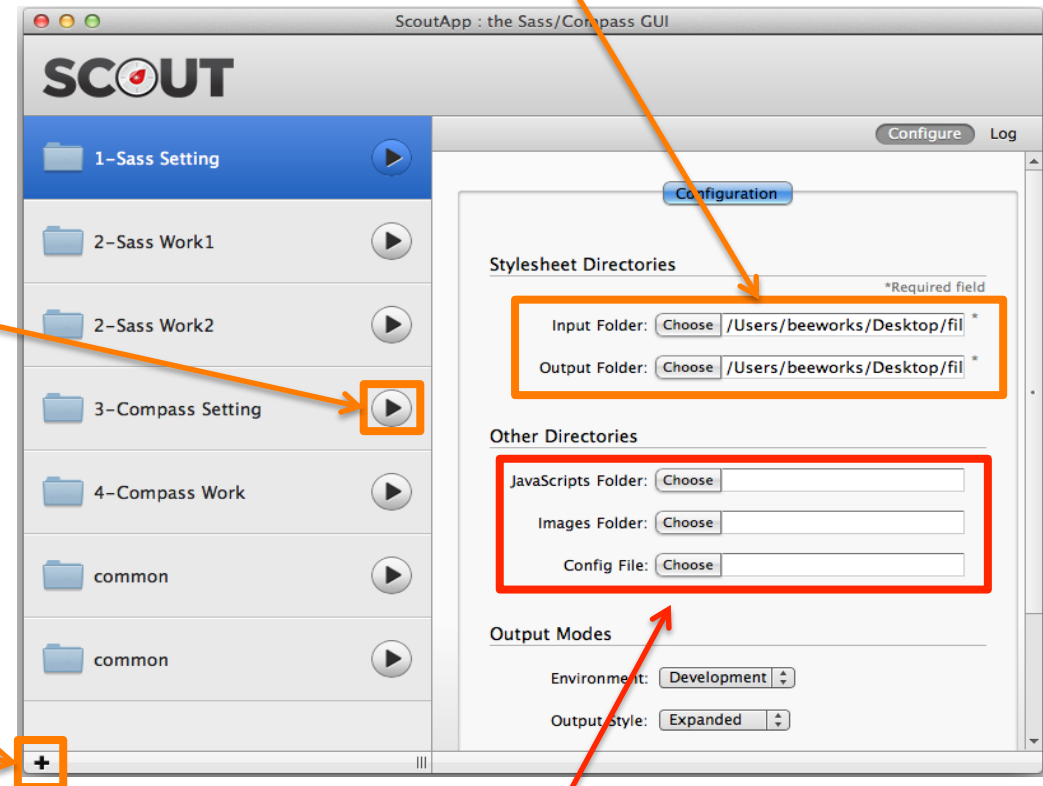
② Stylesheet ディレクトリ設定

Input Folder: /fileset/3-Compass Setting/scss/

Output Folder: /fileset/3-Compass Setting/css/

④ 実行

① プロジェクトの追加 /fileset/3-Compass Setting/



③ Other Directries設定

JavaScript Folder: /fileset/3-Compass Setting/js/

Images Folder: /fileset/3-Compass Setting/img/

Config File: /fileset/3-Compass Setting/config.rb

Compassの設定

ここではコンパイルができるかの確認だけ行います。
以下のCSSファイルが出力されればOKです。

`/fileset/3-Compass Setting/css/style.css`

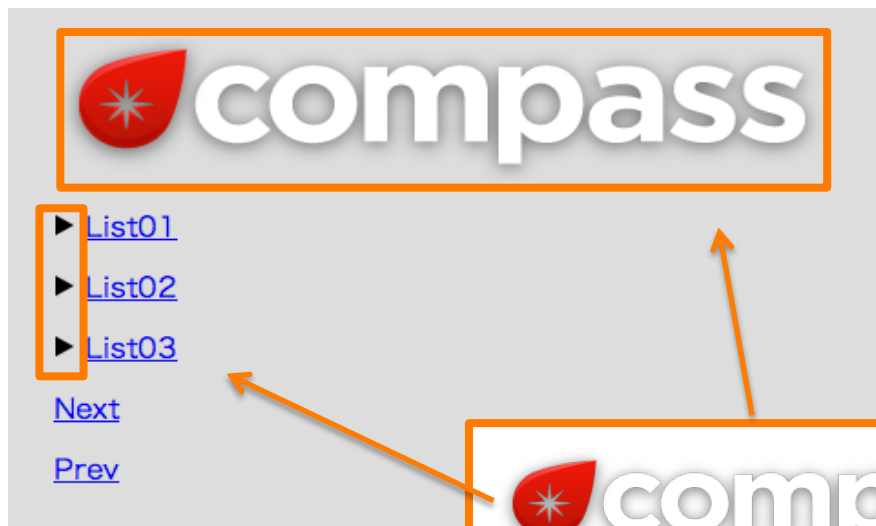
いくつかCompass専用のスタイルを記述しています
ので中身を確認してみてください。

[Work2]
Compassを使ったCSS Sprite実践

Work2. Compassを使ったCSS Sprite

■まずはこちらのHTMLをブラウザで見てください。

/fileset/4-Compass Work/index.html



```
.logo {  
  background-image: url('../img/sprite01-se1c8f4c267.png');  
  background-repeat: no-repeat;  
  display: block;  
  height: 88px;  
  width: 423px;  
  background-position: 0 0;  
  -webkit-background-size: 423px auto;  
  -moz-background-size: 423px auto;  
  -o-background-size: 423px auto;  
  background-size: 423px auto;  
}
```

CSS

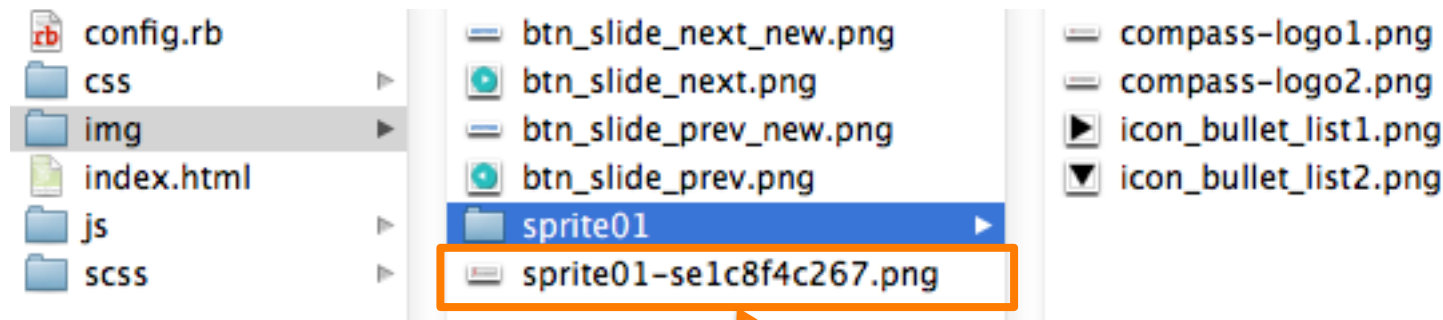


ここが一枚画像

Work2. Compassを使ったCSS Sprite

■ 次に/img/フォルダを確認

/sprite01/フォルダに単体の画像があります



合成された画像ファイル

Compassでは画像の合成、画像のパスやサイズを取得が可能なので、その機能を組み合わせる事でCSS Spriteが簡単に実装できます。

Work2. Compassを使ったCSS Sprite

■ CSS Sprite用ミックスイン

Compassには“CSS spriteを実装するミックスイン”というものはなく、命令を組み合わせて作る必要があります、
今回使用するミックスインはBeeworksオリジナル版です。

```
$sprites-map01: sprite-map("sprite01/*.png");
$sprites-img01: sprite-url($sprites-map01);

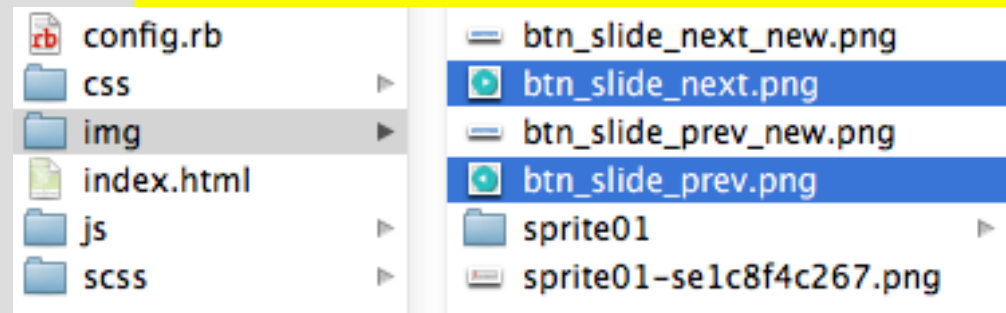
@mixin sprite-background($name, $sprites-map, $sprites-img, $ratio:1) {
  background-image: $sprites-img;
  background-repeat: no-repeat;
  $width: image-width(sprite-path($sprites-map));
  display: block;
  height: image-height(sprite-file($sprites-map, $name)) / $ratio;
  width: image-width(sprite-file($sprites-map, $name)) / $ratio;
  $ypos: round(nth(sprite-position($sprites-map, $name), 2));
  background-position: 0 ($ypos / $ratio);
  @include background-size(($width / $ratio) auto);
}
```

Work2. Compassを使ったCSS Sprite

■ 課題



画像はこちらを使用



[Next][Prev]のテキストリンク部分をCSS Spriteを使って



の画像で指定してください。

Work2. Compassを使ったCSS Sprite

■作業の流れ

- Scoutにプロジェクト登録 (/fileset/4-Compass Work/)
- 2つの画像を/sprite01/に入れる
- style.scssの修正 : [Next][Prev]のスタイルを指定している箇所にspriteの指定を入れてコンパイル
- HTMLから[Next][Prev]のテキスト部分を削除

Work2. Compassを使ったCSS Sprite

■ 次の課題



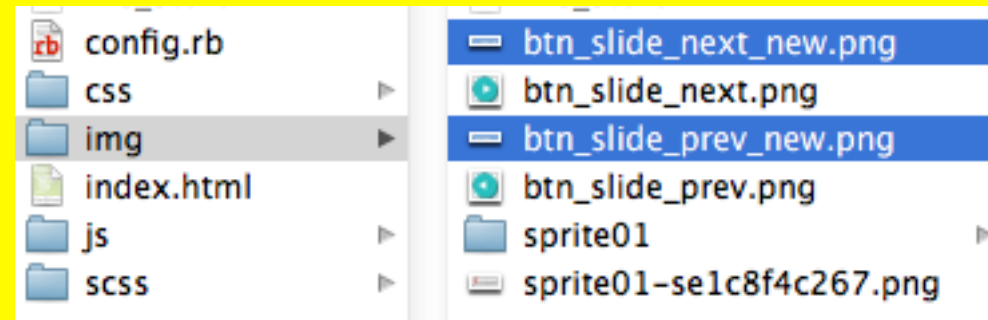
の画像を

Next ▶

◀ Prev

に変更してください。

※画像ファイル名は変えても同じでもOK



Work2. Compassを使ったCSS Sprite

制限時間 : 5min

Work2. Compassを使ったCSS Sprite

■ちなみに、

このミックスインでは引数の最後の値でデバイスピクセル比を指定する事ができます。

```
@include sprite-background(compass-logo1, $sprites-map01,  
$sprites-img01, 1);
```

↓

```
@include sprite-background(compass-logo1, $sprites-map01,  
$sprites-img01, 2);
```

とすれば倍率が2倍になるので、Retina対応が可能となります。
※これがBeeworksオリジナル

Work2. Compassを使ったCSS Sprite

解答:

/fileset/4-Compass Work/scss/
_answer.scss

Work2. Compassを使ったCSS Sprite

解説:

```
.btnNext a {  
  @include sprite-background(btn_slide_next_new, $sprites-  
map01, $sprites-img01, 1);  
}  
  
.btnPrev a {  
  @include sprite-background(btn_slide_prev_new, $sprites-  
map01, $sprites-img01, 1);  
}
```



のファイル名(拡張子なし)

ミックスイン呼び出し



のファイル名(拡張子なし)

ここまでのまとめ

Compassは「スタイルシートのjQuery」等とも呼ばれるぐらい使用頻度が高いもの。

実質Sassを使う際にはCompassも合わせて使う事を前提として良い。

[Work3]納品時の作業

70

Work3.納品時の作業

最後に、実際に案件で使うシーンとして、納品時に行う作業を確認しましょう。

ここで行う作業：

- CSSファイルの圧縮
- 不要ファイルの精査と削除

Work3.納品時の作業

/fileset/5-Last Work/を使います。

まずはScoutに以下のデータをセットしてください。

Project: /fileset/5-Last Work/**common**/

Input Folder: /fileset/5-Last Work/common/scss/

Output Folder: /fileset/5-Last Work/common/css/

JavaScript Folder: /fileset/5-Last Work/common/js/

images Folder: /fileset/5-Last Work/common/img/

Config File: /fileset/5-Last Work/common/config.rb

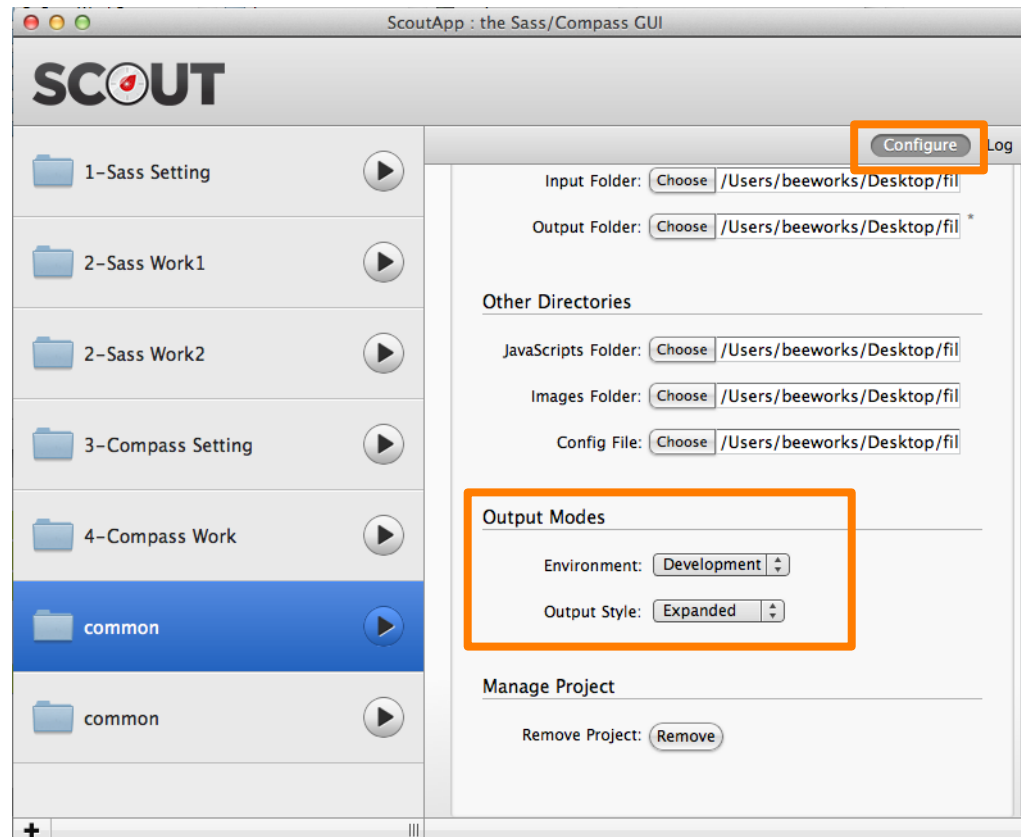
/css/フォルダにstyle.cssが生成されたら/index.htmlをブラウザで確認。

ここから納品用データの作成に入ります。

Work3.納品時の作業 - CSSファイルの圧縮

■CSS出力方法の指定

Scoutの設定に戻って、同じプロジェクト内のconfigureの画面に切り替え→[Output Modes]を設定します。



Work3.納品時の作業 - CSSファイルの圧縮

■ Output Modesについて

Environment: :CSSのコメント表示方法

```
/* line 19, ../scss/_layout.scss */
```

```
#nav {  
  margin-bottom: 20px;  
}
```

```
/* line 23, ../scss/_layout.scss */
```

```
#nav #globalNav {  
  float: left;  
  width: 16.66667%;  
}
```

Development: 表示あり

Production: 表示なし

Output Style: Expanded

:改行／インデント位置の設定

Nested

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

Compact

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

Compressed

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

Expanded

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

開発用

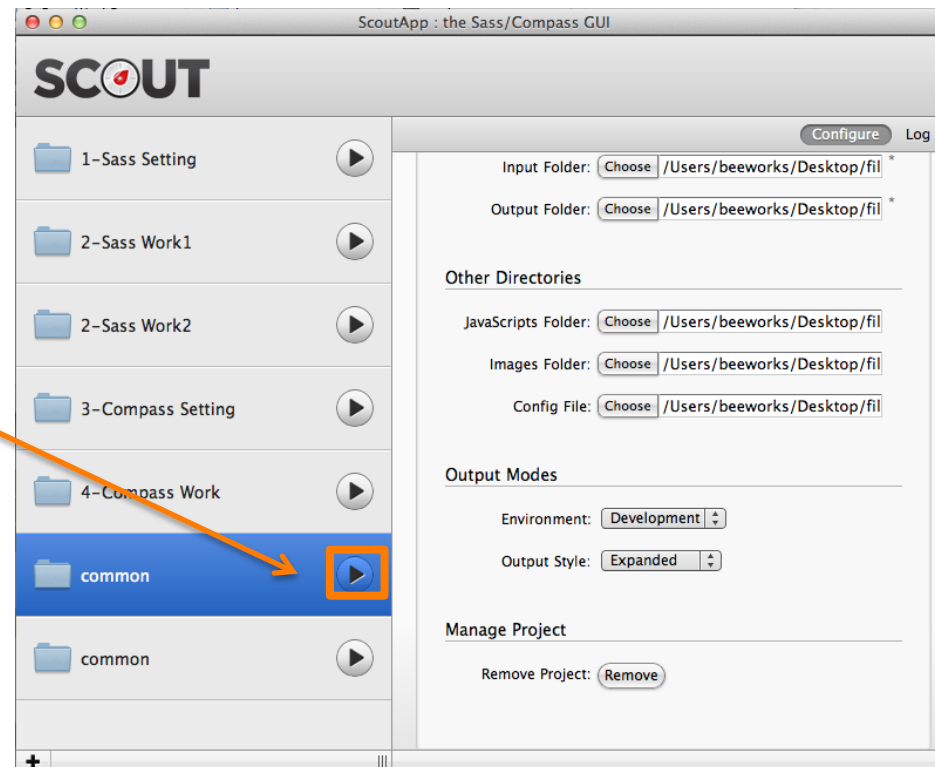
納品用

Work3.納品時の作業 - CSSファイルの圧縮

■では、以下の条件で変換作業開始

Enviroment: Production
Output Style: Compresed

設定変更後は一度実行を
停止してから、再度実行



Work3.納品時の作業 - CSSファイルの圧縮

制限時間: 3min

Work3.納品時の作業 – 不要ファイルの削除

■ 以下はブラウザでの表示には関係なく、納品時には不要ファイルとなるので削除（Sassファイルを納品するケースでは別）。

/fileset/5-Last Work/common/config.rb	・・・設定ファイル
/fileset/5-Last Work/common/scss/	・・・Sassファイル
/fileset/5-Last Work/common/img/sprite01/	・・・Sprite元画像

CSS圧縮と不要ファイル削除が終わったらブラウザでページの表示崩れやエラーが無いかを再度確認し、問題無ければ
/_納品用データ/に一式をコピーして完了。

全体のまとめ

- Sass/Compassは非常に有用であり、今後のCSS構築の標準的な手法となる可能性が高い
- この手法はサイト規模が大きくなるほど効果が高い。(モジュールタイプの大規模サイトなど)
- CSSガイドラインと同様に、複数人で運用する際のルールが必要。

書籍、参考サイト紹介

▼書籍紹介「Web制作者のためのSassの教科書」

<http://book.scss.jp/>

現場で使えるテクニックなどたくさん書かれています。

▼参考サイト:

・ドットインストール

http://dotinstall.com/lessons/basic_sass

http://dotinstall.com/lessons/basic_compass

・Slideshow - WebデザイナーのためのSass/Compass入門

先生: 石本 光司

<http://www.slideshare.net/schoowebcampus/ishimoto>

ひとまず講習はここまで

[Work4]追加課題

ここからはさらにSassを触ってみたいという人向けの追加課題です。

1.スタイルの追加(難易度★★★☆☆)

/fileset/6-Lesson1/

2.レスポンス対応(難易度★★★★★)

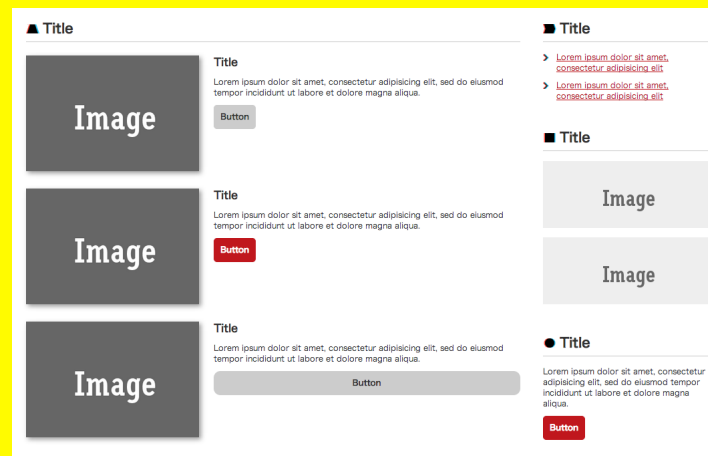
/fileset/6-Lesson2/

追加課題1.スタイルの追加

追加課題1.スタイルの追加

■ /fileset/6-Lesson1/を使用します。
まずは直下にあるHTMLをブラウザで確認して下さい。
/5-Last Work/で使ったものと同じサイトです。

こちらに対して、以下のデザイン変更を適用してください。



/fileset/変更画面イメージ.png

追加課題1.スタイルの追加

■デザイン変更内容の詳細:

- ロゴ画像差替え(/_使用素材/に格納)
- メインコンテンツにあるボタンのうち2番目と3番目のスタイルを変更
 - 2番目:ボタンの背景色をテキストリンクと同じ赤色にして、文字色を白にする
 - 3番目:ボタンを幅いっぱいにして、角丸を10pxにする
- サイドバーにセクションを追加(丸アイコンは/sprite01/に格納済み)
- ボタンのスタイルをサイト共通化

追加課題1.スタイルの追加

■作業のヒント:

- スタイルを追加する際、既存のボタンのスタイルを生かすため `class="btn btn1"` 等とクラスを追加し、スタイルの差がある分だけ `.btn1` に指定すると良いです。
- ブラウザ上でスタイルが当たっている箇所をインスペクタで確認し、`style.css` を表示させればスタイルが当たっているSassファイル内の該当行がわかります。

追加課題1.スタイルの追加

解答

/fileset/6-Lesson1/_answer/

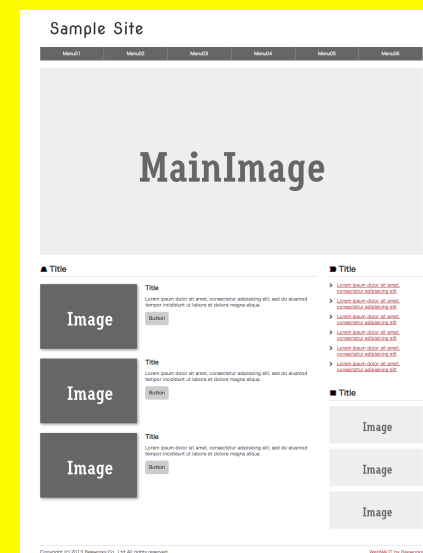
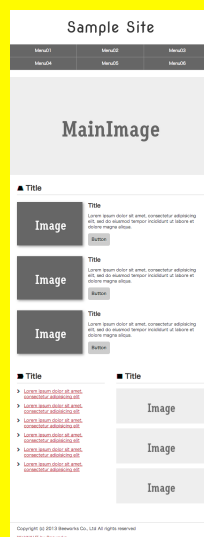
追加課題2.レスポンス対応

90

追加課題2.レスポンス対応

■ /fileset/6-Lesson2/を使用します。
まずは直下にあるHTMLをブラウザで確認して下さい。
/5-Last Work/で使ったものを一部レスポンス化したサイトです。

これを/_answer/index.htmlと同じ動きをするように全体をレスポンス化してください。



91

追加課題2.レスポンシブ対応

■やり方:

mixin.scssでレスポンシブ対応用のミックスイン"media"を用意しているので、そちらを使用します。

次ページにて解説→→→

■ mediaミックスイン解説: スタイル指定例

```
.selector{  
  //PC幅のスタイル  
  font-size: 18px;  
  @include media(tab) {  
    //タブレット幅(1024px〜768px)のスタイル  
    font-size: 16px;  
  }  
  @include media(sp) {  
    //スマホ幅(768px〜480px)のスタイル  
    font-size: 14px;  
  }  
  @include media(iphone) {  
    //iPhone幅(480px以下)のスタイル  
    font-size: 12px;  
  }  
}
```

これを幅毎にスタイルを変化させたい箇所に書き加えながらレスポンシブ化させてください。

追加課題2.レスポンシブ対応

■一部適用済みのものは、layout.scss内のソースにコメントでヒントを入れています。

- ヒント1: tab幅以下ではコンテンツ幅がリキッドになる
- ヒント2: tab幅以下ではロゴがセンター寄せになる
- ヒント3: tab幅以下では左右マージン追加
- ヒント4: sp幅以下では左右マージン0になる

追加課題2.レスポンス対応

解答

/fileset/6-Lesson2/_answer/

Thank you!



↑ Design

**あなたは見分けられる!? 定番
フォント代わりに使える
Google Fonts 27書体** (2014.1.
6)

あけましておめでとうございます! 定番フォント
を愛してやまない、デザイナーTです。今月から
TypeSq ...



↑ Markup

**【Beeworks新卒採用サイト】
それ、本当に使いやすい? ユー
ザーのためのUI改善【改修して
みた】** (2013.12.27)

2013年11月29日、新卒採用サイトをオープンし
ました! 実はこのサイト、公開した後にもいくつ
か改 ...



↑ Direction

**【図解】はじめてのGoogle タ
グマネージャ** (2013.12.19)

こんにちは。プランナーKです。今回は、WEB マ
スターならぜひ活用したいツール「Google タグ
マネー ...



↑ Direction

**【Web担当者必見】CMS導
入、これだけ押さえる! デザイ
ン確認3つのポイント**
(2013.12.10)

こんにちは、ディレクターFです。皆さんは仕事
やプライベートでCMS (コンテンツマネジメント
システム) ...



↑ Design

**スクラップブック界の神アプ
plication? INBOARDの魅力を
徹底解説** (2013.11.19)

こんにちは! お気に入りの画像はざっくり整理派
のデザイナーTです。皆さん、アイデアの引き出
しに役立っているス ...



↑ Direction

**コミュニケーションを促進し、
仕事で円滑&楽しくするツール**
(2013.11.19)

こんにちは、プロジェクトを円滑に進める為に情
報を整理し、共有するディレクターHです。メールで
コミュニケーション ...

WebNAUT

by BEEWORKS

Tweet 118 いいね! 772

Category

- Direction
- Design
- Markup
- Develop
- Others

About WebNAUT

株式会社ビーワークスの運営する研究紹介サ
イト。クリエイティブに関する様々な役立
ち情報を通して、ビーワークスのこだわりや
制作プロセスをご紹介します。
[詳しくはこちらから。](#)

Tag

3D Ai API application CMS CSS
Dreamweaver Google Google Analytics
Google タグ マネージャ HTML IA icon
JavaScript jQuery Twitter

Bootstrap UI ux WebNAUT WEB デ
イレクション なめばら アクセス解析
アンケート調査 インタビュー スマート
フォン ツール ディレクション ディレ
クター デザイナー デザイン デザイ
ンプロセス デザイン理論 プロジェクト
管理 モバイル ランキング レスポンシ
ブ ワイヤフレーム ワークフロ
ー 企画 効率化 営業 基礎 情報共有
情報収集 情報設計 新人 正規表現 組
織運営 育成 配色

<http://webnaut.jp/>
WebNAUTbyBeeworks
@WebNAUT_BW