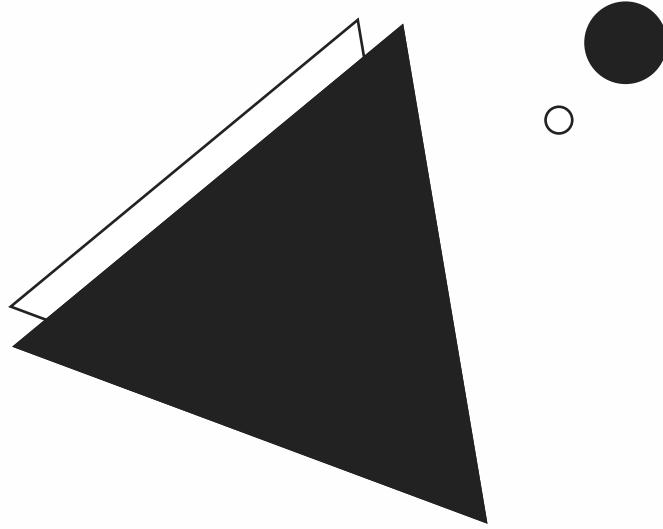


16px

24px



# {ANVIE SOIL

Cording Guidelines  
Style Guide Book

Ver 0.1

---

ANVIE SOILとは

...

---

本仕様について

...

レスポンシブについて

対応OSおよびブラウザについて

実装環境について

---

CSS設計について

...

設計について

ディレクトリ構成について

基本ルールについて

---

HTML

...

設計について

基本ルールについて

ディレクトリ構成について

Pugについて

---

JavaScript

...

設計について

ディレクトリ構成について

---

editorconfigについて

...

### | なにこれ？

ANVIE SOILとは、HTML、CSS、JavaScriptの作業効率最適化及び、メンテナンス性(保守性)向上を目的とした、フロントコーディング規約です。

ANVIE社では原則この規約に乗っ取りコードレビューや質を担保する。

社内外ANVIEの制作に関わる者は本ガイドラインに乗っ取って制作を進めてください。

以下が本規約での「メンテナンス性」の基準となります。

- ・わかりやすい
- ・探しやすい
- ・再利用しやすい
- ・拡張しやすい

参考 : <https://qiita.com/pugiemonn/items/964203782e1fcbb3d02c3>

## | 思想

ANVIE SOILでは、コードの軽量化等よりも、可読性/保守性などに重きを置き作成する。  
**基準として5歳児が見てもわかるコードを意識すること。**  
また、個人の主觀や思想を押し付けはせず、一般的には「こう」するといった一線を意識すること。  
以下基本的な規約

- W3Cに基づいたコーディング
- 対応ブラウザを考慮した機能選定
- コードの拡張性配慮
- SEOへの配慮と意識
- オレオレ記載は撲滅させる

## | 仕様

- サイト文章定義 ...HTML5
- CSS ...CSS3 プリプロセッサはSCSSを採用
- Charsets ...UTF-8を使用する

## | レスポンシブについて

- PC ...1024px以上
- TAB ...769px以上
- SP ...481px以上

## 対応OS及びブラウザについて

{ ANVIE SOIL }

	Ver	Win8	Win10	mac High Sierra	mac Mojave	iOS 12.0系	iOS 13.0系	And8	And9
 Chrome	最新	●	●	●	●	●	●	●	●
 FireFox	最新	●	●	●	●	—	—	—	—
 Safari	最新	—	—	●	●	●	●	—	—
 Edge	最新	●	●	—	—	—	—	—	—
 IE ※要相談	最低 11	—	—	—	—	—	—	—	—

— 選定基準はシェア率を参照する

— IEについては、要相談。  
基本切っていくスタイル  
対応するのであれば、最低11以上

— Google Chromeを標準想定

● ...対応 ▲ ...要相談 — ...未対応

## 環境

- Gulp, WebPack, Git, Github, Scss, Pug, Es6を主に使用し制作する。
- Gulp  
ビルドシステム・タスクランナー
- Webpack  
モジュールバンドラー
- Scss  
CSSのメタ言語 CSSを最大限に拡張する
- Git  
ソースコードの履歴を記録し管理する。バージョン管理システム
- GitHub  
Gitで管理されているソースコードをクラウド上で管理するサービス
- Pug  
HTMLを効率的に書くためのjs製テンプレートエンジン
- ES6  
ECMAScriptの新しい規格...新しいjavascriptの書き方
- その他  
この規約は「吉本式設計」にかなりインスピアされています。

## | ディレクトリ構成について

全体的なディレクトリ構成になります。

srcDir (frontend Dir) のものが、assetsDirにコンパイルされる。

原則xxx.htmlファイルはassets以下を読み込む。

assets Dir = サイトの装飾に必要なアセット

### DIR ...コンパイル後の想定

```
└── .editorconfig  
└── src(frontend)  
    └── assets  
        ├── css  
        │   └── main.css  
        ├── js  
        │   └── main.js  
        └── img  
            └── sample.jpg  
└── html  
    └── index.pug  
└── gulp  
    └── gulpの設定ファイル等  
└── gulpfile.js  
└── package.json  
└── index.html
```

## | ファイル命名規則について

- 半角英数字を使う
- 数字から始めない
- 単語続きは「\_」で区切る
- 「その」ページを表すファイル名にする(htmlなど)
- 機能を表す簡潔な名前にする
- モジュール化のファイルは文頭「\_」をつける.scss)

## | 画像ファイル命名規則について

- 機能ごとにディレクトリを分け、  
命名は性質を表す命名にする。
- レスポンシブの場合、文頭にプレフィックスをつける
- 命名の順番は、「性質\_レスポンシブ.jpg」など

```
img
  └── logo
      └── logo.svg
  └── icon
      └── icon_twitter.svg
      └── icon_arrow.svg
      └── icon_arrow_sp.svg
  └── page
      └── index
          └── index_mv.jpg
      └── about
          └── about_xx.jpg
  :
```

## | 設計について

ディレクトリ構成はFLOCSSを採用、命名規則にはchainable BEMを使用する。

本規約は「わかりやすい」「メンテしやすい」を軸に制作します。

## | 基本ルール

— IDは使用しない。ただしJs等で要素に指定する場合は可。

スタイルを当てるのは禁止する。

— inlineでのスタイルは原則禁止

— importantは禁止。ただしプラグインなどをどうしても打ち消す際などには可。

コメントを残すこと。

— IDについて

基本的にjsで使う想定から、単語が2つ続くときはキャメルケースで書く。

その他クラス命名規則はBEMの説明欄をご覧ください。

## | ディレクトリ構成について

### - FLOCSS

cssの設計思想の一つで、OOCSS、SMACSS、BEMなどのコンセプトが取り入れられている。

FLOCSSの基本構成は、Foundation、Layout、Objectになります。

### - Foundation

制作における基本的なスタイルを定義する。

reset.scssやbase.scssなど。

### - Layout

サイト共通となるレイアウトBlock要素。

header.scss、footer.scssなど

### - Object

本来ObjectをさらにComponent、Project、Utilityの3つに分類するが、

本規約ではComponent、Projectのみとする。

理由はUtilityとは便利クラスという位置付けのため、これを採用するとコードが  
ブラックボックス化する恐れがあるため。

...便利 = ちょっと足りない部分とかのクラスを用意する→

わかりにくく→本軸とズレる

### SCSS

```
├─ foundation
│  ├─ _base.scss
│  ├─ _reset.scss
│  └─ :
└─ layout
    ├─ _header.scss
    ├─ _footer.scss
    ├─ _navigation.scss
    └─ :
└─ object
    ├─ component
        ├─ _button.scss
        ├─ _hamburger.scss
    └─ project
        ├─ _index.scss
        ├─ _about.scss
```

## CHAINABLE BEMについて

B...Block E...Element M...Modifier

従来のBEMのModifierに対する課題を解消した命名ルールです。

クラス名の基本構成は、[接頭辞]-[Block]\_\_[Element]\_-[Modifier] となる。

原則ケバブケースで書くこと。

### 接頭辞

セクション、ブロックにあたる要素にクラスを付ける際に、接頭辞を付けることで、どういった役割をもっているのか、どのCSSファイルに記述されているのか、どれぐらいの詳細度をもっているのかをクラス名だけで判断できるようになる。

FLOCSSを採用しているので、各ディレクトリのプレフィックスをつける。

```
l-header {  
  color: #fff;  
}
```

```
c-button {  
  color: #fff;  
}
```

```
p-about {  
  color: #fff;  
}
```

### BLOCK

ページ構成のルートとなる要素を表します。

### ELEMENT

「Element」は必ずしも、親要素となる「Block」の子要素に使用するわけではなく、あくまでも「Block」の構成要素として使用します。「Element」に子要素がある場合は、その「Element」が「Block」になります。

### Modifier

BとEの状態の変化を表すもの

さらに詳しい記載は下記からご確認ください

<https://bit.ly/33jThpP>

## | 基本ルール

- 小文字で記述する。
- HTMLタグは必ず小文字で記述する。
- 可動性を重視するのであれば、親要素で囲むことも可。
- インデント
  - 半角スペース2
  - CMSなどによって、動的に生成される箇所については可。
- コメントアウト
  - セクションごとにコメントを残すこと。

```
<!-- MV -->  
<div class="p-index__mv">...  
<!-- MVここまで -->
```

## | 静的LP等の場合

- テンプレートエンジンを使用する(pug)

## | WPの場合

- phpを使用

## | ディレクトリ構成について

### - layout

ページのレイアウトに関するhtmlを記  
\_page.pugで要素をincludeしてhtmlを生成する。

### - mixin

便利関数をまとめる場所

### html(Pug)

```
├── _layout
│   ├── _page.pug
│   ├── _head.pug
│   ├── _header.pug
│   └── ...
├── _mixin
│   ├── _xxx.pug
│   └── ...
└── index.pug
└── about.pug
```

## | Pugについて

読み方は(パグ) 犬のパグと一緒に。

元々「jade」と呼ばれる言語だったが、商標登録等問題が起きPugへ変更された。

以下特徴として、

- 閉じタグが不要
- ファイルを分割できる
- 変数等プログラムちっくな書き方ができる
- 要素をテンプレート化できる
- cssと書き方を統一できる
  - classは「.」、idは「#」...と書くことができる
- 閉じタグが無いため、インデントして記載していく

### 参考

<https://qiita.com/soarflat/items/a80bdca813ae83bc9ebc>

[https://qiita.com/cotolier\\_risa/items/135d168eddd6ae6c3409](https://qiita.com/cotolier_risa/items/135d168eddd6ae6c3409)

[https://www.tam-tam.co.jp/tipsnote/html\\_css/post12789.html](https://www.tam-tam.co.jp/tipsnote/html_css/post12789.html)

### Pug

```
.l-wrapper
  main.p-index
    .p-index__mv
      h1.p-index__mv-heading
        |タイトル
```

### html

```
<div class="l-wrapper">
  <main class="p-index">
    <div class="p-index__mv">
      <h1 class="p-index__mv-heading">
        タイトル
      </h1>
    </div>
  </main>
</div>
```



## | 設計について

JavaScriptはできる限り外部ファイル化する。  
ページ、機能ごとにファイル、モジュールを分ける。  
必要に応じてページ単位で機能を呼び出す。  
最終的にbundleしたものをhtml側で読み込む。

- 秒数、クラス名、window処理中に変更がないものについては  
変数(変数名は大文字)に入る (オブジェクト管理でも良い)
- キャamelケースで書く
- 規格はES6
- class構文ベース
- 外部ライブラリ、プラグインを使用する場合は、使用箇所にコメントを残す
- 基本的にはモジュール管理だが、必要に応じてCDNも可
- jQueryを使用する場合一貫して使用すること。

## | ディレクトリ構成について

### - main.js

entry部分すべてをまとめて分配するファイル  
どのページに振り分けるのかなどを記載する

### - init

modulesで作ったものをまとめて、main.jsへ中継する場所

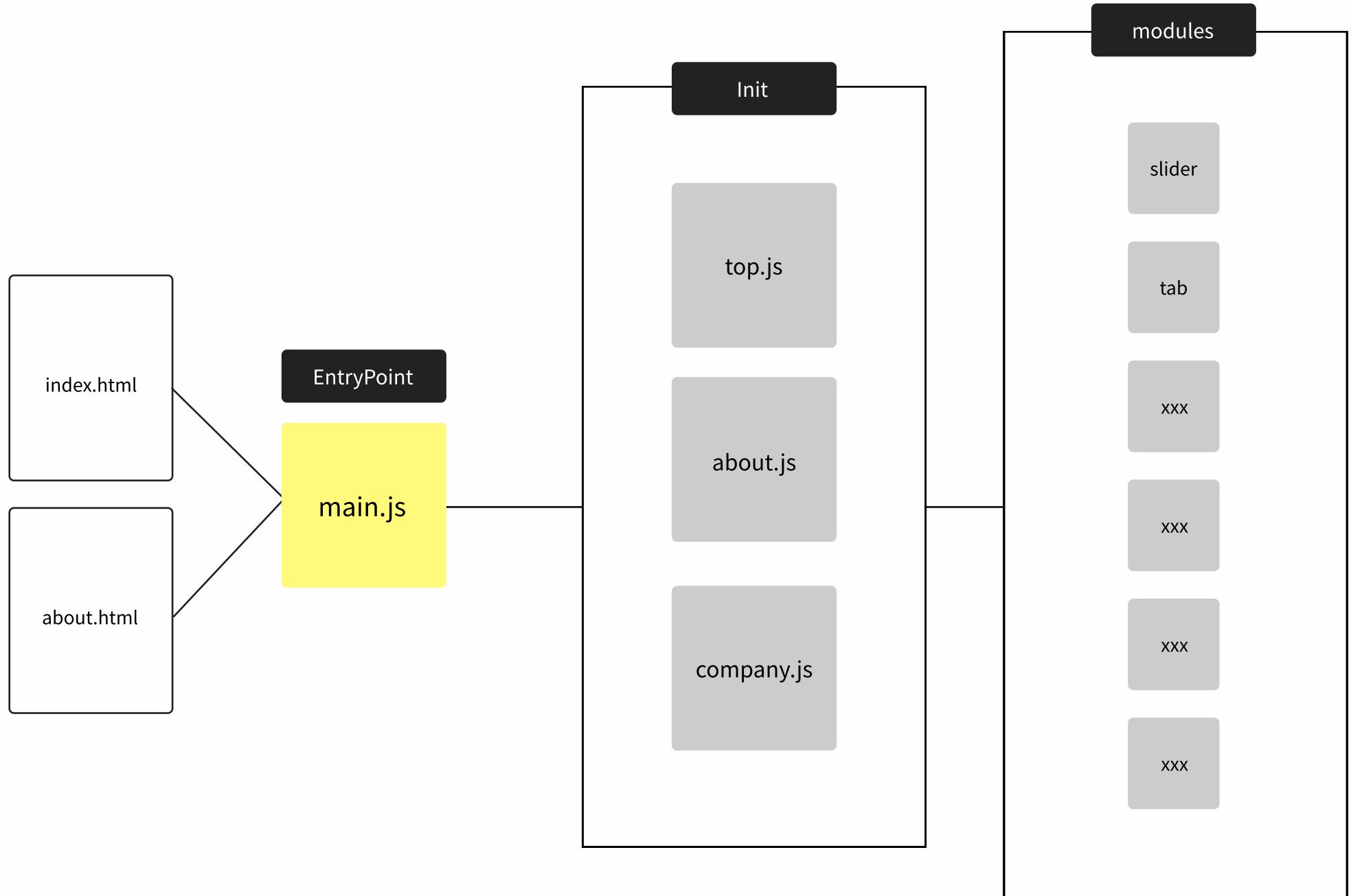
### - modules

各機能ごと、component単位で作る  
ディレクトリ名を機能名に、機能のエントリーをindex.jsとする。  
もし複数種類あった場合、機能ごとのファイルを作成し、index.jsで読み込む



js

```
├── init
│   ├── common.js
│   └── page
│       ├── top.js
│       └── about.js
└── modules
    ├── slider
    │   └── index.js
    └── tab
        └── index.js
└── main.js
```



## | EditorConfig

エンジニア界隈では謎に宗教文化が存在しその宗派は無限にあります。  
コードの統一化という軸に対して矛盾してしまうことが多々発生します。

”インデントいくつ？、タブ派？、Vim? Atom? VS CODE? WebStorm?”

使用しているエディターやタブ数などの違いがあるので、  
命名等ルールを統一したとしても、ツールや思想の違いが出てしまい結局統一性が保てません。  
それら思想の違いで、割とマジで喧嘩や人間関係悪化にまで発展することがあります。  
それらの負「editorcongif」は解決できます。(多分...)

タブ数xx個、改行コード○○...など記載したファイルを読み込むことで、  
どの環境でも同じコードを書く環境が整います。

終わり

{ANVIE SOIL

---

| END

Github リポジトリ

[https://github.com/anviedd/ANVIE\\_SOIL](https://github.com/anviedd/ANVIE_SOIL)