

# Enduring CSS

高津戸壯 @Takazudo

# 自己紹介

- 高津戸壯 (たかつど たけし)
- 株式会社ピクセルグリッド
- フロントエンドエンジニア
- @Takazudo
- エディタ: MacVim



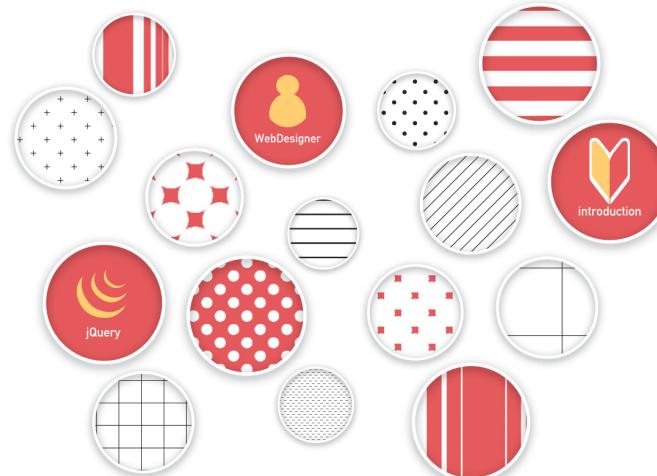
著者: takazudo@gmail.com

[改訂版]

# Webデザイナーのための jQuery入門

魅力的なユーザーインターフェースを手軽に作る

高津戸壯／小原司



「何かを取ってきて…」  
「それに何かする!」

jQueryの考え方は超簡単

HTML、CSSをマスターしたら読んでみよう!  
アコーディオン、タブコンテンツ、スライドショー、タッチ、スクロール  
いろいろなサンプルを作りながらしっかり学べる

技術評論社



# アンケートで いただいたお悩み

- class命名のコツ
- CSSを複数人で触る際の設計
- 部分的にデザインが少し異なるパートが出てきた時のモジュール化方法
- モジュール化していく際、どこまでをモジュールとして考えればよいか
- 設定が複雑になるにつれ、チーム内で共有するハードルがあがり、活用できない

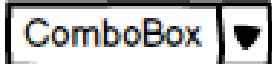
# なぜCSS設計論が必要なのか？

- どのようにでも書ける
- 知らないスタイルが当たる
- 上書き地獄
- チーム間での意思疎通

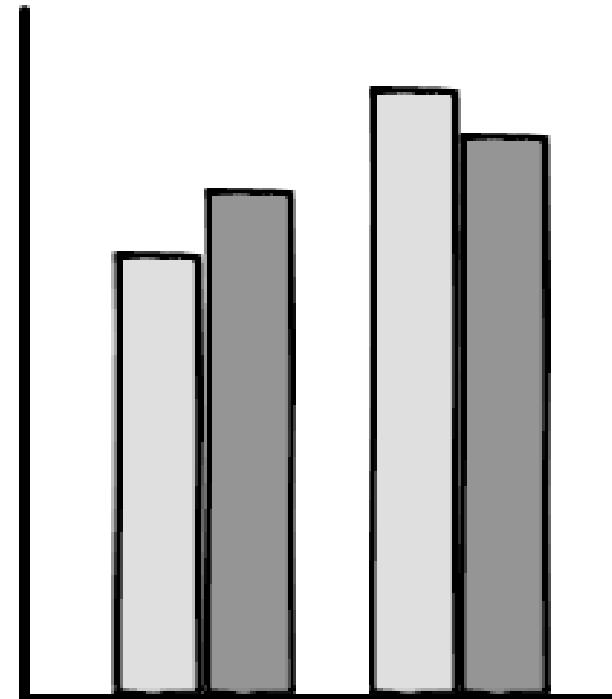
→ OOCSS, SMACSS, BEM

# 大体共通している概念

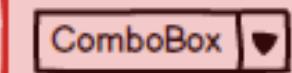
- モジュール化
- 命名規則
- 繙承的な話



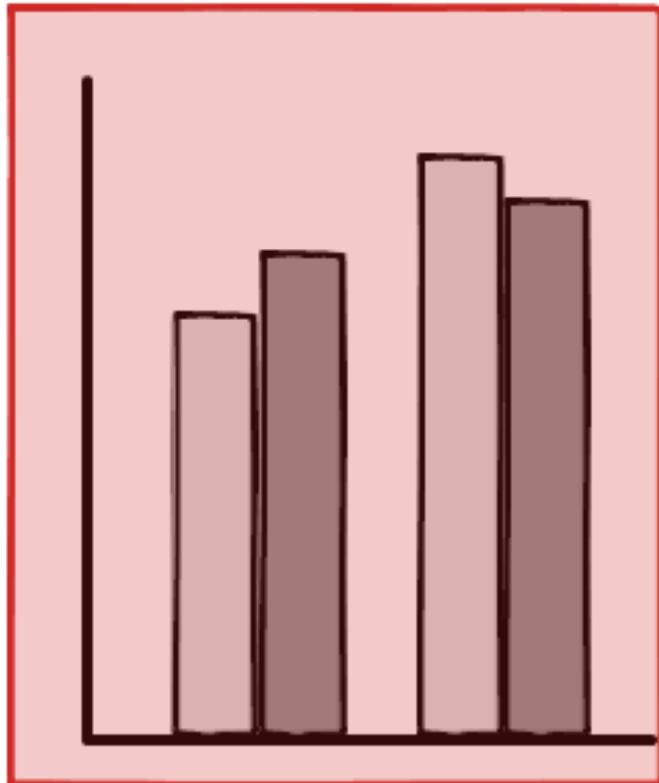
Name (job title)	Age	Nickname	Employee
Giacomo Guilizzoni Founder & CEO	37	Peldi	<input type="radio"/>
Marco Botton Tuttofare	34		<input checked="" type="checkbox"/>
Mariah MacLachlan Better Half	37	Patata	<input type="checkbox"/>
Valerie Liberty Head Chef	:)	Val	<input checked="" type="checkbox"/>
Guido Jack Guilizzoni	6	The Guids	<input type="checkbox"/>



The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog.

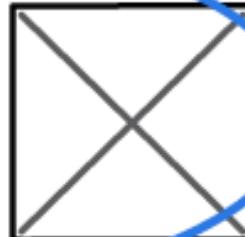


Name (job title)	Age	Nickname	Employee
Giacomo Guilizzoni Founder & CEO	37	Peldi	<input checked="" type="radio"/>
Marco Botton Tuttofare	34		<input checked="" type="checkbox"/>
Mariah MacLachlan Better Half	37	Patata	<input type="checkbox"/>
Valerie Liberty Head Chef	:)	Val	<input checked="" type="checkbox"/>
Guido Jack Guilizzoni	6	The Guids	<input type="checkbox"/>

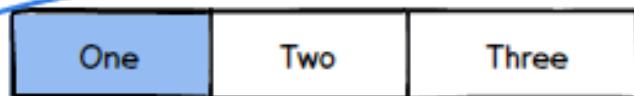


The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog.

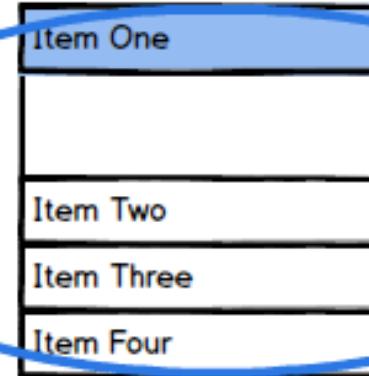
The quick brown fox jumps over  
the lazy dog. The quick brown  
fox jumps over the lazy dog. The  
quick brown fox jumps over the  
lazy dog. The quick brown fox  
jumps over the lazy dog.



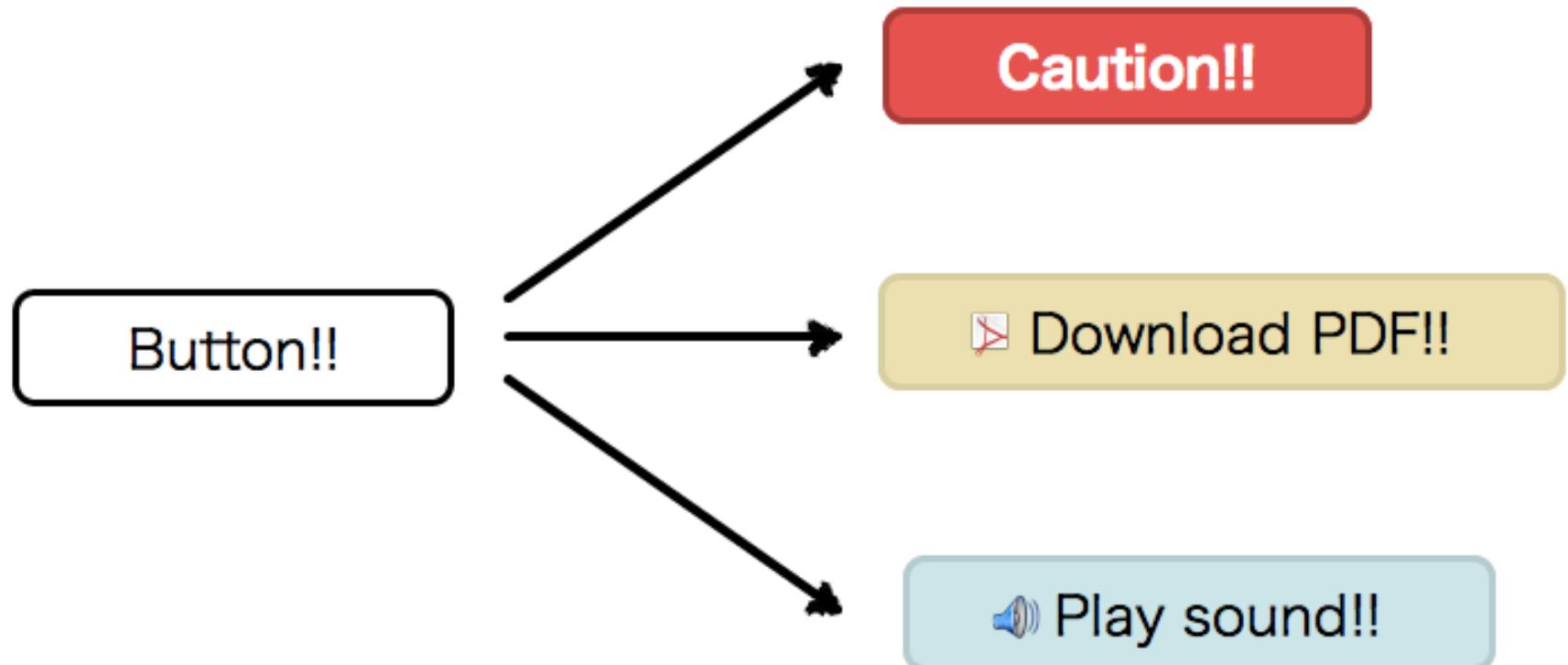
◀ FEB 2008 ▶						
S	M	T	W	T	F	S
1	2					
3	4	5	6	7	8	9
10	11	12	13	M	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	



[Home](#) > [Products](#) > [Xyz](#) > [Features](#)



- block
- block\_element
- block\_element\_modifier
- dialog
- dialog\_close-button
- dialog\_menu
- dialog\_menu\_hidden



```
<button class="button button_caution">...</button>
<button class="button button_pdf">...</button>
<button class="button button_play">...</button>
```

# すべてが解決するか？

- どうモジュールを切ればいいのか
- class名をどうつければいいのか
- 複雑になってしまうのだが？

具体的にどう設計するかっていう話はまた別

eCSS

# Enduring CSS

Architect and maintain large-scale  
CSS codebases

**Ben Frain**

# Enduring CSSとは

- Ben Frainの著書
- Enduring: 長続きする、永続的な、不朽の、我慢強い、辛抱強い
- 設計方法論＋アドバイス集的な内容
- 考え方を必要に応じて取り込めば良さそう

# ECSSいわく

自分には既存の設計論がぴったりマッチしなかった

- 複雑になりすぎでは？
- サイトがスケールしたら結局破綻するのでは？
- パフォーマンス突き詰めすぎでは？



- こう考えたらよいのでは？というもの
- Webアプリを想定している

# ECSSの基本的な考え方

- ・全体で一つとして考えない
- ・分けて考える

1. モジュール化
2. 名前空間
3. アセットの分離
4. クラス名の命名規則

# 1.モジュール化

- モジュールとは一体？

# ECSSでのモジュール

ざっくり言うと…

最も大きい単位の機能領域

具体的にどう分けてるの？

<http://ecss.io>

- トップページ
- チャプタページ
- 目次

- ・そんな細かくはしない
- ・ネストしたりもなるべくしない

## BENEFITS OF ENDURING CSS



### Isolated, self-quarantining styles

Learn how to isolate and contain styles so modules become self-quarantining and simpler to transport into different contexts and projects



### Modern Tooling

Learn how to use PostCSS and modern linting tools to analyse and maintain CSS across a large codebase



### Scale To Any Size

Whether one developer or one hundred, ECSS provides a dependable and maintainable approach to writing style sheets



### Avoiding abstraction and specificity

Understand the disadvantages of specificity and abstraction on large scale responsive projects and why 'flat' specificity and style isolation may be preferential



### One key selector to rule them all

Understand how to write a CSS superset that will provide a 'single source of truth' for every key selector in your project



### Handling state

Consider how to handle state more accessibly with ARIA attributes as opposed to class changes

ECSS

## BENEFITS OF ENDURING CSS



### Isolated, self-quarantining styles

Learn how to isolate and contain styles so modules become self-quarantining and simpler to transport into different contexts and projects



### Modern Tooling

Learn how to use PostCSS and modern linting tools to analyse and maintain CSS across a large codebase



### Scale To Any Size

Whether one developer or one hundred, ECSS provides a dependable and maintainable approach to writing style sheets



### Avoiding abstraction and specificity

Understand the disadvantages of specificity and abstraction on large scale responsive projects and why 'flat' specificity and style isolation may be preferential



### One key selector to rule them all

Understand how to write a CSS superset that will provide a 'single source of truth' for every key selector in your project



### Handling state

Consider how to handle state more accessibly with ARIA attributes as opposed to class changes

## BENEFITS OF ENDURING CSS



### Isolated, self-quarantining styles

Learn how to isolate and contain styles so modules become self-quarantining and simpler to transport into different contexts and projects



### Modern Tooling

Learn how to use PostCSS and modern linting tools to analyse and maintain CSS across a large codebase



### Scale To Any Size

Whether one developer or one hundred, ECSS provides a dependable and maintainable approach to writing style sheets



### Avoiding abstraction and specificity

Understand the disadvantages of specificity and abstraction on large scale responsive projects and why 'flat' specificity and style isolation may be preferential



### One key selector to rule them all

Understand how to write a CSS superset that will provide a 'single source of truth' for every key selector in your project

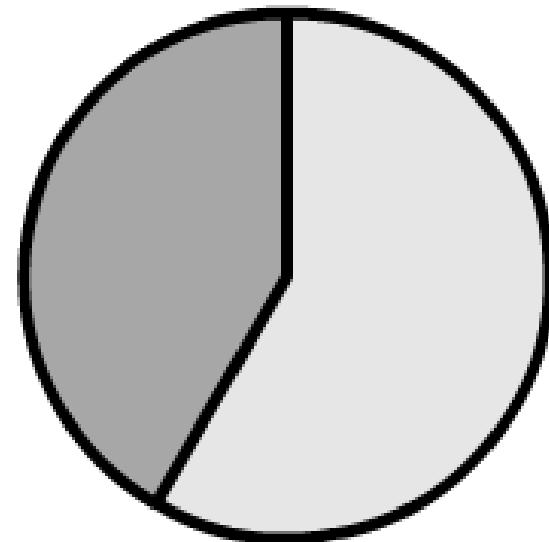


### Handling state

Consider how to handle state more accessibly with ARIA attributes as opposed to class changes

# Super Big Heading!!

The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog.



# Super Big Heading!!

Name (job title)	▲ Age	Nickname	Employee
Giacomo Guilizzoni Founder & CEO	37	Peldi	<input checked="" type="radio"/>
Marco Botton Tuttofare	34		<input checked="" type="checkbox"/>
Mariah MacLachlan Better Half	37	Patata	<input type="checkbox"/>
Valerie Liberty Head Chef	:)	Val	<input checked="" type="checkbox"/>
Guido Jack Guilizzoni	6	The Guids	<input type="checkbox"/>

# 同じようなものも別物 なぜか？

- そっちのほうが分かりやすいから
- 複雑化を防ぐことができるから
- 同じようなCSSを何度も  
書かないといけないのでは？
- それでもいい。gzipとかすればいい

## 2.名前空間

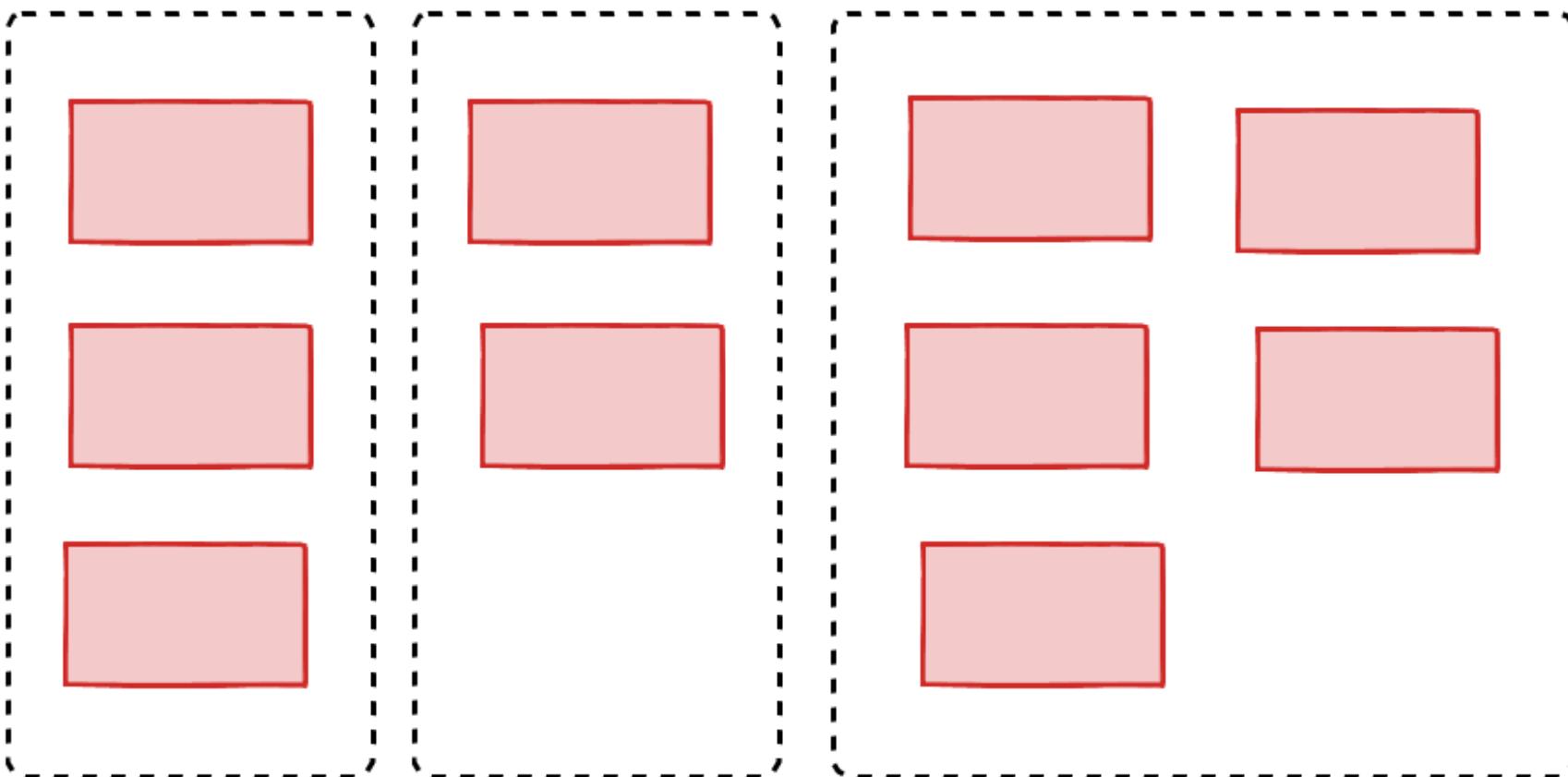
- モジュールを名前空間でグルーピングする

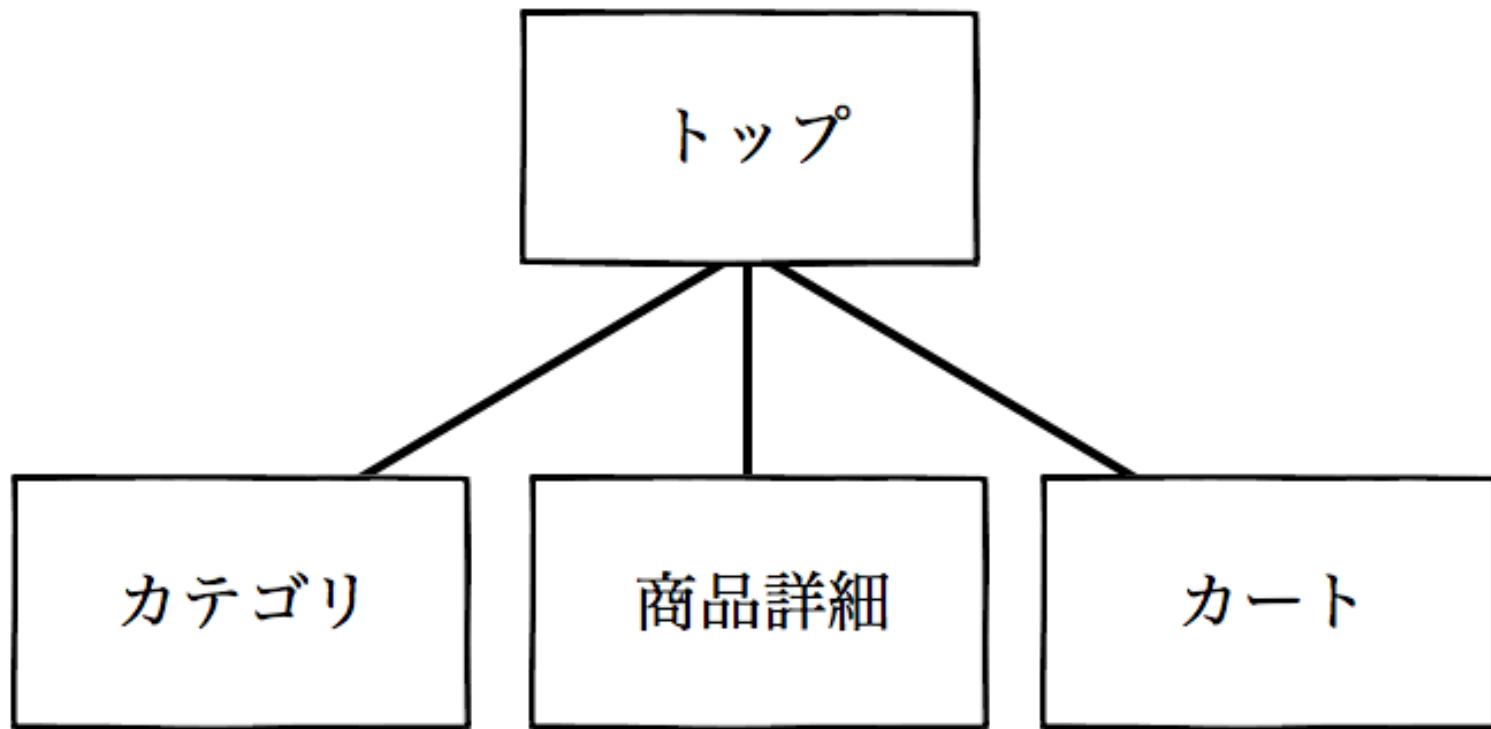


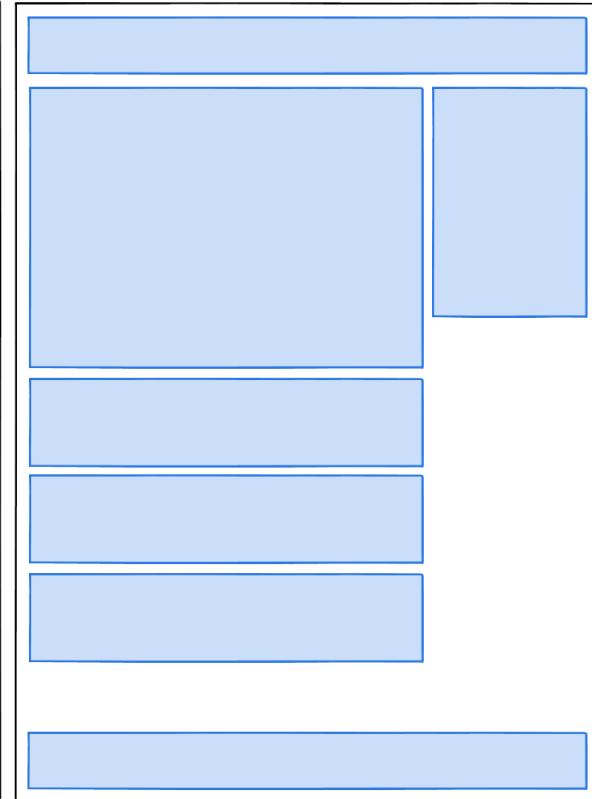
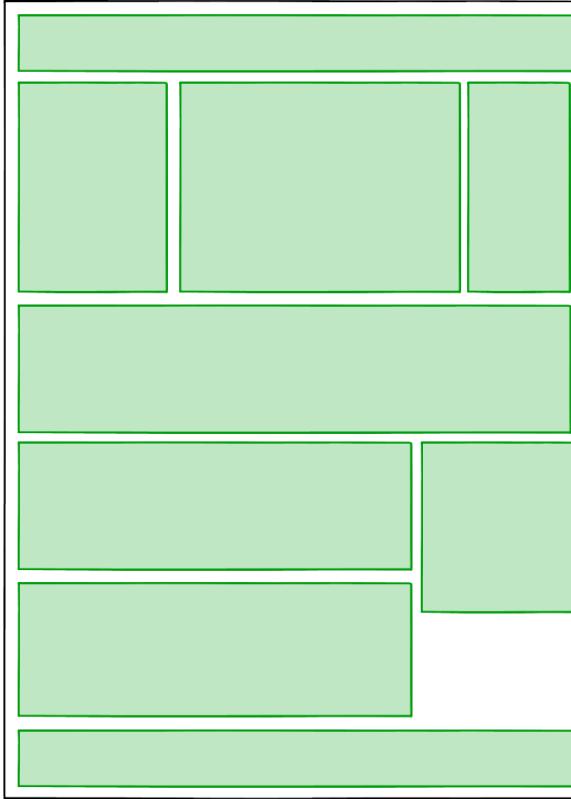
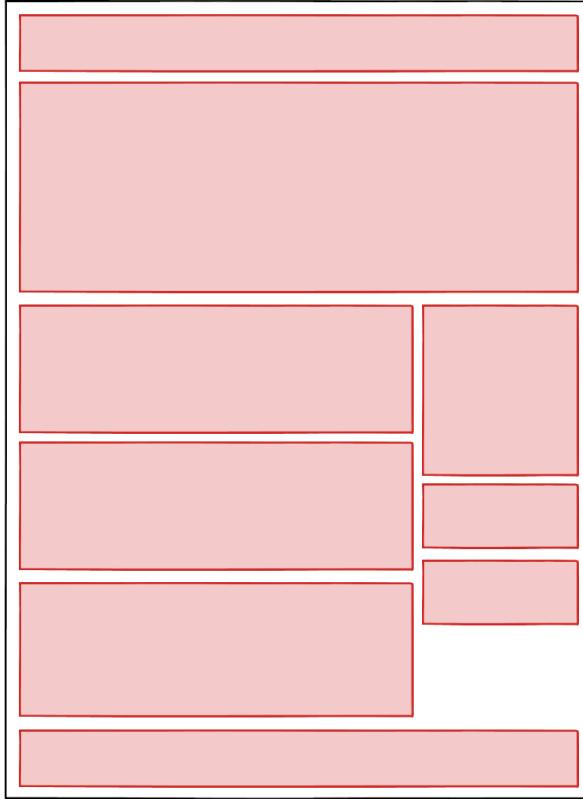
**A**

**B**

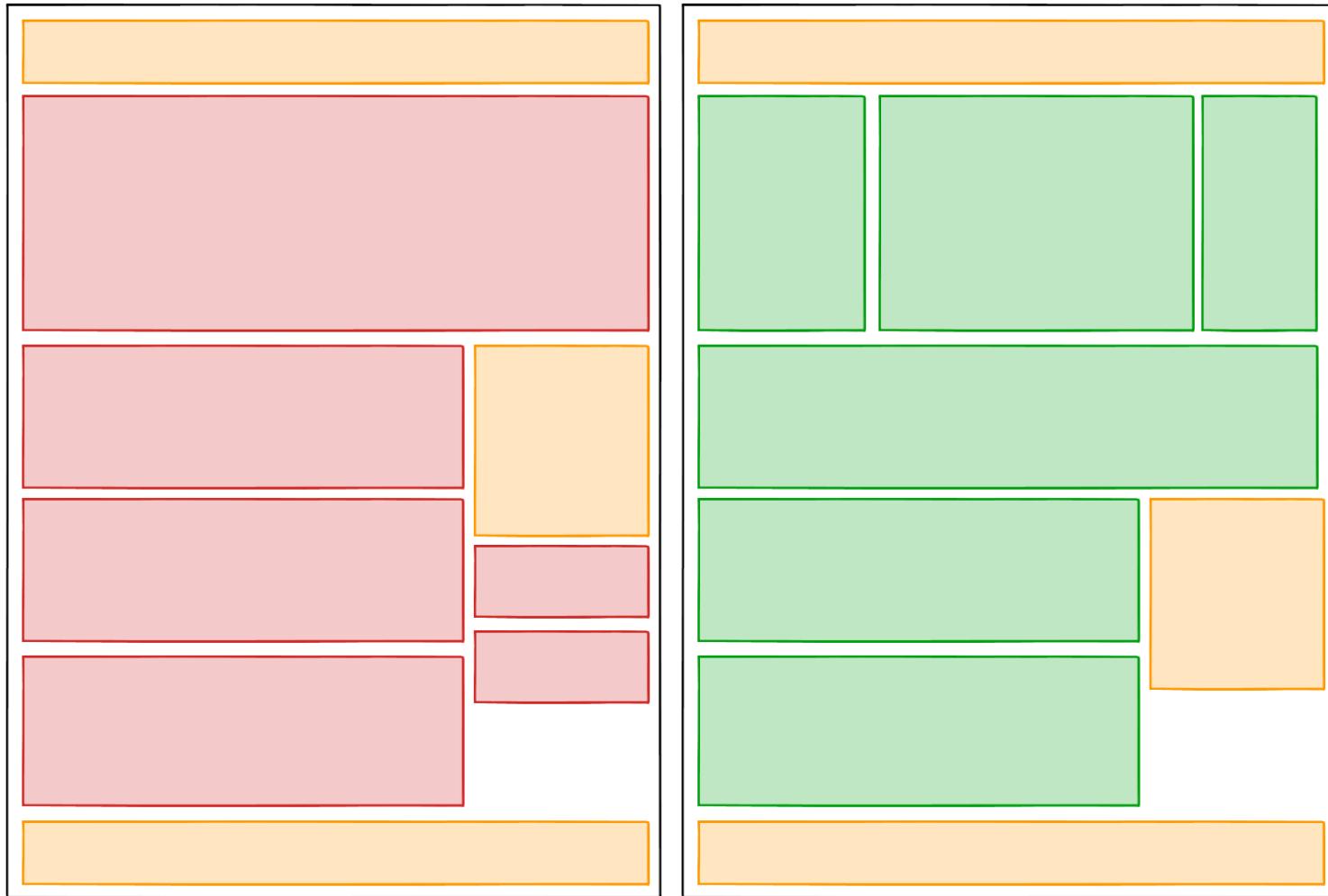
**C**







topPage, productDetail, shoppingCart



common

# 名前空間の分け方例

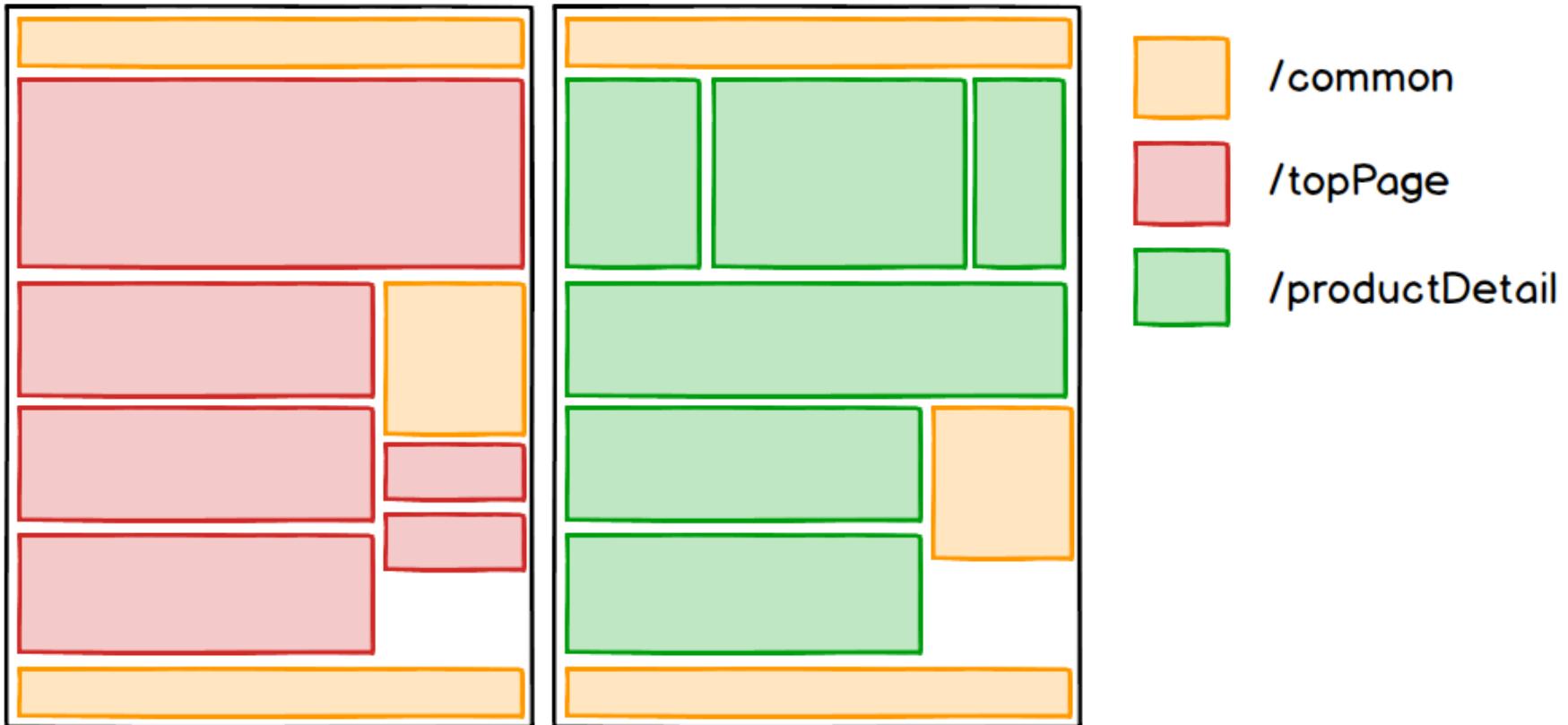
- CMSで出しているテンプレートが違う
- ここだけWordPressで出している
- サイト全体で共通
- 管理する部署が違う

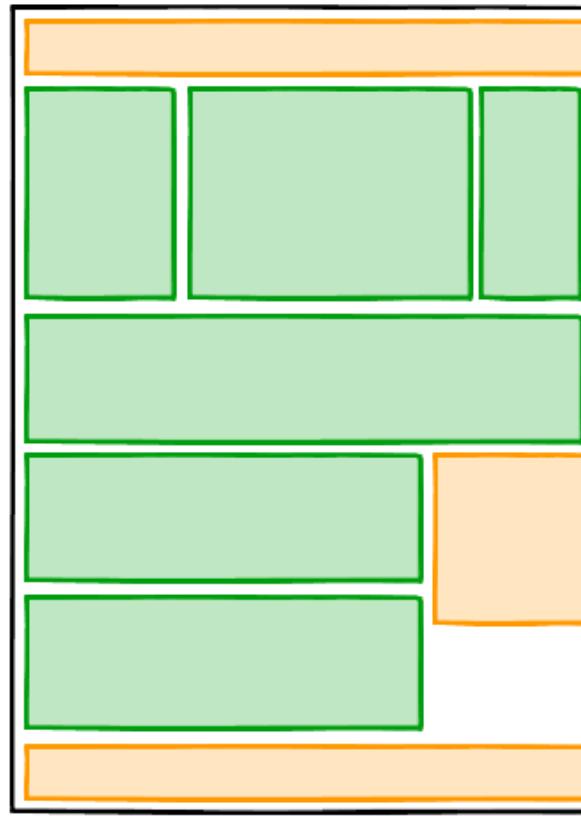
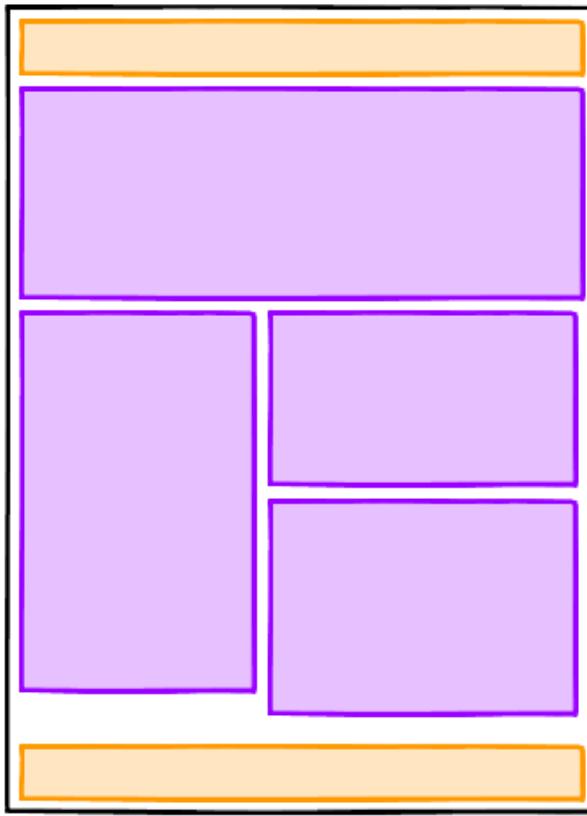
# 3.アセットの分離

- 名前空間ごとに読み込むCSS、JS、画像類を完全に分離

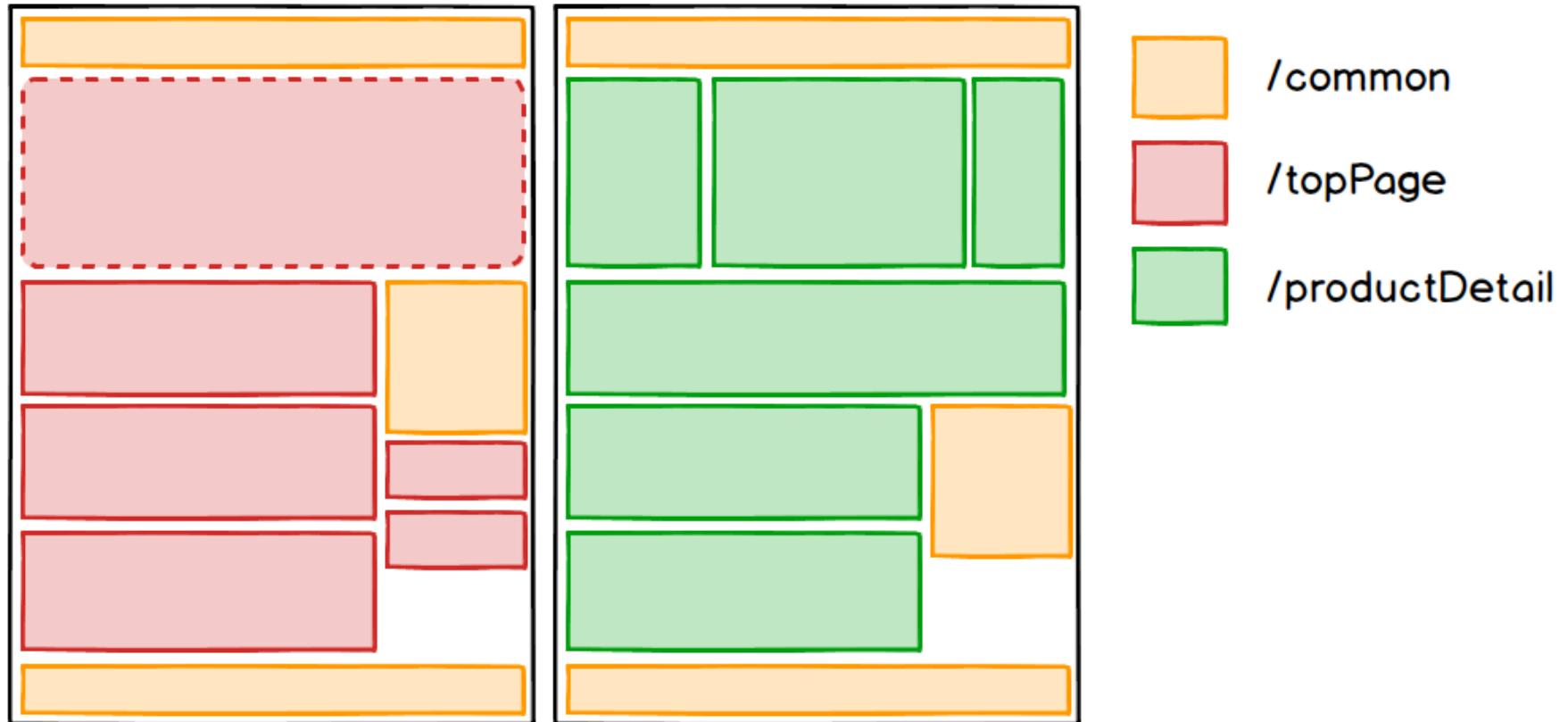
# 分離すると何が嬉しい？

- CSS、JS、画像の干渉を防げる
- 名前空間単位でまるごと消せる
- どのファイルをいじればいいのか明白
- どこにファイルを置けばいいのか明白





- /common
- ~~/topPage~~
- /productDetail
- /topPage2



# 分離しなかったら？

- このCSS消していいんだろうか…
- この画像どこに置けばいいんだろう…
- このモジュール変更するのが怖い

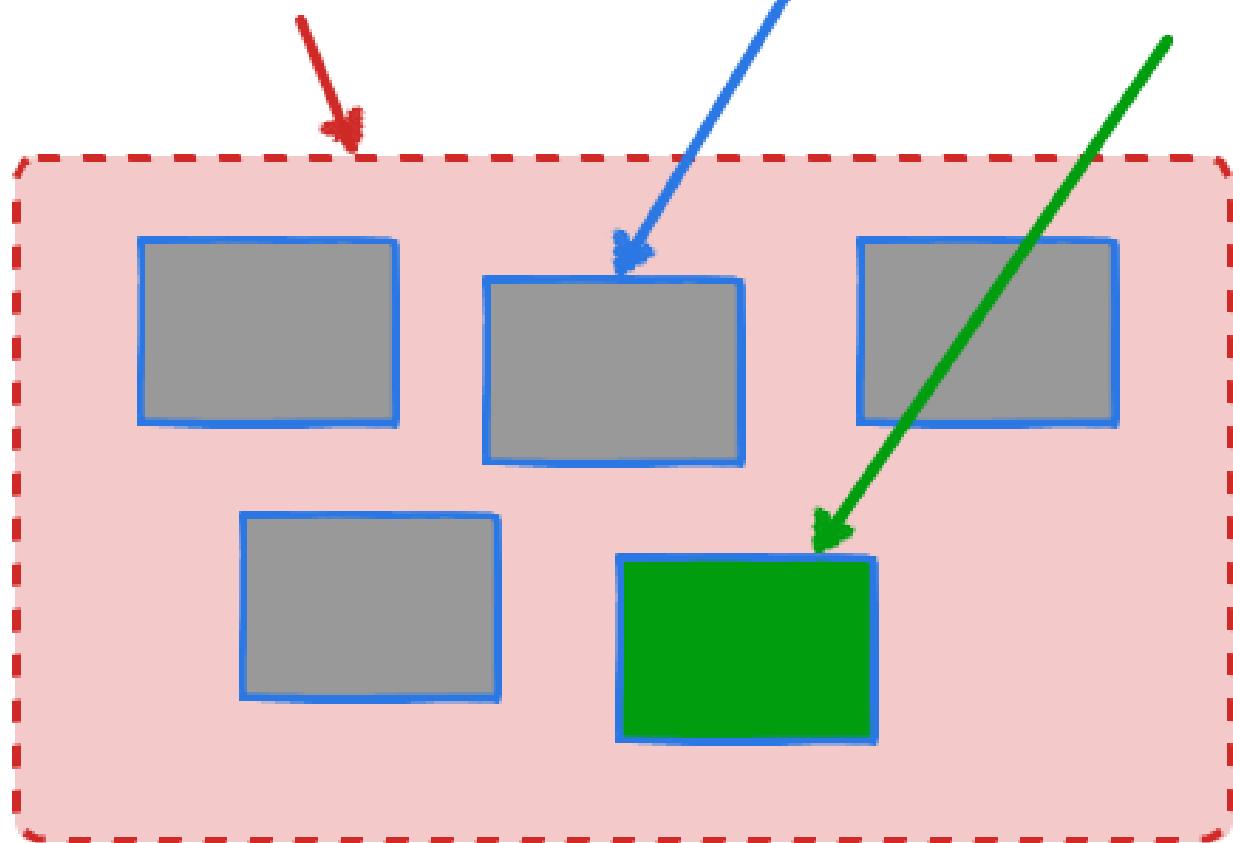
いや、そもそも変更出来ないし消せない

# 4. クラス名の命名規則

- BEM的命名規則に名前空間の考え方を足したもの

**Block**

**Element  
Modifier**



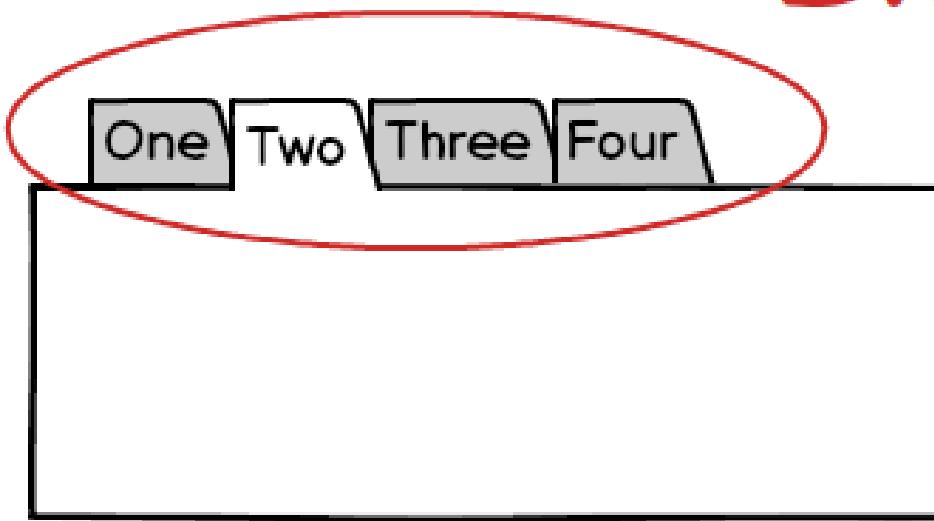
One

Two

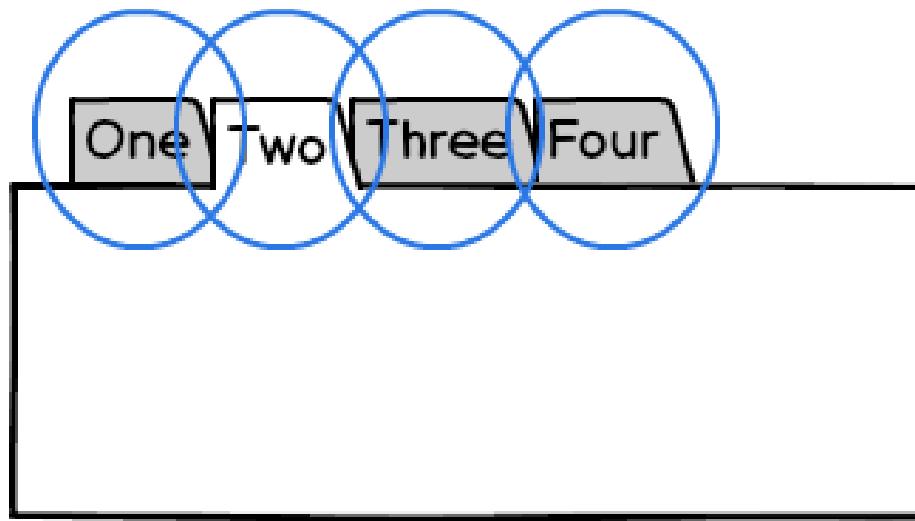
Three

Four

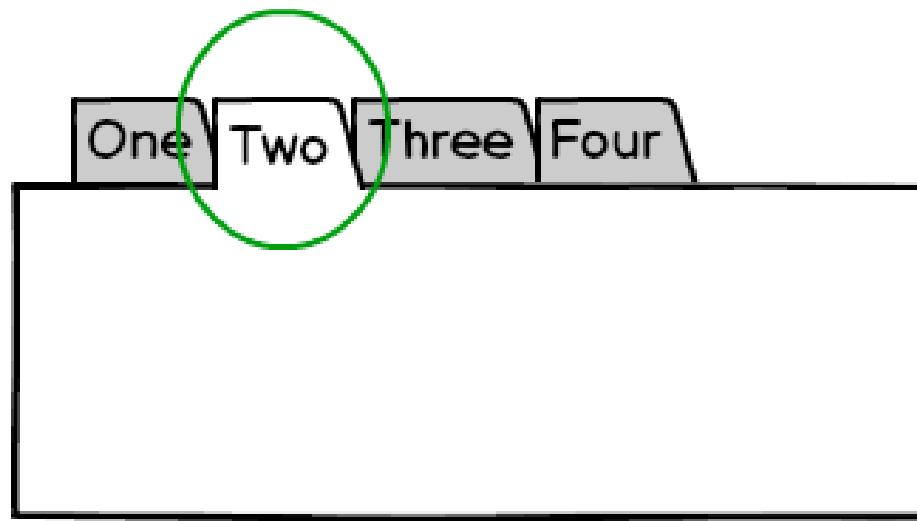
# Block



# Element



# Modifier



# BEMでは……

Block      .tab

---

Element    .tab\_\_item

---

Modifier   .tab\_\_item--active

```
<ul class="tab">
  <li class="tab__item"><button>One</button></li>
  <li class="tab__item tab__item--active"><button>Two</button></li>
  <li class="tab__item"><button>Three</button></li>
  <li class="tab__item"><button>Four</button></li>
</ul>
```

ECSSでは……

**.ns-Module\_ChildNode-variant**

---

Module        .tp-Tab

---

Child node    .tp-Tab\_Item

---

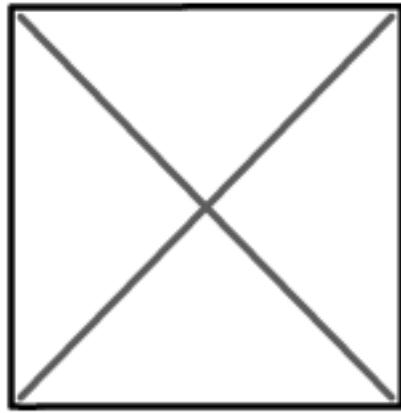
Variant        .tp-Tab\_Item-active

```
<ul class="tp-Tab">
  <li class="tp-Tab_Item"><button>One</button></li>
  <li class="tp-Tab_Item tp-Tab_Item-active"><button>Two</button></li>
  <li class="tp-Tab_Item"><button>Three</button></li>
  <li class="tp-Tab_Item"><button>Four</button></li>
</ul>
```

改めてECSSのサイトを見てみる

- トップページ
- チャプタページ
- 目次

# モジュール名決定の難しさ



## Heading

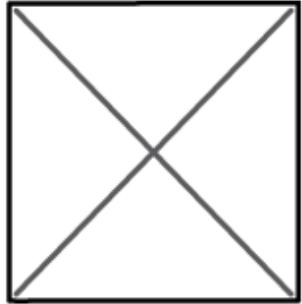
The quick brown fox jumps  
over the lazy dog. The quick  
brown fox jumps over the lazy  
dog. The quick brown fox  
jumps over the lazy dog.

media? imageBlock? productItem?

# 汎用的な名前 media にしたとする

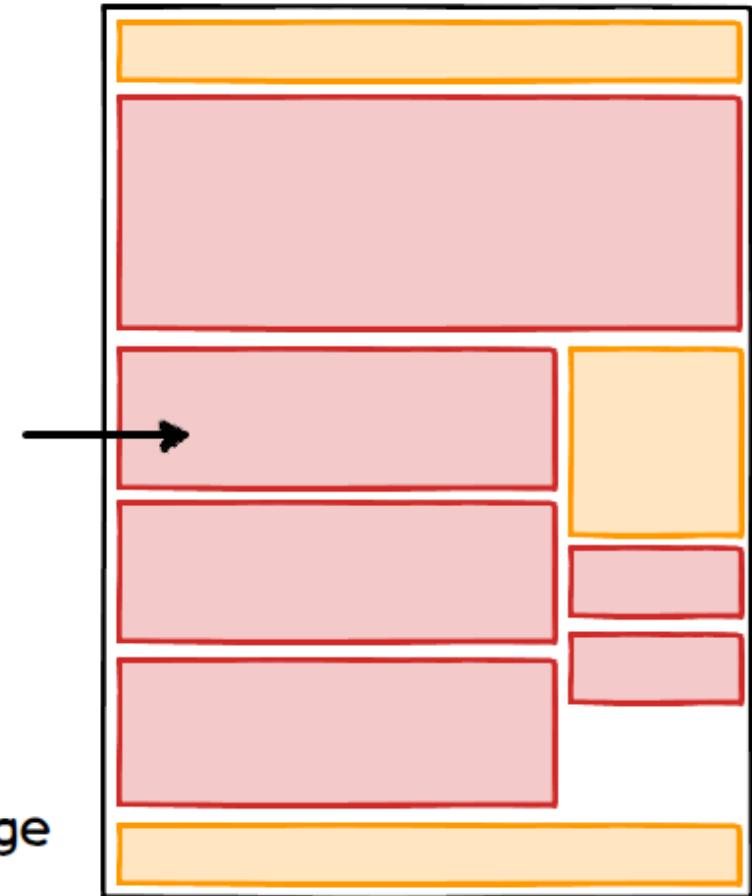
- モディファイアを多用して複雑化
- どこで使われているか分からない
- 結果CSSを変更できない
- 当然このモジュールのCSSは消せない
- 似たのが出たら**media2**？
- その使い分けを把握するのは困難
- 何回も使いまわせるのでCSS容量的には  
  ハッピーではある

# 名前空間の嬉しさ



**Heading**

The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog.



/topPage

tp-Media

# tp-Media

- トップページだけで使う想定で作っている
- どこで使われるのか明確
- どこかで使われるかもと考える必要がほぼ無い
- いらなくなったらトップページだけ見て消せる
- **tp-FeaturedProduct**とか具体的にしてよい
- 名前空間というスコープ内で考えることができる
- 同じようなモジュールが出ても別モジュールとして考えることになるのでCSS容量的には増大

# まとめ

- ECSSは「分けて」考える
- だからいくらでもスケールできる
- 管理と名前付けにシンプルさをもたらす
- 無理のない管理を実現する
- ミニマムなCSSを目指すものではない
- 最高のパフォーマンスは求めていない

# 実際どう？

- 名前付けに頭を悩ませることが減る
- 怖くて触れないを避けられそう
- 管理が別れるサイトで特に有効そう
- ビルドを頑張らねばならなそう
- 一人で管理している場合はそんなにかも

ありがとうございます

たい、Flexible Box チーム開発に効く環境構築術 Snap.svg で快適 SVG アニメーション Web マップの実装手法 JavaScript 再入門 ちょ  
実践！AngularJS フロントエンドのサウンド実装 ピクセルグリッド的ブックガイド 制作業務のコミュニケーション テンプレートエンジンのススメ 最適な  
ナショナルエンジニアのためのデザイン入門 ソーシャルコーディングのススメ JavaScript 開発のためのテスト入門 エンジニアに聞く、仕事の行方 フロントエ  
キュリティ ピクセルグリッドスタッフの情報収集 クライアントとのコミュニケーション Compass で簡単、CSS スプライト作成 快適な  
e, Marionette マークアップ・エンジニアのための SVG 入門 大規模プロジェクト運用のコツ Web の文字の読みやすさ 値ある CSS の設計と運用のため  
の仕事術 技術編 今、どんな働き方していますか？ CodeGrid を振り返る 貴い Media Queries 管理 ルーキーに捧ぐ、codeGrid おすすめ記事 プロキ  
ーズに ピクセルグリッド技術サーバー Code Grid の作り方 ピクセルグリッドが訪ねる、開発の現場 静的 HTML のためのテンプレートエンジン Web ブラ  
est, Asia レポート 開発会宿レポート ブラウザにおける書き最新事情 注目したいのはこの技術 仕事を書の読み方 CSS ブロハイ編 フロントエンド最新技  
ント 自由自在 これからグリッドレイアウト 自作ツール作成から学ぶノウハウ AWS で始めるサーバーレス・アーキテクチャ three.js 使いこなし AWS  
サイトの公開 これから始める React.js 実践、SVG スプライト Falcor で実現する効率的な fetch 実践、ユニットテスト はじめての d3.js レスポンシ  
ビティのコントロール Enduring CSS の設計思想 ピクセルグリッドのフォント選定と実装のプロセス これから始める React.js 発展編 今までの印刷  
Falcor で実現する効率的な fetch サーバー編 根っこから学ぶ CSS3 アニメーション どうする？ ブラウザ機能格差の解消 実践！スマホサイトの viewpo  
t のこと 覚えた！ハマった！エンジニアの近況 表示速度改善 Data URI スキーム coffeeScript をめぐる議論 コードで学ぶスマホサイト制作 今、  
ピクセルグリッドの仕事術 初心者のための jQuery jQuery conf レポート WebGL の基礎 ピクセルグリッド的ブックガイド CoffeeScript で学  
coffeeScript の世界 おしゃれな shadow DOM おすすめディベロッパ用語会議 Web アプリ開発のための JavaScript ライブラリ

- ECSS
- Enduring CSSの設計思想
- 知っておきたいHTMLテンプレート設計法

# オマケ

- サイト全体で共通のモジュールは？
- 同じようなモジュールがあるよ
- mixinやextendしたいのですが？
- WAI-ARIA
- 複雑なCSSの記述は避けるべきである
- ビルドについて