

# RESTORE

# THE

フロントエンドに**秩序**を取り戻す方法

@amagitakayosi

# ORDER

あくなき探求  
エンジニアを募

190

**早口でいきます**

今日話さないこと

**最新ライブラリの使い方**

**最高に美しい設計**

**Node.jsの話**

今日話すこと

**リファクタリング**

**コード規模とアーキテクチャ**

**フレームワーク導入**

**テスト環境構築**

スピリット



@amagitakayosi



Hatena



Hatena Blog

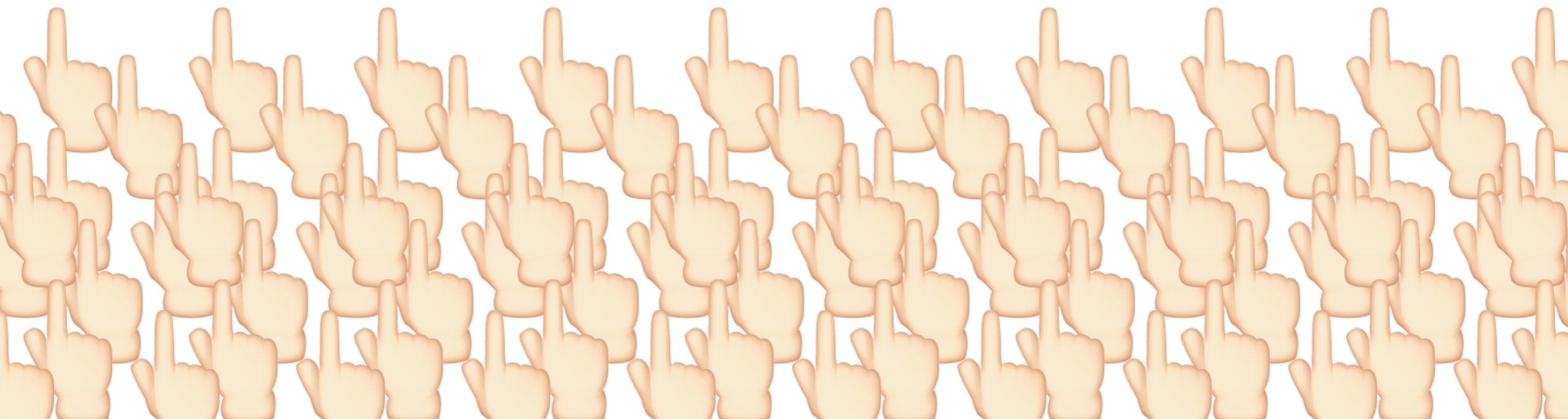


Hatena Blog

# 今すぐ登録！

---

<http://blog.hatena.ne.jp/register>





Hatena Blog

2015-11-07

# はてなブログ

はてなブログは4周年! ありがとう!! キャンペーン  
限定オリジナルトートバッグをゲットしよう!

# 4周年



デザインは仮です。予告なく変更する場合があります。

B! Bookmark

0

ツイート

0

いいね!

0

はてなブログは4周年! ありがとう!! キャンペーン

2015-07-27

# 記事編集画面（PC版）をリニューアルしました。広いテキストエリアと、スッキリした編集サイドバーで執筆に集中できます

[新機能](#) [機能変更](#)

はてなブログでは、PC版の記事編集画面をリニューアルしました。広々としたテキストエリアを中央に配置し、スッキリとしたデザインで、記事を書くことに集中できます。





This repository

Search

Exp



hatena / **Hatena-Epic**

# 記事編集画面コーディング #879

 **Merged**

**amagitakayosi** merged 540 commits into `staging/master`

 Conversation **92**

 Commits **250+**

 Files changed



**ueday** commented on 27 May

テンプレート眺めたところだと何やってるかわからな

**+11,329** **-12,888** 

---

**Labels**



**このトークでは**

**編集画面リニューアルに伴う**

**JS大改造の内容を**

**紹介します！！！！**

はてなブログの

設計思想



## schema

cho45 authored on 29 Jul 2011



## first commit

cho45 authored on 29 Jul 2011

**2011-07-29 開発開始**

# なるべく薄く

- Perl
- Plackベース / ORMなし
- JS
  - Vanilla JS

Be the first to clip this slide

ぼくのかんがえたさい

きょうのうえぶあふり

けーしょんふれーむ

わーく

◀ 21 of 91 ▶



Recommended

More from this author



Spark Streaming + Amazon Kinesis  
Yuta Imai  
1,968 views



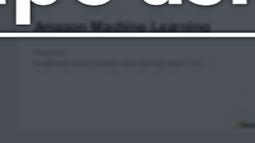
映画にでてくるハッカーに  
なりたい - YAPC Asia 2010  
Hiroh Satoh  
1,563 views



オンラインゲームの仕組み  
と工夫  
Yuta Imai  
1,055 views



Intro. to JavaScript  
Hiroh Satoh  
1,285 views



Amazon Machine Learning  
Yuta Imai  
10,631 views



Nseg49 mysql  
Masahiro Tomita  
1,394 views

# 参考: 元祖設計者のスライド

Share Like Download

<http://www.slideshare.net/cho45/yapc-asia-2011>

in | 0 f | 3 t | 19 g+ | 0

Published on Oct 14, 2011

Published in: Technology

Speaker Deck Published on Aug 21, 2015

# Perlの上にも三年 ずっとイケてるサービスを 作り続ける技術 hitode909

share

 **hitode909**  
8 Presentations

★ Star this Talk 9 Stars

Published in Technology

Stats 19,506 Views

### Share

Twitter, Facebook

Embed

Direct Link

Download PDF

Perlの上にも三年～ずっとイケてるサービスを作り続け

# 参考: サーバサイド設計の話

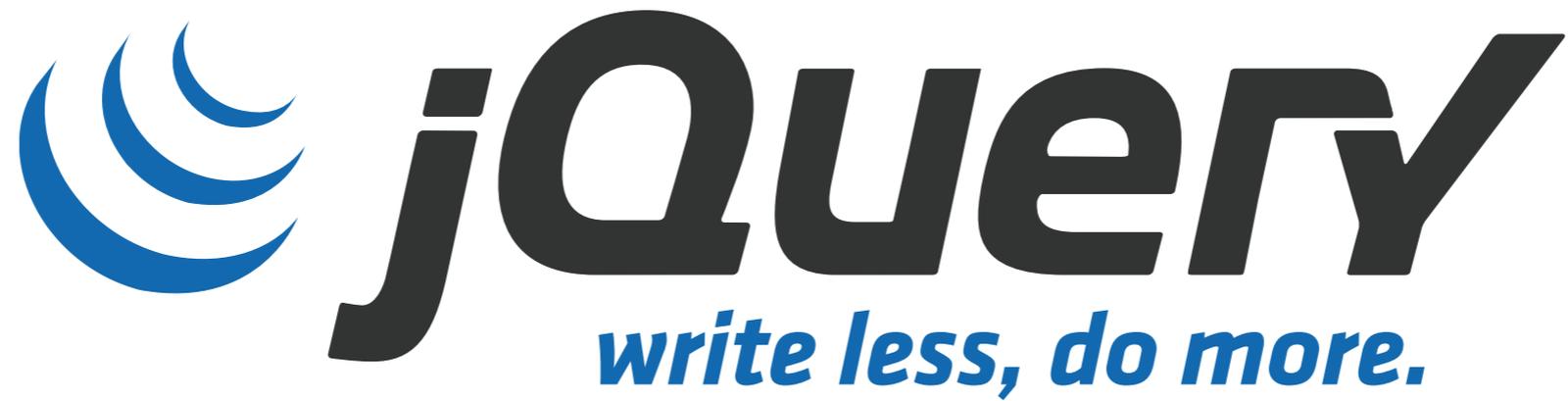
発表者は、この三年間はてなブログというイケてるPerlのサービスの開発に携ってきました...

<https://speakerdeck.com/hitode909/perlfalsehang-nimosan-nian-zututoiketerusabisuwozuo-risok-keruji-shu>

救いを求め、藁にもすがる思いで、朝も夜も読み続けたオブジェクト指向入門.....

偶然発見して、これだと思ったドメイン駆動設計.....

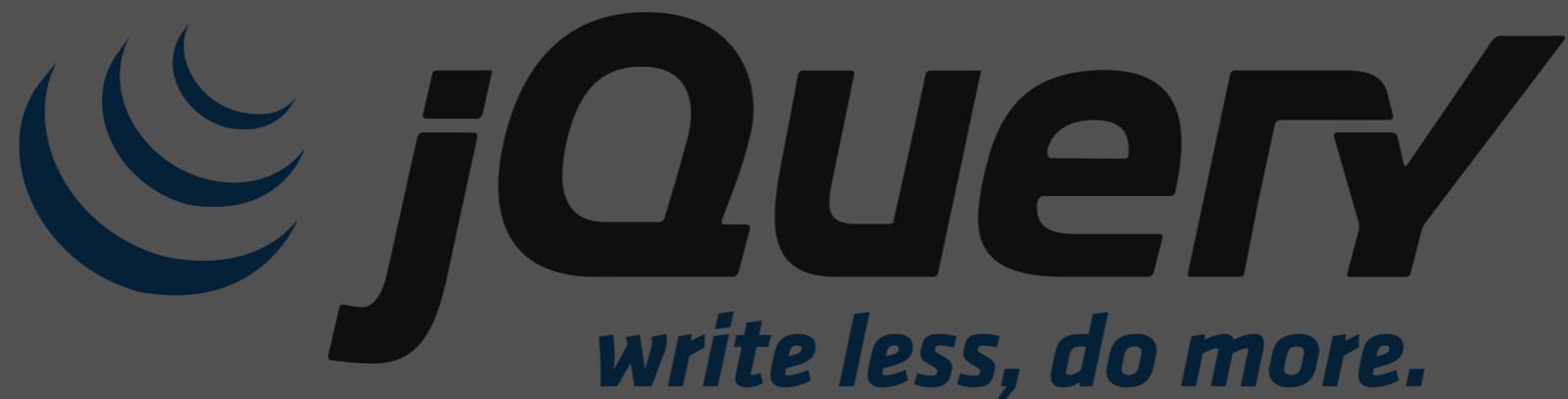
苦しみの軌跡と、現在最高の設計を紹介します.....!!!!!!



+

UNDERSCORE.JS





そして4年が経過した.....

UNDERSCORE.JS

---



# 問題発生



# 問題1

巨大な

フアイル

## 問題2

統一感のない

スタイル

# 問題3

テスト不可

地道に改善

していくぞ

!!!!!!

**Mission 1:**

**SPLIT**

**THE**

**CODE**

# はてなブログのJS

- Total **22633** lines (2015-04-03時点)
- そんなに巨大ではないが……

# admin-user-blog-edit.js

# 5524行

(2015-04-06時点)

```
(function($){
  Hatena.Diary.Pages.Admin["user-blog-edit"] = function () {
    if (window.parent != window) {
      Hatena.Diary.Pages.createForParent();
      Hatena.Diary.Pages.send("ready");
    }

    var title      = $("#title");
    var count      = $("#character-count");
    var container  = $("#editor");
    var form       = $("#edit-form");
    var main       = $("#editor-main");
    var supportContainer = $("#editor-support-container");
    var supportItems = $("#items");
    var curations  = $("#editor .curation-tab-content");

    var formState = new Hatena.Diary.FormState();
    formState.observeForm(form);

    var editor = new Hatena.Diary.Editor(form);
    editor.bind("initialize", function () {
      formState.setFormInitialState();
    });
    editor.trigger("initialize");

    editor.bind("change", function () {
      formState.checkFormState();
      var c = editor.getCharacterCount();
      count.text(Hatena.Locale.textN("edit_form.character_count_unit", c, 0));

      if (window.parent != window) Hatena.Diary.Pages.send("change", formState.isConfirmEnabled());
    });
    setTimeout(function () { editor.focus(); }, 100);

    title.on("keydown", function (e) {
      if (keyString === "Enter") {
        editor.trigger("submit");
        return false;
      }
    });

    editor.form.submit(function () {
      var submitButton = editor.form.find("submit");
      var disable = function() {
        editor.form.addClass("unedited");
        submitButton.attr("disabled", "disabled");
      };
      var enable = function() {
        editor.form.removeClass("unedited");
        submitButton.removeAttr("disabled");
      };

      var maxLength = 455368; // 2^16 + 16 までとする
      if (editor.getBytesCount() > maxLength) {
        disable();
        editor.bind("change", enable);
        alert(Hatena.Locale.text("edit_form.body_length_exceeded_error"));
        return false;
      } else { // 文字数制限にひっかからず、投稿に成功するまで
        Hatena.Diary.Util.preventDuplicateSubmit($editor.form);

        if (window.parent != window) Hatena.Diary.Pages.send("submit");
      }
    });

    Hatena.Diary.Editor.Support.currentEditor = editor;
    Hatena.Diary.currentCategoryEditor =
      new Hatena.Diary.Pages.Admin["user-blog-edit"].CategoryEditor(form, "#entry-categories");

    new Hatena.Diary.Pages.Admin["user-blog-edit"].OgImageEditor(form, "#og-image");

    var datetimeEditor = new Hatena.Diary.Pages.Admin["user-blog-edit"].DatetimeEditor(form, "#datetime-input");
    Hatena.Diary.Pages.Admin["user-blog-edit"].observeForScheduledEntry(datetimeEditor);

    // setupGooglePicker()では、EditorSidebar.setup()で呼ばれるtriggerイベントをDOMに設定しているので、
    // EditorSidebar.setup()よりも前に呼ばれる必要がある。
    Hatena.Diary.Pages.Admin["user-blog-edit"].setupGooglePicker();

    Hatena.Diary.Pages.Admin["user-blog-edit"].EditorSidebar.setup();

    Hatena.Diary.Pages.Admin["user-blog-edit"].setupResizeEditor({
      main: main,
      editor: editor,
      curations: curations,
      supportItems: supportItems
    });

    new Hatena.Diary.Pages.Admin["user-blog-edit"].AmazonSearch({
      container: $("#editor-amazon-search")
    });

    // 初心者タブがあるなら、自動的に初心者タブを表示する
    var $beginnerTab = $(".curation-bar-itemlist .beginner");
    if ($beginnerTab.length > 0) {
      if (!$beginnerTab.hasClass("active")) {
        $beginnerTab.trigger("click");
      }
    }

    $(".twitter-open-button").click(function () {
      $(".curation-bar-itemlist .twitter").trigger("click");
      return false;
    });

    if ($("#editor-twitter").hasClass("enabled")) {
      new Hatena.Diary.Pages.Admin["user-blog-edit"].Twitter({
        container: $("#editor-twitter")
      });
    }

    new Hatena.Diary.Pages.Admin["user-blog-edit"].HatenaBookmark({
      container: $("#editor-hatena-bookmark")
    });

    Hatena.Diary.Pages.Admin["user-blog-edit"].Fotolife.init();
    Hatena.Diary.Pages.Admin["user-blog-edit"].Paint.init();

    if ($("#editor-itunes.enabled").get(0)) {
      new Hatena.Diary.Pages.Admin["user-blog-edit"].Itunes({
        container: $("#editor-itunes")
      });
    }

    if ($("#editor-archive.enabled").get(0)) {
      new Hatena.Diary.Pages.Admin["user-blog-edit"].Archive({
        container: $("#editor-archive")
      });
    }

    if ($("#editor-instagram.enabled").get(0)) {
      new Hatena.Diary.Pages.Admin["user-blog-edit"].Instagram({
        container: $("#editor-instagram")
      });
    }

    if ($("#editor-500px.enabled").get(0)) {
      new Hatena.Diary.Pages.Admin["user-blog-edit"].500px({
        container: $("#editor-500px")
      });
    }

    if ($("#editor-vimeo.enabled").get(0)) {
      new Hatena.Diary.Pages.Admin["user-blog-edit"].Vimeo({
        container: $("#editor-vimeo")
      });
    }

    if ($("#editor-youtube.enabled").get(0)) {
      new Hatena.Diary.Pages.Admin["user-blog-edit"].Youtube({
        container: $("#editor-youtube")
      });
    }

    if ($("#editor-mill.enabled").get(0)) {
      new Hatena.Diary.Pages.Admin["user-blog-edit"].Mill({
        container: $("#editor-mill")
      });
    }

    new Hatena.Diary.Pages.Admin["user-blog-edit"].Gourmet({
      container: $("#editor-gourmet")
    });

    new Hatena.Diary.Pages.Admin["user-blog-edit"].Nicovideo({
      container: $("#editor-nicovideo")
    });

    new Hatena.Diary.Pages.Admin["user-blog-edit"].Gist({
      container: $("#editor-gist")
    });

    new Hatena.Diary.Pages.Admin["user-blog-edit"].Flickr({
      container: $("#editor-flickr")
    });

    if ($("#promotion-entry").get(0)) {
      new Hatena.Diary.Pages.Admin["user-blog-edit"].Promotion({
        element: $("#promotion-entry-container"),
        buttons: $("#promotion-button"),
        cancelButton: $("#promotion-cancel-button"),
        pager: $("#promotion-page-markers")
      });
    }

    Hatena.Diary.Pages.Admin["user-blog-edit"].EditorSidebar.setup();

    Hatena.Diary.Pages.Admin["user-blog-edit"].setupOgDescriptionInput(editor);

    var embedBox = new Hatena.Diary.Pages.Admin["user-blog-edit"].EmbedBox({
      container: $(".js-editor-embed-box")
    });

    // URLがpasteされるとイベント発生する
    // URLがpasteされるとイベント発生する
  };
}

```



# 編集画面のJS

- ・ formを管理するクラス
- ・ 記法ごとの機能  
(WYSIWYG, Markdown, はてな記法)
- ・ サイドバー等のコンポーネント

# 従来のJS分割

- ・ 名前空間でモジュール化
- ・ ファイルを concat して配信

# 名前空間 in JS

- ・ モジュールをObjectにまとめる
  - ・ グローバル変数の濫用を避ける
- 古の**ベストプラクティス**

# 名前空間を用意して

```
// index.js
```

```
window.Hatena = {};
```

# 名前空間に突っ込んで

```
// Foo.js
```

```
Hatena.Foo = {  
  ·· init : function () { ... },  
  ·· show : function () { ... },  
};
```

```
// Bar.js
```

```
Hatena.Foo.init();  
Hatena.Bar = {  
  ·· init : function () {  
    ·· ·· Hatena.AAA.bar();  
  },
```

**index.js**

**+**

**foo.js**

**+**

**bar.js**

**編集画面のJS  
分割したい.....**



# 問題発生



# 依存関係むずかしい

- ・ concatの順番変えたら壊れる
- ・ 変数名を変えると壊れる

```
// Foo.js
Hatena.Foo = {
  · · init : function () { ... },
  · · show : function () { ... },
};
```

```
// Bar.js
Hatena.Foo.init();
Hatena.Bar = {
  · · init : function () {
  · · · · Hatena.AAA.bar();
  · · },
};
```

**Fooが未定義だと  
エラー**



**index.js**

**+**

**foo.js**

**+**

**bar.js**

**index.js**

**+**

**Bar.js**

**+**

**Foo.js**

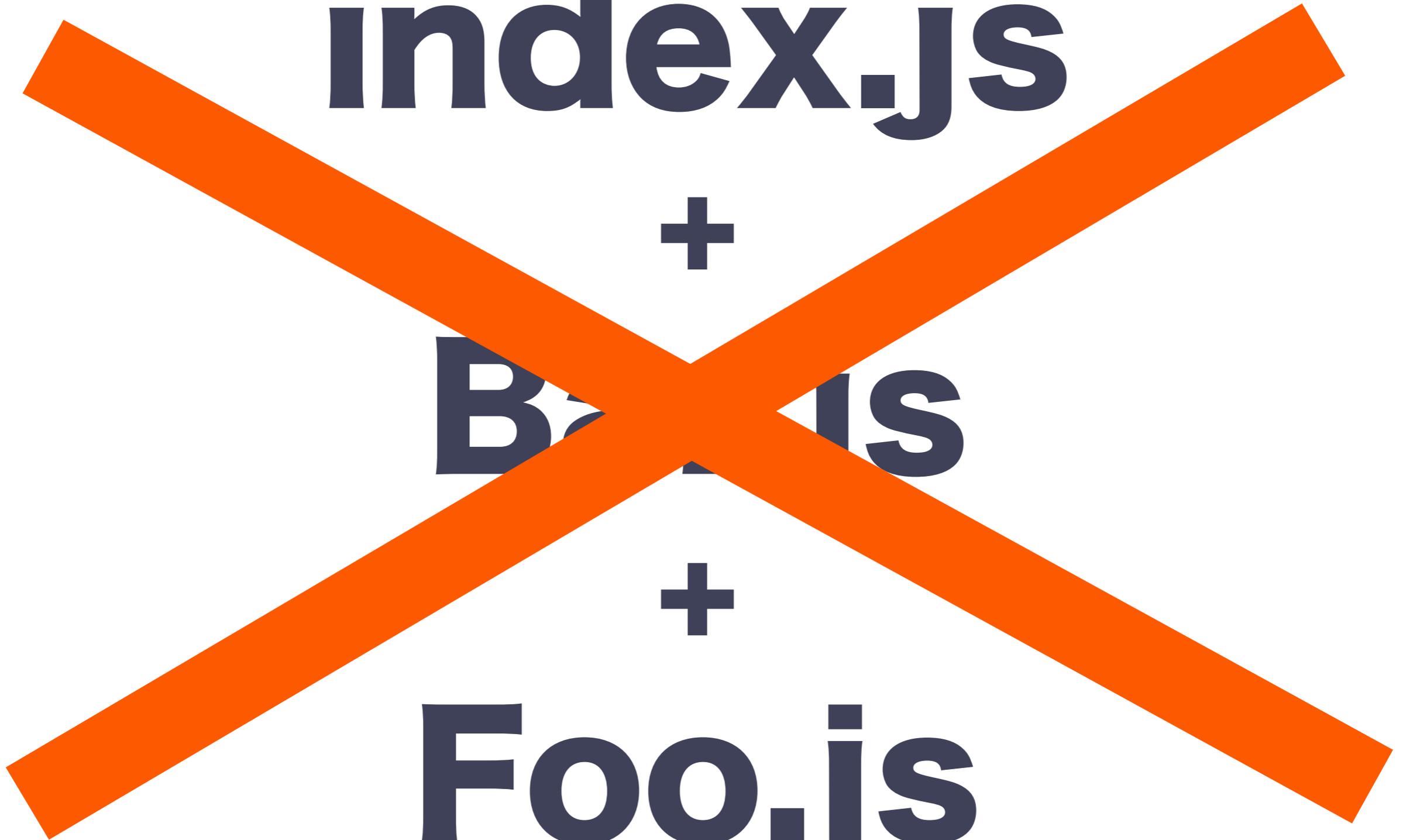
**index.js**

**+**

**Basics**

**+**

**Foo.js**



# 未定義の変数を検出できない

```
/* global Hatena */
Hatena.Util.loadNickName = function (user) {

  foo();

  var userInfo = Hatena.Util.loadUserInfo(user);

  return userInfo.nickName;

};
```

# 未定義の変数を検出できない

```
/* global Hatena */
```

```
Hatena.Util.loadNickName = function (user) {
```

```
· foo();
```

未定義のglobal変数は  
lintでチェックできるけど.....

```
· var userInfo = Hatena.Util.loadUserInfo(user);
```

```
· return userInfo.nickName;
```

```
};
```

# 未定義の変数を検出できない

```
/* global Hatena */  
Hatena.Util.loadNickName = function (user) {  
  
    foo();  
  
    var userInfo = Hatena.Util.loadUserInfo(user);  
  
    return userInfo.nickName;  
  
};
```

名前空間の中は  
lintでチェックできない！

実行するまで

バグがわからない！

**concat っらい.....**



browserify

# browserify

- CommonJS形式でモジュールを管理
- 競合と比べ、シンプルに使える
  - RequireJS : 書き方が面倒
  - Webpack : 大艦巨砲主義

# 依存管理が楽になる

- ・ concat 順を考えないで良い🙏
- ・ 存在しないファイルをロードするとエラー吐いてくれる

# ビルド時にエラー出してくれる！

```
→ nodefest git:(master) ✗ browserify -o bundle.js Foo.js  
Error: Cannot find module './Bar' from '/Users/amagitakayosi/nodefest'
```

# browserify導入

- ・ 2015-05 導入開始
- ・ 少しずつCommonJSに変換
- ・ ビルドしたJSを  
既存のJSとconcat

# CommonJS形式に 書き直して

```
// src/js/Foo.js  
var Foo = function () { ... };  
module.exports = Foo;
```

# 名前空間につっこんで

```
// src/js/index.js
window.Hatena = window.Hatena || {};
window.Hatena.Diary = {
  · Foo : require('./Foo'),
};
```

# ビルドしたJSを これまでのJSとconcat

```
"vendor/overthrow-dist/overthrow.js",
```

```
"bundle.js", ← ビルドしたJS
```

```
"base.js",
```

```
"admin.js",
```

```
"admin-feedback-iframe.js",
```

```
"admin-globalheader.js",
```

```
"admin-invite-new.js",
```

```
"admin-register.js",
```

```
"admin-register-iframe.js",
```

これまでのJS

落とし穴



# 共通化は正しく行う

```
// admin/foo.js
```

```
Hatena.Diary.Admin['Foo'] = function () {  
  Hatena.Diary.Admin['Bar']();  
  piyo();  
};
```

```
// admin/bar.js
```

```
Hatena.Diary.Admin['Bar'] = function () {  
  hoge();  
  fuga();  
};
```

# 共通化は正しく行う

```
// admin/foo.js
Hatena.Diary.Admin['Foo'] = function () {
  Hatena.Diary.Admin['Bar']().
  piyo();
};

// admin/bar.js
Hatena.Diary.Admin['Bar'] = function () {
  hoge();
};
```



# 共通化は正しく行う

- ・ 他の名前空間を参照してると  
分割しづらい
- ・ 先にリファクタリングすべし

**現在の状況**

# admin-user-blog-edit.js

5524行



2333行

```
(function($){
  Hatena.Diary.Pages.Admin["user-blog-edit"] = function () {
    if (window.parent != window) {
      Hatena.Diary.Pages.createForParent();
      Hatena.Diary.Pages.send("ready");
    }

    var title = $("#title");
    var count = $("#character-count");
    var container = $("#editor");
    var form = $("#edit-form");
    var main = $("#editor-main");
    var supportContainer = $("#editor-support-container");
    var supportItems = $("#items");
    var curations = $("#curations");

    var formState = new Hatena.Diary.Pages.Admin["user-blog-edit"].FormState({});
    formState.observe($("#form"));

    var editor = new Hatena.Diary.Editor.Support.currentEditor({});
    editor.bind("initialize", function () {
      formState.setFormInitialState();
    });
    editor.trigger("initialize");

    editor.bind("change", function () {
      formState.checkFormState();
      var c = editor.getCharacterCount();
      count.text(Hatena.Locale.textN("edit_form.character_count_unit", c, 0));

      if (window.parent != window) Hatena.Diary.Pages.send("change", formState.isConfirmed());
    });
    setTimeout(function () { editor.focus() }, 100);

    title.on("keydown", function(event) {
      if(keyString(event) == "RET") {
        editor.focus();
        return false;
      }
    });

    editor.form.submit(function() {
      var submitButton = editor.form.find("submit");
      var disable = function() {
        editor.form.addClass("unedited");
        submitButton.attr("disabled", "disabled");
      };
      var enable = function() {
        editor.form.removeClass("unedited");
        submitButton.removeAttr("disabled");
      };

      var maxLength = 455368; // 2^16 * 18 までとする
      if (editor.getBytesCount() > maxLength) {
        disable();
        editor.bind("change", enable);
        alert(Hatena.Locale.text("edit_form.body_length_exceeded_error"));
        return false;
      } else { // 文字数制限にひっかからず、投稿に成功するまで
        Hatena.Diary.Util.preventDuplicateSubmit($editor.form);

        if (window.parent != window) Hatena.Diary.Pages.send("change", false);
      }
    });

    Hatena.Diary.Editor.Support.currentEditor = editor;
    Hatena.Diary.currentCategoryEditor =
      new Hatena.Diary.Pages.Admin["user-blog-edit"].CategoryEditor(form, "#entry-categories");

    new Hatena.Diary.Pages.Admin["user-blog-edit"].OgImageEditor(form, "#og-image");

    var datetimeEditor = new Hatena.Diary.Pages.Admin["user-blog-edit"].DatetimeEditor(form, "#datetime-input");
    Hatena.Diary.Pages.Admin["user-blog-edit"].observeForScheduledEntry(datetimeEditor);

    // setupGooglePicker()では、EditorSidebar.setup()で呼び出されるtriggerイベントはDOMに設定しているので、
    // EditorSidebar.setup()よりも前に呼び出される必要がある。
    Hatena.Diary.Pages.Admin["user-blog-edit"].setupGooglePicker();

    Hatena.Diary.Pages.Admin["user-blog-edit"].EditorSidebar.setup({
      main: main,
      editor: editor,
      curations: curations,
      supportItems: supportItems
    });

    new Hatena.Diary.Pages.Admin["user-blog-edit"].AmazonSearch({
      container: $("#editor-amazon-search")
    });

    // 初心者タブがあるなら、自動的に初心者タブを表示する
    var $beginnerTab = $(".duration-bar-itemlist .beginner");
    if ($beginnerTab.length > 0) {
      if (!$beginnerTab.hasClass("active")) {
        $beginnerTab.trigger("click");
      }
    }

    $(".twitter-open-button").click(function () {
      $(".duration-bar-itemlist .twitter").trigger("click");
      return false;
    });

    if ($("#editor-twitter").hasClass("enabled")) {
      new Hatena.Diary.Pages.Admin["user-blog-edit"].Twitter({
        container: $("#editor-twitter")
      });
    }

    new Hatena.Diary.Pages.Admin["user-blog-edit"].HatenaBookmark({
      container: $("#editor-hatena-bookmark")
    });

    Hatena.Diary.Pages.Admin["user-blog-edit"].Fotolife.init();
    Hatena.Diary.Pages.Admin["user-blog-edit"].Paint.init();

    if ($("#editor-itunes.enabled").get(0)) {
      new Hatena.Diary.Pages.Admin["user-blog-edit"].Itunes({
        container: $("#editor-itunes")
      });
    }

    if ($("#editor-archive.enabled").get(0)) {
      new Hatena.Diary.Pages.Admin["user-blog-edit"].Archive({
        container: $("#editor-archive")
      });
    }

    if ($("#editor-instagram.enabled").get(0)) {
      new Hatena.Diary.Pages.Admin["user-blog-edit"].Instagram({
        container: $("#editor-instagram")
      });
    }

    if ($("#editor-evernote.enabled").get(0)) {
      new Hatena.Diary.Pages.Admin["user-blog-edit"].Evernote({
        container: $("#editor-evernote")
      });
    }

    new Hatena.Diary.Pages.Admin["user-blog-edit"].Pixiv({
      container: $("#editor-pixiv")
    });

    new Hatena.Diary.Pages.Admin["user-blog-edit"].Gourmet({
      container: $("#editor-gourmet")
    });

    new Hatena.Diary.Pages.Admin["user-blog-edit"].Nicovideo({
      container: $("#editor-nicovideo")
    });

    new Hatena.Diary.Pages.Admin["user-blog-edit"].Globe({
      container: $("#editor-globe")
    });

    new Hatena.Diary.Pages.Admin["user-blog-edit"].Flickr({
      container: $("#editor-flickr")
    });

    if ($("#promotion-entry").get(0)) {
      new Hatena.Diary.Pages.Admin["user-blog-edit"].Promotion({
        element: $("#promotion-entry-container"),
        buttons: $(".promotion-button"),
        cancelButton: $(".promotion-cancel-button"),
        pager: $(".promotion-page-markers")
      });
    }

    Hatena.Diary.Pages.Admin["user-blog-edit"].setupEditorSidebarHome();

    Hatena.Diary.Pages.Admin["user-blog-edit"].setupOgDescriptionInput(editor);

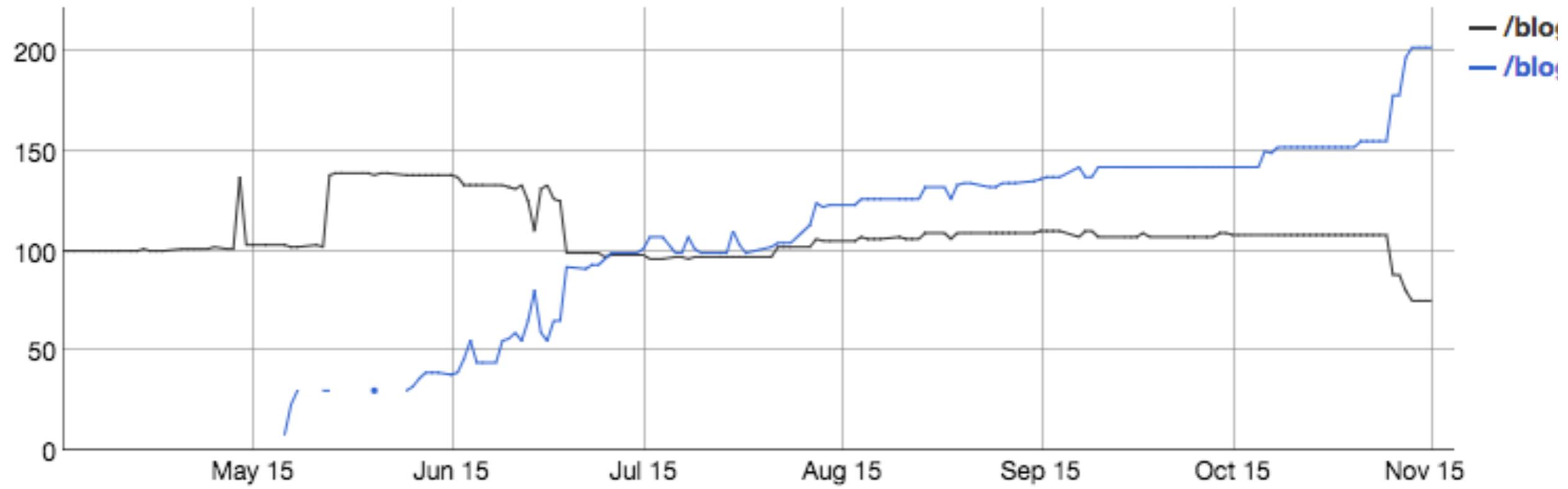
    var embedBox = new Hatena.Diary.Pages.Admin["user-blog-edit"].EmbedBox({
      container: $(".js-editor-embed-box")
    });

    // URLがpasteされるとイベント発生する
    // URLがpasteされたらURLのバリデーションを行う
  }
});
```



# js\_compare 単なるJS vs Browserify化したJS

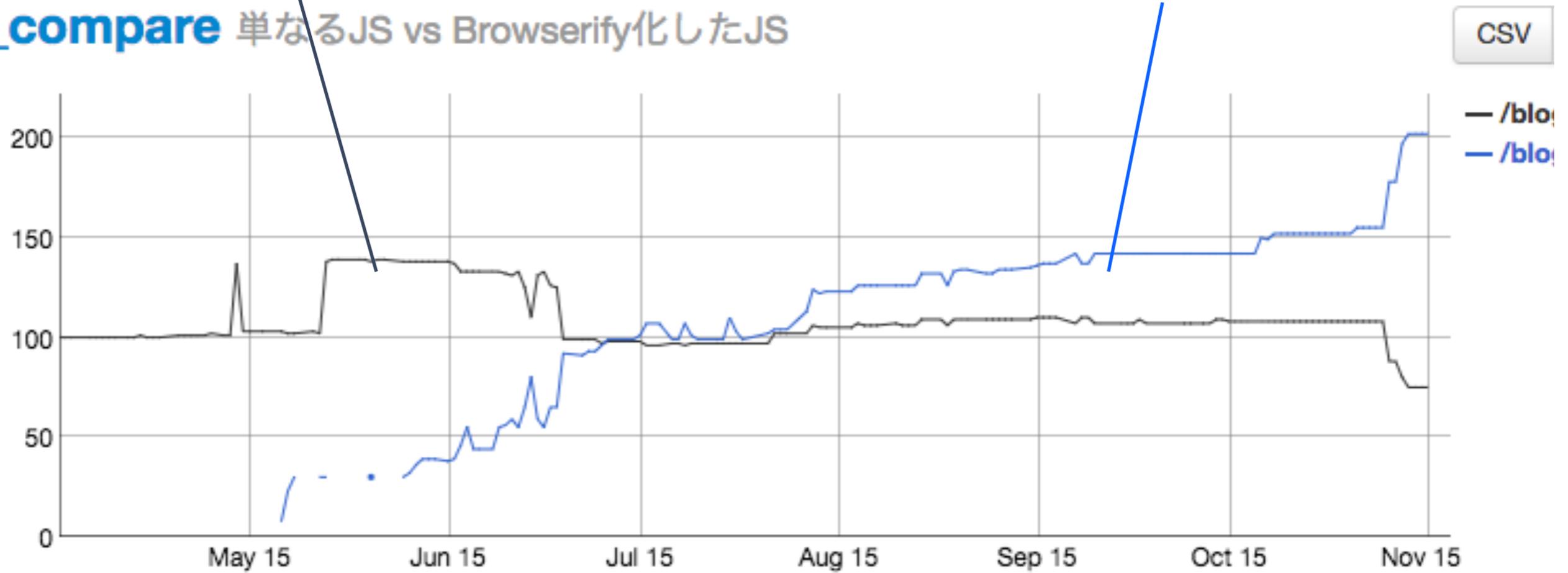
CSV



concatしてるファイル数

browserify管理下の  
ファイル数

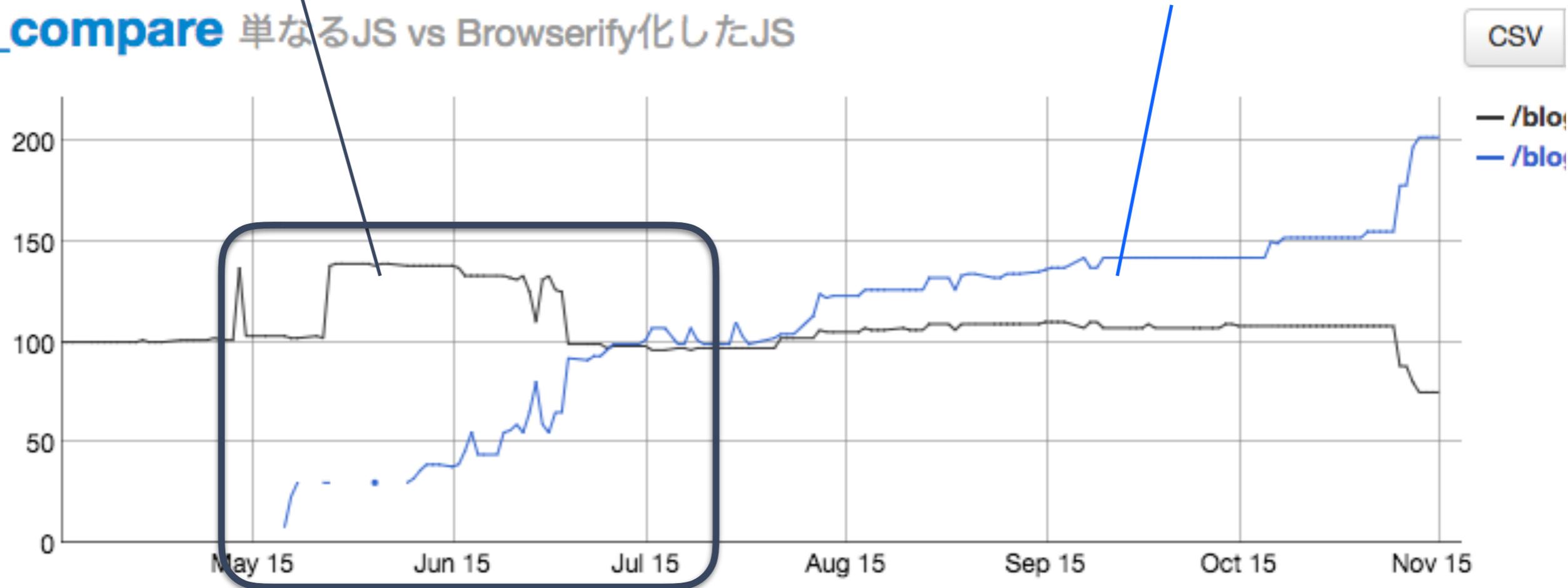
js\_compare 単なるJS vs Browserify化したJS



concatしてるファイル数

browserify管理下の  
ファイル数

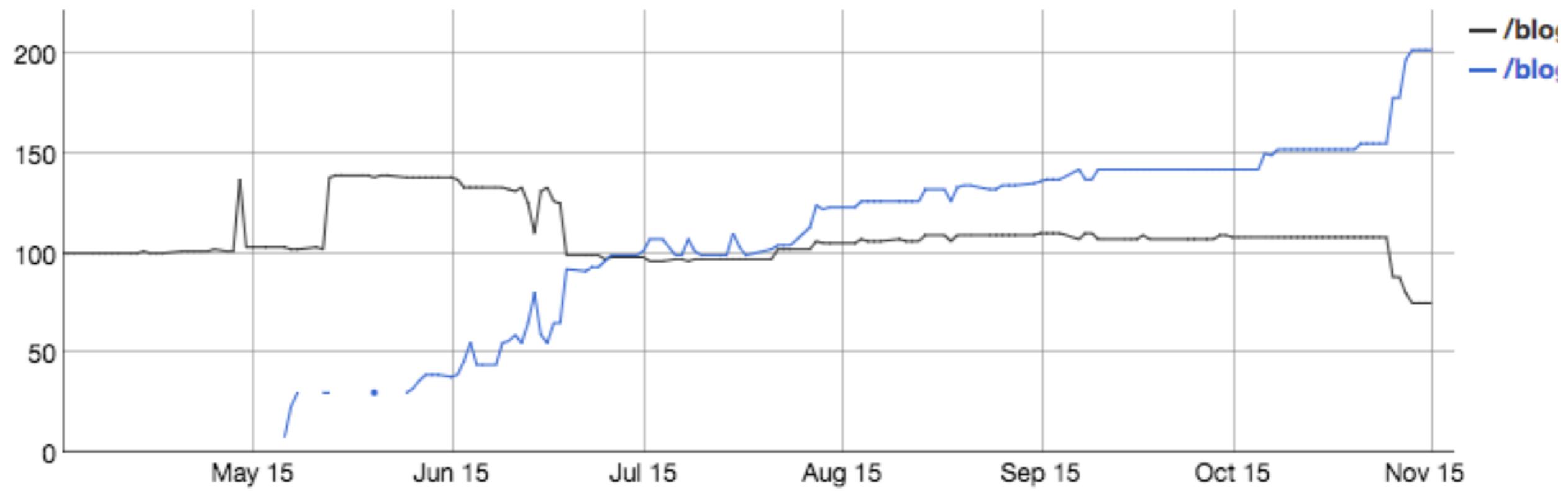
js\_compare 単なるJS vs Browserify化したJS



細かくモジュール化すると  
元ファイルより多くなる

# js\_compare 単なるJS vs Browserify化したJS

CSV



browserify管理下の

concatしてるファイル数

ファイル数

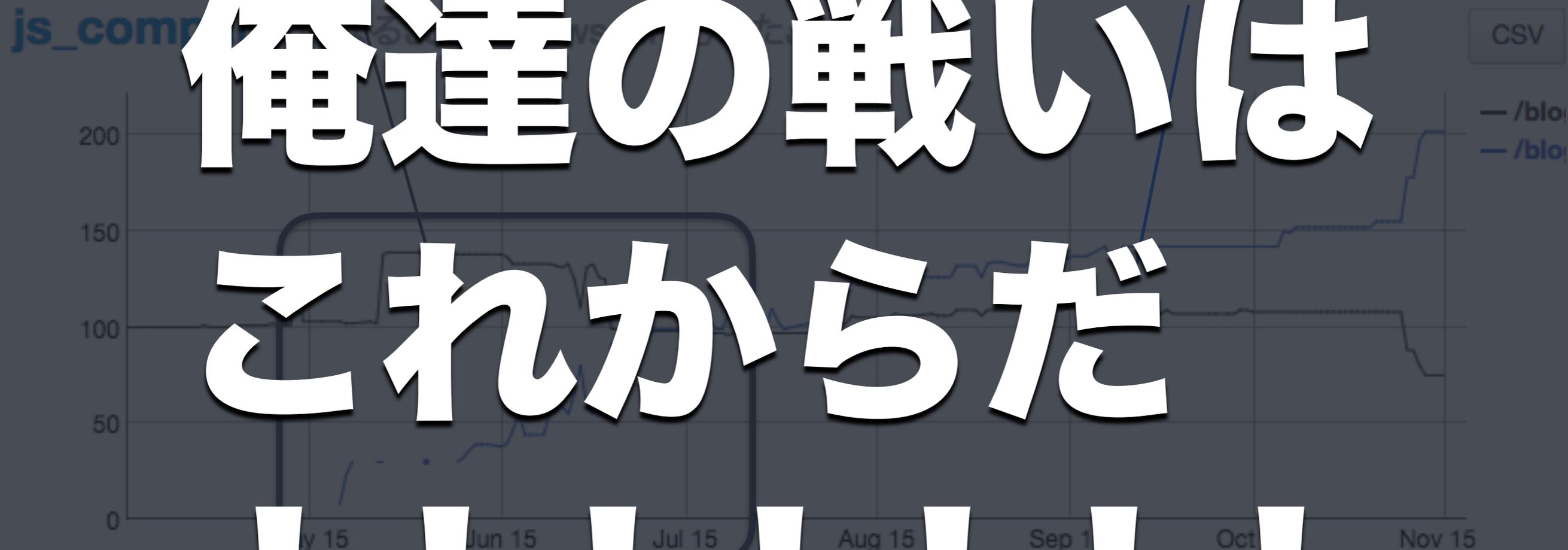
# 俺達の戦いは

# これからだ

# !!!!!!

細かくモジュール化すると

元ファイルより多くなる



**Mission 2:**

**KEEP**

**THE**

**STYLE**

**設計スタイル**

**コーディングスタイル**

# Vanilla Architecture

VanillaJSでのコンポーネント設計

# 既存JSの問題点

- ・ データとDOMが密結合
- ・ コンポーネント間の連携がムズい
- ・ サーバとの連携がダルい

# データとDOMが密結合

- ・ DOM属性にデータを保存
- ・ あちこちでDOM操作
  - ・ データ間の整合性が壊れやすい

# コンポーネント間の連携

- ・ データフローに決まりがない
- ・ 循環参照で呼び合ったり
- ・ 最悪DOMイベントを使う

# サーバとの連携

- ・ HTML構造からデータを作る
  - ・ JSONとかではない
- ・ DB更新の度にサーバ側で描画



## 編集サイドバー

完了



写真を投稿

タブを表示する



絵を描く

タブを表示する



カテゴリー

タブを表示する





# 編集サイドバー

完了



写真を投稿



タブを表示する



ここをクリックすると



タブを表示する

DBが更新されて



カテゴリー



タブを表示する



写真を投稿

サーバでHTMLを描画し  
\$.ajaxで受け取って  
\$.replaceWith()する



絵を描く



タブを表示する



カテゴリー



タブを表示する

スタートレスで

美しい設計

**JSで描画すれば**

**本来は不要**



# VanillaJSのまま再設計

- ・ データとDOMを分離
- ・ 機能毎にViewModelを作る
- ・ 機能同士はイベントでやりとり

# データとDOMを分離

- ・ 初期化時にデータを全て用意
- ・ DOM操作は専用メソッドで行う

# DOMを都度さわらずに

```
Foo.prototype = {  
  
  getComment : function () {  
    return this.$comment.data('comment');  
  },  
  
  setComment : function (comment) {  
    this.$comment.data('comment', comment);  
  },  
  
};
```

# DOMを都度さわらずに

```
Foo.prototype = {  
  getComment: function () {  
    return this.$comment.data('comment');  
  },  
  setComment: function (comment) {  
    this.$comment.data('comment', comment);  
  }  
};
```



# 初期化時にデータを用意

```
var Foo = function ($el) {  
  · this.$el      = $el;  
  · this.$comment = $el.find('.comment');  
  · this.comment = this.$comment.data('comment');  
};
```

```
Foo.prototype = {  
  · · · getComment : function () {  
  · · ·   · · · return this.comment;  
  · · · },  
  · · · setComment : function (comment) {  
  · · ·   · · · this.comment = comment;  
  · · · },  
};
```

# DOM操作は専用メソッドで

- ・ JSでレンダリング
- ・ データ変更の度に全部再描画
  - ・ 整合性、メンテしやすさを優先
  - ・ パフォーマンスは度外視

```
var COMMENTS = _.template('<li><%- comment %></li>');
```

```
...
```

```
render : function () {  
  this.$comments  
    .empty()  
    .append(COMMENTS({  
      comments : this.comments,  
    }));  
},
```

# 機能毎にViewModel

- ・ コンポーネントを親と子に分割
  - ・ Smart / Dumb Component

# 親コンポーネント

- ・ データ、ビジネスロジックを管理
- ・ EventEmitterを継承
  - ・ DOM Eventを極力つかわない！

# 子コンポーネント

- ・ 親への参照を持つ
- ・ データの描画とイベント発行のみ

# 例: Sidebar

The image shows a screenshot of the Hatena Blog editor interface. The main area is a text editor with a title field, a toolbar with various icons, and a large text area. The sidebar on the right is titled "編集サイドバー" (Editing Sidebar) and contains a list of options with toggle switches. The options are:

- よく使う機能は、ショートカットとしてタブに追加できます。
- 写真を投稿
- カテゴリ
- 編集オプション
- 過去記事貼り付け
- 引用貼り付け
- Amazon商品紹介
- YouTube貼り付け
- Twitter貼り付け
- Instagram貼り付け
- はてなブックマーク貼り付け
- Googleフォト貼り付け
- ニコニコ動画貼り付け
- pixiv貼り付け

At the bottom of the editor, there are two buttons: "公開する" (Publish) and "下書き保存する" (Save Draft).

### > 編集サイドバー



よく使う機能は、ショートカットとしてタブに追加できます。



写真を投稿

カテゴリー

編集オプション

過去記事貼り付け

引用貼り付け

Amazon商品紹介

YouTube貼り付け

Twitter貼り付け

Instagram貼り付け

はてなブックマーク貼り付け

Googleフォト貼り付け

ニコニコ動画貼り付け

# Sidebar

# SidebarButtons

# Config

## 編集サイドバー

よく使う機能は、ショートカットとしてタブに追加できます。

写真を投稿

カテゴリー

編集オプション

過去記事貼り付け

引用貼り付け

Amazon商品紹介

YouTube貼り付け

Twitter貼り付け

Instagram貼り付け

はてなブックマーク貼り付け

Googleフォト貼り付け

ニコニコ動画貼り付け

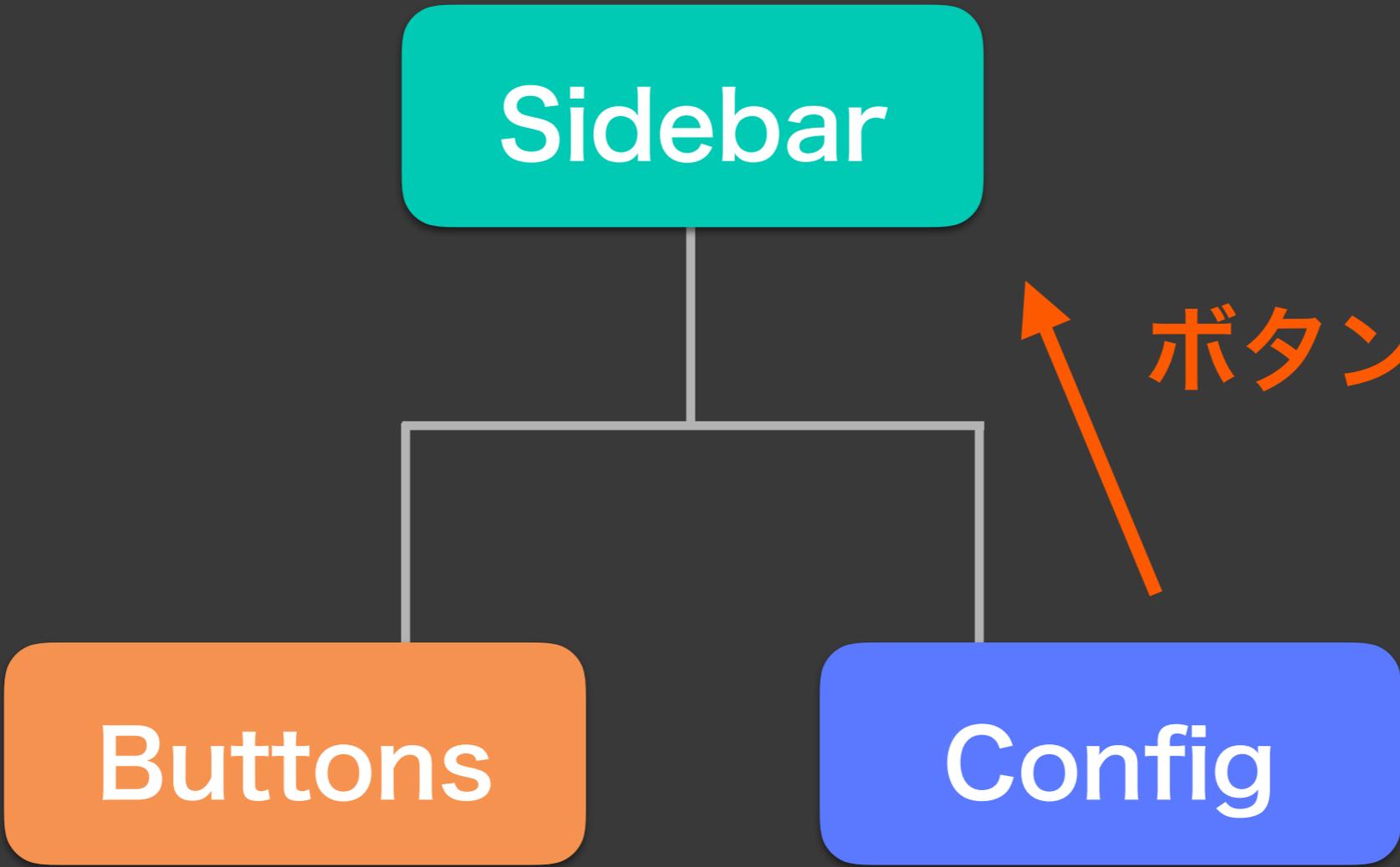
pixiv貼り付け

**Sidebar**

```
graph TD; Sidebar[Sidebar] --- Buttons[Buttons]; Sidebar --- Config[Config];
```

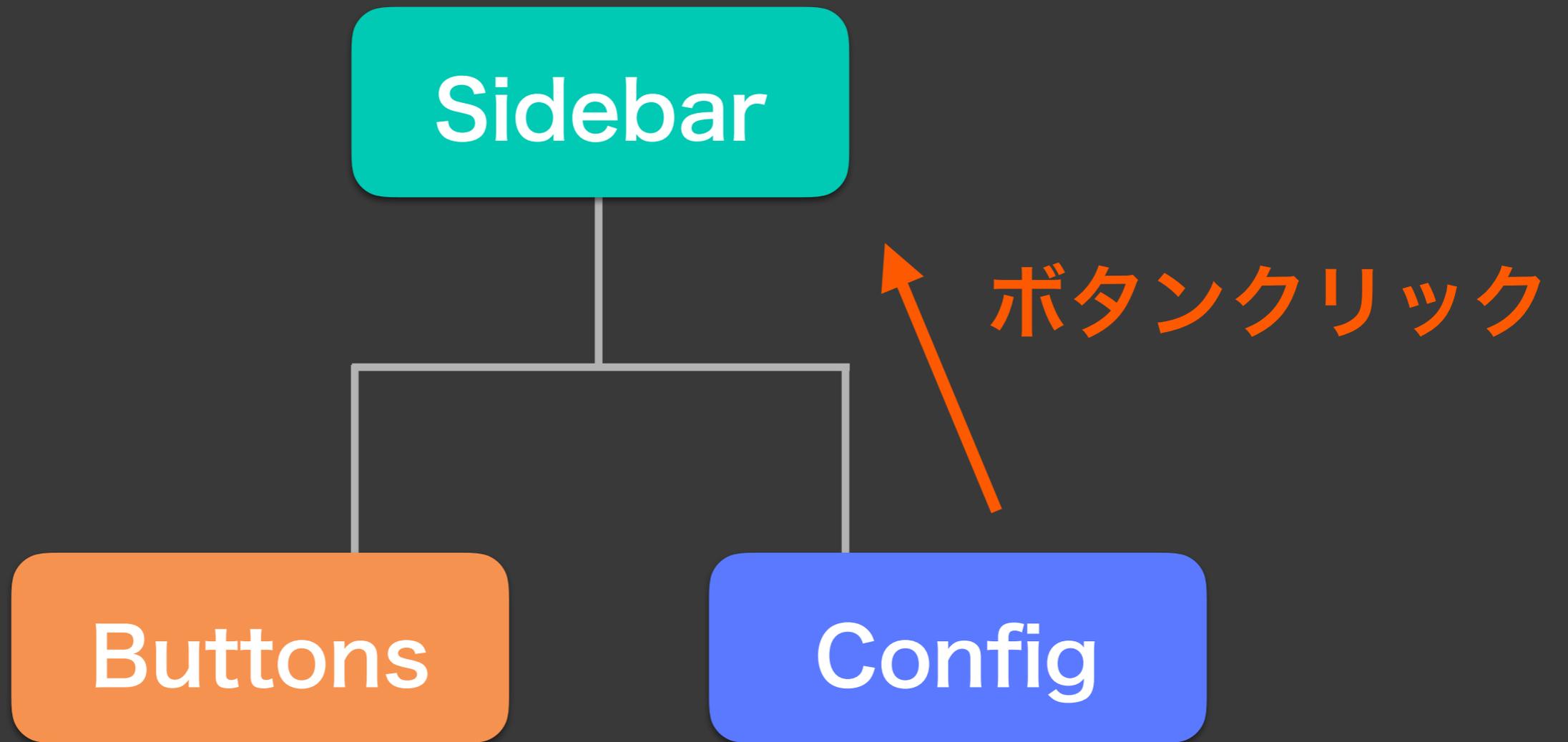
**Buttons**

**Config**

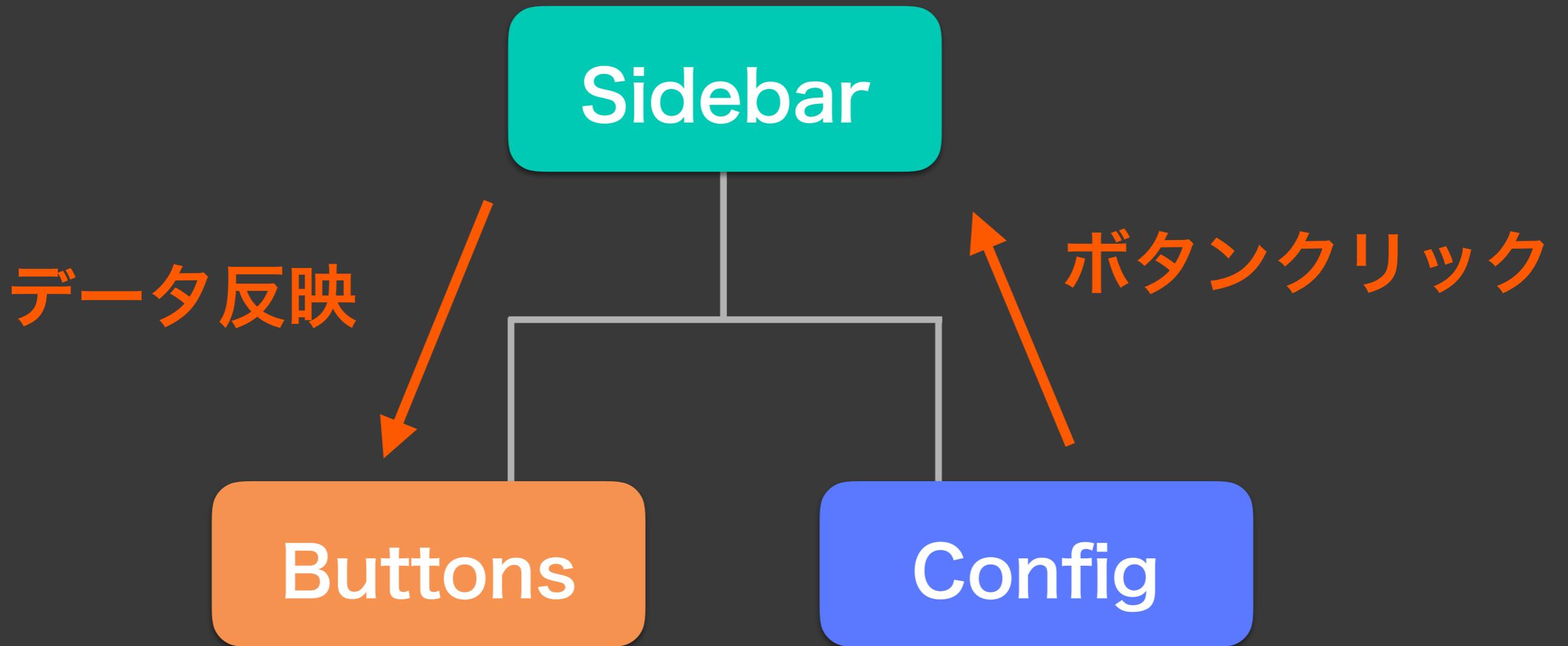


ボタンクリック

データ変更 + emit('change')



データ変更 + emit('change')



# コンポーネント間のやりとり

The image shows a screenshot of the Hatena Blog editor interface. The main content area features a large teal 'Fotolife' component at the top and a large orange 'MarkdownEditor' component below it. The editor includes a title field, a rich text toolbar, and a main text area. On the right side, there is a sidebar for photo uploads, titled '写真を投稿' (Upload Photos), which includes a '+ 写真を投稿' button and a gallery of selected images. The interface is in Japanese and includes navigation and utility buttons at the bottom.

# コンポーネント間のやりとり

- ・ 上位のViewModelをつかってイベントを発生させる
- ・ または `$window.trigger()`

# 上位のViewModelを作成

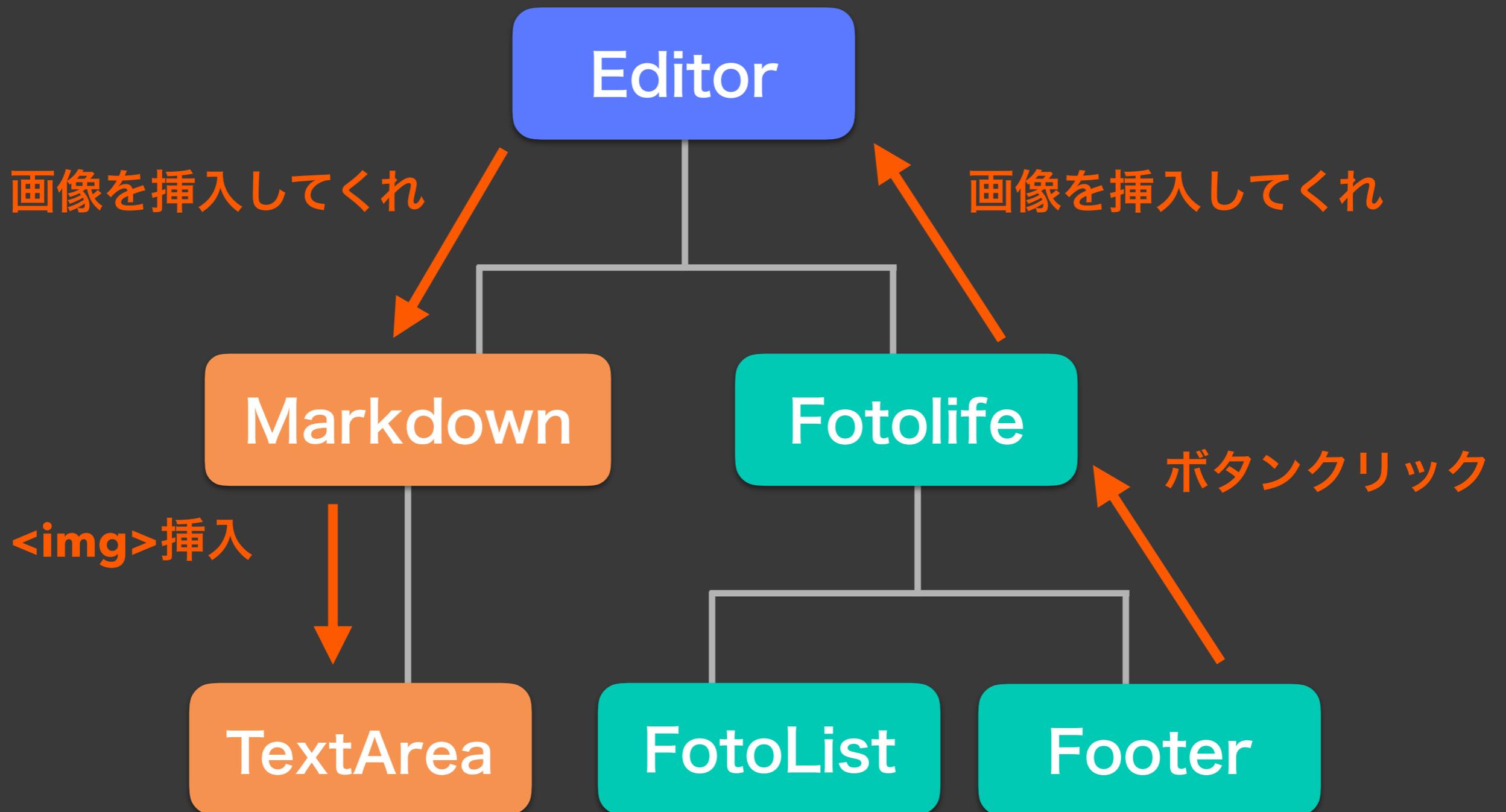
The image shows a screenshot of the Hatena Blog editor interface. The main editing area is highlighted with a blue border and contains the text "MarkdownEditor" in large orange letters. To the right, a sidebar for photo uploads is highlighted with a cyan border and contains the text "Fotolife" in large cyan letters. The word "Editor" is written in large blue letters across the top of the editing area. The interface includes a title field, a rich text toolbar, and a "公開する" (Publish) button at the bottom left.

MarkdownEditor

Fotolife

Editor

公開する



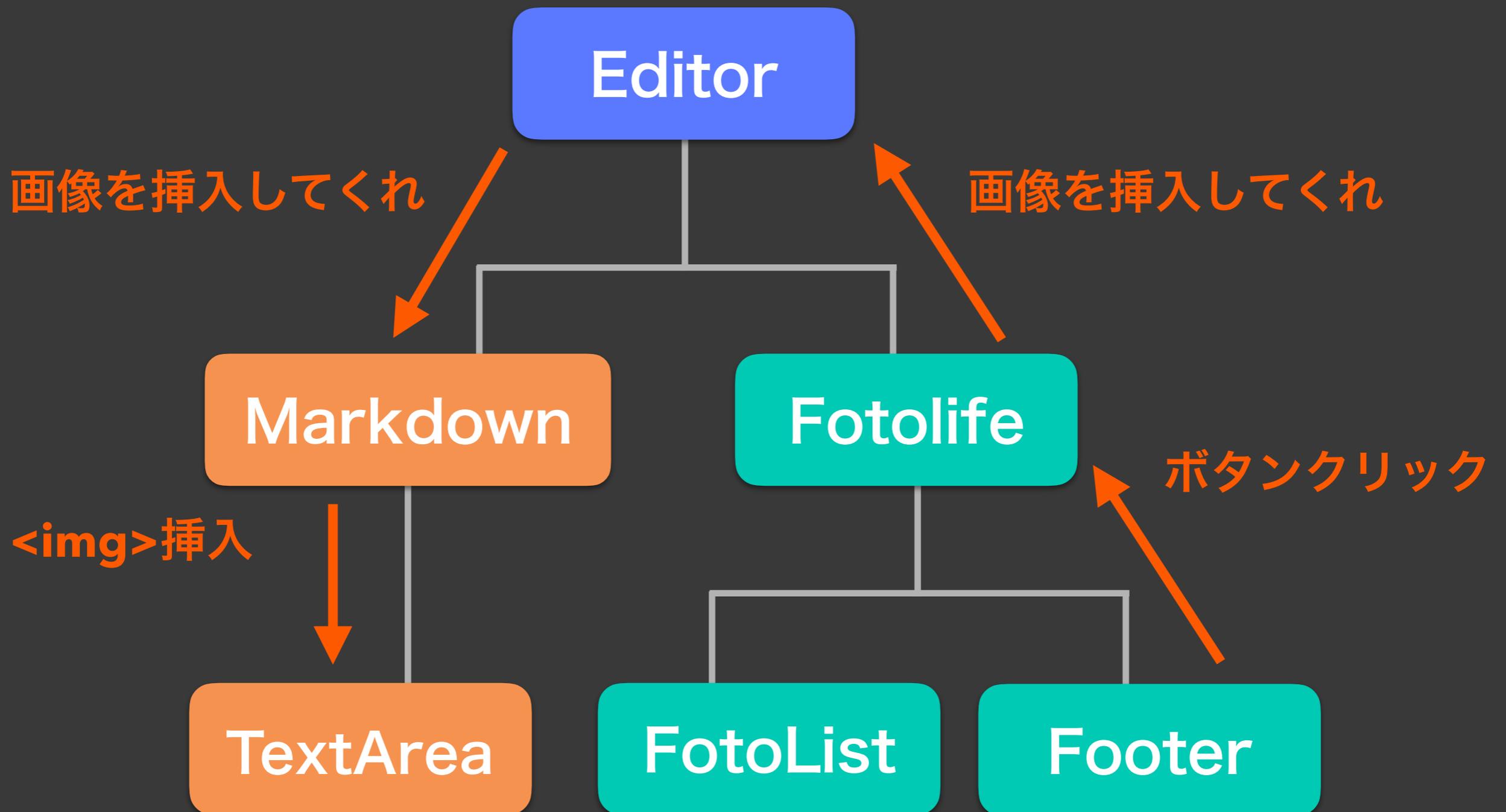
# 抽象的なイベント名にする

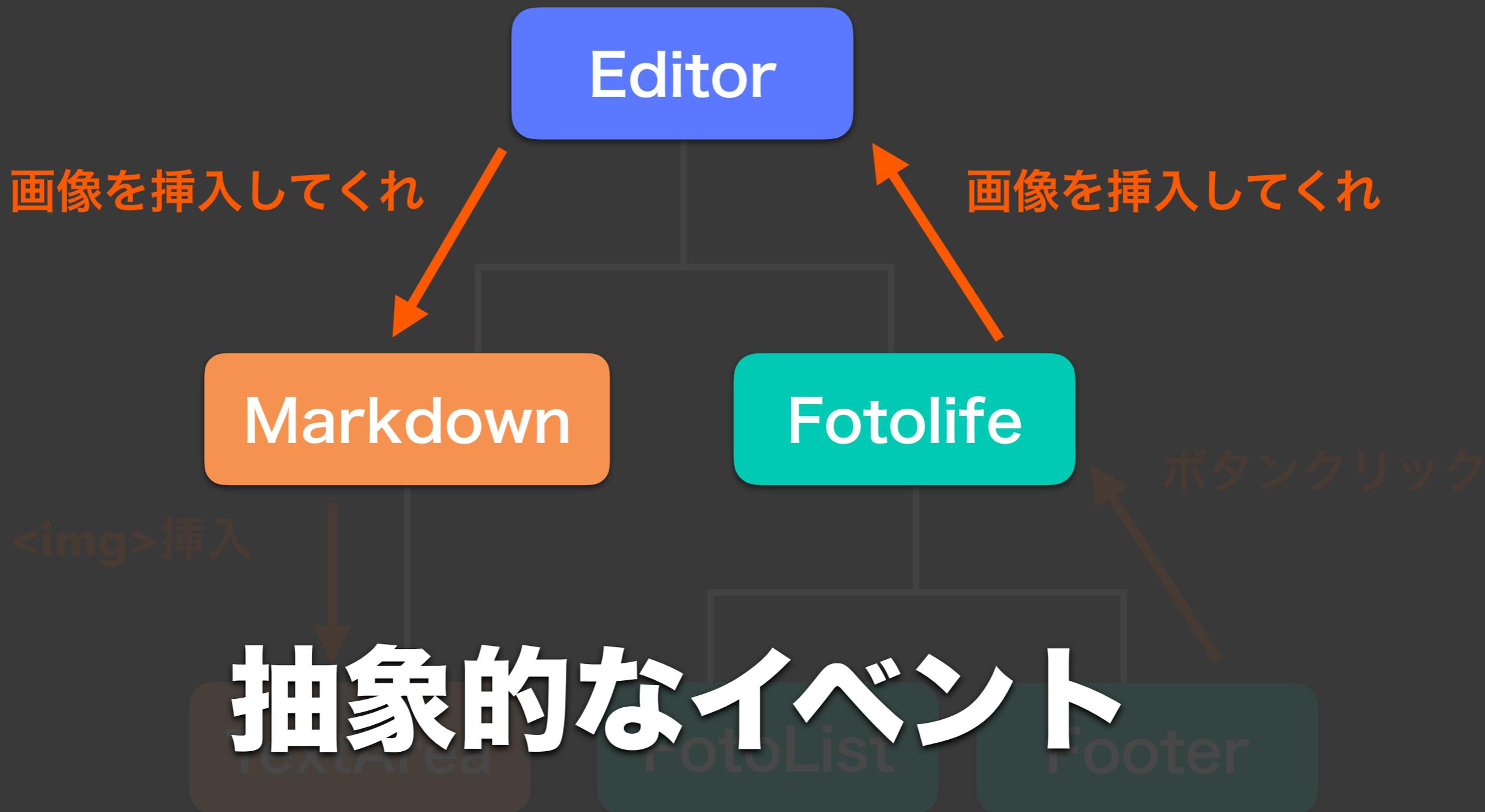


`buttonClicked`



`insertImage`





**データとDOMの**

**不整合は**

**起きにくくなった！**

が

# けっこう大変💧

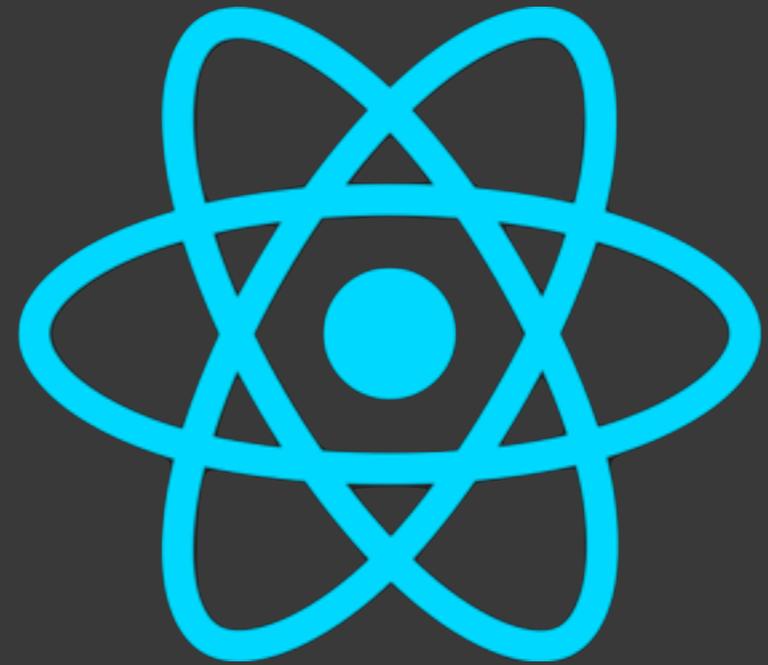
- ・ 設計の合意とるのが大変
- ・ DOM操作ダルい
  - ・ ライブラリにやってほしい

**Viewフレームワーク**

**使いたい………！**



**vs**

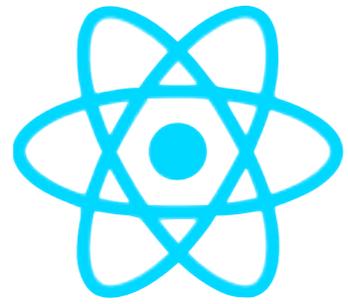




✓ 古き良きViewModel

✓ 日本語ドキュメント

✗ いろんな書き方できる



# React

✓ 覚えることが少ない

✓ 情報が豊富

✗ JSX

# 1ヶ月ほど議論

- ・ 学習コストは？
- ・ 必要な時に捨てられるか？
- ・ 新機能を両方で実装したり

新機能でViewModelフレームワーク使いたい.md

VM.md

Raw

# 新機能でViewModelフレームワーク使いたい

## 動機など

- 
- リファクタしててこういう処理書いたりして、「何故人間がDOMの表示とかイベント監視まで管理せねばならないのか??」という気持ちになる

## Vue.jsとReactの比較

### Vue.js

#### pros

- 日本語ドキュメントが充実
- 古き自きMVMのVMって感じ

<> Code

Revis

Embed URL

<script

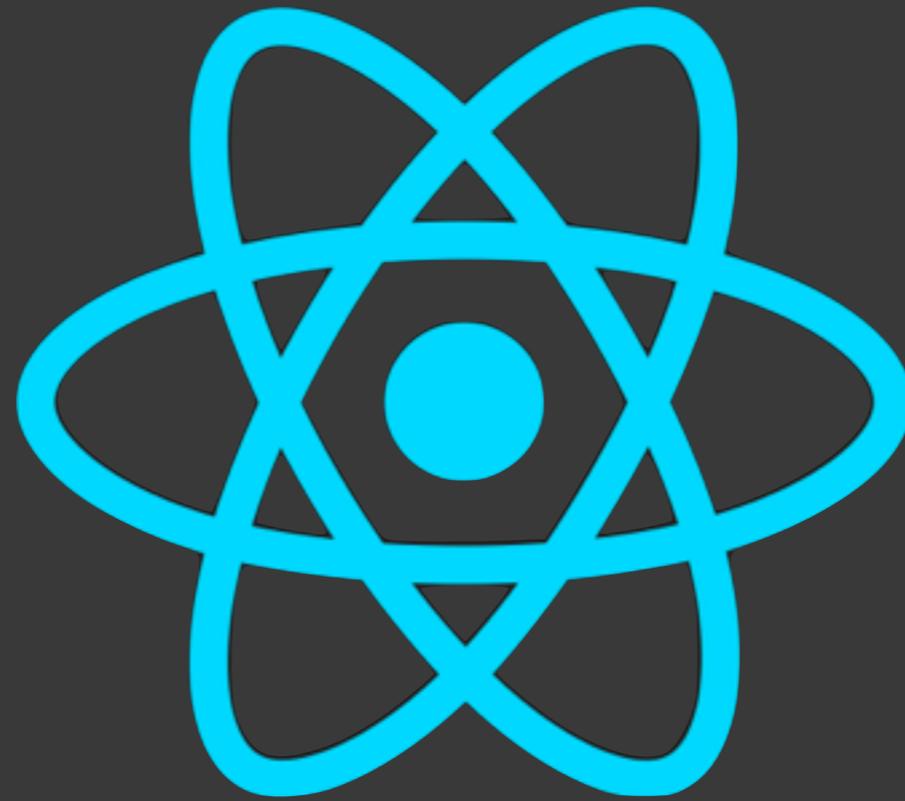
SSH clone U

git@ghe.

You can clone  
or Subversio

Clon

Down



# Reactを採用

- ・ よりシンプルに書ける
- ・ React → Vue は簡単そう

# デザイナーが触れない？

- ・ JSXくらい触れるはず
- ・ CSSで自在にスタイルできる  
よい設計を目指す

# BABEL 導入

- JSXのために導入という口実
- stage2 以下は使用禁止
- 勉強会, issueで知見をシェア

- Introduction
- 1. 自己紹介
- 2. ECMAScriptとは?
- 3. ECMA-262
- 4. TC39
- 5. ECMAScript 6 / 2015の呼び方
  - 5.1. ES6のエディタ
- 6. Ecma標準とISO標準の違い
- 7. ECMAScript 7 / 2016
- 8. ES.nextのエディタ
- 9. ES.nextと策定プロセス

## 5段階のStage

- 0 Strawman - アイデア
- 1 Proposal - 提案
- 2 Draft - ドラフト
- 3 Candidate- 仕様書と同じ形式
- 4 Finished - 策定完了

次期ECMAScriptは1年ごとに出るので、その時までにStage 4となったものが次期ECMAScriptに入る。

**参考: ECMAScriptとは何か**  
**by @azuさん**

**<http://azu.github.io/slide-what-is-ecmascript/>**

- 9.1. 5段階のStage
- 10. TC39 F...
  - 10.1. 例)
  - 10.2. 何で1年ごとにリリースするの?
  - 10.3. 誰がプロポーザルを書いてるの?
- 11. 新しいプロポーザルを提案するには
- 12. ECMAScriptのGitHub
- 13. 仕様についてはどこで議論?
  - 13.1. 最近の面白い議論の流れ
- 14. 結局ECMAScript 2016って何が入る...
  - 14.1. ECMAScriptに静的型って入る...

# Arrow Function

- return間違えやすい

```
const a = 100;
```

```
( () => ( a ) )(); // 100  
( () => { a } )(); // undefined  
( () => ({ a }) )(); // { a : 100 }
```

## 社内勉強会のようす

ES6/7の雑な紹介 by fand/amagitakayosi

Published November 4, 2015

# Arrow Function

- return間違えやすい

```
const a = 100;
```

```
( () => ( a ) )(); // 100  
( () => { a } )(); // undefined  
( () => ({ a }) )(); // { a : 100 }
```

## Share

# 最新ジャバスクリプト会場 #9508

Edit

Open

hitode909 opened this issue on 15 Sep · 17 comments



hitode909 commented on 15 Sep

Owner



ES6やReactなどなんかいい情報があったら知りたい



yohei commented on 15 Sep

Owner



<http://www.slideshare.net/teppei/effective-es6> teppei さんのいい資料



hitode909 commented on 15 Sep

Owner



そういえばヤブシーのES6のやつ聞きたいのだった



hitode909 commented on 15 Sep

Owner



<https://kangax.github.io/compat-table/es6/> feature name のところみれば何がつかえるかわかる?



amagitakayosi commented on 15 Sep

Owner



だいたいteppeiさんののでよさそうだけどbabel公式にもガイドある

<https://babeljs.io/docs/learn-es2015/>

Labels

None yet

Milestone

No milestone

Assignee

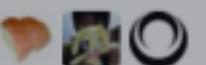
No one — assignees

Notifications

Unsubscribe

You're receiving notifications because you commented.

3 participants



Lock conversation

# issueで知見を集める

# React Architecture

React使用時のコンポーネント設計

# 基本的な考え方

- ・ Fluxフレームワークは使いたくない
- ・ EventEmitterだけで頑張る
- ・ 規模によって、段階的に  
Fluxっぽくしていく

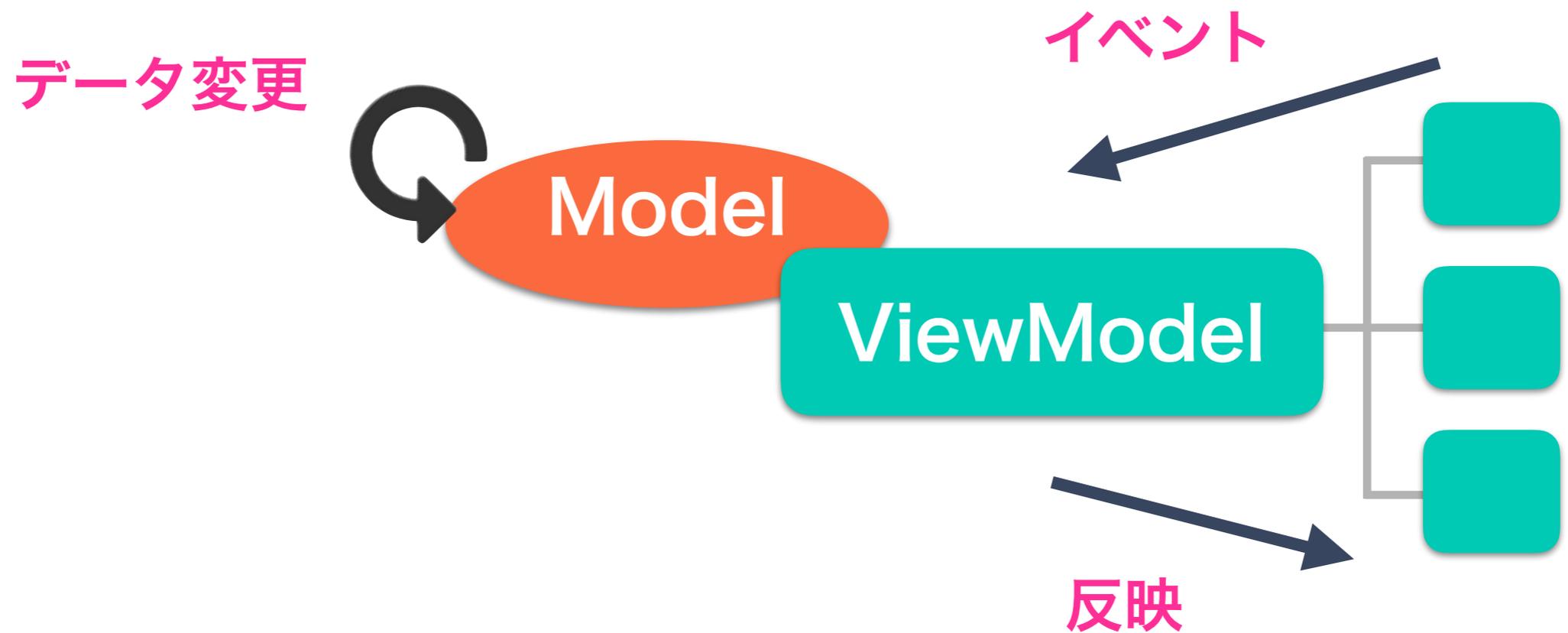
# 小さな機能のとき

- ・ ViewModelだけで完結
- ・ 以前のコードの `render()` を置き換えるだけ

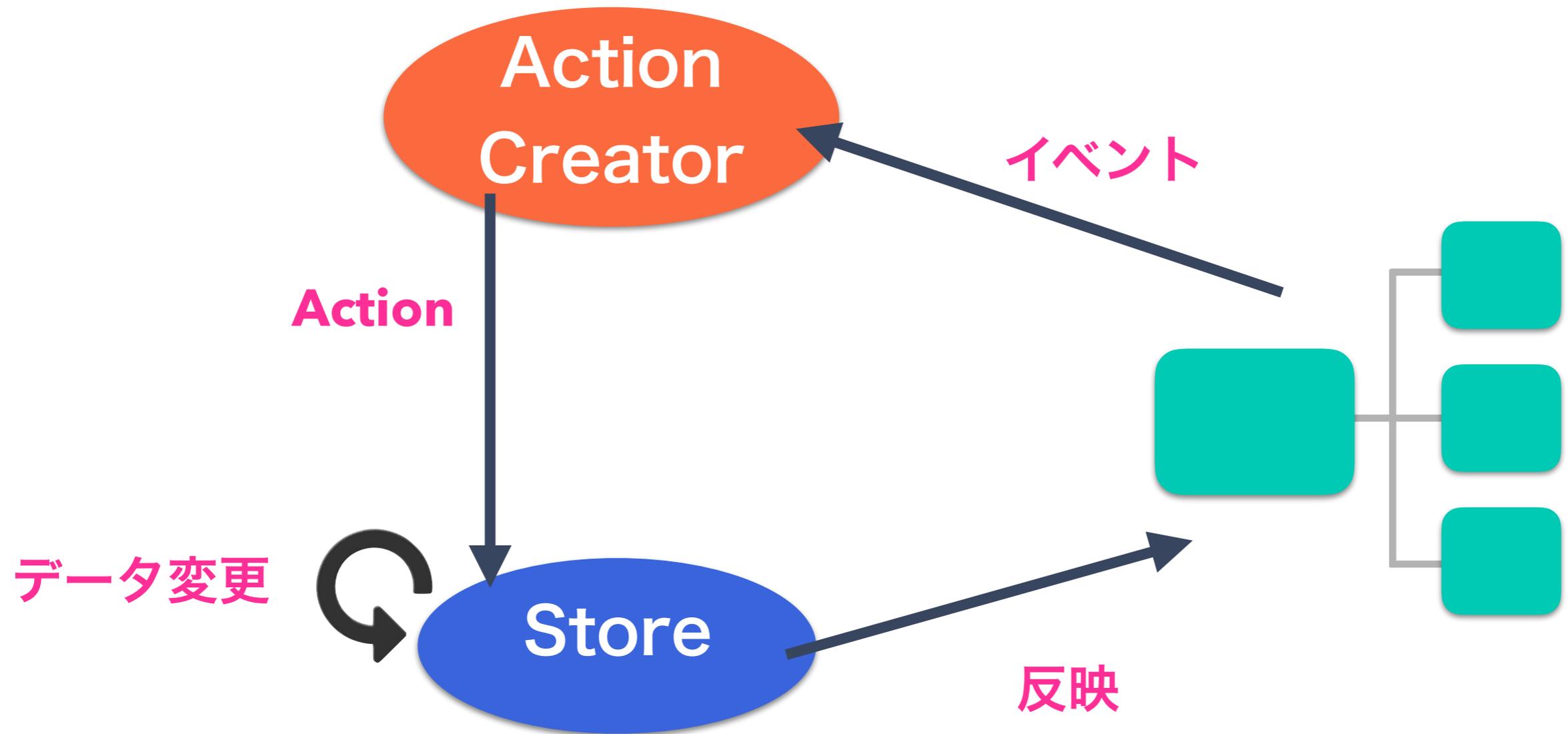
# 大きな機能のとき

- EventEmitterで部分的にFlux
- Store, ActionCreatorが複数ならDispatcherを導入

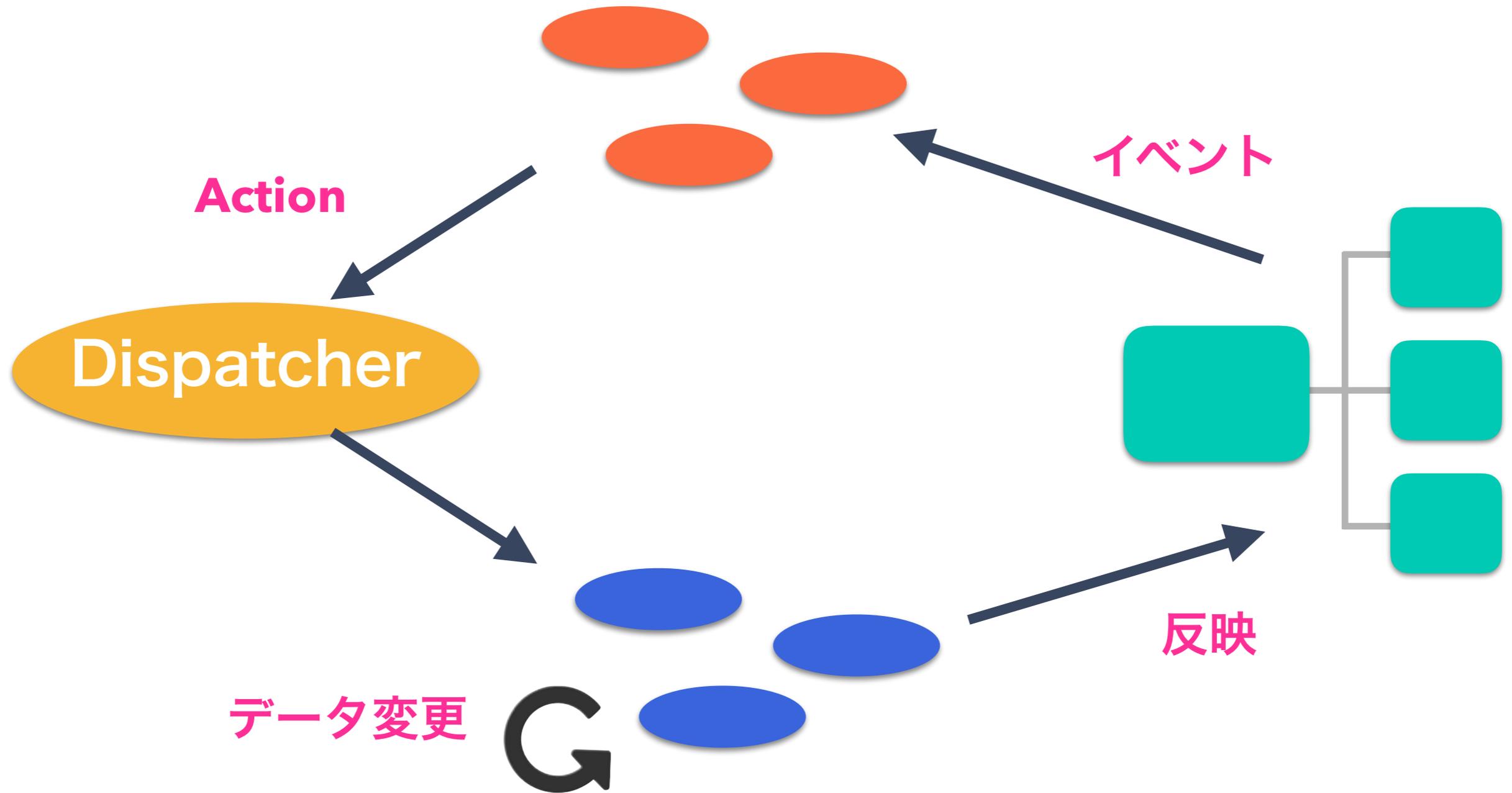
# 小規模 : ViewModel



# 中規模：Flux風



# 大規模 : Dispatcherを導入



# Coding Convention

コーディングスタイルのチェック

従来はJSHintでチェック

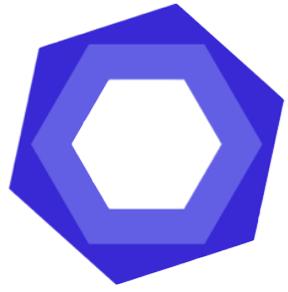
独自ルールはPerlで書いてた

✗ ルールを微調整しにくい

✗ JSX / ES6 に非対応



ESLint



# ESLint

- ✓ 標準ルールが充実
- ✓ 設定がカスタマイズしやすい
- ✓ 独自ルールが簡単に書ける

## 日本語禁止するESLintルール

no-japanese.js

Raw

```
1  /**
2   * @fileoverview Rule to forbid writing Japanese
3   * @author amagitakayosi
4   */
5
6  "use strict";
7
8  var path = require('path');
9
10 //-----
11 // Rule Definition
12 //-----
13
14 module.exports = function(context) {
15   var rule = {
16     "Program": function checkForForbiddenCharacters(node) {
17
18
19       var excludePath = context.options[0].excludePath;
20       var excludePathFull = path.resolve(__dirname, '..', excludePath) + '';
21       if (context.getFilename().match(excludePathFull)) {
22         return;
23       }
24
25       // 許容する文字を指定
26       var allowedChars = context.options[0].allowedChars || '';
```

参考: 独自ルール例

<https://gist.github.com/fand/c13213f81bfce11f4132>

# 導入の方針

- ・ 全ての標準ルールでエラーが出るように設定
- ・ 既存コードでエラー消えるまでルールを緩める

# ●のついた行は 全部警告

```
37 var messenger = Messenger.createForWindow(curationWindow, url);
38
39 • messenger.addListener('insertLines', function(data) {
40     self.insertLines(data);
41 });
42 },
43
44 • _validate: function(data) {
45     • _each(['html', 'hatena', 'markdown'], function(mode) {
46         • if (!data[mode]) throw "missing " + mode;
47     });
48 },
49
50 • _setup: function() {
51     • if (this._insertLinesImplement) return;
52
53     // モードに合わせて実装の関数を設定
54     var isPC = Hatena.Diary && Hatena.Diary.Pages.device() === 'pc' && Hatena.Diary.Editor.currentEditor;
55
56     var isTouchEditor = Hatena.Diary && Hatena.Diary.Pages.device() === 'touch';
57
58     var isChildFrame = window.opener;
59
60     // ?ios=1というパラメータついでるときiOS
61     var isiOS = this.isiOS();
62     // ?android=1というパラメーターのときAndroid
63     var isAndroid = this.isAndroid();
64
65     if (isiOS || isAndroid) {
66         // 規定のURL開く
67         • this._insertLinesImplement = this._insertLinesOpenURL;
68     } else if (isPC) {
69         // PC編集画面
70         • this._insertLinesImplement = this._insertLinesPC;
71     } else if (isTouchEditor) {
72         // touch編集画面
73         • this._insertLinesImplement = this._insertLinesTouch;
74     } else if (isChildFrame) {
75         // 親フレームにpostMessageで送信
76         • this._insertLinesImplement = this._insertLinesMessenger;
77     } else {
78         • throw "handling failed";
79     }
80
81 },
82
83 • _insertLinesPC: function(data) {
84     var editor = Hatena.Diary.Editor.currentEditor;
85     var mode = editor.mode;
86     var lines = data[mode];
87     editor.insertLines(lines);
88 },
89 • _insertLinesTouch: function(data) {
90     var syntax = $('#edit-entry').attr('data-entry-syntax');
91
92     // HTMLモードで、contentEditable無効のとき(Androidのデフォルトブラウザなど)、ただのtextareaなので、はてな記法を使う
93     • if (syntax === 'html' && !Hatena.Diary.Pages.AdminTouch['user-blog-edit'].contentEditableEnabledAndEditorModeIsHTML)
94         syntax = 'hatena';
95     }
96
97     • var lines = (data[syntax] || data['hatena'] || data['markdown']);
98
99     • Hatena.Diary.Pages.AdminTouch['user-blog-edit'].insertText(lines.join("\n"));
100 },
101 • _insertLinesMessenger: function(data) {
102     var messenger = Messenger.createForParent();
103     messenger.send('insertLines', data);
104     window.close();
105 },
106 • _insertLinesOpenURL: function(data) {
107     • var html_lines = _map(data.html, function(line) {
108         • return '<p>' + line + '</p>';
109     });
110     • location.href = 'hatenablog:/entry/body/insert?' + $.param({
111         • hatena: data.hatena.join("\n"),
112         • html: html_lines.join(""),
113         • markdown: data.markdown.join("\n\n"),
114         • version: 1
115     });
116     }
117 };
118
119 module.exports = EditorConnector;
120
```



**miloue707** 2:30 PM

eslint入れたらひたすら怒られてて便利



**amagitakayosi** 2:31 PM ★

酷いオギスしみたいなのあったら教えて

**怒られて便利！**

**TEST**

**THE**

**COMPONENT**

**Mission 3:**

# 4月までのJSテスト

- ・ CasperJSのE2Eのみ
- ・ 記事投稿、広告の表示だけ

# JSが壊れやすい💧

- ・ リファクタリングでエンバグ
- ・ リリース後エラーでまくって  
Revertした事も……

JavaScriptテストの疑問、お ×

← → ↻ 0-9.sakura.ne.jp/pub/frontrend/start.html ☆ ☰

Q 「で、今手元にあるコードをどうしたら良いの？」

A 「まずはE2Eテストフレームワークを導入してみては」

kyo ago ∅ ‹ ›

<http://0-9.sakura.ne.jp/pub/frontrend/start.html>

**「E2Eから始める」と言うけれど……**

- ・ **シナリオが複雑**

**「ある順序でページ遷移した時に  
勝手にタブが閉じてしまう」**

**バグのテストしたくない……**

**テストしたい！**

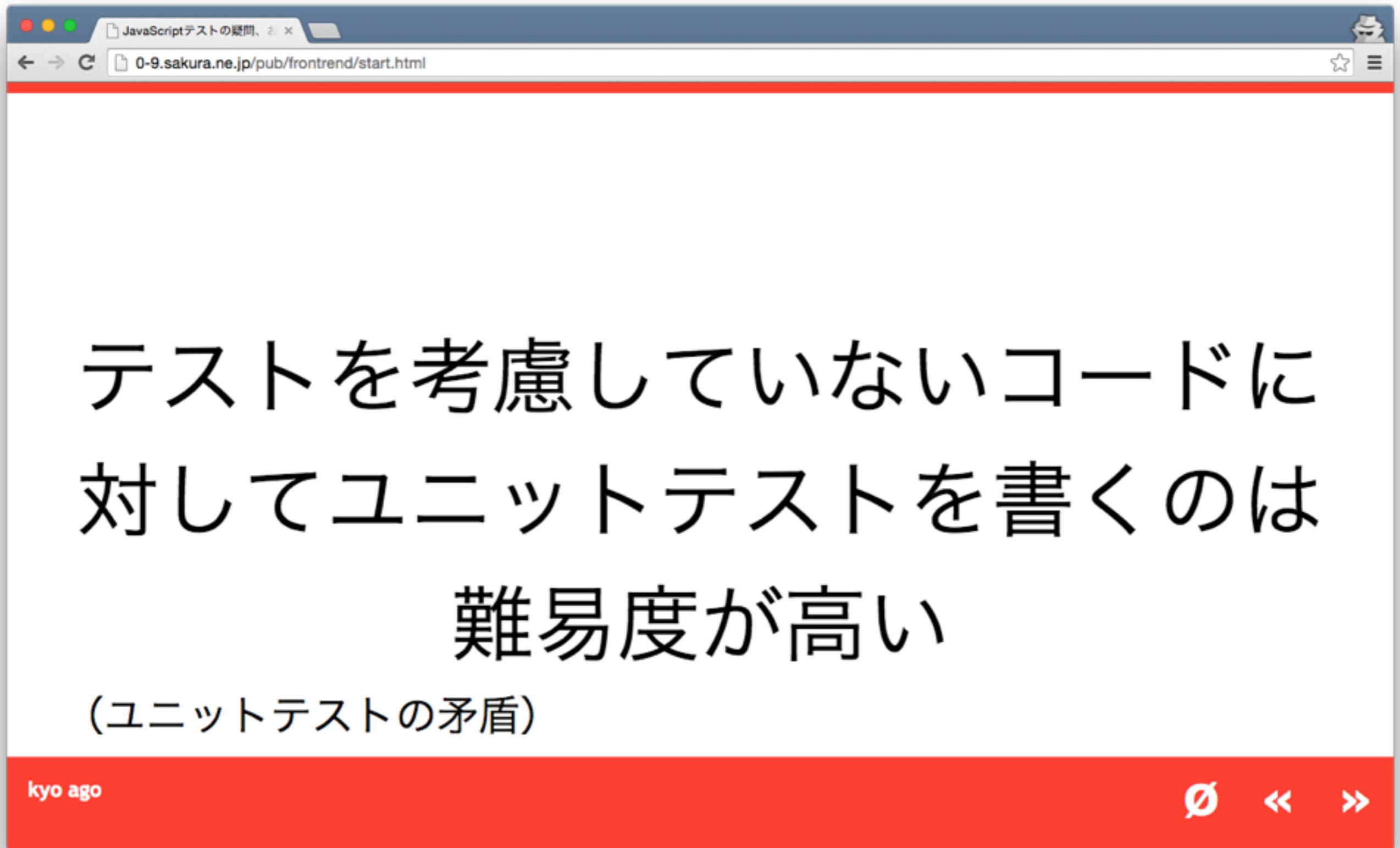
# 単体テストからはじめよう

 **amagitakayosi** commented on 5 Jun

E2Eテストむずいけど単体テスト書けばええやんということに気づいた

# Unit Test

単体テスト環境構築



JavaScriptテストの疑問、お ×

0-9.sakura.ne.jp/pub/frontrend/start.html

テストを考慮していないコードに  
対してユニットテストを書くのは  
難易度が高い

(ユニットテストの矛盾)

kyo ago

⊘ ‹ ›

<http://0-9.sakura.ne.jp/pub/frontrend/start.html>

**理想**

**テスト書く**



**テスト落ちないよう  
リファクタリング**

現實

**テストバブルな設計で**

**リファクタリング**



**テスト書く**

# テストブルな設計を心がける

- ・コード分割は大前提
- ・View操作は分離する
  - ・DOM触るモジュールはテストしづらい



- 元はAngular用のテストランナー
- ブラウザを使ってテストできる
- 単体テスト向け
  - ブラウザ操作は充実していない

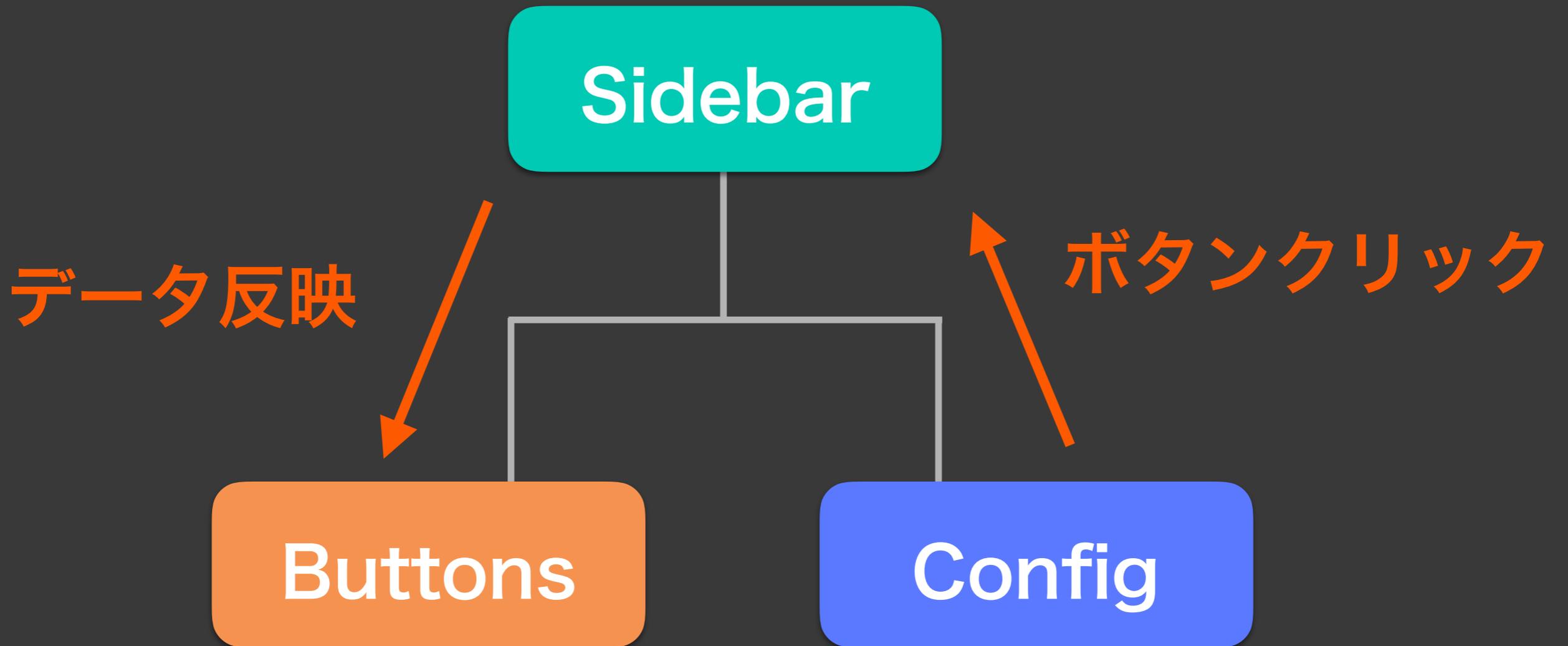
# 少しずつテスト増やす

- ・ 新しいモジュールはテスト必須
- ・ リファクタリング→テスト追加
- ・ Util系モジュールを優先

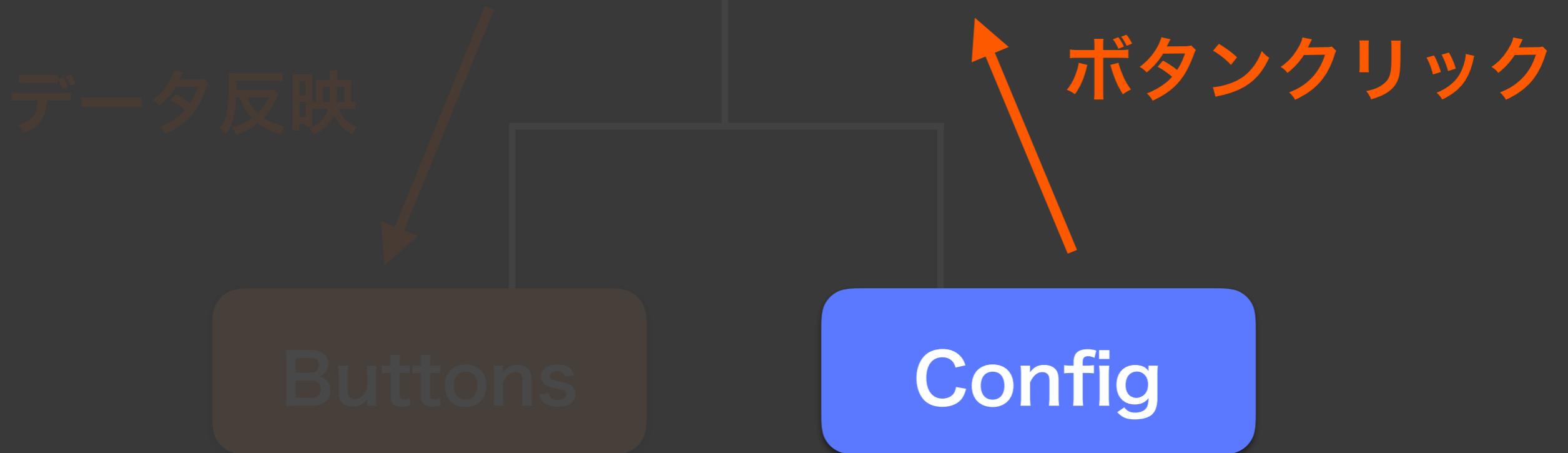
# Viewのテスト

- EventEmitterのおかげでテストしやすくなった！
- イベントは発行されるか？
- DOM要素が描画されてるか？

# データ変更 + emit('change')



# イベントが 正しく発行されるか？



データ変更 + emit('change')

Sidebar

データ反映

ボタンクリック

イベントに対し

データが変更されるか？

# データは emit('change') 正しく描画されるか？

データ反映



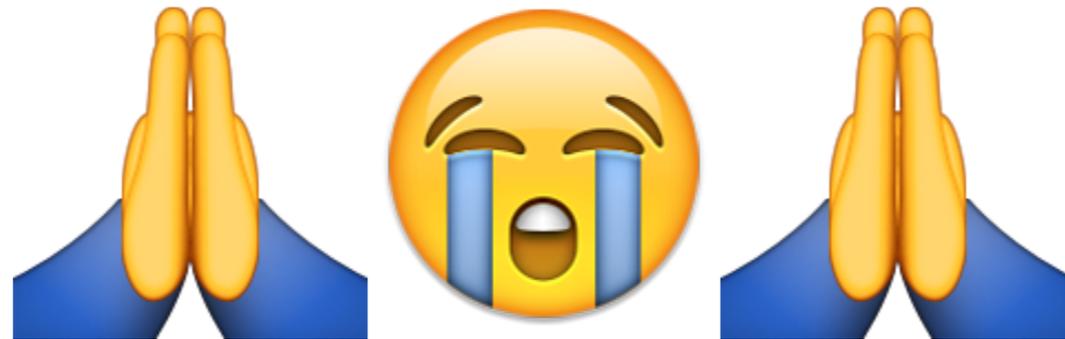
Buttons

ボタンクリック

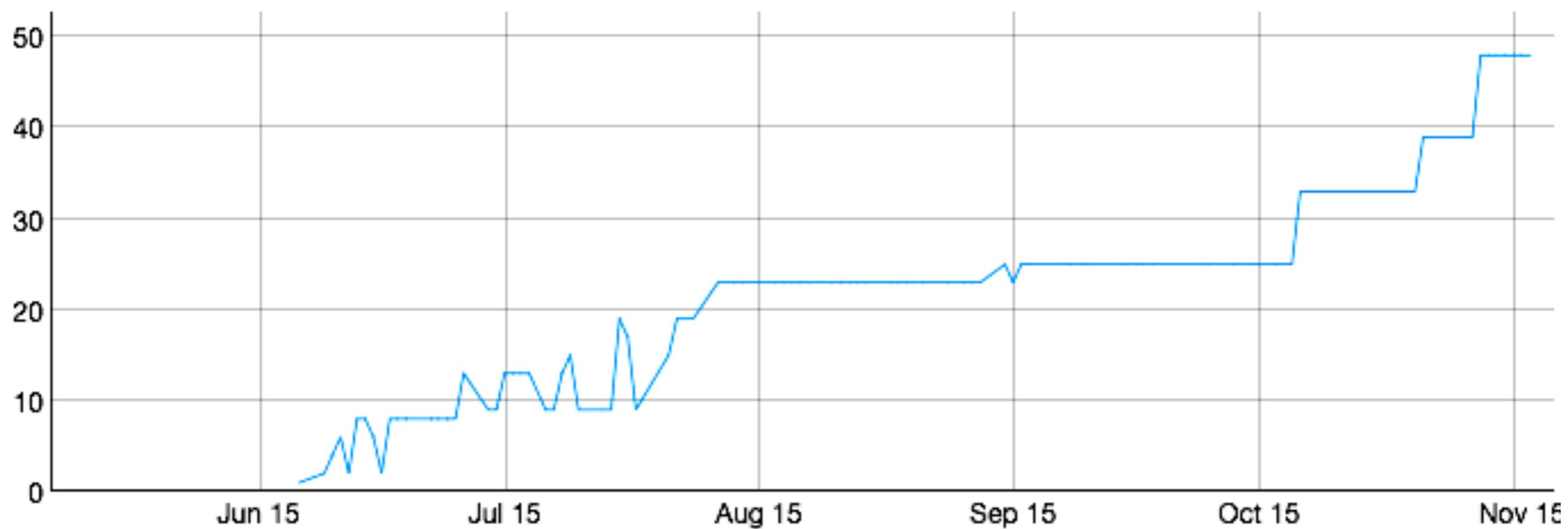


Config

# テスト書ける



# テストファイル数



27 of 127 SUCCESS (0.771 secs / 0.246 se

まだまだまだまだ

): Executed 127 of 127 SUCCESS (0.771 secs / 0.246 sec

発展途上

# E2E Test

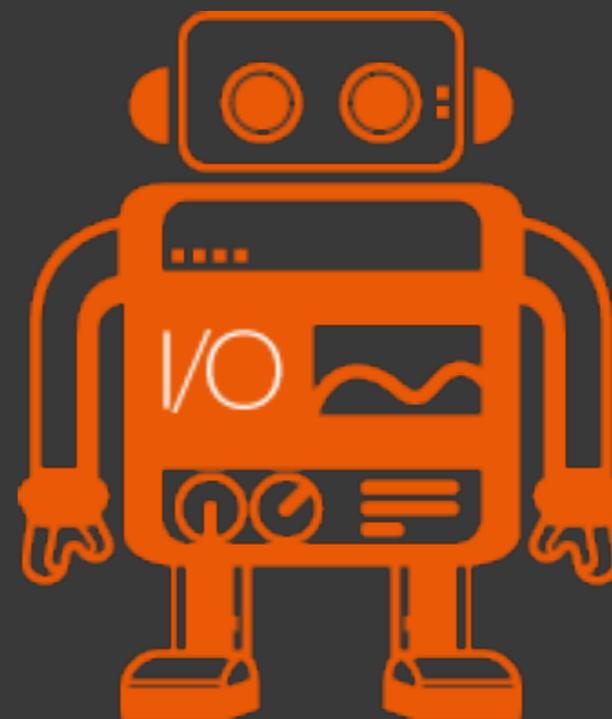
E2Eテスト環境のリニューアル

# 脱CasperJS

- Reactを使うと  
PhantomJSでエラー出る！
- PhantomJS、難しい

# 他の手段を検討

# NIGHTMARE



# NIGHTMARE

## Electronベースのスクレイパー

- ✓ シンプルなAPI
- ✗ 細かいAPIが足りない
- ✗ メソッドチェーンが面倒



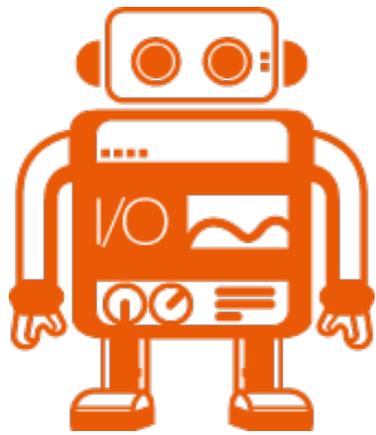
# Protractor

元はAngularJSのテスト用

✓ ユーザー、情報が多い

✓ 同期的に書ける

✗ Angular専用APIは不要



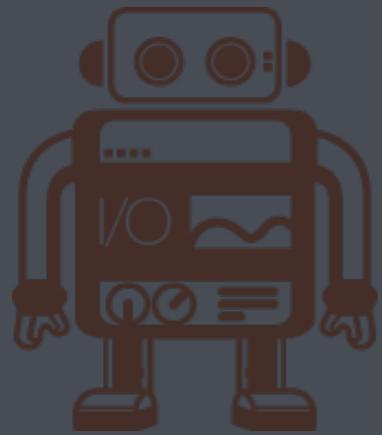
WEBDRIVER 

## Seleniumのバイインディキング

✓ 同期的に書ける

✓ 割とドキュメント充実

 CLIテストランナー付き



WEBDRIVER I/O

Seleniumのバイインディキング

# 採用



同期的に実行する



割とドキュメント充実



CLIテストランナー付き

# 一番の理由

- Selenium系なので  
他ライブラリに乗り換えやすいはず
- Protractor, testium 等

# ES6 generatorで テストを書く

```
describe('管理画面トップ', () => {  
  
  it('編集画面へのリンクがある', function* () {  
    yield browser.url('http://xxxxxxxxxx');  
  
    assert(  
      yield browser.isExisting('.new-entry'),  
      'リンクが存在する'  
    );  
  
    yield browser.click('new-entry');
```

# ES6 generatorで

## テストを書く

```
yield browser.click('.new-entry');
```

```
assert.equal(  
  yield browser.getTitle(),  
  '編集画面',  
  '編集画面に遷移する'  
);
```

```
});
```

```
});
```

## ダッシュボード

- 記事を書く
- 記事の管理
- カテゴリー
- コメント
- アクセス解析
- 設定
- デザイン
- インポート
- ブログメンバー
- 購読中のブログ
- アカウント設定
- グループ
- Proにアップグレード
- ヘルプ

お知らせ

### マイブログ

 [Redacted]'s diary

すべての人に公開

[ブログを表示](#) | [メインブログ](#)

[記事を書く](#) [記事の管理](#) [設定](#)

新しいブログを作成

### 購読中のブログ

購読中のブログはありません。

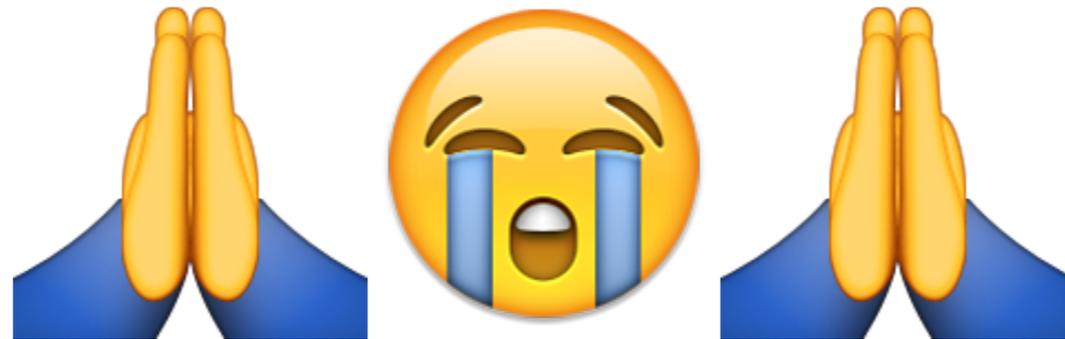
ブログに表示されている「購読者になる」ボタンを押すと、更新を簡単にチェックできます。気になるブログを見つけたら読者になってみましょう！（無料）

- [注目のブログから探す](#)

購読中のブログ

新着エントリー

テスト動いた！



2015-10-30

## React入れたらCasperJSのE2Eテスト壊れたのでPhantomJS2入れたらPhantomJSのユニットテスト壊れたのでwebdriverioでE2Eテスト書き直した話

これまで VanillaJS / jQuery で頑張ってたプロジェクトに React 入れて、Reactは最高！などと言っていたのだけど、E2Eテストが落ちている事に気づいた。

**参考: おもしろエピソード**

どうやら PhantomJS1系で見られる現象のようだった。

<http://amagitakayosi.hatenablog.com/entry/2015/10/30/083000>

e2e - CasperJS で Reactjs のサイトをテスト - Qiita

E2EテストにはCasperJSを使っていた。

Casper の dependencies は `"phantomjs": ">=1.8.2"` となっている。

# RESTORE

# THE

フロントエンドに**秩序**を取り戻す方法

@amagitakayosi

# ORDER

かくくして

秩序は

守られた

# 今日話したこと

- ・ リファクタリング勘所
- ・ 漸進的にアーキテクチャを進化させる
- ・ フレームワーク導入
- ・ テスト環境構築

今日話したこと

良い設計を

目指すための

土台作り

- ・ リファクタリングが所
- ・ 目的や技術が異なる
- ・ フレームワーク導入
- ・ テスト実装構築

# 今日話さなかったこと

- ・ 美しいライブラリ設計の共有
- ・ 非エンジニアとの設計
- ・ 爆速でサービスつくる開発フロー

今日話さなかったこと

まだまだ

・ はてなブログ  
・ 社内ライブラリの設計方法

発展途上

・ サクヤク

めげず

に  
改善する

スピリット

良いサービスを

作っていきなせ!

!!!

!!!

The background features a repeating pattern of light blue diamonds arranged in a staggered grid. The diamonds are semi-transparent and have a subtle gradient, creating a clean and modern aesthetic.

**ありがとうございました**

**株式会社はてなでは  
最高のアーキテクチャを目指し  
あくなき探求をつづける  
エンジニアを募集しています！**