

# Vue.jsプロジェクトの 爆発させかた

2018.06.26 マジ卍 #ichigayageek HANATANI Takuma(@potato4d)

# 自己紹介

## HANATANI Takuma(@potato4d)

- フリーランスのWebエンジニア
  - FrontEnd / Node.js / PHP/ AWS / etc...
- Vue.js日本語公式ドキュメントメンテナ
- Vue.js Japan User Group スタッフ
- nuxt/docs 英/和ドキュメント貢献
- Nuxt tech book Author



# JS History

---

- Vue.js 2015 ~
  - Rails & Vue, SPA, SPA + SSR, Nuxt.js, with TypeScript
- React 2016 ~
  - SPA, PHP & React, Rails & React, with TypeScript, with Flow
- Angular 2017 ~
  - SPA, ほぼ触ってない

Vue:React:Angular=14:4:1 な感じ



# 帝都研究所 BOOTH出張所

HANATANI Takuma



## Nuxt tech book

超高生産性をあなたに



著 : potato4d  
表紙絵 : 月夜色

すべての Vue.js ユーザー に贈る

SPA 開発の新たなデファクトとなる  
超高生産のフレームワークをあなたに。

Tech Books

技術書典4

【技術書典4新刊】 Nuxt  
tech book 【PDF版】

Digital

1,600 JPY

Add to Cart

Love it!

47

ツイート

いいね! 21

- JavaScript
- 技術書典
- Vue.js
- 技術書典4
- 技術書典4新刊
- Nuxt.js

8712-db0c2178aeb2/f/824745/e53a17ee-32aa-4a5e-90af-3685f193d4a6.png

<https://potato4d.booth.pm/items/824745>

今日話すこと



# Vue.jsプロジェクトの 爆破方法

**Why?**



Vue.jsを褒める情報は  
いくらでもあるので.....

## テーマ

フロントエンドを代表する(?)5つのライブラリ/フレームワークを対象に下記のような内容で登壇していただきます。

- なぜそれを選んだか。
- プロダクトで実際に使ってみて嬉しかったこと、つらかったこと。
- 思ったより簡単だったこと、**思ったより罨だったこと。**
- ドキュメントやリリースノートからはなかなか得られない知見。
- 次は何を使いたいか。
- (登壇者がこれを話したい！というものがあればそれを優先)

明日役に立たないけど  
いつか役に立つ話

# 改めて今日話すこと

---

- 「Vue.jsは誰でも簡単に作れるけど小規模向け」は本当か
- 過去のプロジェクトの反省からみる Vue.js の定説の評価
  - Vuex があってもなくても良いは本当か？
  - デザイナーでも触れて一緒に仕事できるは本当か？
  - バックエンド中心のWebアプリケーションへのちょい入れに強いは本当か？
- Vue.js を新規で使う場合に選択すべきこと

# 想定ターゲット(1)

---

- モダン JavaScript と設計について最低限知っている人
  - 「React, Vue, Angular ってやつが人気なんですよ」
  - 「Vue.jsは簡単にかけてハードルが低い初心者向けのやつですよ」
  - 「Fluxパターンってシングルトンで一方向なやつですよ」

**みたいな雑な認識程度でも話の半分はわかるのでOK**

## 想定ターゲット(2)

---

- Vue.js でつらい思いをしそうな人・した人
  - 状態管理で悩んでいる人
  - プロジェクトの規模感・チーム感でどうしたら良いかわからない人

**これからの開発を助けになれるはず**

# 今日話すこと

---

- 「Vue.jsは誰でも簡単に作れるけど小規模向け」は本当か
- 過去のプロジェクトの反省からみる Vue.js の定説の評価
  - Vuex があってもなくても良いは本当か？
  - デザイナーでも触れて一緒に仕事できるは本当か？
  - バックエンド中心のWebアプリケーションへのちょい入れに強いは本当か？
- Vue.js を新規で使う場合に選択すべきこと

「結局Vue.jsって小規模向けなんでしょ」



A. そうでもない

A. そうでもないが

「ちゃんと開発」するのにコツが必要

Vue.js は PHP とよく似ている

あえて明言する 特徴

# Vue.js is

---

- 「雑に組もうとするととことん雑に組むことが快適にできる」
- 「雑に組むと簡単にプロジェクトを爆発させる」ことができる
- 「爆発しないプロジェクトを作るうとすると極端に難しくなる」
- 「油断すると爆発するけど PHP ほど爆発して困った話をしたがない」

# Vue.js is

---

- 「雑に組もうとするととことん雑に組むことが快適にできる」
- 「雑に組むと簡単にプロジェクトを爆発させる」ことができる
- 「爆発しないプロジェクトを作ろうとすると極端に難しくなる」
- **「油断すると爆発するけど PHP ほど爆発して困った話をしたがない」**

困った話 with 定説

# 今日話すこと

---

- 「Vue.jsは誰でも簡単に作れるけど小規模向け」は本当か
- **過去のプロジェクトの反省からみる Vue.js の定説の評価**
  - Vuexがあってもなくても良いは本当か？
  - デザイナーでも触れて一緒に仕事できるは本当か？
  - バックエンド中心のWebアプリケーションへのちょい入れに強いは本当か？
- Vue.js を新規で使う場合に選択すべきこと



過去のプロジェクトの爆発からみる

Vue.js の定説の評価

巻きで 3 つ紹介

1 / 3

「Vuex あってもなくても良い説」

A. 80% ぐらいの確率で嘘になる

Vuex使わず爆発した話

# Vuex を使わないで爆発した話(1)

---

- 2016年ぐらいの開発の話
- Vuex を Flux 的に Single source of truth にする時代のほうが多かった頃の話
- Single source of truth を遵守するのはやりすぎだと思い Pure JavaScript Object での Store 層を作成
- データとして確定した情報のみを格納。一時ステートはVueコンポーネントに持たせた
- require/import は参照渡しにになるので書き換えるとうまい具合に双方向バインディングと噛み合ってくれる

## Vuex を使わないで爆発した話(2)

---

- 初期開発は JS Object らしい取り回しと効率の良さがうまく効いていた
- 双方向バインディングの恩恵もあって、Reactivity を全面に活用したコードが書けていた



## Vuex を使わないで爆発した話(3)

---

- 仕様が変わったり、特定の API リソースの極端な肥大化によって Pure JS オブジェクトから読み取れないことが多くなってくる
- 使われていない確証を得ることが難しく、段々と開発がつらく.....
- 大きくなるアプリケーションは Vuex を入れておくべきだった

爆発させないために

# 「Vuex を使わないで爆発」を防ぐ(1)

---

- SPA でも Vuex なくても JS Object やストアパターンでいけるは大体嘘
- とりあえず Vuex は必ず入れておいて、ステートの利用を考える
- Vuex は公式に **Single source of truth** に拘る必要はないと明言しているので、ローカルステートの是非で Vuex を入れるか決めないで組み合わせる
- 自分が関わるプロジェクトは現在全て Vuex + ローカルステートで運用中

## 「Vuex を使わないで爆発」を防ぐ(2)

---

- ローカルステートを全く使わない開発は疲弊する。それをやるなら React で良いので Vue はローカルステートをうまく扱うものとして使うこと
- ローカルステートを使うことを許容すると「そのコンポーネントでしか使われない」という状態をさっさと破棄できるので開発レベル差異を吸収できる
- Vuex はドメイン層と事実を Store に、プレゼンテーション層と状態をローカルステートに持たせるべき
- とはいえどのみちVue は脳内の RAM 容量が大きい人向けのフレームワークなのは間違いない

## 余談) Vuex を使っても爆発させる方法

---

- Single source of truth を完全に遵守すると、Redux ほど名前空間を細かく分割しない都合上大体 state が過労死する
- createPostForm って中にフォームの要素いっぱい入ってるけど投稿作ってる箇所とか5個以上あるしどこのフォームだよ
- プレゼンテーション層の都合が何故かドメイン層に書かれていたりして破壊してくる
- レイヤーは分けましょう

2 / 3

「デザイナーでも触れて一緒に仕事できる」

説

A. マジ松



デザイナーと一緒に仕事すると  
爆発したりしなかったりする

## デザイナーと一緒に仕事すると爆発したりしなかったりする(1)

---

- 正直デザイナー + マークアップエンジニアができる人は Vue コンポーネントはかける
- ejs / erb / jade / Smarty / mustache とテンプレートエンジンを渡ってきたマークアップの人間であれば書くにあたって何も問題はない
- 「デザインは一流だけどマークアップは不慣れ」でも Scoped CSS で臭いものに蓋をできるのは最高
- ここでも「ここでしか使われていないもの」を簡単に破棄できる差異吸収が効いてくる

## デザイナーと一緒に仕事すると爆発したりしなかったりする(2)

---

- だいたい問題が発生するのは Node 環境の話なので環境の構築は代行する
- Storybook の stories の書き方まで教えて云々カンヌン.....とかやりだすと泥沼化する
- 適当な Sandbox コンポーネントを作るか stories は FE が書いてそこで無限に書いてもらったほうが結果的に幸福になる
- テストを書いてたらテストはぶっ壊れるので直す
- これさえやれば爆発しない (テストは爆発するかも)

3 / 3

Vue.js 既存の Web アプリの  
エッジ利用 JS として最強説

A. だいたいあつてる

Web アプリに Vue.js を  
差し込むと爆発しなくなる話

# Web アプリに Vue.js を差し込むと爆発しなくなる話(1)

---

## テンプレート記法がそこそこモダンで使いやすい

- サーバーサイド中心の(Slim や Blade などから HTML を出力している)プロジェクトに一部 JS を入れるなら Vue.js は最高
- [v-cloak] を使ってテンプレートの露出を防ぎつつ、バックエンドのテンプレートエンジンと組み合わせやすいのが素敵 (やりすぎると爆発する)
- デリミタ( `{{}}` )を自由に変えられる( `[]` など)のでどのテンプレートエンジンともかち合わない



# Web アプリに Vue.js を差し込むと爆発しなくなる話(2)

---

## 状態管理を雑にできるのが良い

- 双方向バインディングを利用すると適当にデータをぶち込むと適当にリアルタイムで反映される
- `this.setState()` みたいな概念がなく、適当にグローバルに状態を展開するとそれだけでストア層を展開できる
- `window.store` を作り出して薄いストア層を作るのがおすすめ

# Web アプリに Vue.js を差し込むと爆発しなくなる話(1)

---

## ローカルステートという概念があるので new Vue しやすいのが良い

- ローカルステートを許容する技術であるため気軽に new Vue をして薄い独自ストアと組み合わせると快適に開発ができる
- SPA の場合は Vuex がないと爆発したが、ちょい入れならそこまでクリティカルな影響はない
- 多くの書き方を許容している利点を活かしてプロジェクト規模にマッチした開発スタイルを取ると秩序を作りやすい

## その他に.....

---

以下の話は時間の都合上盛り込む余裕がなかったので懇親会で

- プラグインやルーティングフックがオレオレ構成の話
  - utils ディレクトリができたり管理システムが崩壊して爆発する

## その他に.....

---

以下の話は時間の都合上盛り込む余裕がなかったので懇親会で

- プラグインやルーティングフックがオレオレ構成の話
- utils ディレクトリができたり管理システムが崩壊して爆発する
- utils の爆発の話はコチラ→ [https://slides.com/potato4d/vuejs\\_meetup7](https://slides.com/potato4d/vuejs_meetup7)

## その他に.....

---

以下の話は時間の都合上盛り込む余裕がなかったので懇親会で

- Vue + TypeScript で幸せになるうとする話
  - Vuex + TypeScript で mapGetters / mapActions が any になって爆発する話

# その他に.....

---

以下の話は時間の都合上盛り込む余裕がなかったので懇親会で

- テスト
  - 初見でやると大体爆発する

その他にも聞いてみたい話は  
お気軽にどうぞ

# 今日話すこと

---

- 「Vue.jsは誰でも簡単に作れるけど小規模向け」は本当か
- 過去のプロジェクトの反省からみる Vue.js の定説の評価
  - Vuex があってもなくても良いは本当か？
  - デザイナーでも触れて一緒に仕事できるは本当か？
  - バックエンド中心のWebアプリケーションへのちょい入れに強いは本当か？
- **Vue.js を新規で使う場合に選択すべきこと**



Vue.js を新規で  
導入する場合には選択すべきこと

# Vue.js を新規で導入する場合に選択すべきこと

---

- SPA 編
- SPA じゃない編

# Vue.js を新規で導入する場合に選択すべきこと

---

- **SPA 編**
- SPA じゃない編

# SPA 編

---

## 最低限やっておくべきこと

- 読み取り専用のアプリケーションでもない限りは Vuex は必ず導入する
- オレオレレイヤーを作り出しそうな部分については極力よその実装を参考にして「書いたやつしかわからない」を避ける
- デザイナーにコンポーネントを作ってもらうときは書くまでのハードルを最大限下げる
- Nuxt.js を使う

消費したくないなら **Nuxt.js**



# NUXT

## Universal Vue.js Applications

[GET STARTED](#)[GITHUB \(V1.4.0\)](#)

SHUTTLEWORTH  
FUNDED

# Nuxt.js とは

---

- Vue based なフルスタックフロントエンドフレームワーク
  - SPA 専用。ちょい入れは不可。
- Vue + Vue Router + Vuex + SSR 環境がデフォルト
- オレオレプラグインレイヤーなども Nuxt.js が吸収
- 規約ベースのアーキテクチャによって規約を守ると誰でも高品質なコードが書ける

SPA 向けの開発セット



# Nuxt.js を使うと嬉しいこと

---

- 規約がオレオレアーキテクチャからプロジェクトを守ってくれる
- 副産物として以下のような便利機能がついてくる
  - ルーティングの自動生成機能（ルーティング管理用のレイヤを消せる）
  - Vuex ストアを扱いやすくするオートローディング（導入が面倒じゃない）
  - 豊富なプラグイン/エコシステム

手っ取り早く  
大体の問題を解消できる

SPA は Nuxt.js で良い

# Vue.js を新規で導入する場合に選択すべきこと

---

- SPA 編
- **SPA じゃない編**

# SPA じゃない編

---

- ちょい入れの場合は大体さっきまで書いた通り
- 素朴な「ストアパターン」で実装する
- どのみち jQuery のコードなどからデータをいじくり回したくなる（体験談）ので Readonly を生み出す Vuex は使わない
- `new Vue ({})` はいくつしても良いのでコンポーネント同士は疎結合で作る

# SPA じゃない編

---

- これから実装する人向けの追加 Tips
- Smarty, Slim, Blade などと一緒に使うならできるだけ一つの HTML 構造をお互いに書き換えるのはやめる
- `<script id="initialState" type="text/x-template"></script>`を用意して、`$('#initialState').innerHTML`で取り出して Vue コンポーネントに流し込む設計で作る

終わりに

Vue.js は可燃性



乙4とってから挑みましよう



はじめて受験される方へ

試験地の選択

試験は各道府県の当センター支部ごとに実施しています(東京は中央試験センター)。このため、試験日、試験会場、願書受付期間等が異なります。詳細は次の試験地の選択から各支部のサイトで確認してください。

試験地の選択 都道府県を選択 確認 地図から選ぶ

各支部からの重要なお知らせ

Table of prefectures: 北海道, 青森, 岩手, 宮城, 秋田, 山形, 福島, 茨城, 栃木, 群馬, 埼玉, 千葉, 東京, 神奈川, 新潟, 富山, 石川, 福井, 山梨, 長野, 静岡, 愛知, 岐阜, 三重, 滋賀, 京都, 大阪, 兵庫, 奈良, 和歌山, 岡山, 広島, 鳥取, 島根, 山口, 徳島, 香川, 愛媛, 高知, 福岡, 熊本, 佐賀, 長崎, 大分, 宮崎, 鹿児島, 沖縄

お知らせ

- 気象警報の発表等に伴う試験延期等の緊急情報の取得方法等について

危険物取扱者試験

- 試験案内
紹介動画
試験日程[全国一覧]
過去に出題された問題
合格発表

消防設備士試験

- 試験案内
紹介動画
試験日程[全国一覧]
過去に出題された問題
合格発表

電子申請はこちらから (インターネット受験申込み)

【注意】 消防設備士試験での各種資格(電気工事士等)による科目免除を希望し、資格証明の書類を必要とする場合は電子申請はできません。(※団体一括申請を除く) 書面申請を行ってください。なお再受験の場合、前回と同じ条件で電子申請できます。
電子申請をした後の取り扱いについて

☆ インターネット受験申込み(電子申請)では受験票は送付(電子メール)

- 電子申請手続きの注意事項(必ず事前にご覧ください)
電子申請(受験申込み)案内
電子申請の際に準備するもの
受験票がダウンロードできない場合

受験票ダウンロードメール(通知)はお知らせメールです。[電子申請はこちらから]の[受験票ダウンロード]で試験日の10日前頃から印刷できます。

本部からの重要なお知らせ

- [2018.6.20] 過去に出題された問題の追加・更新について
過去に出題した問題の一部を掲載しています。(平成30年6月追加・更新)
[2018.4.16] 予防技術検定を受検する予定の皆様へ
平成30年度から予防技術検定の受検手数料が、5,000円から5,700円となります。
受検予定の皆様におかれましては、その旨ご承知いただけますようよろしくお願いいたします。

免状の交付・書き換え等

- 申請手続き一覧
新規免状の交付
写真書換え
本籍等の書換え
再交付
書換・再交付申請書のダウンロード

**Thank you!**