報道各位

2018 年 10 月 2 日
株式会社インプレスR&D
https://nextpublishing.jp/

Vue を開発現場で使いこなすためのノウハウ集

『現場で使える Vue.js tips 集』発行

技術書典シリーズ・10 月の新刊

インプレスグループで電子出版事業を手がける株式会社インプレス R&D は、『現場で使える Vue.js tips 集』(著者:渋田 達也)を発行いたします。

『現場で使えるVue.js tips集』
https://nextpublishing.jp/isbn/9784844398431

著者:渋田 達也
小売希望価格:電子書籍版 1600 円(税別)／印刷書籍版 1800 円(税別)
電子書籍版フォーマット:EPUB3／Kindle Format8
印刷書籍版仕様:B5 判／カラー／本文 114 ページ
ISBN:978-4-8443-9843-1
発行:インプレス R&D

＜＜発行主旨・内容紹介＞＞

【Vue.js を現場で使いこなすための Tips 集！】

本書は、筆者がこれまで携わってきた開発の経験から得た Vue.js の使いこなし Tips 集です。
Form や Vuex などについての即現場でも使えるようなテストコード付きのサンプルコードを多数収録しています。

〈本書の対象読者〉
Vue.js をある程度使いこなしている中級者以上のユーザー
仕事で Vue.js をさらに活用したいプログラマ
(本書は、次世代出版メソッド「NextPublishing」を使用し、出版されています。)

「お問い合わせフォーム」を題材に、開発の現場で役立つノウハウを紹介

第2章　お問い合わせフォームと戦う

Webのお仕事をやっていると各種のフォームを作る場面に出会うのではないでしょうか。筆者も幾度となく経験しています。その都度、作るものに合わせたフォームを作ってきました。

フォームって難しいですよね。入力にエラーがないかバリデーションして、エラーがあればエラーに応じたメッセージを表示して、ものによっては確認画面も存在します。

一応ブラウザにはフォームのバリデーションをサポートするための仕組みが備わっています。Google Developersに「最適なフォームの作成」というページ（https://developers.google.com/web/fundamentals/design-and-ux/input/forms/?hl=ja）があり、このページにはHTMLの属性やConstraint Validation APIを利用したバリデーションの方法などが書かれています。しかし、UI的な要求などを考慮するとこういったデフォルトの機能だけでは達成できないことが多いと思います。そこでこれらの機能を自前で実装する、という決断をすることが多々あるのではないでしょうか。

本章ではWebサービスを作る上で切っても切り離せないフォームについて解説します。今回は前述のGoogle Developerのページに書かれているようなブラウザのAPIを利用したバリデーションやメッセージの表示を利用せずに、自前で実装することにします。これは自前実装という現場が多いであろう、という筆者の肌感覚の独断と偏見による判断です。

というわけで本章では、自前実装のお問い合わせフォームを例に作っていきます。

2.1　フォームは難しい、そしてめんどくさい

冒頭にも書きましたがフォームはやること（要件）が多いです。

・各項目のバリデーション
・バリデーション内容に応じたメッセージの表示
・確認画面

確認画面は実装されないこともありますが、それ以外のふたつはほぼ実装が必須です。これに加えて最近ではUXを考慮し、入力中にバリデーション（リアルタイムバリデーション）を行うフォームが多くなってきました。さらにUXを高めるため、にユーザーの入力が完了するのを待ってエラーメッセージを表示することが推奨されることもあります。このように最低限必要な要件に加え、UXを高めるためなどの理由により追加で要件が発生することもあります。

実装についても考えてみましょう。Vue.jsでは、watchの機能を使えばある程度のリアルタイムバリデーションを行うフォームを簡単に実装できます。例えば次のようなものです。

```
<template>
  <div>
    <input type="text" v-model="name">
    <p>{{ message }}</p>
  </div>
</template>
<script>
export default {
  data() {
    return { name: '', message: '' };
  },
  watch: {
    name(value) {
      if (this.name.length > 10) {
        this.message = '長すぎます';
      } else {
        this.message = '';
      }
    },
  },
};
</script>
```

v-modelで双方向バインディングを行い、watchでその値を監視させています。これで項目ごとのバリデーションとバリデーションの内容に応じたメッセージの表示、加えてリアルタイムバリデーションができています。とても便利です。

これにユーザーの入力が完了したらバリデーションする、という要件を加えましょう。ここでは「ユーザーの入力が完了した」という状態を、フォーカスが外れたときと定義します。

```
<template>
  <div>
    <input type="text" v-model="name" v-on:blur="handleBlurName">
    <p>{{ message }}</p>
  </div>
</template>
<script>
export default {
  data() {
    return {
      name: '',
      nameValidatable: false,
      message: '',
```


ユースケースを基に Vuex の Tips を紹介

第4章　Vuexのtips

本章はVuexについて、筆者のユースケースやどのように使っているかなどを解説します。

4.1　ユースケース

みなさんはVuexをどのように利用しているでしょうか？筆者は次のようなケースで利用しています。

・ページをまたいでデータを共有したいとき
・APIのデータをキャッシュしておきたいとき
・モーダルダイアログやトーストなどグローバルに配置しているコンポーネントの操作

ページをまたいでデータを共有したいときは、フォームの章におけるVuexの使い方と同じなのでここでは触れません。残りのふたつをそれぞれ触れていきたいと思います。

APIサーバーのデータをキャッシュしておきたいとき

これは同じリソースへのリクエストが連続で行わる場合、例えば`/users/1`へのリクエストなどで行ったりすることがあります。「することがあります」と曖昧な書き方にしているのは、キャッシュというものはサービスとしてどの程度行うかを決めるところである、と筆者は考えているからです。そのため、プロジェクトにより方針が変わります。フロントエンドのエンジニアだけで判断せず、APIのエンジニアやサービスの責任者と話して決めていくことが大事です。

モーダルダイアログやトーストなどグローバルに配置しているコンポーネントの操作

こちらはデータというよりもUIの操作としてVuexを使っています。モーダルダイアログやトーストといったコンポーネントは複数箇所で使われるものではないので、それぞれのコンポーネントをページにひとつずつ配置しています。Vuexのactionに表示用のactionを用意しておき、それを呼び出すことで表示を行うようにしたりしてます。これは見てもらう方が早いと思うので、ご興味のある方は下記のサンプルをご覧ください。

https://github.com/mya-ake/vue-tips-samples/tree/master/samples/modal_store

4.2　Vuexの使い方

Vuexの使い方ということで
・モジュールのすすめ
・一部のmapperヘルパーは使わない
・どの程度Vuexに載せるか
・ストアの値を参照するときはgetterを使う
を取り上げようと思います。

モジュールのすすめ

筆者は基本Vuexをモジュールとしてしか使いません。そして`namespaced`は常に`true`としています。その理由は、名前空間をモジュール単位で区切ることができるからです。

単一ストアVuexを利用していた場合はstateやmutation、actionなどそれぞれの名前が被らないようにするのはとても面倒です。プレフィックスを付けるなどの対応をすれば問題ありませんが、いちいちプレフィックスを書く必要が出て来るので書く量も増えてきます。

これがモジュールを区切ることで、モジュールがそのプレフィックスの役目を担ってくれるので、名前を気にするのは同じモジュール内だけで済みます。

ただモジュールモードにも欠点があります。それはmutaionやactionを呼ぶためのキーとなるタイプにモジュール名を付ける必要があるという点です。

例えば次のようなモジュールがあったとします。

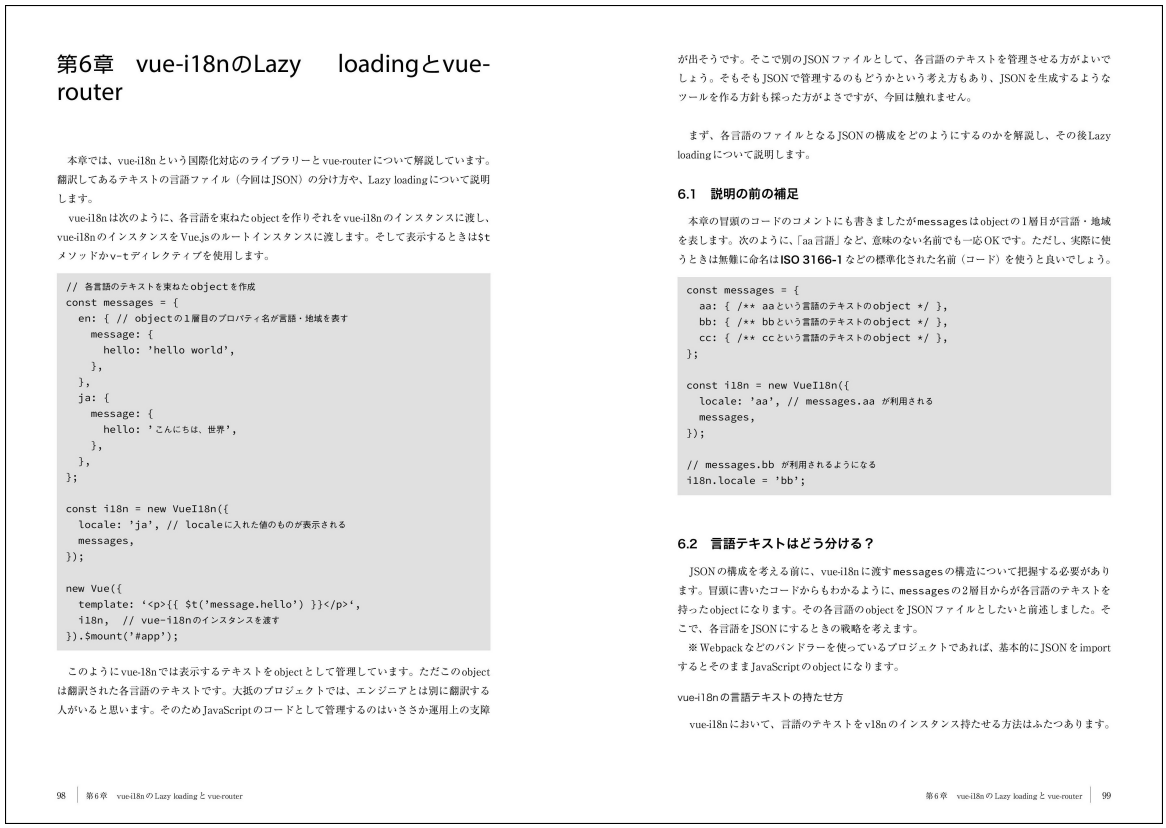
```
const formModule = {
  namespaced: true,
  state: {
    values: null,
  },
  mutations: {
    setValues(state, values) {
      state.values = values;
    },
  },
});

export default new Vuex.Store({
  modules: {
    form: formModule,
  },
});
```

これをコンポーネント内から呼ぶときは、次のようにモジュール名/`mutaion`タイプのようにする必要があります。

```
<script>
export default {
  methods: {
```

国際化対応のライブラリ vue-i18n についても解説



第6章　vue-i18nのLazy　loadingとvue-router

本章では、vue-i18n という国際化対応のライブラリーと vue-router について解説しています。翻訳してあるテキストの言語ファイル（今回はJSON）の分け方や、Lazy loading について説明します。

vue-i18n は次のように、各言語を束ねたobjectを作りそれをvue-i18nのインスタンスに渡し、vue-i18nのインスタンスをVue.jsのルートインスタンスに渡します。そして表示するときは$tメソッドかv-tディレクティブを使用します。

```
// 各言語のテキストを束ねたobjectを作成
const messages = {
  en: { // objectの1層目のプロパティ名が言語・地域を表す
    message: {
      hello: 'hello world',
    },
  },
  ja: {
    message: {
      hello: 'こんにちは、世界',
    },
  },
};

const i18n = new VueI18n({
  locale: 'ja', // localeに入れた値のものが表示される
  messages,
});

new Vue({
  template: `<p>{{ $t('message.hello') }}</p>`,
  i18n,  // vue-i18nのインスタンスを渡す
}).$mount('#app');
```

このようにvue-i18nでは表示するテキストをobjectとして管理しています。ただこのobjectは翻訳された各言語のテキストです。大抵のプロジェクトでは、エンジニアとは別に翻訳する人がいると思います。そのためJavaScriptのコードとして管理するのはいささか運用上の支障が出そうです。そこで別のJSONファイルとして、各言語のテキストを管理させる方がよいでしょう。そもそもJSONで管理するのもどうかという考え方もあり、JSONを生成するようなツールを作る方針も採った方がよさですが、今回は触れません。

まず、各言語のファイルとなるJSONの構成をどのようにするのかを解説し、その後Lazy loadingについて説明します。

6.1　説明の前の補足

本章の冒頭のコードのコメントにも書きましたがmessagesはobjectの1層目が言語・地域を表します。次のように、「aa言語」など、意味のない名前でも一応OKです。ただし、実際に使うときは無難に命名はISO 3166-1などの標準化された名前（コード）を使うと良いでしょう。

```
const messages = {
  aa: { /** aaという言語のテキストのobject */ },
  bb: { /** bbという言語のテキストのobject */ },
  cc: { /** ccという言語のテキストのobject */ },
};

const i18n = new VueI18n({
  locale: 'aa', // messages.aa が利用される
  messages,
});

// messages.bb が利用されるようになる
i18n.locale = 'bb';
```

6.2　言語テキストはどう分ける？

JSONの構成を考える前に、vue-i18nに渡すmessagesの構造について把握する必要があります。冒頭に書いたコードからもわかるように、messagesの2層目からが各言語のテキストを持ったobjectになります。その各言語のobjectをJSONファイルとしたいと前述しました。そこで、各言語をJSONにするときの戦略を考えます。

※Webpackなどのバンドラーを使っているプロジェクトであれば、基本的にJSONをimportするとそのままJavaScriptのobjectになります。

vue-i18nの言語テキストの持たせ方

vue-i18nにおいて、言語のテキストをvi18nのインスタンス持たせる方法はふたつあります。

＜＜目次＞＞

第 4 章　Vuex の tips
4.1　ユースケース
4.2　Vuex の使い方
4.3　まとめ
第 5 章　vue-test-utils でなにをテストするか
5.1　最低限のコンポーネントテスト
5.2　テストの方針
5.3　テストコードの実例解説
5.4　まとめ
第 6 章　vue-i18n の Lazy loading と vue-router
6.1　説明の前の補足
6.2　言語テキストはどう分ける？
6.3　言語テキストの Lazy loading
6.4　まとめと課題

＜＜著者紹介＞＞

渋田　達也(しぶた　たつや)
福岡で活動している web 系のエンジニア。サーバーレスなどのバックエンドのこともしたりするが最近はフロントエンドがメインとなっている。

＜＜販売ストア＞＞

電子書籍:
Amazon Kindle ストア、楽天 kobo イーブックストア、Apple Books、紀伊國屋書店 Kinoppy、Google Play Store、honto 電子書籍ストア、Sony Reader Store、BookLive!、BOOK☆WALKER
印刷書籍:
Amazon.co.jp、三省堂書店オンデマンド、honto ネットストア、楽天ブックス
※　各ストアでの販売は準備が整いしだい開始されます。
※　全国の一般書店からもご注文いただけます。

【株式会社インプレス R&D】 https://nextpublishing.jp/
株式会社インプレス R&D（本社：東京都千代田区、代表取締役社長：井芹昌信）は、デジタルファーストの次世代型電子出版プラットフォーム「NextPublishing」を運営する企業です。また自らも、NextPublishing を使った「インターネット白書」の出版など IT 関連メディア事業を展開しています。
※NextPublishing は、インプレス R&D が開発した電子出版プラットフォーム（またはメソッド）の名称です。電子書籍と印刷書籍の同時制作、プリント・オンデマンド（POD）による品切れ解消などの伝統的出版の課題を解決しています。これにより、伝統的出版では経済的に困難な多品種少部数の出版を可能にし、優秀な個人や組織が持つ多様な知の流通を目指しています。

【インプレスグループ】 https://www.impressholdings.com/
株式会社インプレスホールディングス（本社：東京都千代田区、代表取締役：唐島夏生、証券コード：東証 1 部 9479）を持株会社とするメディアグループ。「IT」「音楽」「デザイン」「山岳・自然」「旅・鉄道」「学術・理工学」を主要テーマに専門性の高いメデイア&サービスおよびソリューション事業を展開しています。さらに、コンテンツビジネスのプラットフォーム開発・運営も手がけています。

【お問い合わせ先】

株式会社インプレス R&D NextPublishing センター
TEL 03-6837-4820
電子メール: np-info@impress.co.jp