Rails 5.1 + Webpacker + Vue.js 入門
2017 年 5 月 22 日の技術セミナーのまとめ

2017 年 5 月 23 日

株式会社オイアクス　黒田努

前フリ

- 前月末に Ruby on Rails 5.1 が出た。正式に webpack を取り込んだ点が目玉。
- これから Rails と JavaScript の組み合わせ方が大きく変化していく。
- しかし、従来のアセットパイプライン（sprockets）と webpack は共存できる。
- Rails 5.1 からはデフォルトで jQuery が組み込まれなくなったので、使いたい場合は自分で `jquery-rails` を Gemfile に追加する必要がある。
- 本日のライブコーディングで使用する Rails アプリケーションのソースコードは
  `https://github.com/oiax/tamachi_vue`
  から取得できる。初期状態には `ver0` というタグが打ってある。ここからスタート。
- ユーザーの登録・編集フォームで「得意なプログラミング言語」の選択状態により other_language を入力するためのテキストフィールドがトグルされる UI が jQuery で実現されている。
- これを Vue.js で書き直す、というのが今回のライブコーディングの目標。

初期状態（ver0）のコードの説明

- Rails 5.1 で新しいヘルパーメソッド `form_with` が導入された。従来の `form_tag` と `form_for` を統合するもの。`local: true` オプションを付けないと xhr（Ajax）でリクエストが飛ぶので要注意。Vue.js で使うときは、`local: true` オプションを付ける。
- テキストフィールドをトグルする UI 関連のコードは `app/assets/javascripts/users.js` に書いてある。
- ラジオボタンに対して click イベントを結びつけ、イベント発生時にラジオボタンの選択状態を調べてテキストフィールドの表示・非表示を切り替えている。

Step 1: webpack と Vue.js の導入

- webpack とは、簡単に言えば JavaScript のソースコードの中で `import` 文を使えるようにするための仕組み。
- `import` 文が使えると、JavaScript のソースコードを複数のモジュールに分割できるようになる。複雑なフロントエンド開発を行うときには便利だが、すべてのブラウザが `import` 文に対応していないので、それらのモジュールをまとめるツールとして webpack が生まれた。
- Rails と webpack をつなぐ架け橋が Webpacker という名前の Gem パッケージ。
- 前提条件として Yarn が必要。Yarn は npm の代替品。
- `Gemfile` に `webpacker` を追加して `bundle install`。
- `rails webpacker:install` して `rails webpacker:install:vue` するといろんなファイルができて、必要なパッケージ群が node_modules ディレクトリの下に配置される。
- `app/javascript/packs` ディレクトリに「エントリー」と呼ばれる JavaScript コードを置く。これは、HTML 文書から直接的に読み込む対象となる JavaScript コードを指す webpack 用語。モジュールは別のディレクトリに置く。

Step 2: jQuery の除去

- `Gemfile` から `jquery-rails` を削除して、`bundle install`
- `app/assets/javascripts/application.js` から `//= require jquery` を除去
- `app/assets/javascrpts/users.js` を削除
- jQuery と Vue.js は共存できるので残しておいても構わない。

Step 3: app/javascript/packs/users/form.js を作る

```
import Vue from "vue/dist/vue.esm"

document.addEventListener("DOMContentLoaded", () => {
  new Vue({
    el: "#user-form"
  })
})
```

- vue/dist/vue.esm は実際には node_modules/vue/dist/vue.esm.js を指す。これを Vue という名前（定数）でインポートする。
- el オプションの値は CSS セレクタ。user-form という id を持つ HTML 要素に Vue コンポーネントをマウントする。
- app/views/users/new.html.erb に javascript_pack_tag を追加。
- app/views/users/edit.html.erb に javascript_pack_tag を追加。
- bin/webpack-dev-server を起動。
- rails s を起動。
- ブラウザを開いてユーザー新規登録フォームをリロード。特に変わった感じはしないけれど、v-show ディレクティブを使ってみると、確かに Vue.js が機能していることがわかる。

Step 4: v-model ディレクティブ等の追加

```
<%= f.text_field :name, class: "form-control", style: "width: 300px",
 "v-model" => "user.name" %>
…
<%= f.radio_button :language, "ruby", "v-model" => "user.language" %> Ruby
…
<%= f.radio_button :language, "php", "v-model" => "user.language" %> PHP
…
<%= f.radio_button :language, "other", "v-model" => "user.language" %> その他
…
<div class="form-group" v-show="user.language === 'other'">
…
```

```
import Vue from "vue/dist/vue.esm"

document.addEventListener("DOMContentLoaded", () => {
  const app = new Vue({
    el: "#user-form",
    data: {
      user: {
        name: "",
        language: undefined,
        other_language: ""
      }
    }
  })
})
```

- app/views/users/_fields.html.erb に v-model ディレクティブおよび v-show ディレクティブを追加。
- これで、新規登録フォームでテキストフィールドをトグルできるようになる。

Step 5: fom_with に local:true オプションを設定

- form_with はデフォルトで xhr によるリクエストを行う。
- Vue.js とは相性が悪いので local: true オプションを追加する。

Step 6: Turobolinks の除去

- Turbolinks は Vue.js と相性が悪いので除去。
- Gemfile から turbolinks を削除して bundle install
- app/assets/javascripts/application.js から //= require turbolinks を削除
- app/views/layouts/application.html.erb から Turbolinks 関連の記述を削除

Step 7: フォームの初期データを DOM ツリーから取る

```
import Vue from "vue/dist/vue.esm"

document.addEventListener("DOMContentLoaded", () => {
  const language = document
    .querySelector("[v-model='user.language']:checked")

  new Vue({
    el: "#user-form",
    data: function () {
      return {
        user: {
          name: document.querySelector("[v-model='user.name']").value,
          language: language ? language.value : undefined,
          other_language: document
            .querySelector("[v-model='user.other_language']").value
        }
      }
    }
  })
})
```

Step 8: v-cloak で隠す

```
<div class="panel panel-default">
  <div class="panel-body">
    <div id="user-form" v-cloak>
      <%= form_with model: @user, local: true do |f| %>
        <%= render "fields", f: f %>
        <%= f.submit "登録", class: "btn btn-primary" %>
      <% end %>
    </div>
  </div>
</div>
```

```
[v-cloak] {
  display: none;
}
```

- v-cloak ディレクティブを付けておくと、Vue.js による再描画が終わるまでテンプレートを隠しておける。
- **（セミナーでは不言及）**ただし、あらかじめ CSS を設定しておく必要がある。

Step 9: `vue-data-scooper` の導入

```
import Vue from "vue/dist/vue.esm"
import VueDataScooper from "vue-data-scooper"

Vue.use(VueDataScooper)

document.addEventListener("DOMContentLoaded", () => {
  new Vue({
    el: "#user-form"
  })
})
```

- ここから先は、私の「独自研究」。
- 私が作った Vue.js 向けのプラグイン `vue-data-scooper` を使うと、Step 7「フォームの初期データを DOM ツリーから取る」でやったことが自動化される。
- `yarn add vue-data-sooper` を実行。

Step 10: `vue-rails-form-builder` の導入

- 私が作った Gem パッケージ `vue-rails-form-builder` を使うと、Step 4 で行った `v-model` ディレクティブの追加手順を省略できる。
- `form_with` を `vue_form_with` で置き換え、`v-model` ディレクティブをすべて削除する。

Step 11: `form_with` のグローバル設定を変更

```
Rails.application.config.action_view.form_with_generates_remote_forms = false
```

- デフォルトで `form_with` は xhr によるリクエストを行うが、Rails のグローバル設定で変更できる。
- 上記のコードを `config/initializers/action_view.rb` に貼り付ければ、`form_with` に local: true をいちいち付けなくてもよくなる。