WordPress REST API と Vue.js を使ったフロントエンド開発

> 世界中で利用される CMS である WordPress と、リアクティブ系 JavaScript ライブラリのひとつである Vue.js を使ったウェブ・フロントエンド開発手法をサンプルサイトを元に具体的なソースコードとともに解説します。

ひろましゃ@WordBench札幌

自己紹介

名前	田中 広将（ひろましゃ）
コミュニティー	WordBench札幌
WordPress暦	10数年
OSC 北海道暦	7年
職業	SIer所属のインフラエンジニア
属性	パソコンとか詳しい人

“ よろしくお願いいたします。 ”

本日のお話

話題の流れ

1. WordPress と WordPress REST API の概要
2. Vue.js の概要
3. WP REST API + Vue.js の適用例紹介（サンプルサイトを用いて解説）

> 本日のスライドとソースコードは、OSC 北海道 2017 サイトでリンクを公開しますので、ゆっくりご覧になっていってください。

サンプルサイト

本日のために製作したサンプルサイトを元に以下の解説をします。

- WordPress REST API
  - REST API カスタムエンドポイントの作成方法。
  - カスタム投稿の内容を JSON で出力する。

サンプルサイト

- Vue.js
  - WordPress から AJAX で取得した JSON を「データーバインディング」機能を用いて HTML に表示する。
  - 操作（イベント）を契機に CSS3 アニメーションを発生させる。
  - イベントに対してデーターをさまざまな角度から画面に反映させる。

目標

- これらのテクニックは個々に利用可能ですので、今回解説するサンプルサイトと同様の構成のサイトだけでなくさまざまなサイトに応用することができると思います。
- WordPress + Vue.js の勘所をつかんでいって頂ければと思います。

> まずはサンプルサイトを実際に操作して完成形を紹介します。

WordPress と WordPress REST API の概要

WordPress とは

- WordPress は世界的に使われているコンテンツマネージメントシステムです。
  - 世界の人気サイト、一説によれば 20% 以上で使われており CMS ではトップシェアと言われています。
  - 10数年近くの歴史があるオープンソースソフトウェアで、コミニュティーベースで開発が進められ、自由に使うことが出来ます。

WordPress の大まかな機能

WordPress は HTML を出力する

- WordPress は「テーマ」と呼ばれるサイトの HTML テンプレート（雛形）に対して、表示するコンテンツを管理画面から操作して、ソースコードの修正なしにウェブサイトをコントロールするソフトウェアです。
- テーマは HTML/CSS と、WordPress の「テンプレートタグ」によって構成されるテンプレートファイル群が実態です。

WordPress は HTML を出力する

- テーマはサイト制作者が自由にコーディングすることができるので、どのようなデザインのウェブサイトでも WordPress を使って製作することができます。

テンプレートファイルの例

```
<div class="jumbotron">
    <div class="row">
        <div class="col-md-4">
            <a href="#" class="thumbnail pull-left">
                <img src="<?php the_post_thumbnail_url() ?>
            </a>
        </div>
        <div class="col-md-8">
            <h1><?php the_title(); ?></h1>
            <div class="ld"><?php the_content(); ?></div>
            <p>
                <a class="btn btn-primary btn-lg">
                    カート入れ
                </a>
            </p>
        </div>
    </div>
</div>
```

WordPress は HTML を出力する

“ WordPress の管理画面を実際に操作してみます。 ”

WordPress REST API

- WordPress REST API は、WordPress 4.4 から追加された、WP の歴史から見ると比較的新しい機能です。
- この機能により、WordPress はテーマによる HTML の出力に加え、JSON 形式でサイトのコンテンツを返すことができるようになりました。

JSON 形式とは

- 人が見ても何も嬉しくない出力ですが、JSON（JavaScript Object Notation）は JavaScript が解釈しやすい形式です。

```
{
    "id": 34,
    "title": "手づくりバゲット",
    "excerpt": "パンってパンの形をしてるだけで幸せよね。",
    "permalink": "http://example.com/osc2017do/item/pan1",
    "thumbnail": "bread_01-242x242.png",
    "price": 500
}
```

JSON 形式とは

- ブラウザ上で JavaScript のプログラムを動作させ、WordPress から JSON でコンテンツを取得し、HTML に展開することで人に見える形にします。
- この手法をとると、ブラウザの再描画なしにページを遷移させるなどの、リアルタイムで動きのあるモダンなウェブサイトの動作をさせることができます。
- 本日はこの JavaScript プログラミングに Vue.js というライブラリを活用して、WordPress から取得した JSON コンテンツを処理しウェブサイトを製作していきます。

エンドポイント

- 必要なコンテンツを JSON で取得するための URL のことをエンドポイントと呼びます。
- WordPress にエンドポイントを追加すると wp-json/ 配下に指定したパスで URL が生成されます。(追加の方法は後述します)

```
http://example.com/wp-json/osc2017do/v1/items/1
```

> 実際にアクセスして JSON を表示する操作をしてみます。

Vue.js の概要

Vue.js とは

- Vue.js は近年活用が進んでいるリアクティブ系の JavaScript ライブラリのひとつで、データーに対する画面（HTML）の描画を得意としています。
- 従来の HTML を直接操作するプログラミング手法と異なり、プログラムで持つデーター（変数）と HTML を、HTML への簡単な追記で「バインディング」することにより、驚くほど簡単に処理を画面に反映することができます。

Vue.js とは

- HTML とのバインディング（拘束）を行ってしまえば、JS から変数の値を更新するだけで、画面が自動的にに更新されます。

html

```
<h3>
  <a v-bind:href="item.permalink">{{ item.title }}</a>
</h3>
```

javascript

```
this.item.permalink = "http://example.com/items/1";
this.item.title = "タイトル";
```

Vue.js とは

- Vue.js はプログラムもコンパクトでドキュメントも簡潔にまとめられており 1日程度あればその全容をつかむことができます。
- 近年 Vue.js の活用は進み、その他のリアクティブ系 JavaScript ライブラリである React.js や Angular と並ぶ地位を確立しています。

“ Vue.js のサイトを紹介します。 ”

なぜ WordPress + Vue.js なのか

- Vue.js は単純に HTML の script タグで vue.js ファイルを読ませれば良いだけなので導入が簡単です。

`html`

```
<script type='text/javascript' src="/js/vue.min.js">
</script>
```

なぜ WordPress + Vue.js なのか

- Vue.js は HTML に対して簡単な追記を行う形でプログラミングを行っていくため、WordPress のテンプレートファイルを壊さずそのまま使えます。

`index.php`

```
<?php get_header(); // WordPress テンプレートタグ ?>
<h3>
  <a v-bind:href="item.permalink">{{ item.title }}</a>
</h3>
```

なぜ WordPress + Vue.js なのか

- 他のライブラリでは、ライブラリを導入するためにタスクランナーの導入といった技術の追加が必要になるケースが多く、やりたいことに対するコストと釣り合わないケースがあります。（簡単な処理に対して大掛かりになることがある）
- また、WordPress のテンプレートファイルから処理したい部分を別ファイルに切り出す必要があり、無理が効かない場合があります。（テンプレートタグを持つ他の CMS でも同様です）

WP REST API + Vue.js の適用例

WordPress REST API を用いて商品をサイトに表示

“ サンプルサイトの商品表示部分を見てみます。 ”

WordPress REST API を用いて商品をサイトに表示

- Vue.js は変数の内容を HTML バインディングの機能で簡単に画面に表示できます。
- このことを利用して、REST API で取得した JSON をそのまま変数に格納し、それを HTML にバインディングしていきます。
- WordPress ではこの REST API リクエストに対してデーターを返す処理を追加します。

商品を格納する WordPress カスタム投稿タイプの設定

- 投稿タイプとは WordPress におけるコンテンツの入れ物です。
- コンテンツの形に応じて、投稿タイプを増やすことができます。

“ サンプルサイトの商品登録画面を見てみます。 ”

商品を格納する WordPress カスタム投稿タイプの設定

- 商品を入れる投稿タイプとしてアイテム投稿タイプを追加します。

functions.php

```php
/**
 * WordPress 初期設定.
 */
add_action('init', function() {
    /**
     * 商品を格納する投稿タイプ(アイテム)定義
     */
    register_post_type('item', array(
        'labels' => array('name' => 'アイテム'),
        'public' => true,
        'has_archive' => true,
        // タイトルと本文と抜粋とアイキャッチを使う
        'supports' => array(
            'title', 'editor', 'excerpt', 'thumbnail')
    ));
});
```

JSON コンテンツ取得用のエンドポイントの追加

- 指定ページの商品を4件ずつ取得するカスタムエンドポイントを WordPress に追加します。

```
http://example.com/wp-json/osc2017do/v1/items/1
```

> サンプルサイトのカスタムエンドポイントを見てみます。

functions.php

```php
add_action('rest_api_init', function() {
    // /wp-json/osc2017do/v1/items/[paged:ページ数]
    register_rest_route(
            'osc2017do/v1',
            '/items/(?P<paged>\d+)', array(
        'method' => 'GET',
        'args' => array(
            'paged' => array(
                'default' => 1
                // ページ数は数値であること
                ,'sanitize_callback' => 'absint'
            )
        ),
        'callback' => function($r) {
            // ここで返却したい JSON を作成します
        }
    ));
});
```

- callback の中で、通常の WordPress のテンプレートファイル/テンプレートタグと同様の記述で WordPress から投稿を取得し、JSON 形式にして return します。

```
'callback' => function($r) {
    $result = array();
    // WordPress から商品一覧情報取得
    $query = new WP_Query(array(
        // カスタム投稿 item から取得
        'post_type' => 'item',
        // 1ページは4件
        'posts_per_page' => 4,
        // エンドポイントからで指定されたページ数
        'paged' => $r->get_param('paged')
    ));
    // 続く..
```

```php
    //..続き
    // 取得した投稿を1件づつ JSON に格納
    $items = array();
    while($query->have_posts()) {
        $query->the_post();
        $item = array();
        $item['id'] = get_the_ID();
        $item['title'] = get_the_title();
        $item['excerpt'] = get_the_excerpt();
        $item['permalink'] = get_the_permalink();
        $item['thumbnail'] =
            get_the_post_thumbnail_url(
                get_the_ID(), 'item');
        $item['price'] = SCF::get('price');
        $items[] = $item;
    }
    $result['items'] = $items;
    // PHP の配列を return すると JSON になります。
    return $result;
}
```

JSON コンテンツ取得用のエンドポイントの追加

- 商品コンテンツを返す REST API のカスタムエンドポイントができました。

```
http://example.com/wp-json/osc2017do/v1/items/1
```

> ここまでの実装が WordPress のサーバー側の世界です。

Vue.js のセットアップ

- WordPress のテーマディレクトリに利用する .js ファイルを格納します。
- WordPress テンプレートファイルで Vue.js を使うためにライブラリを読み込ませます。
- AJAX を使うためのライブラリ（axios）を読み込ませます。
- Vue.js を用いた自分のプログラムを読み込ませます。（cart.js にしました）
- REST API のエンドポイントを JavaScript から読める形で input hidden に定義します。

Vue.js のセットアップ

- ここでは footer.php に追記しています。

`footer.php`

```
<!-- Vue.js 読み込み -->
<script type='text/javascript'
  src="<?php echo get_theme_file_uri('/js/vue.min.js'); ?>'
<!-- axios 読み込み(AJAX用) -->
<script type='text/javascript'
  src="<?php echo get_theme_file_uri('/js/axios.min.js'); ?
<!-- 自分の Vue.js を使った JavaScript(今日はこれをかきます) -->
<script type='text/javascript'
  src="<?php echo get_theme_file_uri('/js/cart.js'); ?>"></
<!-- REST API のエンドポイントを JavaScript から読める形で定義 -->
<input type="hidden" id="resturl"
  value="<?php echo get_rest_url(null, '/osc2017do/v1') ?>'
```

Vue.js のセットアップ

- Vue.js に処理させるエリアを決める id をテンプレートファイルの HTML に付与します。（#cart にしました）

`header.php`

```
<div class="container" id="cart">
```

“ container はデモサイトのコンテンツを囲う div になっています。 ”

Vue.js から REST API を呼び出し

- 新規作成した cart.js に Vue.js を使った JavaScript を記述し、REST API で取得した JSON を処理して画面に反映していきます。
- Vue.js のライブラリ仕様より決められた位置に処理をかいていきます。

cart.js

```
new Vue({
    // el には、Vue.js 処理対象となる html の範囲を id で設定します。
    el: '#cart',
    // data には、処理するデーターを定義
    // ここにおいた変数は HTML とバインディング(展開)できます。
    data: {
        // 商品一覧を格納する変数
        items: []
    },
    // Vue.js 初期化時に必ず呼ばれます。
    created: function () {
        // 初期表示(1ページ目)
        this.query(1);
    },
```

```
        // 処理
    methods: {
        // 商品を REST API で取得する
        query: function(paged) {
            // エンドポイントに対して AJAX で JSON をリクエスト
            // wp-json/osc2017do/v1/items/[paged:ページ数]
            axios.get(jQuery("#resturl").val() +
                "/items/" + paged).then((response) => {
                // data に定義した変数に取得した JSON を
                // そのまま格納します。
                this.items = response.data.items;
            })
        }
    }
});
```

- 取得した JSON データーを items 変数にそのまま格納するところに注目してください。
- data 内に置かれた変数は HTML にバインディングすることができます。

```
// data には、処理するデーターを定義
// ここにおいた変数は HTML とバインディングできます。
data: {
    // WordPress REST API のエンドポイント
    resturl: '',
    // 商品一覧
    items: []
},
```

HTML に Vue.js のデータバインディング記法を追記

- 商品一覧を出力したいテンプレートファイルの HTML に Vue 形式のデータバインディング記法を追記します。
- `v-for` と `v-bind` と `{{ 変数名 }}` の部分が Vue.js のバインディング記法です。
- `items` は cart.js で定義した変数名と一致させバインドします。

page-items.php

```
<div v-for="item in items">
    <div class="thumbnail">
        <img v-bind:src="item.thumbnail" />
        <div class="caption">
            <h3><a v-bind:href="item.permalink">
                {{ item.title }}</a></h3>
            <div class="excerpt">{{ item.excerpt }}</div>
            <div class="price">{{ item.price }} 円</div>
            <div class="register">
                <a href="#">カート入れ</a>
            </div>
        </div>
    </div>
</div>
```

- Vue.js の data で定義された配列変数を回します。

```
<div v-for="item in items">
```

- 属性への値の出力

```
<img v-bind:src="item.thumbnail" />
```

- 要素への値の出力

```
<div class="excerpt">{{ item.excerpt }}</div>
```

> “ このような記法で HTML と Vue.js をバインドした後は、Vue.js の data に定義してある変数を修正した瞬間に HTML に自動的に反映するようになります。 ”

WordPress REST API を用いて商品をサイトに表示

> サイトを表示して確認してみます。
>
> 初期表示を行っている `this.query(1)` の 1 を 2 に変えることで次のページが表示できます。

カートサイドバーの開閉

- カートサイドバーの開閉は Vue.js と CSS の組み合わせで実装されています。
- data に開閉状態を保持する変数を追加し、HTML 属性の class とバインディングします。

“ サイドバーの動きを確認してみます。 ”

開閉用のデータバインディングと CSS の設定

- Vue.js のデーターとしてカートサイドバーの開閉情報を格納する変数（`sidebarShow`）を追加します。

`cart.js`

```
new Vue({
    // 省略..
    data: {
        // WordPress REST API のエンドポイント
        resturl: '',
        // サイドバーの表示状態（最初は閉じている）
        sidebarShow: false,
        // 商品一覧
        items: []
    },
    // 省略..
```

開閉用のデータバインディングと CSS の設定

- HTML に sidebarShow 変数をバインドして、開閉状態を元にサイドバーに CSS のクラス（`sidebar-open`）を付与します。
- `v-bind:class` は指定した変数が true のときに、指定した CSS の class（`sidebar-open`）が付与されます。

`sidebar.php`

```
<div class="sidebar"
    v-bind:class="{ 'sidebar-open': sidebarShow }">
    <h3>カートの中身</h3>
</div>
```

開閉用のデータバインディングと CSS の設定

- サイドバーに指定した CSS (`sidebar`)は初期状態で `right: -340px;` とすることで画面上見えない状態としておき、`sidebarShow` が true になり `sidebar-open` が付与されたときに `right: 0px;` とすることで表示させます。

`style.css`

```
.sidebar {
    right: -340px;
}

.sidebar-open {
    right: 0px;
}
```

開閉用のデータバインディングと CSS の設定

- このとき CSS に transition プロパティーを指定することで postion の変化を検知しブラウザがアニメーション効果をつけてくれます。

`style.css`

```
.sidebar {
    right: -340px;
    transition: left .5s, right .5s;
}
```

ユーザーのクリック（イベント）で sidebarShow を true/false にする

- Vue.js のイベント記法である `v-on:click` を HTML の動作させたい部分に配置します。
- click 時に呼び出す `toggleSidebar` メソッドを cart.js に追加します。

`header.php`

```
<a href="#" v-on:click="toggleSidebar">カート</a>
```

- 閉じる処理を行っている画面を暗くしているエレメントに対しても同様に `v-on:click` を付与します。

`footer.php`

```
<div class="sidebar-overlay"
    v-bind:class="{ 'sidebar-overlay-open': sidebarShow }"
    v-on:click="toggleSidebar">
</div>
```

- `toggleSidebar` メソッドで呼ばれるたびに `sidebarShow` が true/false を交互に設定される処理をかきます。

`cart.js`

```
new Vue({
    // 省略
    methods: {
        // サイドバーの表示をトグルする.
        toggleSidebar: function() {
            // true だったら false に
            // false だったら true に
            this.sidebarShow = !this.sidebarShow;
        }
    }
    // 省略
}
```

カートサイドバーの開閉

> サイドバーの開閉ができました。もう一度動作をみてみます。

- Vue.js のコーディングは以下の順番で処理を追加すると理解しやすいです。
  1. 処理するデーターとなる変数の追加
  2. HTML にバインドやイベント（`v-bind`, `v-on:click`）を追加
  3. データを処理するメソッドを追加

カートサイドバーに商品を入れる処理

- カートサイドバーに商品を入れる処理も、カート内のデータを処理する変数の追加と `v-on:click` のイベントで実装されています。
- カート内に入れた商品はサイドバー上に HTML バインディングで表示します。

> 商品を入れる処理の動きを確認してみます。

処理するデーターとなる変数の追加

- カートに入れた商品を格納する変数（`baggage`）を追加します。

```
new Vue({
    // 省略..
    data: {
        // 省略..
        // サイドバーの表示状態（最初は閉じている）
        sidebarShow: false,
        // 商品一覧
        items: [],
        // カートに入れた商品
        baggage: []
    },
    // 省略..
```

HTML イベント（v-on:click）を追加

- `v-on:click` をボタンに追加し、さきほど追加したサイドバーを開閉する toggleSidebar メソッドを呼び出します。
- どの商品が押されたかを識別するため WordPrsss 上のユニークな id となっている `item.id` をメソッドの引数として追加します。

page-items.php

```
<div v-for="item in items">
    <div class="thumbnail">
        <img v-bind:src="item.thumbnail" />
        <div class="caption">
            <h3><a v-bind:href="item.permalink">{{ item.tit
            <div class="excerpt">{{ item.excerpt }}</div>
            <div class="price">{{ item.price }} 円</div>
            <div class="register">
                <a href="#"
                    v-on:click="toggleSidebar(item.id)">
                    カート入れ
                </a>
            </div>
        </div>
    </div>
</div>
```

選択された商品を変数に格納

- `toggleSidebar` メソッドのサイドバー開閉に処理を追加して、変数 `baggage` に引数で選択された商品の情報ひとつを、JSON で取得した商品情報 `items` からそのままコピーして追加します。

cart.js

```
new Vue({
    methods: {
        toggleSidebar: function(id) {
            this.sidebarShow = !this.sidebarShow;
            // id 指定なしの場合は何もしない。
            if(!Number.isInteger(id)) return;
            // id を元に JSON で取得した items をから検索して
            // 商品を特定する。
            var search = this.items.filter(
                    (item, index) => {
                if(item.id == id) return true;
            });
            // 見つけた商品を baggage に追加する。
            this.baggage.push(search[0]);
        }
    }
}
```

- 変数 baggage に格納されたデーターをサイドバーのHTML バインドします。

sidebar.php

```
<ul class="list-group">
    <li v-for="item in baggage">
        <a v-bind:href="item.permalink">
            {{ item.title }}
        </a>
    </li>
</ul>
```

“ この処理の追加により「カート入れ」ボタン押下後にサイドバーに商品が追加されるようになります。
次にカートから商品を削除する処理を追加してみます。 ”

カートから商品を削除する処理を追加

- カートから削除する[x]を追加します。Vue.js 側に `remove` メソッドを実装し、`baggage` 変数から選択された商品を削除します。

`sidebar.php`

```
<ul class="list-group">
    <li v-for="item in baggage">
        <a v-bind:href="item.permalink">
            {{ item.title }}
        </a>
        <a href="#" v-on:click="remove(item.id)">x</a>
    </li>
</ul>
```

cart.js

```
new Vue({
    methods: {
        remove: function(id) {
            // baggage から id に該当するデータを削除(splice)
            for(var i = 0; i < this.baggage.length; i++) {
                if(this.baggage[i].id == id) {
                    // データを削除すれば自動的に画面に反映される!
                    this.baggage.splice(i, 1);
                    break;
                }
            }
        }
    }
}
```

カートから商品を削除する処理を追加

> このように、一度 Vue.js のデーター定義と HTML をバインディングしてしまえば、変数の操作だけで自動的に HTML に反映されます。

カートに入っている商品数を表示する

- `v-bind` や `{{ }}` による HTML のバインディングには JavaScript の式を書くことができます。
- カートに入っている商品（baggage）を表示するのは次のように簡単です。

`header.php`

```
<a href="#" v-on:click="toggleSidebar">
    カート
    <span class="badge">{{ baggage.length }}</span>
</a>
```

カートに入っている商品数を表示する

> カートから商品を出し入れして商品数が更新されるのを確認してみます。
>
> たったこれだけの記述で、カートに対する商品（`baggage`）の追加・削除処理の全てのパターンで数が自動的に更新されます。
>
> たとえば、カートに入っている全ての商品を消す処理をかけば、何もしなくても 0 が表示されます。（Vue.jsの威力!）

カートに入っている商品の「カート入れ」ボタンを非活性に

- カート入れボタンに `v-bind:class="contains(item.id)"` を追加し、メソッド内でカート内に商品があるかを判定して CSS の disable クラスを返します。

page-items.php

```
<div v-for="item in items">
    <div class="thumbnail">
        <img v-bind:src="item.thumbnail" />
        <div class="caption">
            <h3><a v-bind:href="item.permalink">{{ item.tit
            <div class="excerpt">{{ item.excerpt }}</div>
            <div class="price">{{ item.price }} 円</div>
            <div class="register">
                <a href="#"
                    v-on:click="toggleSidebar(item.id)"
                    v-bind:class="contains(item.id)">
                    カート入れ
                </a>
            </div>
        </div>
    </div>
</div>
```

cart.js

```
new Vue({
    methods: {
        contains: function(id) {
            // baggage から商品を検索
            var search = this.baggage.filter((item, index)
                if(item.id == id) return true;
            });
            // 結果がなければ CSS の disable クラスを出力しない
            if(search.length == 0) {
                return { disabled: false };
            }
            // あれば CSS の disable クラスを出力
            return { disabled: true };
        }
    }
}
```

カートに入っている商品の「カート入れ」ボタンを非活性に

> 先程のカートに入った数の表示と同様に、`baggage` を検索して活性・非活性を決定することで、どのような UI 変更追加がされても常に正しい状態を画面に表示することができます。
>
> 今回は時間の関係で紹介できませんでしたが、ページネーションの活性非活性処理も、同様の考え方を推し進めた Vue.js の算出プロパティーという処理で行っています。

まとめ

まとめ

- WordPress と Vue.js はとても相性が良いです。
- 特にサイトの多くの部分を WordPress により処理させ、一部を JavaScript で処理したい場合に、テンプレートファイルの構成の修正を最小限に抑え、簡単に導入がすることができます。

まとめ

- Vue.js のデーターバインディングは強力です。変数とHTML を拘束することにより、プログラムを簡潔にし、どのような操作を追加・修正しても紛れのない画面を出力することができます。
- データーとなる変数の追加、HTML へのバインディング、メソッドによる変数の処理というパターンで製作できます。

ご清聴ありがとうございました

“ ありがとうございました。 ”

- 引き続きブースの方におりますので質問などありましたらお願いいたします。
- OSC 北海道より配布されているアンケートに何か書いていただけたらフォローアップできるかもしれません。
- 資料は後ほど OSC 北海道 2017 サイトにリンクを掲載します。