

Vue.jsではじめるMVVM入門



June 9, 2016

Toru Enokido
Design Strategies Office.
DeNA Co., Ltd.

自己紹介

榎戸 徹 @55enokky



2014年 1月～10月
怪盗ロワイヤルフロントエンドを担当



2014年 10月～
KenCoMフロントエンドを担当

本日のアジェンダ

- ❑ Vue.js 概要
 - ❑ Vue.js とは
 - ❑ Vue.js の特徴
 - ❑ Vue.js がサポートするブラウザ
- ❑ Vue.js における MVVM
 - ❑ ViewModel とは
 - ❑ Model とは
 - ❑ View とは
- ❑ ToDoアプリを作ってみる
 - ❑ ディレクティブとは
 - ❑ methods とは
 - ❑ computed プロパティとは

Vue.js 概要

Vue.jsとは？

Vue.js は MVVM と呼ばれる設計パターンを採用している JavaScriptフレームワークです。
MVVMフレームワークでは Model、View、ViewModel の3つでアプリケーションを構築します。

Vue.jsの特徴

Vue.jsは双方向データバインディングを実現することに特化しているため、jQuery を使うととても複雑になってしまう処理をシンプルかつわかりやすく書くことができます。

AngularJSはフルスタックなフレームワークで、何でもできますが、独特な文法や制約によって学習コストが高いとされています。

一方でVue.jsは、軽量で他のライブラリに依存しないため、必要なところだけ Vue.js で作り、他のライブラリやフレームワークを組み合わせるという方法も可能です。

Vue.js がサポートするブラウザ

Vue.js がサポートするブラウザは最近のモダンブラウザのみです。

Internet Explorer 8 以下には対応していないため

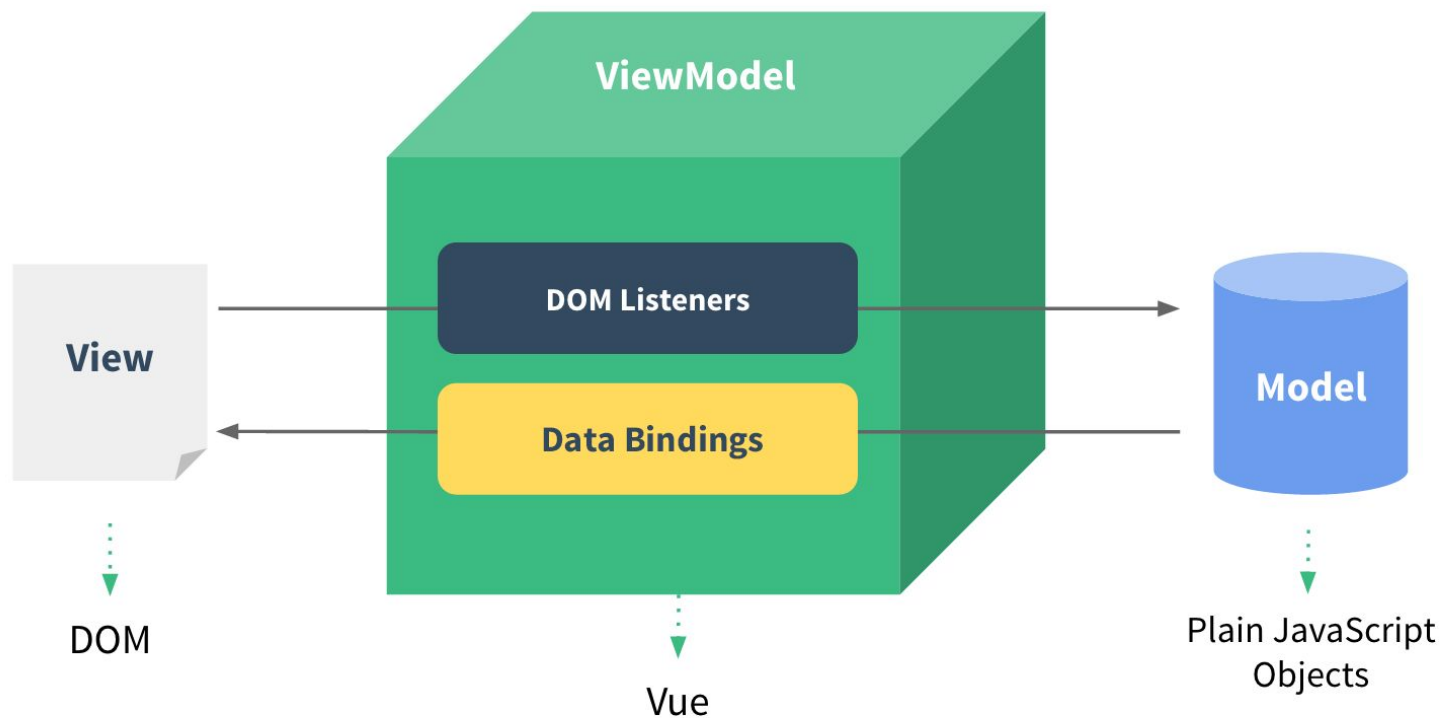
実際に使う際にはサポートブラウザを確認しましょう。

(実装にECMAScript 5の機能が使われており、これをサポートしているブラウザで動作するようになっていきます)

Vue.js におけるMVVM

MVVMとは？

MVVMとは**モデル(M)**と**ビュー(V)**間のやり取りを**ビューモデル(VM)**を介して行うアーキテクチャのことです。



(引用: <https://jp.vuejs.org/guide/overview.html>)

ViewModel

ViewModel とは Vue.js のインスタンスのことで、
View と Model をとりもつオブジェクトです。

Model

Model とは dataオブジェクト と methodsオブジェクト のことです。

dataオブジェクトの属性は v-text や v-model といったディレクティブによってバインドされます。

View

View とは el で指定されたDOM要素のことです。

elでDOM要素を指定することで、ViewModel の適用範囲を指定できます。

ToDoアプリを作ってみる

完成品の確認

今回は



<http://codepen.io/>


を使ってみます。


codepen の設定

1. <http://codepen.io/> にアクセス
2. 画面右上の  をクリック
3. JSパネルの設定アイコン()をクリック
4. Add External JavaScript に <http://cdn.jsdelivr.net/vue/1.0.24/vue.min.js> と入力します

Add External JavaScript

These scripts will run in this order and before the code in the JavaScript editor. You can also link to another Pen here, and it will run the JavaScript from it. Also try typing the name of any popular library.





ViewModel の作成

ViewModel の作成

js

```
new Vue({  
  
});
```

画面(View)の作成

ToDo アプリの機能

- 登録したToDo一覧が見れる
- 完了したToDoにチェックを入れられる
- 新規ToDoが登録できる
- 完了したToDoが削除できる
- ToDoの総数が表示されている

画面(View) の作成

html

```
<div id="my-app">
  <p>
    Task:
    <input type="text">
    <button>Add</button>
  </p>
  <hr>
  <ul>
    <li>
      <input type="checkbox">
      牛乳を買う
      <button>Delete</button>
    </li>
    <li>
      <input type="checkbox">
      プロテインを買う
      <button>Delete</button>
    </li>
    <li>
      <input type="checkbox">
      スポーツドリンクを買う
      <button>Delete</button>
    </li>
  </ul>
</div>
```

画面(View) の作成

CSS

```
ul {  
  margin: 0;  
  padding: 0;  
  list-style-type: none;  
}  
ul > li {  
  margin: 5px;  
  text-indent: 0;  
}
```

Modelを書いていく

Modelの作成 - ToDoリストの登録

js

```
data: {  
  todos: [  
    { task: '牛乳を買う', isCompleted: false },  
    { task: 'プロテインを買う', isCompleted: false },  
    { task: 'スポーツドリンクを買う', isCompleted: true }  
  ],  
  methods: {  
  }
```


登録したToDoリストを表示してみる

ディレクティブとは

View をコントロールするための特殊なコマンドです。

Vue.js ではHTMLに `v-***` で始まる属性を記述することで様々なDOM操作を行うことができます。

v-for

配列をレンダリングするためのディレクティブです。
形式は item in items となります。

html

```
<li v-for="todo in todos">
  <input type="checkbox">
  {{ todo.task }}
  <button>Delete</button>
</li>
```

v-model

form input 要素で双方向データバインディングするためのディレクティブです。
Checkbox で使用する場合は boolean 値が返ります。

html

```
<input type="checkbox" v-model="todo.isCompleted">
```

v-bind

html の属性値をバインドするためのディレクティブです。
v-bindは省略することも可能です。

html

```
<span v-bind:class="{ 'complete': todo.isCompleted }">{{ todo.task }}</span>
```

CSS

```
ul > li > .complete {  
  text-decoration: line-through;  
  color: #ddd;  
}
```

新たにToDoを追加できるようにする

新たにToDoを追加できるようにする

フォームから ToDo を追加するための data を追加し、View と結びつけます。

html

```
<input type="text" v-model="newTask">
```

js

```
data: {  
  newTask: '',  
}
```

v-on

イベントを受け取るためのディレクティブです。
@を使って省略することも可能です。

html

```
<button v-on:click="addTask()">Add</button>
```


methodsとは

methods とは ViewModel のメソッドを定義するオブジェクトです。

ここで定義したメソッドは v-on をつかって呼び出すことができます。
メソッド内で this を使用した場合は ViewModel を指します。

Add ボタンをクリックした時のメソッドを追加する

フォームから ToDo を追加するためのメソッドを追加します。

js

```
methods: {  
  addTask: function() {  
    this.todos.push({  
      task: this.newTask,  
      isCompleted: false  
    });  
    this.newTask = '';  
  }  
}
```

ToDoを削除できるようにする

\$remove

Vue.js で配列の要素を削除するためのメソッドです。

html

```
<button v-on:click="deleteTodo(todo)">Delete</button>
```

Delete ボタンをクリックした時の メソッドを追加する

js

```
deleteTodo: function(todo) {  
  this.todos.$remove(todo);  
}
```

ToDoの数を表示する

computed プロパティ

動的なプロパティを生成するメソッドを持つオブジェクトで、複雑な計算が必要な場合は computed プロパティを使用します。

(methodsとの違いはgetやsetの定義ができます)

ToDo の数を表示するための view を追加する

html

```
<p>Remaining Tasks: {{ remains }}/{{ todos.length }}</p>
```


computed プロパティに メソッドを追加する

js

```
computed: {  
  remains: function() {  
    var count = 0;  
    for(var i=0; i<this.todos.length; i++) {  
      if(!this.todos[i].isCompleted) {  
        count++;  
      }  
    }  
    return count;  
  }  
}
```

やってみよう

やってみよう

1. 新規登録フィールドが空の時は登録できないようにしてみよう！
 - addTask関数のなかでnewTaskが空の場合は登録しない処理を追加してみよう。
2. エンターキーを押すだけで新規タスクが登録できるよう機能を追加してみよう！
 - キー修飾子 (<https://jp.vuejs.org/guide/events.html#キー修飾子>)を使ってキーイベントを取得してみよう。

答え合わせ

答え合わせ

1. 新規登録フィールドが空の時は登録できないようにする

js

```
addTodo: function() {  
    if( this.newTask == '' ) return;  
    ...  
}
```

答え合わせ

2. エンターキーを押すだけで新規タスクが登録できるよう機能を追加する

html

```
<input type="text" v-model="newTask" v-on:keyup.enter="addTask()">
```