第2回
フロントエンド入門
Vue.js編

2017年5月24日

オトナのプログラミング勉強会

協力 **未来会議室**

自己紹介

- 村上　卓
- 29歳
- フリーランス
- Angular/Ruby On Rails

この勉強会について

オトナのプログラミング勉強会

http://otona.pro

- 2016年8月から開始
- 月2回（第1水曜、第3水曜）
- ジャンル（？）
  - プログラム言語、機械学習、Web系...

5月はGWのためずらしました

前回の復習

前回の資料はこちら

http://otona.pro/post/20170315vuejs/

Vue.jsの特徴

- ビューに特化
- 軽量
- Virtual DOM (仮想DOM)
- コンポーネントシステム
- 日本語ドキュメント

Hello (html)

テンプレートの書き方 {{}} でプロパティ名を囲む

```
<div id="app">
  {{ message }}
</div>
```

Hello (js)

Vueインスタンスを生成

```
new Vue({
  el: '#app',
  data: {
    message: 'Hello Vue!'
  }
});
```

実行

Hello Vue!が表示される

- elオプションにDOMを指定
- dataオプションにリアクティブにするオブジェクト

コンポーネント

コンポーネントとは

部品化(DOMの抽象化)

- 再利用可能
- カプセル化
- 疎結合

Webコンポーネント

4つの仕様の総称

- Templates
  - クライアントサイドのテンプレート(templateタグ)
- HTML Imports
  - HTML/CSS/JSをまとめてロードする仕様
- Custom Elements
  - 独自要素の登録/利用
- Shadow DOM
  - DOMツリーのカプセル化

Vue.jsのコンポーネント

- Webコンポーネントではない
- HTMLを拡張
- Custom Elementsの仕様で設計

コンポーネントを作ってみよう

ひな形　https://jsfiddle.net/sugumura/gcg9xpro/

単純なコンポーネント

```
<div id="main">
  <my-component></my-component>
</div>
```

```
Vue.component('my-component', {
  template: '<h1>カスタムタグ</h1>'
})
```

コンポーネントの展開

タグが下記のように展開されている

```
<div id="main">
  <h1>カスタムタグ</h1>
</div>
```

カスタムタグの命名

小文字のハイフン区切りがカスタムタグの仕様

データの受渡し

```
<my-component message="Hello!"></my-component>
```

```
Vue.component('my-component', {
  props: ['message'],
  template: '<h1>{{ message }}</h1>'
})
```

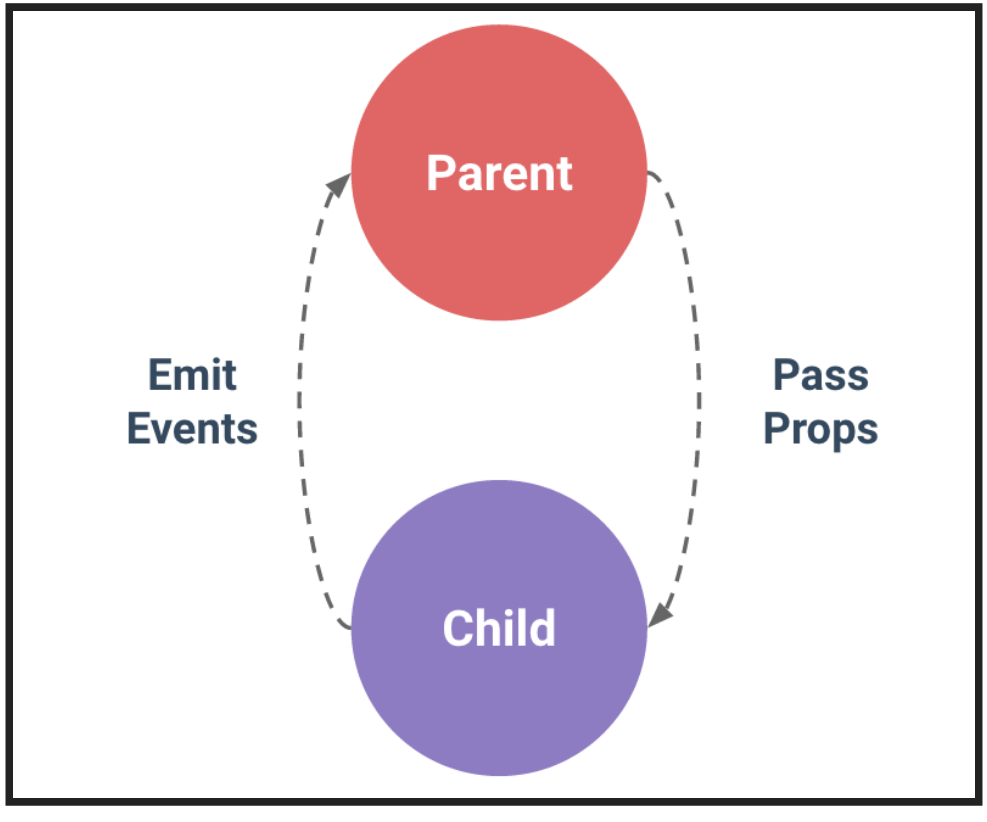
バインディング

バインディングしたプロパティを渡す場合

```
<input v-model="modelMessage">
<my-component v-bind:message="modelMessage"></my-component>
```

```
new Vue({
  el: '#main',
  data: {
      modelMessage: ''
  }
})
```

コンポーネントの構成



ref:
https://vuejs.org/v2/guide/components.html#Composir

Components

単方向データフロー

- one-way-down
- 親から子にデータが流れるだけ
- 子から親へはイベントを経由

カスタムイベント(子)

```
Vue.component('my-button', {
    props: {
      label: {  // バリデーション
      type: String, // 文字列
      required: true  // 必須
    }
  },
  template: '<button v-on:click="clickLabel">{{ label }}</butt
  methods: {
    clickLabel: function () {
      // イベント発火
      this.$emit('click', this.label);
    }
  },
})
```

カスタムイベント(親)

```
<div id="main">
  <my-component v-bind:message="message"></my-component>
  <my-button label="a" v-on:click="labelClick"></my-button>
  <my-button label="b" v-on:click="labelClick"></my-button>
</div>
```

```
new Vue({
  el: '#main',
  data: {
      message: 'click my buttons',
  },
  methods: {
      labelClick: function (label) {
        this.message = label + ' button click!';
    }
  }
});
```

コンポーネントを作る練習をしよう

スピナー

ローディング中にぐるぐる回るやつ

ひな形

https://jsfiddle.net/sugumura/xvt2Lh46/

スピナー(js)

```
Vue.component('my-spinner', {
  template: `<div class="my-spinner rotate"></div>`
});
```

スピナーのCSS

四角を表示し、下線を透過

```
.my-spinner {
  display: inline-block;
  width: 16px;  // 横幅
  height: 16px;  // 縦幅
  border: 2px;  // 線の太さ
  border-style: solid;  // 線の種類
  border-color: black black transparent;  // 色
  border-radius: 100%;  // 角を丸める半径
}

.my-spinner.rotate {
  animation: rotate-anim 1s 0s infinite linear;
  animation-fill-mode: both;
}
```

アニメーション

360度回転

```
@keyframes rotate-anim
{
  0%   { transform: rotate(0deg) scale(1); }
  50%  { transform: rotate(180deg) scale(0.8); }
  100% { transform: rotate(360deg) scale(1); }
}
```

パラメータ化をしよう

サイズや色が固定でしか扱えないのでパラメータ化しましょう

プロパティ

```
Vue.component('my-spinner', {
// ...
  props: {
    loading: {  // 表示・非表示のフラグ
      type: Boolean,
      default: true
    },
    color: {  // スピナーの色
      type: String,
      default: '#000000'
    },
    size: { // スピナーのサイズ
      type: Number,
      default: 16,
    }
```

算出プロパティ

style要素として表示に反映させる

```
Vue.component('my-spinner', {
// ...
  computed: {
    style: function () {
      return {
        height: this.size + 'px',
        width: this.size + 'px',
        borderColor: this.color + ' ' + this.color + ' ' + 'tr
      };
    }
  }
// ...
```

テンプレート

```
Vue.component('my-spinner', {
  template: `<div class="my-spinner rotate"
    v-bind:style="style"
    v-show="loading"></div>`,
// ...
```

```
<div id="main">
  <my-spinner color="#11FF11" v-bind:size="30"></my-spinner>
</div>
```

スピナー完成

スピナーを他のコンポーネントに組み込もう

SpinnerImage

画像読み込み中にスピナーを
表示するようなコンポーネント

長いので完成品

https://jsfiddle.net/sugumura/4jdw8gz6/

ポイント1

- dataはコンポーネントでは関数

```
data: function () {
  return {
    isLoad: this.loading,
    src: this.url
  };
},
```

ポイント2

- propsに定義したものは直接methodsで書き換えてはいけない(単方向フロー)

```
props: {
  loading: {
    type: Boolean,
    default: true
  }
},
data: function () {
  //..
    isLoad: this.loading,
},
methods: {
  load: function() {
    this.isLoad = false;
  },
```

コンポーネントの設計(私見)

- 汎用性の高いコンポーネントを作るのは難しい
  - 特にビューを伴う場合
- スコープを大きいものは作りやすい
  - 動くものを作ってからコンポーネント化すると良い
- レイアウトにコンポーネントを使うと見通しが良い
- 再利用性よりスコープを小さくするためにコンポーネント化することが多い

レイアウトのコンポーネント

ヘッダー・フッターをコンポーネント化

```
<div id="app">
  <app-header>Header</app-header>
  <app-footer>Footer</app-footer>
</div>
```

コンテンツ配信

slotを利用することで親で指定したコンテンツを配信できる

```
// footer省略
var appHeader = Vue.extend({
    template: `<header>
      <slot>Headerプレースホルダー</slot>
  </header>`
});

new Vue({
  el: '#app',
  components: {
    'app-header': appHeader
  }
});
```

コンポーネントの登録

Vue.componentを利用するとグローバル登録
Vueインスタンスのcomponentsに指定するとローカル登録

```
new Vue({
  // ...
  components: {
    'my-component': Child
  }
})
```

単一ファイルコンポーネントについて

- vueファイルにテンプレート/スタイル/スクリプトを記述する
- ビルド処理必須
- スクラッチから作成できる場合は要検討

単一ファイルコンポーネント - Vue.js

ライフサイクルダイアログ

- Vue.jsの生成から破壊までをまとめた図
- 挙動に困ったときに見ると解決するかも...？

https://vuejs.org/v2/guide/instance.html#Lifecycle-Diagram

時間が余ったらその場の要望で何かコーディングします

例

- 動的コンポーネント
- アニメーション
- 動的セレクトリスト

質疑等

おつかれさまでした