Vue.js

マッキー

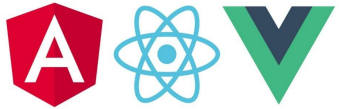
目次

- JS FrontEnd FWについて
- jQueryとの違い
- Vue.jsについて
- Vue.js vs jQuery
- まとめ

JS FrontEnd FWについて

【特徴】
仮想DOM・・・DOMを構築できるオブジェクト
データバインディング・・・データの更新に伴い、ビューを更新する仕組み
コンポーネント志向・・・UIを切り出して部品化、再利用がしやすい

【有名どころ】
Angular.js・・・フルスタック、大抵のことは何でもできる、導入コスト高
React.js・・・ライブラリが豊富、JSX記法に馴染めるか、導入コスト中
Vue.js・・・HTMLとJSが独立、既存にも組み込みやすい、導入コスト小



jQueryとの違い

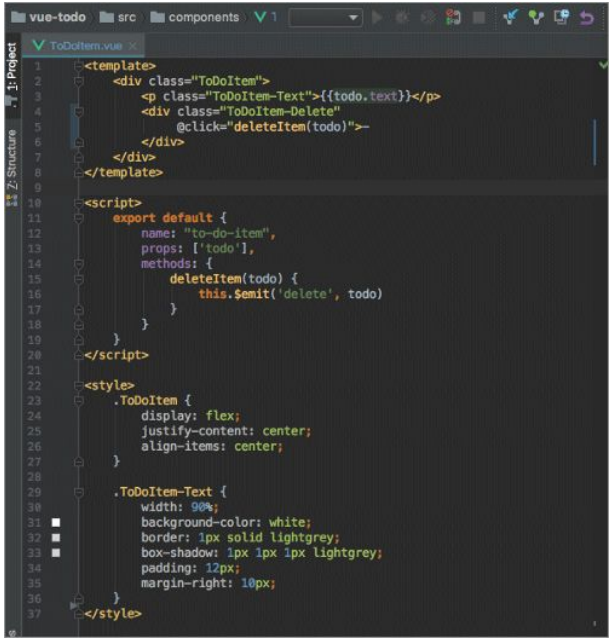
jQueryはセレクタ操作に特化したライブラリ

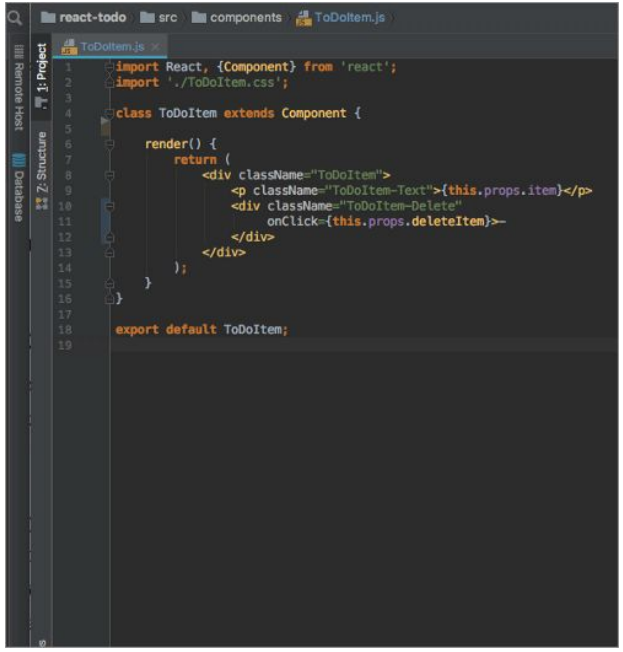
DOMの一部を少しだけ弄るには手軽に扱える

画面内で発火する複数のイベントに応じて
画面の表示パターンが変わるような場合は、全体の管理が大変

Vue.jsについて

・学習コストが低い
・テンプレート、ロジック、スタイルを分離して記述できる





vue-todo › src › components › V 1

ToDoItem.vue

```vue
<template>
    <div class="ToDoItem">
        <p class="ToDoItem-Text">{{todo.text}}</p>
        <div class="ToDoItem-Delete"
             @click="deleteItem(todo)">-
        </div>
    </div>
</template>

<script>
    export default {
        name: "to-do-item",
        props: ['todo'],
        methods: {
            deleteItem(todo) {
                this.$emit('delete', todo)
            }
        }
    }
</script>

<style>
    .ToDoItem {
        display: flex;
        justify-content: center;
        align-items: center;
    }

    .ToDoItem-Text {
        width: 90%;
        background-color: white;
        border: 1px solid lightgrey;
        box-shadow: 1px 1px 1px lightgrey;
        padding: 12px;
        margin-right: 10px;
    }
</style>
```

react-todo › src › components › ToDoItem.js

ToDoItem.js

```jsx
import React, {Component} from 'react';
import './ToDoItem.css';

class ToDoItem extends Component {

    render() {
        return (
            <div className="ToDoItem">
                <p className="ToDoItem-Text">{this.props.item}</p>
                <div className="ToDoItem-Delete"
                     onClick={this.props.deleteItem}>-
                </div>
            </div>
        );
    }
}

export default ToDoItem;
```

Vue.js vs jQuery

実際に書いてみた

まとめ

簡単なUI操作はjQueryでもいいかも

モダンなフロントエンドを作りたい

SPAをつくりたい

Vue.jsから始めるのがいいかな