React + MobX + TypeScript + styled-components
を使って簡単なカウンターを作ろう

一部資料ないのソースコードが読みにくいと思うので、その際はGitHub
のリポジトリから確認、コピペしてください
(https://github.com/kobakazu0429/react-mobx-counter)

yarnとは

JavaScript(Node.js)におけるパッケージ管理の役割
もともとnpm(Node Package Manager)というものがあり、
それの上位互換として生まれた

- とにかく早い

Reactとは

FacebookがOSSとして公開している、JavaScriptのライブラリ
Virtual(仮想) DOMを使うため、

- 高速レンダリング
- JSからのアクセスが容易

TypeScriptとは

JavaScriptにはない型(int, floatなど)を取り入れられる

メリット

- 変数に変な値が入らない
- null安全（nullがあるとコンパイルできない）である
- コンパイル時にエラーを吐くのでデバッグの高速化につながる

デメリット

- 本来JavaScriptには型がないので生のJavaScriptにコンパイル（トランスパイル）する必要がある
    - 今回はBabelを用いる

はじめに

まずは以下のコマンドで `yarn` が使えるかの確認をお願いします

```
$ yarn -v
1.7.0
```

それでは早速作っていきましょう！

```
$ yarn global add create-react-app
$ create-react-app --scripts-version=
            react-scripts-ts <your-app-name>
$ cd <your-app-name>
```

```
$ tree

├── README.md
├── images.d.ts
├── node_modules // インストールしたモジュールが保存されている
├── package.json
├── public // reactはここをベースにする
│   ├── favicon.ico
│   ├── index.html
│   └── manifest.json
├── src
│   ├── App.css
│   ├── App.test.tsx
│   ├── App.tsx
│   ├── index.css
│   ├── index.tsx
│   ├── logo.svg
│   └── registerServiceWorker.ts
├── tsconfig.json
├── tsconfig.prod.json
├── tsconfig.test.json
├── tslint.json
└── yarn.lock
```

```
$ cat package.json
```

```
{
  "name": "counter",
  "version": "0.1.0",
  "private": true,
  "dependencies": {
    "react": "^16.4.1",
    "react-dom": "^16.4.1",
    "react-scripts-ts": "2.16.0"
  },
  "scripts": {
    "start": "react-scripts-ts start",
    "build": "react-scripts-ts build",
    "test": "react-scripts-ts test --env=jsdom",
    "eject": "react-scripts-ts eject"
  },
  "devDependencies": {
    "@types/jest": "^23.1.2",
    "@types/node": "^10.3.6",
    "@types/react": "^16.4.2",
    "@types/react-dom": "^16.0.6",
    "typescript": "^2.9.2"
  }
}
```

多分こんな感じです。
RubyでいうGemfileみたいなもので、
ここにモジュールのバージョンなどが書き込まれます。

では今回使う他のモジュールのインストールをしてみましょう

```
$ yarn add --dev mobx mobx-react mobx-react-devtools
                 styled-components
// 一行で入力してください
```

ではもう一度中身の確認

```
$ cat package.json
```

```
{
  "name": "counter",
  "version": "0.1.0",
  "dependencies": {
    "mobx": "^5.0.3",
    "mobx-react": "^5.2.3",
    "mobx-react-devtools": "^5.0.1",
    "react": "^16.4.1",
    "react-dom": "^16.4.1",
    "react-scripts-ts": "2.16.0",
    "styled-components": "^3.3.3"
  },
  "scripts": {
    "start": "react-scripts-ts start",
    "build": "react-scripts-ts build",
    "test": "react-scripts test --env=jsdom",
    "eject": "react-scripts eject"
  },
  "devDependencies": {
    "@types/jest": "^23.1.1",
    "@types/node": "^10.3.4",
    "@types/react": "^16.4.1",
    "@types/react-dom": "^16.0.6",
    "typescript": "^2.9.2"
  }
}
```

こんな感じになったと思います。

現在の環境と同じ環境を作りたい場合は`package.json`があるディレクトリで`$ yarn`と打つと自動的にインストールされます

便利！！

一度サーバーを起動させてみましょう

```
$ yarn start
```

```
// package.json
{
  "name": "counter",
  "version": "0.1.0",
  "private": true,
  "dependencies": {
  },

  "scripts": {
    "start": "react-scripts start",
    "build": "react-scripts build",
    "test": "react-scripts test --env=jsdom",
    "eject": "react-scripts eject"
  },

  "devDependencies": {
  }
}
```

`$ yarn ○○○` は `package.json` の `scripts` に書かれたコマンドのショートカットです

```
// ./src/App.tsx
 1 import * as React from 'react';
 2 import './App.css';
 3
 4 import logo from './logo.svg';
 5
 6 class App extends React.Component {
 7   public render() {
 8     return (
 9       <div className="App">
10         <header className="App-header">
11           <img src={logo} className="App-logo"
              alt="logo" />
12           <h1 className="App-title">
              Welcome to React</h1>
13         </header>
14         <p className="App-intro">
15-          To get started, edit <code>src/App.tsx
  -           </code> and save to reload.
  +           Hello, React
16         </p>
17       </div>
18     );
19   }
20 }
21 export default App;
```

多分更新ボタン(F5, ⌘ + R, Ctrl + R)などを押さなくても、自動で更新されたと思います。

これをホットリロードといい、`webpack-dev-server` というものが更新を検知して、ブラウザも自動的に更新してくれます。

便利！！

解説

- `1: import * as React from 'react';`
  - `node_modules` にある `react` から `*(全て)` をインポートし `as(として) React` 使えるようにしています
- `6: class App extends React.Component {`
  - 1行目で読み込んだ `ReactからComponent` というものを `extends(継承)` し、 `App` というClass(塊)を作っています
- `21: export default App;`
  - `export` すると他のファイルから `import './App.css';` のように参照することができ、 `import` を使う
  -

今回は`create-react-app`で生成したので、不要なファイルが多くわかりにくいので以下のファイルを削除します。

- `src/App.test.js`
- `src/registerServiceWorker.js`
- `src/logo.svg`
- `public/manifest.json`
- `public/favicon.ico`

./package.json

```
- private": true,
```

./tsconfig.json

```
 "compilerOptions": {
+  "experimentalDecorators": true,
+  "allowSyntheticDefaultImports": true
```

./tslint.json

```
 {
   "extends": ["tslint:recommended", "tslint-react", "tsl
   "linterOptions": {
     "exclude": ["config/**/*.js", "node_modules/**/*.ts"
   },
+  "rules": {
+    "interface-name": [true, "never-prefix"]
+  }
 }
```

`./src` 内で

```
$ mkdir components
$ mkdir stores

$ touch components/Counter.tsx
$ touch stores/CountStore.ts
```

```
$ tree

├── components
│   └── Counter.tsx
├── stores
│   └── CountStore.ts
├── App.css
├── App.tsx
├── index.css
└── index.tsx
```

`./src/index.tsx`

```tsx
import * as React from "react";
import * as ReactDOM from "react-dom";
import App from "./App";
import "./index.css";

ReactDOM.render(<App/>, document.getElementById("root"));
```

./src/App.tsx

```tsx
import { Provider } from "mobx-react";
import * as React from "react";
import Counter from "./components/Counter";
import CountStore from "./stores/CountStore";

const stores = new CountStore();

class App extends React.Component {
  public render() {
    return (
      <Provider count={stores}>
        <Counter />
      </Provider>
    );
  }
}

export default App;
```

./src/stores/CountStore.ts

```
import { action, computed, observable } from "mobx";

export interface CountStoreType {
  num: number;
  getDoubleCount: () => void;
  onIncrement: () => void;
  onDecrement: () => void;
}

export default class CountStore {

}
```

./src/stores/CountStore.ts

```
export default class CountStore {
  @observable private num = 0;

  @computed // observableが変わると同時に変わる
  get getDoubleCount() {
    return this.num * 2;
  }

  @action.bound // vueでいうmutatinos stateを変えれる
  public onIncrement() {
    this.num = this.num + 1;
  }

  @action.bound
  public onDecrement() {
    this.num = this.num - 1;
  }
}
```

./src/components/Counter.tsx

```
import { inject, observer } from "mobx-react";
import * as React from "react";
import { CountStoreType } from "../stores/CountStore";

interface Props {
  count?: CountStoreType;
}

@inject("count")
@observer
export default class Counter
   extends React.Component<Props> {

}
```

./src/components/Counter.tsx

```
+ import styled from "styled-components";

export default class Counter extends React.Component<Props
  public render() {
    const { count } = this.props;
    // <=> const count = this.props.count;
    return (
      <Wrapper>
        Counter : {count!.num} <br />
        <button onClick={count!.onIncrement}> + </button>
        <button onClick={count!.onDecrement}> - </button>
        <br /> GetDoubleCount: {count!.getDoubleCount}
      </Wrapper>
    );
  }
}

const Wrapper = styled.div`
  text-align: center;
  background-color: rgba(186, 203, 255, 0.8);
`;
```