React 2023

2023/02/10

馬場一樹

状態管理

Reactとは

- JavaScript で UI を作るためのライブラリ
- Svelte/Vue/Angular などもあるけど一番よく使われているのが React
-

React登場前のUI開発: jQueryの時代

- jQuery は UI を作るための部品集
- HTMLのDOM編集するため状態が HTML にある
- 大規模な UI を JS で記述するには不向き
    - 逆に LandingPage みたいなやつなら jQuery でも良いと思う

ReactはUIの状態をJSで管理することができる

例えば、モーダルが開いているか閉じているかを知りたいとき、

- jQueryの場合はモーダルのDOMがあるかどうかを確かめる
- Reactの場合はJSの変数でモーダルの状態を管理する

React の状態管理の方法

- React Hook
  - useState / useEffect / useContext など
  - Reactに備わっている機能
- Redux
  - デファクトスタンダードだったライブラリ
  - React Hook 以前からある
  - いわゆる「単一方向データフロー」
    - GitHub - facebook/flux: Application Architecture for Building User Interfaces
- Recoil / jotai
  - 最近よく聞く

React における状態

- 個々のComponent内部の細かい状態の管理
  - modalの開閉状態の管理など
  - useState, useReducer などを使う
- グローバルな状態の管理
  - サインインユーザーの情報など
  - useContext, Redux, Recoil, jotai などを使う

globalな状態管理

- サインイン情報などシステム全体で使いたい状態
- useState でこれをやると各コンポーネントに引数でstateを渡していく必要がある（いわゆるバケツリレー）
- useContext を使うとバケツリレーする必要がなくなる

useContextの欠点

```
ReactDOM.createRoot(document.getElementById("root")).render(
  <React.StrictMode>
    <ThemeContext.Provider>
      <UserContext.Provider>
        <RouterProvider router={router} />
      </UserContext.Provider>
    </ThemeContext.Provider>
  </React.StrictMode>
);
```

複数のContextを定義するとネストが深くなり管理が煩雑になる。

useContext の代替手段

- Recoil, jotai など
- preact, solid.js における signal も似たような使い方ができる

Reduxは何者なのか？

- Recoil, jotai と同じようにグローバルな状態の管理ができる
- 単一方向データフローによりAPIからデータを取得する (Query) や、Form による Create/Update のデータをサーバーに反映する (Mutation) も管理できる

だったら Redux を使えば一番良いのか？

- レンダリングが増えてパフォーマンスが悪くなる可能性がある
- 単一方向データフローは初心者には理解が難しく、またほとんど何もしないコードが増えていく傾向にある
- Tanstack query や SWR などキャッシュをうまいことやってくれるライブラリが出てきた

最近のはやり？

- コンポーネント内部の細かい状態管理
  - useState, useReducer
- global な状態管理
  - Recoil, jotai
- API通信周り
  - Tanstack query, SWR, など
- Form の状態管理
  - React hook form, 素のHTML5 validation, など
  - 【React】フォームは状態管理せずに実装できるよ - Qiita
  -

Reactの状態管理の歴史について

- Reactにおける状態管理の動向を追ってみた

フレームワーク

React の代表的なフレームワーク

- Next.js
- Remix

- Routes や SSR (Server side rendering) などがやりやすい
- 状態の管理は前述の通り複雑なので組み込まれていない
    - ただし SWR は Next.js と同じチームが作っている
-

フレームワーク登場以前のReact

- SPA が多い
- 設計者の力量によってはスパゲッティコードになる
- 統一された企画がないのでプロジェクトに参画しづらい
-

Next.js

- Vercel が作っている
- Vercel で動かすことを前提とした機能がいくつかある
  - あるいはVPS や CloudRun などのサーバーで動かす
  - 実質 SSR か静的HTML出力の2択になるので Firebase などの他のホスティングサービスで Next.js を動かすのはちょっと怖い
  - 静的HTML出力の場合かなり機能が制限される
- パフォーマンスのために機能が複雑になりがち

Remix

- React Router の作者が作ってる
- Spotify に買収された
- React Router v6 を使いつつ効率的に SSR できる
- サーバーサイドも実装可能
  - 1つのjsxファイル中にサーバー側のコード (Loader) とクライアント側のコードを書ける
  - ちなみに deno の fresh も似たような感じ
-

自分としてはフレームワークがあまりすきになれない

- フレームワークは色々な事情を考慮して作るので自分のユースケースで考えたとき過剰になる場合が多い
- そのフレームワークの流儀に従う必要がある
- Next.js にいたってはサーバー環境まで束縛されている感じが強い
  - ベンダー依存がなんとなく不安

個人的に注目しているもの: preact-cli

- preact のフレームワーク
- preact の公式が作っている
- SPA, SSR, pre-rendering ができる
- 癖が少なくてちょうどいい感じ
- preact
  - react 互換だが 3kb という小ささ
  - 100%の互換性をうたっており、React 製のライブラリが使える
  - className を class と書くことができる等、後発ゆえの細かい改善が色々ある
  - signal, createContext など React にはない機能もある
    - この機能で Recoil / jotai などが不要になる

その他

新機能: Suspense

- Component ごとに SSR できる機能
- Component を遅延ロードすることでページの表示を高速化する
- 詳しくは
  - Suspenseはどのような機能なのか | ReactのSuspense対応非同期処理を手書きするハンズオン

Reactの学習ロードパス

- SPA を作ってみる
  - Tutorial v6.8.1 | React Router
  - React Router の Tutorial が丁寧な Hands-on なのでこれをやる
  - Vite, React Router について学べる
- 上記に API 通信機能を組み込む
  - オープンなAPIを使って機能を作ってみる
    - Cataas … 猫の画像をランダムに返してくれる API
    - News API … 世界中のニュースをまとめている API
  - fetch や axios などを使って通信する
  - React Router の loader で通信する or Tanstack Query や SWR を使ってみる
- Remix をやってみる
  - React Router v6 を理解していればほとんどわかったようなもの