



React

全4回で学ぶReact.js入門講座 4回目

2025/7/01

Kazuma SEKIGUCHI

前回のアジェンダ

- useEffectの使い方
- 再レンダリングの起きる条件
- memoによるレンダリング最適化
- useCallbackによるレンダリング最適化
- 変数のmemo化
- グローバルなState管理の方法（Contextの利用）

今回のアジェンダ

- カスタムフックとは何か
 - カスタムフックの必要性
- ロジックの再利用
- カスタムフックの作成、利用
- ビルド
- npm serveでの確認

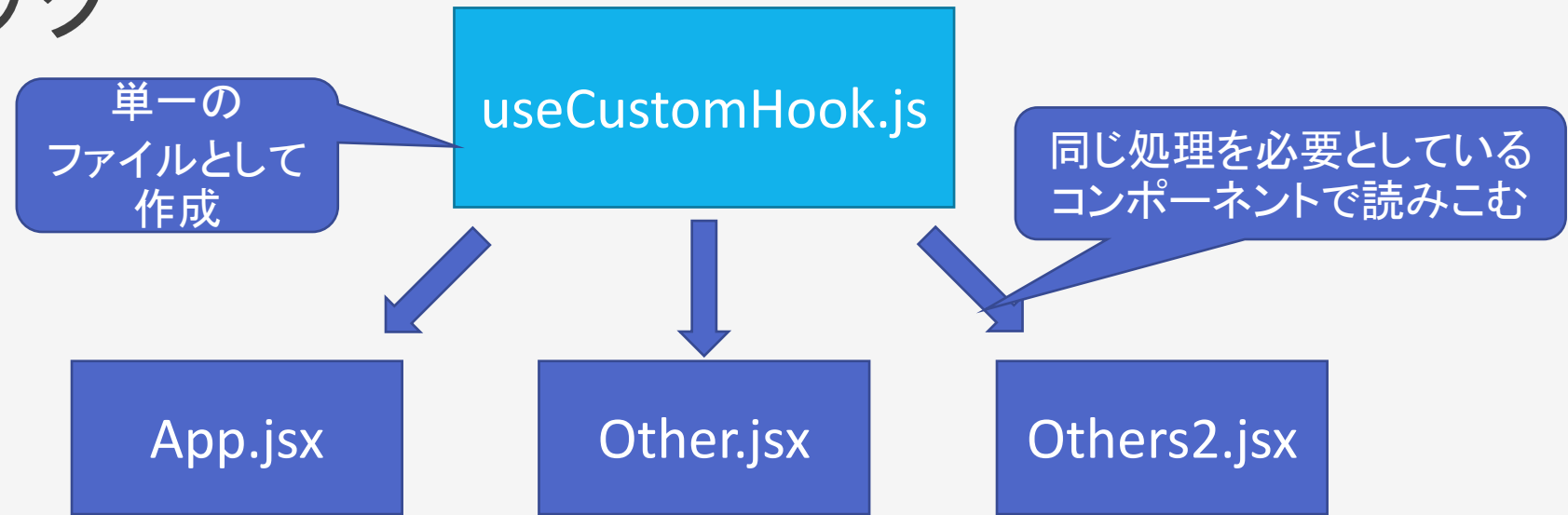
カスタムフック

- 任意の処理をまとめてHooks（機能みたいなもの）を作成することが可能
 - 自分で作成できるHookなのでカスタムフック
 - ReactのuseState()やuseEffect()などは利用可能
 - フック名（関数名）はuseから始まること
- 処理を分離できるため、複数のコンポーネントから利用が可能になる
 - ロジック部分を分離して利用することが可能

カスタムフックの利点

- 大規模になればなるほど同じような処理を行う場面は増える
 - ユーザーデータの取得
 - 検索結果の表示
 - お問い合わせフォームの表示
- できれば、1つにまとめておいた方が、修正や管理がしやすい
 - バグがあったときにも1つだけ修正すれば、適用される
 - 分離していないと全てのコンポーネントを修正しないといけない、かつ大体見落として残してしまう
- ファイルからロジックを分離できるので、見通しが良くなる
 - 状況によっては一覧性に乏しい状況にはなる

カスタムフック



- どこかのコンポーネントに書いて、他のコンポーネントでも使い回したい場合などに別ファイルとして抜き出しておく
 - 使いたいコンポーネントで読み込むことで、利用が可能
- 通常ファイル名やHook名をuseOOOと、useから始める
 - 他のHooksもuseから始まっているため
 - 拡張子はコンポーネントと異なり、jsとする例が多い

カスタムフック

- Reactのコンポーネント同様、useEffect,useStateなどのuse系機能は全て利用可能
 - JSのものも当然利用可能であるため、機能の制限は少ない

• 呼び出す側

カスタムフックのファイル

```
import { useFetchUsers } from "../hooks/useFetchUsers";  
export const App = ()=>{  
  const { userlist,isLoading,isError,onClickFetchUsers} = useFetchUsers();
```

カスタムフックがreturnしてくる
ものを受け取る

JSONデータの読み込み

- サーバーサイドと連動する場合、JSONでデータを受け取るケースは多い
 - JSONでデータ交換をすることで、データのやりとりが容易
 - サーバーサイドはJSONだけ出力するようにしてくれればOK
- 受け取るReact側にJSONを受け取る機能は存在しない
 - JSに存在するfetchを使う
 - axiosライブラリーを利用する
 - Reactなどでサーバーと連動する場合のHTTPクライアントの定番ライブラリー
 - <https://github.com/axios/axios>

JSONデータの読み込み

- サーバーサイドからの読み込みの場合、時間が掛かる点、タイムアウトするケースがある点、失敗するケースがある点を考慮する必要がある
 - fetchではタイムアウト処理が実装されていないため、自分で実装する必要がある点は注意が必要
 - axiosの方がその点では楽
 - エラー時はcatchで処理が可能

```
const controller = new AbortController();  
const signal = controller.signal;
```

```
const timeoutId = setTimeout(() => {  
  controller.abort();  
}, 5000);
```

```
fetch(url, { ...options, signal })  
  .then(response => {  
    clearTimeout(timeoutId);  
    // レスポンスの処理  
  })  
  .catch(error => {  
    clearTimeout(timeoutId);  
    if (error.name === 'AbortError') {  
      // タイムアウトエラーの処理  
    } else {  
      // その他のエラーの処理  
    }  
  });
```

fetchでのタイムアウト処理例

fetchでデータを受け取る

- 最近 Fetch API でデータ取得する例が多い

```
fetch("http://localhost:3000/data.json")  
  .then(response => response.json())  
  .then((json) => {  
    console.log(json)  
  })  
  .catch(() => setIsError(true))  
  .finally(() => setIsLoading(false));
```

データの受け取り先を指定

response に受け取ったデータが存在。
json() で JSON 形式として受け取る

json に受け取った JSON データが格納されるので、後で行いたい処理を記述

行いたい処理内容

エラーが起きたときの処理を記述

全ての処理が正常に完了したときの処理を記述

受け取ったデータをmapで処理する

- 配列をループ処理し、各要素に対して指定された関数を適用した結果を新しい配列に格納して返す
 - JSONデータは通常配列データであるため、mapとの相性が良い

すべてを処理し
終わった配列が
格納される

json.peopleという
配列

それにmapを
利用

```
const users = json.people.map(user => ({  
  name:user.name,  
  age:user.age,  
  hobby:user.hobby.join(',')  
}));
```

配列の1つの
データ

新しいオブジェクト
を作成

ビルド

- 今までに作成したReactのアプリはそのままではWebサーバー上では動作しない
- Webサーバー上で動作させるためにはビルドが必要
 - 必要なファイルをまとめて1つのファイルにしたり、ファイルの依存関係进行处理して、Webサーバーで動作できるようにしてくれる
 - `npm run build`コマンドを実行することで、`dist`フォルダー内に本番用のファイルが生成される
- ビルドしてできあがったデータをWebサーバーにアップロードすれば、問題無く動作する

ビルド

- npmコマンドでビルドを行う
 - npmでビルドができるのは、reactが準備してくれているため
 - 自分でreactを設定した場合は、ビルド設定をする必要がある

```
npm run build
```

- reactのフォルダーで上記コマンドを実行するとdistフォルダーが自動的に作成される
 - その中のファイルやフォルダーをアップロードすればOK
 - あちこちにReactで作成した、という名称が残っているので必要に応じてHTMLなどの修正をしても良い

ビルドされたデータを確認する

- ビルドされると、index.htmlが存在するので、それをダブルクリックすれば実行できそう
 - これをしてもきちんとした表示はされない（エラーになる）
- Webサーバーを通さないと正常に表示できないため、必要であれば、npmでWebサーバーをインストールして確認する

```
npm install -g serve
```

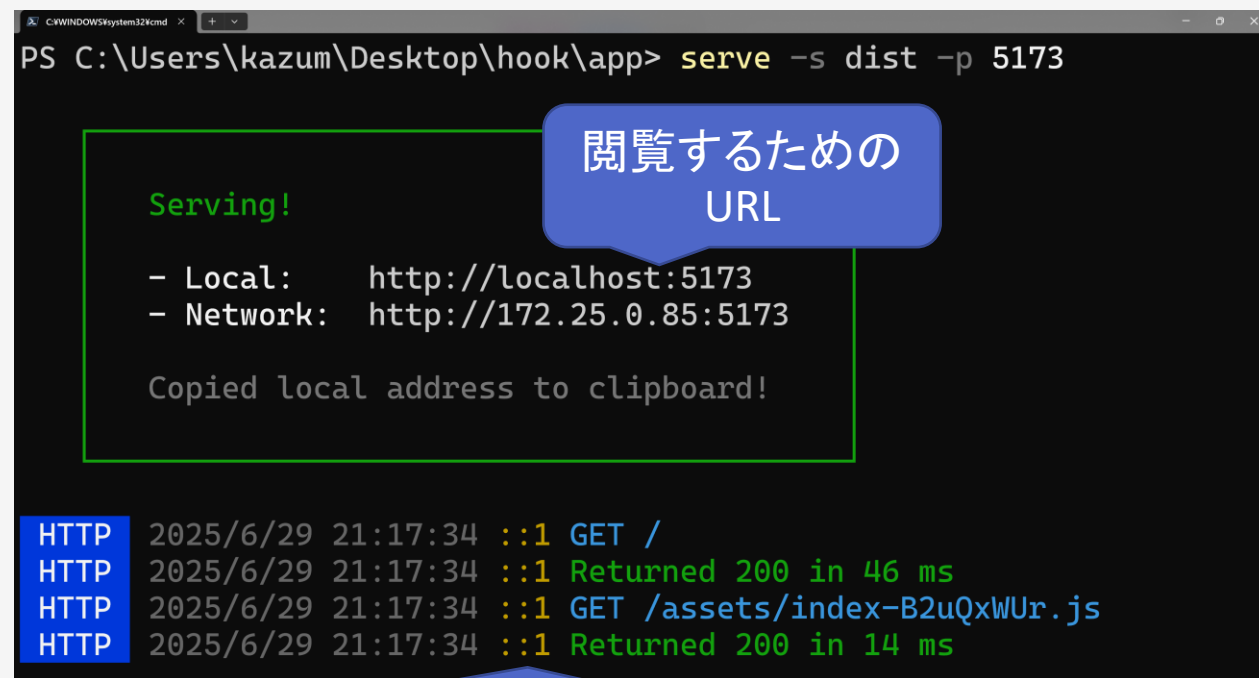
- npmコマンドでserveをインストールする
 - -gを付けているためグローバルインストールになる
（1回インストールすれば以後は恒常的に利用可能になる）
 - 既にインストールしている場合は、この作業は不要

ビルドされたデータを確認する

```
serve -s dist -p 5173
```

デフォルトは3000で起動するが、
pオプションで利用するポートを
変更可能

- インストールが完了したら、reactのフォルダーで上記コマンドを打つことでWebサーバーが立ち上がり、確認すべきURLが表示される
 - そこにブラウザでアクセスすれば表示され、機能も動作する
- 停止するときは、ctrl+C (command + C) で止まる



```
PS C:\Users\kazum\Desktop\hook\app> serve -s dist -p 5173

Serving!

- Local:    http://localhost:5173
- Network:  http://172.25.0.85:5173

Copied local address to clipboard!

HTTP 2025/6/29 21:17:34 ::1 GET /
HTTP 2025/6/29 21:17:34 ::1 Returned 200 in 46 ms
HTTP 2025/6/29 21:17:34 ::1 GET /assets/index-B2uQxWUr.js
HTTP 2025/6/29 21:17:34 ::1 Returned 200 in 14 ms
```

閲覧するための
URL

アクセスしたときのアクセ
スログ (閲覧すると増えて
いく)

この先の学習

- 公式ドキュメントを再度確認
 - チュートリアルも恐らくさほど苦無く読めるはず
- プロジェクトを作成して実際に作ってみる
 - 書籍検索・管理アプリ
 - メモ/日記アプリなどなど
- 周辺エコシステムの学習
 - Jotai、React Router、Tailwind CSS
 - TypeScript
 - Next.js

ありがとうございました。
よきReactライフを。