SPAを制作しているときに、Reactの状態管理でログイン状態かログアウト状態かを取得したかったため、useReducertとuseContextで実装しました。

方針

- ログイン状態を判断するstateをグローバルで管理する
- そのstateには、ログイン状態であればユーザーのidを挿入し、ログアウト状態であればnullを挿入する

Reducerを作成する（UserAuthReducer.ts）

Reducer用のファイルを作成し、下記コードを作成しました。

```
export type StateType = {
    id: string;
};

export type ActionType = {
    type: string;
    payload: string;
};

export const UserAuthReducer:any = (state: StateType, action: ActionType) => {
    switch (action.type) {
        case 'setId':
            return {
                ...state,
                id: action.payload
            };

        case 'setOutId':
            return {
                ...state,
                id: null
            };

        default:
            return {
                state
            };
    }
}

export const initialState = {id: null};
```

- TypeScriptなので、型も指定する
- Reducerのdispatchのタイプが ‘SetId’ であれば、stateのidに dispatchのpayloadに渡す**string**(id)を挿入
- Reducerのdispatchのタイプが ‘SetOutId’ であれば、stateのidに dispatchのpayloadに渡す**null**を挿入
- initialStateは、stateのidの初期値を指す

Contextを作成する（UserAuthContext.ts）

私はContext用のファイルを作成しましたが、親コンポーネントのファイルに直接書く方もいるようですので、自由です。

```
import React, { createContext } from 'react';

export const UserAuthContext = createContext({} as {
  state: any;
  dispatch: React.Dispatch<React.SetStateAction<any>>;
});
```

ここでもちゃんと型を指定しないとTypeScriptに怒られました。

親コンポーネントにProviderを設置（app.tsx）

stateを適用したいコンポーネントたちをContext.Providerで囲みます。

```
//React
import React,{useReducer} from 'react';
import ReactDOM from 'react-dom';

//ContextProvider
import {UserAuthReducer, initialState } from './reducers/UserAuthReducer';
import {UserAuthContext} from './UserAuthContext';

//Components
import Search from './Search';
import Top from './Top';
import Login from './Login';

const App: React.FC = () => {
```

```
  const [state, dispatch] = useReducer(UserAuthReducer, initialState)
    return (
    <UserAuthContext.Provider value={{ state, dispatch }}>
         <div id="global-container">
              <Login />
              <Top />
              <Search />
         </div>
    </UserAuthContext.Provider>
  )
}

if (document.getElementById('app')) {
    ReactDOM.render(<App />, document.getElementById('app'));
}
```

ログイン時にstateを設置する処理を作成（Login.tsx）

以下をログイン成功時の処理と一緒に行えばOK。

```
//ログイン時の情報をres.dataという形で取得しています。
dispatch({
     type: 'setId',
     payload: res.data.id,
});
```

ログアウト時はこんな感じ。

```
dispatch({
     type: 'setOutId'
});
```

- 最初のReducerで設定した動作を行ってくれます。
- typeが ‘setId’ であれば、payloadの内容を、stateのidに挿入します。
- typeが ‘setOutId’ であれば、stateのidにnullを挿入します。