React研修2025 第2回
React中級編

2025年3月9日

Keita Ito

カスタムフック

https://ja.react.dev/learn/reusing-logic-with-custom-hooks

カスタムフックを使用しない例

```
export function Component1() {
  const [isOpen, setIsOpen] = useState(false);
  const open = () => setIsOpen(true);
  const close = () => setIsOpen(false);
  const toggle = () => setIsOpen((prev) => !prev);
  return (
    <div>
      <button onClick={open}>Open</button>
      <button onClick={close}>Close</button>
      <button onClick={toggle}>Toggle</button>
      {isOpen && <div>Opened</div>}
    </div>
  );
}
```

```jsx
export function Component2() {
  const [isOpen, setIsOpen] = useState(false);
  const open = () => setIsOpen(true);
  const close = () => setIsOpen(false);
  const toggle = () => setIsOpen((prev) => !prev);
  return (
    <div>
      <button onClick={open}>Open</button>
      <button onClick={close}>Close</button>
      <button onClick={toggle}>Toggle</button>
      {isOpen && <div>Opened</div>}
    </div>
  );
}
```

カスタムフックを使用する例

```
export function useOpen() {
  const [isOpen, setIsOpen] = useState(false);
  const open = () => setIsOpen(true);
  const close = () => setIsOpen(false);
  const toggle = () => setIsOpen((prev) => !prev);
  return { isOpen, open, close, toggle };
}
```

```jsx
export function Component1() {
  const { isOpen, open, close, toggle } = useOpen();
  return (
    <div>
      <button onClick={open}>Open</button>
      <button onClick={close}>Close</button>
      <button onClick={toggle}>Toggle</button>
      {isOpen && <div>Opened</div>}
    </div>
  );
}
```

```jsx
export function Component2() {
  const { isOpen, open, close, toggle } = useOpen();
  return (
    <div>
      <button onClick={open}>Open</button>
      <button onClick={close}>Close</button>
      <button onClick={toggle}>Toggle</button>
      {isOpen && <div>Opened</div>}
    </div>
  );
}
```

カスタムフックの利点

再利用可能なロジックをカスタムフックに切り出すことで、コードの重複を減らすことができる。

フェッチライブラリの使用

- 標準API: `fetch`
- ライブラリ: `axios`, `react-query`, `swr`

`fetch` API

```
export function Component() {
  const [data, setData] = useState(null);
  useEffect(() => {
    fetch("https://api.example.com/data")
      .then((res) => res.json())
      .then((data) => {
        setData(data);
      });
  }, []);
  if (!data) return <div>Loading...</div>;
  return <div>{data}</div>;
}
```

`fetch` APIの問題点

- エラーハンドリングが煩雑
- ローディング状態の管理が煩雑
- キャンセル処理が煩雑

axios ライブラリ

```
import axios from "axios";

export function Component() {
  const [data, setData] = useState(null);
  useEffect(() => {
    axios.get("https://api.example.com/data").then((res) => {
      setData(res.data);
    });
  }, []);
  if (!data) return <div>Loading...</div>;
  return <div>{data}</div>;
}
```

`react-query` ライブラリ

```
import { useQuery } from "react-query";

export function Component() {
  const { data, isLoading } = useQuery("data", async () => {
    const res = await fetch("https://api.example.com/data");
    return res.json();
  });
  if (isLoading) return <div>Loading...</div>;
  return <div>{data}</div>;
}
```

`react-query` ライブラリの特徴

- キャッシュ機能
- エラーハンドリング
- キャンセル処理
- クエリの再実行

swr ライブラリ

```
import useSWR from "swr";

export function Component() {
  const { data, error } = useSWR(
    "https://api.example.com/data",
    async (url) => {
      const res = await fetch(url);
      return res.json();
    }
  );
  if (error) return <div>Error</div>;
  if (!data) return <div>Loading...</div>;
  return <div>{data}</div>;
}
```

`swr` ライブラリの特徴

- キャッシュ機能
- エラーハンドリング
- キャンセル処理
- クエリの再実行

状態管理ライブラリ

`useState` で管理できる状態が複雑になった場合、状態管理ライブラリを使用する。
バケツ渡しが煩雑になる場合に有用。

- Zustand
- Recoil
- Jotai
- Redux (Toolkit)

Zustand

```
const usePersonStore = create((set) => ({
  firstName: "",
  lastName: "",
  updateFirstName: (firstName) => set(() => ({ firstName: firstName })),
  updateLastName: (lastName) => set(() => ({ lastName: lastName })),
}));

function App() {
  const firstName = usePersonStore((state) => state.firstName);
  const updateFirstName = usePersonStore((state) => state.updateFirstName);

  return (
    <main>
      <label>
        First name
        <input
          onChange={(e) => updateFirstName(e.currentTarget.value)}
          value={firstName}
        />
      </label>

      <p>
        Hello, <strong>{firstName}!</strong>
      </p>
    </main>
  );
}
```

```
export function Component2(){
  const firstName = usePersonStore((state) => state.firstName);
  const updateFirstName = usePersonStore((state)
  => state.updateFirstName);


  return (
    <div>
      <input
        type="text"
        value={firstName}
        onChange={(e) => updateFirstName(e.target.value)}
      />
      <p>Hello, {firstName}!</p>
    </div>
  );
}
```

Recoil

Deprecatedになったので、非推奨。Reactの開発元のFacebookが開発。

```
const textState = atom({
  key: "textState",
  default: "",
});

function TextInput() {
  const [text, setText] = useRecoilState(textState);

  const onChange = (event) => {
    setText(event.target.value);
  };

  return (
    <div>
      <input type="text" value={text} onChange={onChange} />
      <br />
      Echo: {text}
    </div>
  );
}
```

Jotai

```
const countAtom = atom(0);

export function App() {
  const [count, setCount] = useAtom(countAtom);

  return (
    <>
      <div>{count}</div>
      <button onClick={() => setCount((c) => c + 1)}>Increment</button>
      <button onClick={() => setCount((c) => c - 1)}>Decrement</button>
    </>
  );
}
```

Redux

```
export const counterSlice = createSlice({
  name: "counter",
  initialState: {
    value: 0,
  },
  reducers: {
    increment: (state) => {
      state.value += 1;
    },
    decrement: (state) => {
      state.value -= 1;
    },
    incrementByAmount: (state, action) => {
      state.value += action.payload;
    },
  },
});

export const { increment, decrement, incrementByAmount } = counterSlice.actions
```

```
export function Counter() {
  const count = useSelector((state) => state.counter.value);
  const dispatch = useDispatch();

  return (
    <div>
      <div>
        <button onClick={() => dispatch(increment())}>Increment</button>
        <span>{count}</span>
        <button onClick={() => dispatch(decrement())}>Decrement</button>
      </div>
    </div>
  );
}
```