Zero to Hero

Reactマスター

Level 読んだ後 100

Level 0 本書を読む前

対象

→HTML／CSSを学習したビギナー

- ✓JavaScriptの基礎の基礎
- ✓レンダリング
- ✓仮想DOM
- ✓コンポーネント
- ✓useState／useEffect
- ✓カスタムHook
- ✓イベントとfunction

React基礎から応用までカバー

三好 アキ

―目次―

第4章　Reactの基礎を学ぼう

第5章　Reactをもっと使ってみよう

第6章 さらに深くReactを知ろう

更新履歴
• 2024年2月21日：一般発売
• 2024年2月24日：内容を一部追加

はじめに

本書のねらい

Reactの確かな理解とスキルを身につけることが本書の目標です。ウェブアプリケーションの仕組み、JavaScriptの基礎知識から、Reactの基本および発展的内容まで、Reactアプリケーション開発の広い範囲を本書はカバーしています。

本書の一部には、言葉による概念の説明が続く所がありますが、本書の大部分はコードを自分の手で実際に書き、自分の目で実際に働きを確認しながら進めてもらうことを想定しています。自分の手で作ったものが動いたときの喜びは格別です。このような楽しさを読者の方に味わってもらうことが、実はReactをマスターすること以上の本書の最大のねらいといえます。

「自分でもできるんだ」という小さな成功体験と、自分に対する信頼の上昇は前へと進む原動力になり、いずれ私たちをより大きな成功へと導いてくれます。本書を読んだあと、読者の方に「もっと知りたい｜」と思ってもらえたなら、本書のねらいは成功したといえるでしょう。

本書の対象読者

本書を読むときの前提知識はHTMLとCSSだけです。JavaScriptの知識は第3章で基礎から解説します。

私はReact入門書（上記）をすでに数冊書いていますが、これらを読み終えた人に次に読んでもらいたい一冊として本書は書かれています。これらの入門書では、読者に最短距離で成功体験をひとつ作ってもらうことを一番のねらいとしていたので、Reactの基本コンセプトやコードのくわしい説明は省いていました。

一方で本書では、「オブジェクト」や「ファンクション」といった専門用語を積極的に使い、さらにコードのくわしい解説も加えてあります。上記入門書を読んで、「Reactのもっと深い理解を見つけたい」という読者に最適な一冊が本書です。

上記入門書を読まれていない方でも、本書では基礎の基礎から解説をするので安心してください。

本書のおすすめの使い方は、3回程度繰り返すことです。同じ本を何度も読むのは効率が悪く思えますが、複数の本を1回ずつ読むよりも同じ本を繰り返す方が学びは深まります。また、複数回繰り返すことを前提にすると、「一回ですべてを理解しよう」という不要なプレッシャーを自分に与えずにすみます。1回目でうまく理解できずに詰まったところでも、2回、3回と繰り返すと、不思議とスムーズに進められることに気がつくでしょう。これから1回目をはじめる読者の方は、わからないところ、頭にうまく入ってこないところがあっても、とにかく前へ前へと進むこと、とにかく最後のページまで到達することを優先させてください。最後のページまで行き、本書の全体像、つまりReactの全体像がぼんやりとでもつかめたら、どのエリアが自分は得意で、どのエリアがまだ苦手なのかが、よりはっきりと見えるようになるでしょう。

本書の構成

本書は全6章から構成されています。第1章から第3章まではReact、JavaScriptの基礎知識の紹介、そして第4、5章では実際にReactアプリ開発を進めます。最後の第6章では、それまでに触れる機会のなかったReactの発展的事項を紹介します。分量としては、React開発をする第4章と5章が本書の60%以上を占めています。以下、各章で学ぶことをすこしくわしく見ていきましょう。

第1章では、アプリケーションについての基礎知識、アプリケーションの構造やフロントエンド／バックエンドの働きについて学びます。章の後半は、Node.jsやVS Codeのインストールなど、React開発をはじめる準備です。

第2章ではReactの概要や特徴から、コンポーネント、仮想DOM、そしてレンダリングといったReactを使う上でかならず知っておきたい概念を紹介します。

第3章は、オブジェクトや定数宣言などのJavaScriptの基礎の基礎から、組み込みファンクション、ファンクションの構造など、React開発でも必須となるJavaScriptの知識を実際にコードを書きながら説明します。

第4章からは実際にReactアプリ開発を始めます。React + Viteをベースとして、レシピを検索する下記アプリを開発していきます。アクセスして「sushi」や「udon」と調べてみましょう。レシピが分からなければ、「a」や「m」と打っても、その文字を含むレシピが表示されます。ダークモードや検索履歴を表示する機能もついています。

https://react-app-meal-finder.netlify.app

第1章　アプリケーションの仕組みを知ろう + 開発の準備

この章で学ぶこと

アプリケーションの仕組み
フロントエンドの役割
バックエンドの役割
CRUD操作
ターミナルの使い方
Node.js／VS Codeのインストール
npmパッケージのバージョンの確認方法
エラーが起きた時の対処方法

この章ですること

ここではアプリケーションの構造や、フロントエンドとバックエンドの役割など、開発者が知っておくべき基礎知識を紹介します。章の最後では開発ツールの準備をします。この章の内容は私の既刊書に書かれていることと内容がかぶっているので、すでに読まれている方はスキップして第2章に進んで大丈夫です。

第2章　Reactについて知ろう

この章で学ぶこと

Reactの概要
Reactが使われる理由
コンポーネント
仮想DOM
レンダリングの種類
レンダリングのステップ
初回レンダリング
再レンダリング

この章ですること

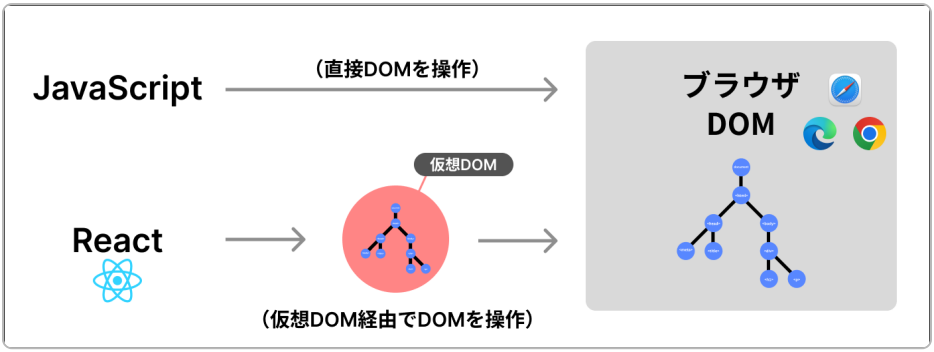
Reactの歴史やReactを支える基本コンセプト、そしてReactの実際の働きを見ていきます。読んでいて理解できないところがあっても、あまりこだわらずに前へと進んでください。第4章以降、React開発を実際に進める中で本章を読み返し、理解を深めてもらうことをねらいとしています。

まずコンポーネントです。次のコードを見てください。これはReactのコードになります。

```
const Header = () => {
    return (
        <header>ヘッダー</header>
    )
}

const Buttons = () => {
    return (
        <button>ボタン</button>
    )
}

const FormAndButton = () => {
    return (
        <form>
            <input/>
            <button>ボタン</button>
        </form>
    )
}

const App = () => {
    return (
        <div>
            <Header/>
            <Buttons/>
            <FormAndButton/>
        </div>
    )
}
```

大きく4つのかたまり、「Header」「Buttons」「FormAndButton」「App」がここには確認でき、それぞれの中にコードが数行書いてあります。

もっとよく見ると、最初の3つのかたまり（Header／Buttons／FormAndButton）が「App」の中に入っているのもわかります。実はこれはそれぞれが「Headerコンポーネント」「Buttonsコンポーネント」「FormAndButtonコンポーネント」で、その3つが「Appコンポーネント」によってまとめられているのです。さらにこれは、先ほど見たレシピアプリの画面のコンポーネントと構成が同じで、ブラウザではわかりませんが、コードの上では「Appコンポーネント」によって全体がまとめられています。



ブラウザの表示を変更するとき、Reactは「実際のDOM」をダイレクトに操作するのではなく、仮想DOMに対して操作を行います。その後、ブラウザ上に表示されている「実際のDOM」と仮想DOMを比較し（差分検出）、必要な箇所だけを変更するのです。Reactアプリが高速かつスムーズに動く理由のひとつは、このような仮想DOMの活用にあります。一方でJavaScriptやjQueryは、DOMに直接操作を加えます。なお、仮想DOMとはReact固有の概念ではなく、他のフレームワークでも使われているものです。

JSX

JSXとはReactで使われる構文です。「JavaScript XML」の略語で、一般的にReactはJSXを使って書きます。JSXはHTMLによく似た構文なので、HTMLを書いたことのある人はすぐに慣れるでしょう。JSXのコードは次のようなものです。

```
const Hello = () => {
    return (
        <div>
            <h1>こんにちは</h1>
        </div>
    )
}
```

`const` や `(` 、 `{` などが使われていますが、 `return` 横のカッコ内はHTMLコードとほぼ同じです。JSXについてくわしくは、後ほどアプリ開発をする中で確認していきましょう。

第3章　JavaScriptを知ろう

この章で学ぶこと

JavaScriptの歴史
オブジェクトと配列
データの種類
定数宣言
組み込みfunction
function
console.log()
map()
slice()
switch
演算子
分割代入
スプレッドオペレーター

この章ですること

Reactを使う上でJavaScriptの知識は必須です。しかし「JavaScript全範囲の知識が満遍なく必要」というわけでありません。本章では、React開発で必要となるJavaScriptの知識を厳選して紹介します。

前章と同じように、この章も一度読んで理解できなくて大丈夫です。次章以降、Reactを実際に開発していく中で理解できることはたくさんあるので、本章にあまり時間をかけすぎないようにしてください。しかし本章の中盤で紹介するfunctionは非常に重要で、次章以降も何度も言及するので、一度は目を通しておいてください。

JavaScriptの簡単な歴史

ブレンダン・アイクという人が、1995年に10日ほどで作りあげたものがJavaScriptです。それ以来、ブラウザで唯一動くプログラミング言語としてJavaScriptは確固たる地位を築いています。

プログラミング言語は、日本語や英語、イタリア語といった自然言語とは異なり、人工的に作られたものです。そのため新しい機能や働きが追加されたり、記法が変わったりします。JavaScriptにも誕生以来さまざまな変更が加えられてきましたが、近年のもっとも大きなアップデートは2015年にありました。これはES6とも呼ばれ、本章でこのあと見ていく定数宣言の `const` やアローfunctionなどは、この時に追加されたものです。Reactは、この2015年のアップデート時に追加された新しい機能や記法を全面的に採用しています。

JavaScriptはもともとブラウザ内でしか動きませんでしたが、JavaScriptを実行するための環境の開発が進められ、「サーバーで動くJavaScript」といわれるNode.jsなどが今では広く使われています。

「データ」について

JavaScriptを含むあらゆるプログラミング言語の本質的な役割は、「データを操作すること」です。

例えば、世界各都市の天気や気温や湿度、また総人口や出生数などは、私たちにもっとも馴染みのあるデータの典型例です。しかしデータだけが大量にあっても、無用の長物となってしまいます。データをなにか意味のあることに使うには「データを分類・整理する」、つまり「データを操作する」というプロセスが不可欠で、そこで使われるのがプログラミング言語なのです。

このことを踏まえて、データについて少し考えてみましょう。まず最初にデータの「種類」と「形」について考えます。

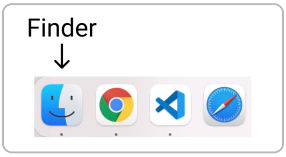
たとえば「食器」というカテゴリーの中に「種類（茶碗、カップ、どんぶり、小鉢など）と、「形（丸、三角、四角など）」があるように、「データ」にも「種類」と「形」があります。最初にJavaScriptのデータの種類と形を見て、次にそのデータを操作する方法を見ていきましょう。

	種類	**形**
食器	茶碗、カップ、どんぶり、小鉢など	丸、三角、四角など
データ	文字列、数値、真偽値、null、undefinedなど	オブジェクト、配列

組み込みfunction

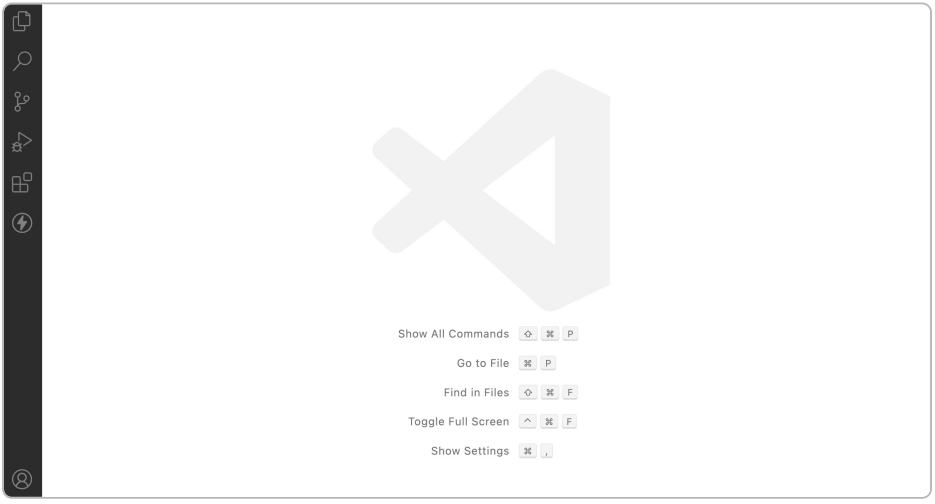
React開発でかならず使う組み込みfunctionを見ていきます。最初に必要なフォルダとファイルを用意しましょう。

場所はどこでも構いませんが、本書ではダウンロードフォルダを使います。Finderでダウンロードフォルダを開きましょう。



そこに右クリックで新規フォルダを作成し、「javascript-test」と名前をつけます。

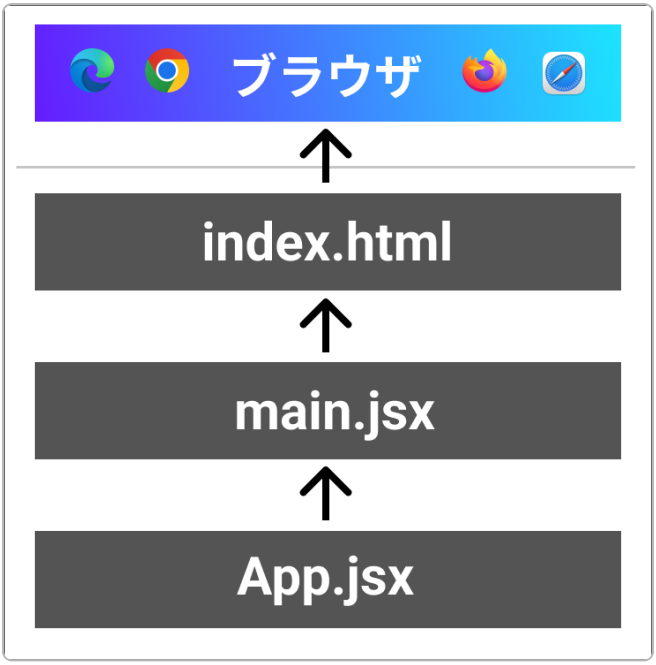
このフォルダをVS Codeで開きます。VS Codeを起動し、上部メニューバー「File」から「Open Folder...」をクリックし、いま作った「javascript-test」を開いてください。次のようになります。



第4章　Reactの基礎を学ぼう

この章で学ぶこと

React + Vite
フォルダ構成の考え方
Reactコード（コンポーネント）の構造
Reactアプリの構造
SPA
CSSの適用方法
fetch()
Promise
非同期処理（async／await）
state
state更新の方法（useState）
テンプレートリテラル
イベントの種類
イベントの書き方のバリエーション
単方向データフロー
ページの設定方法（react-router-dom）
ページの移動
Layoutコンポーネント
ページタイトルの設定



このあと開発を進める中でわかりますが、Reactで新しいページを作るとき、HTML開発時のように `about.html` や `blog.html` といった新しいHTMLファイルは作りません。アプリ内のHTMLファイルは `index.html` ひとつだけです。この `index.html` を下地として、その上にJavaScriptでページを描画・表示します。

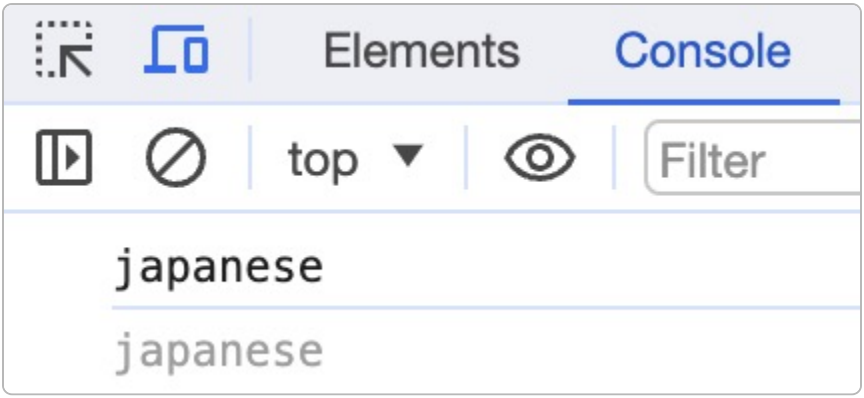
Reactアプリは「SPA（Single Page Application）」とも呼ばれますが、ここにある「Single Page」とは、アプリで使われているHTMLファイルがひとつだけであることを意味しています。このHTMLファイルを起点として、ブラウザにJavaScriptコードを読み込ませ、ページ描画やデータ操作などの機能を実現しているのです。このようにSPAでは大量のJavaScriptが使われており、それを最初にブラウザに読み込ませる必要があるため、アプリの初回表示に時間のかかる傾向があります。

以上、Reactアプリの基本構造を説明したので、再び `App.jsx` に戻ってください。上部に次のコードが見えます。

```
import "./styles/App.css"
```

このコードで `App.css` を取り込んでいるのがわかります。ここで、ReactにCSSを適用する方法を確認してみましょう。

方法は3つあります。1つ目は「インライン方式」で、タグの中に直接書き込みをします。次のコードを `<h1>` タグに追加してください。ここでは見やすいように改行をしています。



ここからわかるのは、`mealName`というstateには、`useState`の右のカッコのデータが初期データとして入っていることです。では、`mealName`のデータを他のものに変える、つまり「データの状態を変化させる」にはどうすればいいのでしょうか。ここで使うのが`setMealName`です。実際の動きを確認するので、次のコードを`<input>`タグに追加してください。本書ではコードがページ端で途切れるのを防ぐため、ここでは改行も行っています。

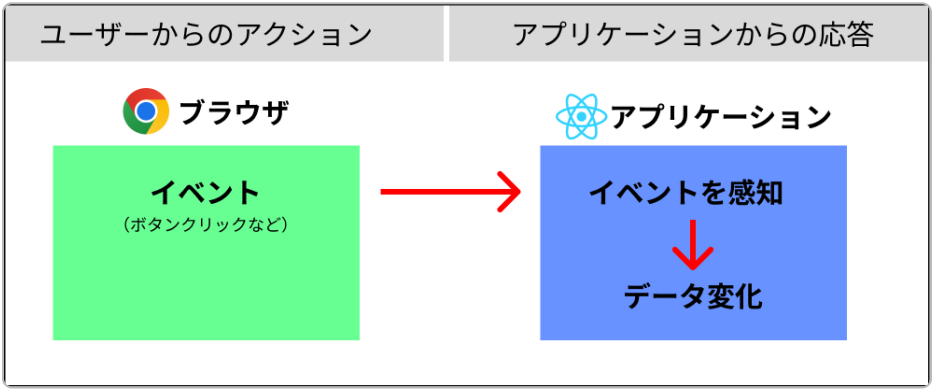
```
// Form.jsx

...

    return(
        <form>                // ⬇追加
            <input onChange={() => setMealName("America")}
                   type="text"
                   name="mealName"
                   placeholder="料理名を英語で入力"
            />
            <button>検索</button>
        </form>
    )
}

export default Form
```

`onChange`とは、「`<input>`タグに文字を入力する」というユーザーのアクション（＝イベント）を感知するJavaScriptのコードです。くわしくは後ほどすぐに説明するので、まずは保存して、ブラウザの「console」を確認してください。次のように表示されています。

以上がstateと、state更新の初歩になります。今の時点でいまいちよくわからなくても、これ以降何度かstateを使ううちに徐々に理解ができるようになります。さて、次に進む前にReactのイベントについて少し解説をしましょう。

「イベント」とは、ブラウザ上でユーザーがする行動／アクションのことです。例として「ボタン（`<button>` タグ）をクリックする」「入力フォーム（`<input>`）に文字を入力する」「マウスをホバーする／重ねる」といったことから、「スクロールする」「ブラウザのサイズを変える」といったことも「イベント」になります。

イベントが発生したとき、アプリはこれを感知する必要があります。イベントによって、アプリ内のデータの状態に変化が起きるためです。例えば先ほど見たように、「`<input>` タグに文字を入力する」というイベントにより、`mealName` 内にあったデータ `japanese` は `American` へと変化しました。



このようなイベントを適切に処理することを「イベント・ハンドリング」と呼びます。イベント・ハンドリングは、「ユーザからのアクション」と「アプリからの応答」を結びつける非常に重要な処理です。

ユーザーの起こしたイベントを感知・認識するため、JavaScriptではイベントに応じて専用のコードを多数用意しています。

```
onchange
onclick
onsubmit
onresize
onmouseover
onscroll
```

Reactでは、これらJavaScriptのコードを少しだけ高機能化したものを使います。ブラウザによって挙動に違いが生じないようにしてあり、表記も次のように大文字を使います。

第5章　Reactをもっと使ってみよう

この章で学ぶこと

stateの整理
useContext
ローディング
エラー処理
try／catch
ダークモードの仕組みと実装
CSS変数
localStorage
useEffectのレンダリング制御方法
デフォルトimportと名前付きimport
配列の扱い方
カスタムHook
条件付きレンダリング
オンラインで公開する方法（Netlify Drop）

この章ですること

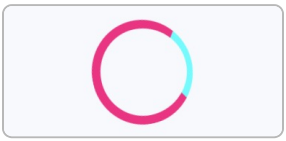
本章の最終的な目標は、ダークモードと検索履歴機能を追加してアプリを完成させ、下記リンクのようにオンラインで公開することです。

https://react-app-meal-finder.netlify.app

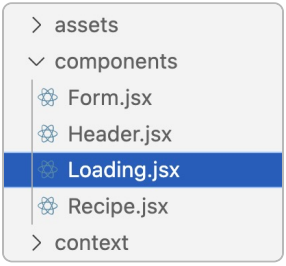
開発を本格的に始める前に、アプリ内のstateの整理と、ローディングの追加、エラー処理の設定をしましょう。

ローディングの設定

アプリでは通常、データ取得中に次のようなローディングが表示されます。



この機能を作りましょう。`Loading.jsx` を `components` フォルダに作ってください。



そこに次のコードを打ちます。

```
// Loading.jsx

const Loading = () => {
    return (
        <div className="loading"></div>
    )
}

export default Loading
```

なお前章でも触れましたが、アローfunctionにおいて `return` 以降のコードが1行の場合は、`return` とカッコを省略して次のようにも書けます。

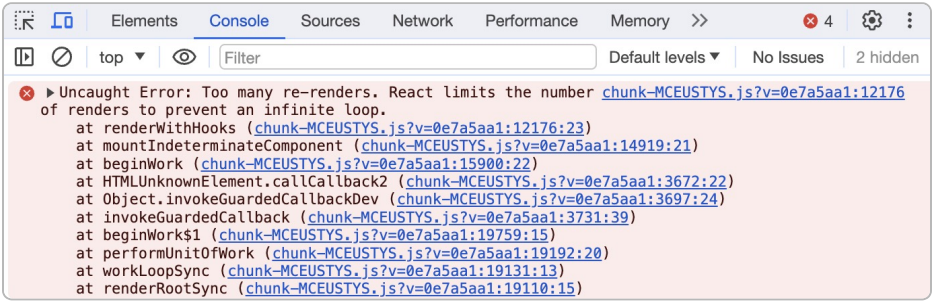
```
// Loading.jsx

const Loading = () => <div className="loading"></div>

export default Loading
```

ここまでで、Reactコンポーネントとは実はJavaScriptのfunctionであると、すでに気がついた人もいると思います。これらReactコンポーネント／functionのさまざまな記法は、最終章でまとめてあります。

localStorageから取得したデータを `localTheme` に格納し、それでstateの `theme` を更新しています。なおここでは、あえて `localTheme` に格納しないで下記のように書いても結果は同じですが、今後のコードの展開上、見通しが悪くなるので定数を使っています。また更新前に `localTheme` があるかどうかも確認したいので `&&` を書いてあります。

```
setTheme(window.localStorage.getItem("color-mode"))
```

これで、アプリが表示されたときにlocalStorageのデータで `theme` が更新されるはずです。保存してブラウザで確認します。すると何も表示がされません。「console」を開くと次のように表示されています。



エラーメッセージに「Too many re-renders」、つまり「レンダリングが必要以上に起きている」とエラーが出ています。「infinite loop／無限ループ」という文字も見えます。なぜここでレンダリングが必要以上に起きているのかを理解するために、第2章で紹介した再レンダリングの起きる条件の1つ目を思い出しましょう。

再レンダリングのきっかけ		再レンダリングされるコンポーネント
コンポーネント内のstateの変更	→	そのコンポーネントが再レンダリング

再レンダリングはstateが更新されると起こります。つまり、いまLayoutコンポーネントで起きているのは、「 `setTheme` が実行されて `theme` が更新される」ということをきっかけにして再レンダリングが始まり、それが無限に繰り返されているのです。

しかし `setTheme` には、最初の一回だけ動いてもらえれば十分です。ここで使うのが `useEffect` になります。次のコードを追加しましょう。

検索履歴の開発 5（条件付きレンダリング）

最後にアプリにある「検索」「モード」「削除」の3つのボタンを整理しましょう。完成見本を見ると、「削除」ボタンは「モード」ボタンの横にあります。

「モード」ボタンはLayoutコンポーネント内にあるので、ここに「削除」ボタンを移動すれば、完成見本と同じ配置になります。その方法でも問題はありませんが、ここでは学習のため、同じコンポーネントを使い回す方法と、コンポーネントを条件に応じて出し分ける方法（条件付きレンダリング）を紹介します。`components` に `Button.jsx` を作ってください。

次のコードを打ちます。

```
// Button.jsx

const SingleButton = () => {
    return(
        <button onClick={}>
            <img src={} alt="icon-image"/>
        </button>
    )
}
```

これがButtonコンポーネントになります。アプリ内の複数箇所で同じデザインのボタンが使われる場合、このように一つのコンポーネントとして作り、それを使い回すことで、例えばデザインの変更など

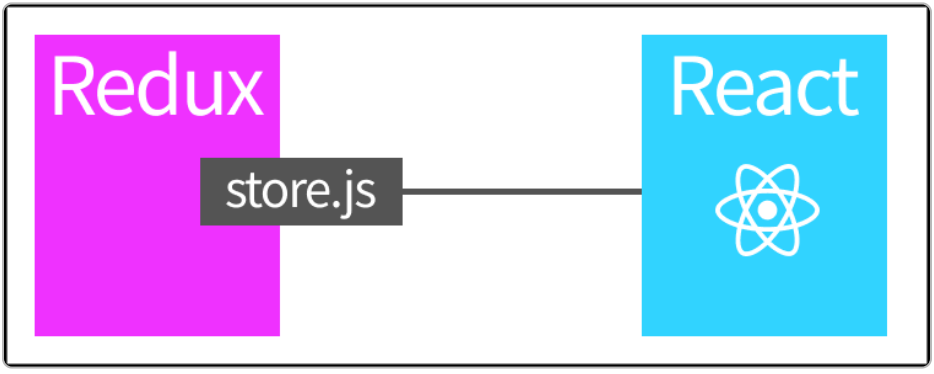
```
export default App
```

保存してください。ブラウザで確認すると、`useReducer`の時に最初に作ったものと同じだとわかります。ここにReduxを導入していきましょう。最初に必要なパッケージをインストールします。

```
npm install @reduxjs/toolkit react-redux
```

1つ目がRedux Toolkit、2つ目がReduxとReactをつなげるパッケージです。ReduxはReact以外でも使える汎用的なstate管理ツールなので、Reactと使うときは二者をつなげるパッケージ`react-redux`を使います。次に、ファイルの整理がしやすいように、Redux関連のファイルを収めるフォルダ`redux`を`src`の中に作りましょう。さらにその中に`store.js`を作ってください。

この`store.js`がRedux側のファイルをまとめ、そしてReactとつなげる接点となります。



著者について

著者: 三好アキ
表紙デザイン: 三好アキ

これまで欧州数ヶ国に滞在し、海外クライアントの案件を多く手がけてきたため、最新のウェブテクノロジーや日本語の情報が少ない静的サイトジェネレーター、Jamstack、ヘッドレスCMSなどの最新情報に精通。最新の知見を活かしながら、ウェブ関連分野の課題解決を行う。

ウェブサイト: https://monotein.com

Twitter: https://twitter.com/monotein_

note: https://note.com/monotein

ビギナー向けの無料メルマガ登録：https://monotein.com/register-newsletter

初版発行日: 2024年2月21日