



React

全4回で学ぶReact.js入門講座 1回目

2025/6/10

Kazuma SEKIGUCHI

自己紹介



関口和真

株式会社コムセントCTO

Webシステム開発、スマートフォンアプリ制作、
サーバー構築、運用など

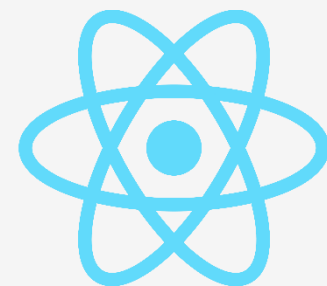
スマートフォンを使ったアプリの制作

サーバーサイドシステムの作成

フロントエンド部分の作成

目標

- Reactはどのようなものであるかを知る
- Reactの基本的な構文を知る
- ReactでのCSS扱いを把握する
- State管理の方法を習得する
- カスタムフックを知る



今回のアジェンダ

- Reactとは何か
- Reactの環境設定
- JSX記法の使い方
- コンポーネントの使い方
- イベントの扱い方

ReactとjQueryに関する質問と回答

- jQueryとの関係性

- 前提として、ReactはjQueryとの相性が悪い
- Reactは仮想DOMを用いて構造を管理しているが、jQueryはDOM自体を変更するように作られているため、jQueryで与えた変更は基本的にReactでは検知できない
- jQueryと併用すると、Reactが正常に動作しない可能性が高い

```
<body>
<main>
<h1>jQuery</h1>
<p>text</p>
```

:



jQueryはDOMそのものを
変化させる

```
<body>
<main>
<h1>jQuery</h1>
<p><img src="" alt=""></p>
```

:

Reactに関する質問と回答

- Reactは難しい
 - 文法を覚えたり、ツールを覚えたりする部分は新しいことを覚える必要があるため、難しく感じるはず
 - 名称がいろいろと出てくるので混乱もしやすい
 - React自体の難しさよりも、JSを知らないために挫折する人の方が多い印象
- ES2018以降を頻繁に利用するため、ES2018などの文法も知っておくことが求められる

使用するソフトウェア



- 今回利用するのはVisual Studio Codeとプラグイン
- JavaScript実行環境としてNode (バージョン20以上)
- ブラウザーはGoogleChrome

- TypeScriptは使わない
- Next.jsは取り扱いわない

Reactとは何か

- 現在のWebはユーザーの操作に合わせて描画が変わる構成が一般的になっている
- Webの本来はHTML+CSSのため、操作に合わせて描画が変わることはほとんどない
- ユーザーの操作に合わせてDOMを変化させることで、ユーザーの動きに合わせて描画を変えることが必要
 - DOMを変化させるためにはJavaScriptを使う

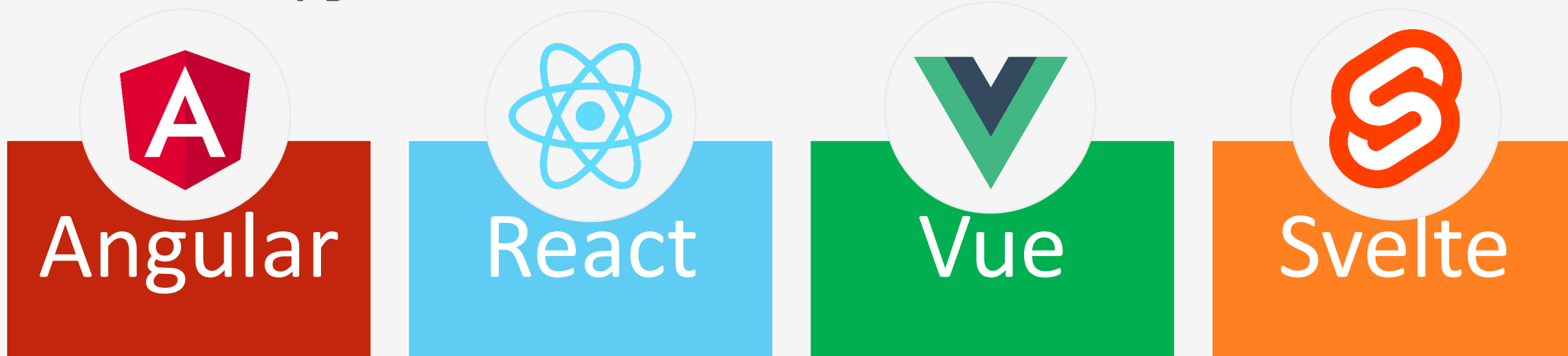
Reactとは何か

- JSを用いることで変化させることはできる
- 画面を再描画することなく、変化させることができる
 - GoogleMapの登場
- JavaScriptによる操作性向上を目指した開発が進む
 - jQueryなどのライブラリーが登場
 - JavaScriptをより簡単に扱うためのライブラリーが登場
- JavaScript依存が強まると、書くコードの量が膨大になってくる

Reactとは何か

- コードが複雑化するほど問題が起きやすくなってきた
 - ある程度統一的に記述できる仕組みが必要
 - 一般的にコードが長いほど問題が起きやすくなる
- 同じコードを書かなくて済む仕組みがあれば便利
- JS自体を機能ごとに複数のファイルに分けて記述していく方が問題が少ない
 - それぞれのファイルの行数を減らすことができる
- フロントエンドを司るライブラリーが登場

Reactとは何か



- 現在はReactが用いられるケースが多い
- Svelteは後発のライブラリーで簡単に記述できる、
ということからシェアが伸びつつある
 - Angularはシェアが落ちつつある
- 日本では比較的Vueが強い印象

Reactとは何か

- Meta社（Facebook）が中心となって開発を進めている
 - 現在の最新バージョンは19.1とバージョンアップも盛ん
 - 2024年春にReact19が正式にリリースされている
- SPA（Single Page Application）が効率的に作成可能
 - SPA:ページが切り替わることなく、画面描画している内容だけが変化するタイプのWebページの作り
- JavaScriptの変数(のようなもの)を変更するだけで勝手にHTML側も連動して変わる
 - データバインディング
- 多くの開発者が使用して実績が積みあがっているので、情報が豊富

Reactとは何か

- 仮想DOMを使って描画を行う
 - DOM=ブラウザーが実際に描画するHTMLの構造
 - 仮想DOM=JSのオブジェクトで書かれた仮想的なDOMのこと。
Reactではここを変更して、実際のDOMとの差分を検知し、違うところだけDOMを書き換えることをしている
- パフォーマンス向上に役立つ
- ページ遷移をJSによる画面の書き換えで表現している
 - ページ遷移を擬似的に扱うこともできる

Reactの仲間

- React Native

- React自体はWeb向けUI構築ライブラリ
- React NativeはReactをベースとしたスマートフォンアプリ作成のためのフレームワーク
 - 1つ記述するだけでiOSとAndroidで動作するアプリを作成できる
- 最近はFlutterなどに押され気味

- Next.js

- Reactをベースとしたサーバーサイドレンダリングや静的Webサイト生成を行うためのフレームワーク
- Reactだけで仕組みも作りたい場合や、サイト全体を構成していきいたい場合などに利用される

NEXT.js

Reactとは何か（デメリット）

- JSXという独特の記法をするため、通常のJSとかなり違う書き方をする
- マイナーバージョンの違いでも動きが変わることがある
 - 書籍などを参考にしていると、マレに挙動が異なることがある
- 比較的覚える用語が多い、かつ独特
- JavaScriptの記述が簡単になるわけではない
 - JS依存になるため、それなりに複雑なJS構文を知っておく必要は生じる

現在のReactでの開発

- React単体での開発はあまり行われていない
 - 別途フレームワークやReactと組み合わせて利用するライブラリーなどと組み合わせて使うことが一般的
 - 有名なところでは、React RouterやJotai、Recoilなど
 - ライブラリーと組み合わせることで実質的なSPA開発が可能
- Viteをバンドルソフトウェア及び開発サーバーとして利用することが多い
 - Viteは高速に動作するため、効率的な開発が可能



Reactを書いてみる

- ReactのPlaygroundなどで簡単に試すことも可能
 - [CodeSandbox](#)
 - [PlayCode.io](#)
- 本格的に書くなら、Node環境を使いつつ、Reactを記述する環境を構築した方が良い
 - 本格的に構築するならViteを使ってReactを構築
 - 最終的に本番ビルドを作成すると、複数のコンポーネントなどをまとめて1つのJSファイルにまとめることが可能
 - 開発時は分割にロードして高速性を維持する

Node.js



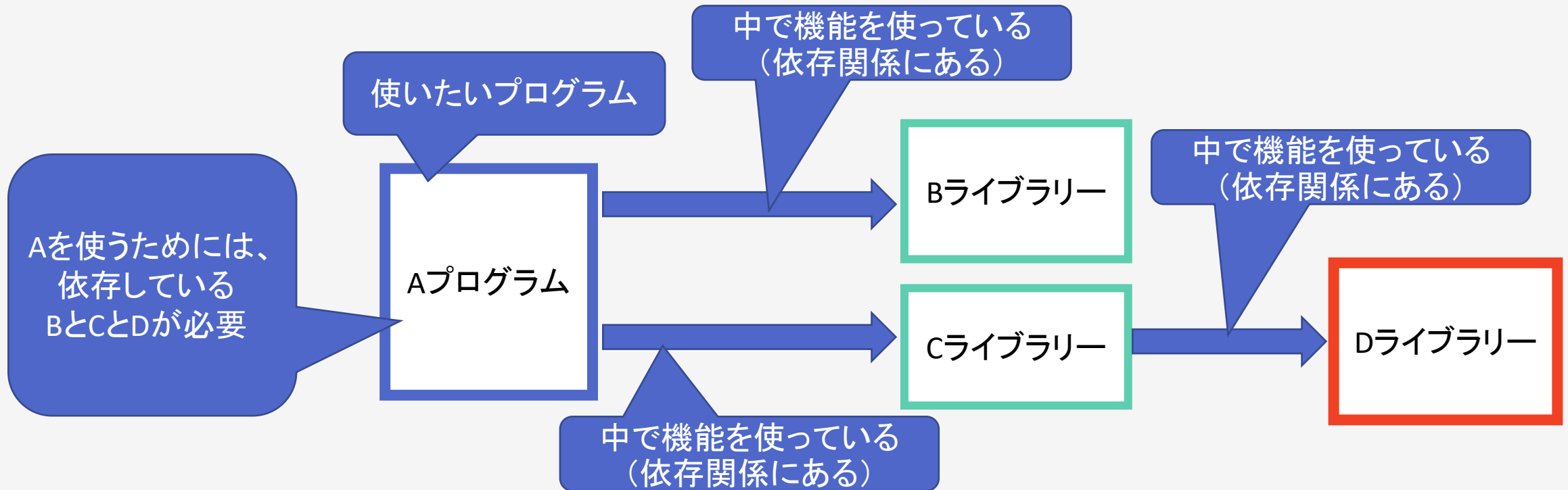
- JavaScript実行環境
 - JS自体はブラウザーで実行可能
 - HTMLやCSSの表示機能を持たず、JSだけを実行する
- JSの普及によりJSだけで実行可能なプログラムを記述したいという要望
 - ブラウザーみたいな表示機能は不要
 - ファイルへの書き込み、読み込みなどの機能が必要
- ChromeのJSエンジンを利用して機能を追加したもの
 - これだけでJSプログラムの実行が可能

Node.js

- フロントエンドの世界ではほぼ欠かせない
 - SassからCSSへの変換、TypeScriptからJSへの変換、バンドルツールの実行などなど、多様なプログラムが提供されている
 - プログラムと言ってもHTMLやCSSのような画面表示機能は持たず、GUIは無いため、全てコマンドで操作する
- 現在は、同様の機能を持つソフトとしてNode.js以外にDenoとかBunなどが登場している

NPM(Node Package Manager)

- Node上で動作するアプリは大量
 - それぞれが依存関係を有していることがある
 - 依存＝他のファイルが無いと動作しない関係



NPM(Node Package Manager)

- 使いたいプログラムをインストールしても使えない
 - 依存関係を解決する必要がある
 - Dをインストールし、Cをインストールし、Bをインストールして、初めてAをインストールできる
 - 中を調べてどういうライブラリーが必要かを調べる必要がある
 - 物凄い大変だし、手間も掛かる
- NPMを使うことで依存関係を自動的に処理してくれる
 - NPM経由でAをインストールしたい、と指定すると、勝手にDをインストールし、Cをインストールし、Bをインストールして、Aをインストールしてくれる

NPM(Node Package Manager)

- パッケージマネージャー
 - 依存関係を解消して、プログラムを使える状態にしてくれるソフト（機能）
 - 不足しているライブラリーやプログラムはネットからダウンロードしてくる
- Linuxなどでは比較的良く採用されている方法
- 実際Windowsなどでもインストール作業は同じようなことをしているが、元々ソフト自体が全ての依存しているファイルをインストーラーが持っているという違いはある

NPXコマンド

- NPMで必要なパッケージをダウンロードした上で実行するためのコマンドの1つがNPX

npx 実行したい内容

- npxだけでNPMコマンドが走り、自動的に必要なパッケージをダウンロードした上で実行してくれるので便利
- Viteで構築するときは、内部的にnpxを利用している

NPMを使ってReactをインストール

- NPMを使う場合、利用するフォルダーを指定する
- GUIは無いため、ターミナルなどから利用する
 - (Windows11の場合) 「スタート」→「すべてのアプリ」→「ターミナル」を選択
 - (Windows10の場合) 「スタート」→「ターミナル」を選択。
「ターミナル」が見つからない場合、「Windowsシステムツール」から「コマンドプロンプト」を選択
 - (macの場合) Finderから「アプリケーション」→「ユーティリティ」→「ターミナル」を選択

NPMを使ってReactをインストール

- フォルダを移動する
 - cdと入力し半角スペースを空けて、フォルダへのパスを記入して、Enter (Return) キー
 - フォルダをドラッグ&ドロップする方が最初は分かりやすいかも
 - mac環境の場合、ドラッグ&ドロップしたら半角スペースが空くので削除した方が無難

現在開いているフォルダー

>のあとに記入

現在開いているフォルダー
(macの場合一部省略される)

\$のあとに記入



```
C:¥Users¥user¥desktop>|
```



```
macbook:desktop macuser $|
```

ユーザー名



```
C:¥Users¥user¥desktop>cd 移動したいフォルダーのパス
```

NPMを使ってReactをインストール

- フォルダを移動したら `npm create vite@latest app -- --template react` と入力



```
C:\Users\user\Desktop\react>npm create vite@latest app -- --template react
```

- appの名前は自由
 - ここに作業環境が作成される
 - 環境構築まで結構時間が掛かる（インターネット回線速度に依存するが、2～3分程度）
- この段階で、`package.json`がダウンロードされる
 - 中にReactのバージョンが書かれているので確認しておく

package.json

- dependenciesにReactが書かれる
 - バージョンが19以上になっていることを確認する
- もしバージョンを指定してインストールしたいなら

```
npm create vite@latest app -- --template react
npm install react@18 react-dom@18
```

とテンプレートを入れてからnpm installコマンドで明示してインストールする

```
6  "scripts": {
7    "dev": "vite",
8    "build": "vite build",
9    "lint": "eslint .",
10   "preview": "vite preview"
11 },
12 "dependencies": {
13   "react": "^19.1.0",
14   "react-dom": "^19.1.0"
15 },
16 "devDependencies": {
17   "@eslint/js": "^9.25.0",
```

NPMを使ってReactをインストール

```
PowerShell 7
PS C:\Users\kazum\Desktop\react19app> npm create vite@latest app --
--template react

> npx
> create-vite app --template react

|
◇ Scaffolding project in C:\Users\kazum\Desktop\react19app\app...
|
Done. Now run:
  cd app
  npm install
  npm run dev

PS C:\Users\kazum\Desktop\react19app> |
```

Done
が表示されれば初期インストールは完了

開発時はここに書かれているコマンドを
実行

NPXを使ってReactをインストール

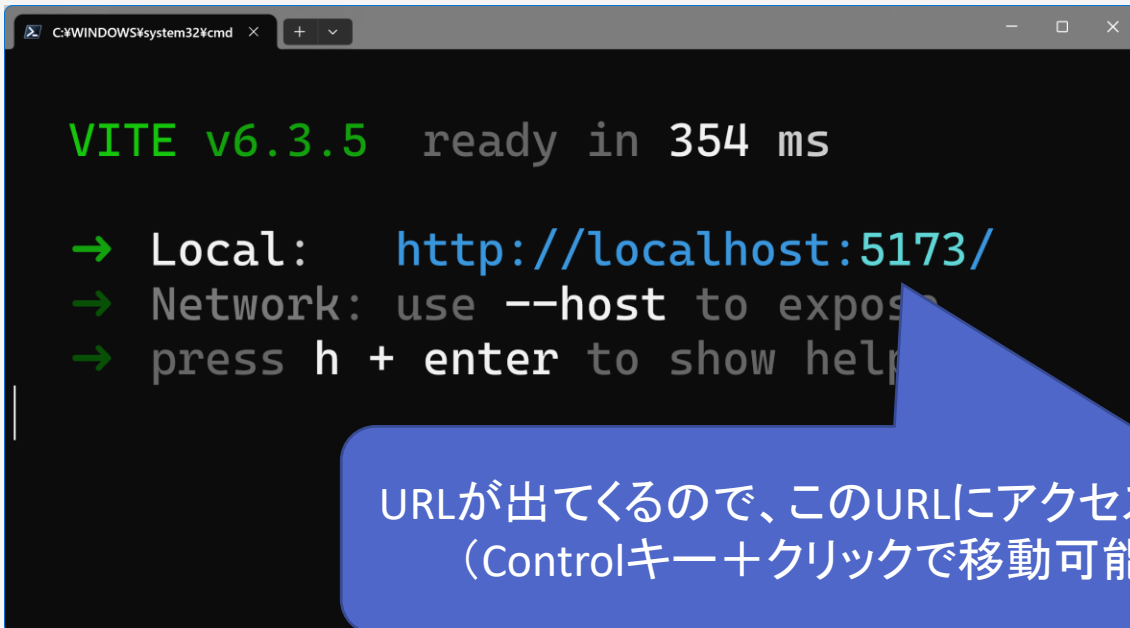
- 出てきたコマンドの通りに入力する

 C:¥Users¥user¥desktop¥react>cd app

 C:¥Users¥user¥desktop¥react¥app>npm start

npxではない点注意

 C:¥Users¥user¥desktop¥react¥app>npm run dev



```
C:\WINDOWS\system32\cmd
VITE v6.3.5 ready in 354 ms
→ Local:  http://localhost:5173/
→ Network: use --host to expose
→ press h + enter to show help
```

URLが出てくるので、このURLにアクセスする
(Controlキー+クリックで移動可能)



このような画面が出てくればOK

起動後のCLI

- ViteでReactを起動させたら後は放置でOK
 - 止めてしまうとブラウザーで表示されなくなる
 - バックグラウンドに放置しておけばOK
 - 開発作業を止めるときはそのままターミナルを閉じてしまえばViteが止まる

jsxを記述

- 構築した段階でsrcフォルダーに最初の表示に利用しているファイル群が生成されているので削除する
 - main.jsxを読み込んで表示をしている
- main.jsxに必要事項を記述すれば表示される
 - main.jsxを保存すれば自動的にブラウザーの表示は更新される

jsxを記述

```
1 import ReactDOMClient from "react-dom/client";
2
3 const App = () =>{
4   return (
5     <>
6       <h1>Reactでの表示</h1>
7       <p>ReactのApp.jsxから表示してます</p>
8     </>
9   );
10 }
11
12 const container = document.getElementById('root');
13 const root = ReactDOMClient.createRoot(container);
14 root.render(<App />);
15
```

react-dom/clientライブラリーから
ReactDOMClient
モジュールをimportする

画面に描画したい内容をreturnの
値として記述する
複数行の場合は、()で括る

returnでは1つのトップレベル要素の中に収める必要が
あるため、タグを使いたくない場合は、<></>を記述

React17では書き方が異なったので
注意

public/index.html内にdiv id="root"が存在するが、そこにconst
Appで記述しreturnした内容を描画するように指定する

jsxを記述

- returnで描画したい内容を指定する
 - 1つのタグに囲まれている必要がある点に注意
 - タグで囲みたくなければ、`<></>`の空タグ記述で囲む
- return内では`{}`で括弧することでJSを記述することが可能
 - `<script>`タグなどは記述不要
- index.jsにすべてを記述すると肥大化してしまうので、ファイルを分けて記述することが可能
 - コンポーネント

コンポーネント

- 拡張子をjsxとして保存する
- const appなどの前にexportを付けた上で、別ファイルとして保存する
 - main.jsxでは別ファイルをimportして読む込むことで、同一ファイルに記述しているかのように利用することが可能

```
1 import ReactDOMClient from "react-dom/client";
2 import { App } from "./App";
3
4
5 const container = document.getElementById('root');
6 const root = ReactDOMClient.createRoot(container);
7 root.render(<App />);
```

main.jsx

別ファイルApp.jsxに記述したconst Appの内容を読み込む
拡張子の記述は不要

イベント

- JSではonclickなどでイベントを利用することが可能
 - Reactでは通常addEventListenerは使用しない
 - Reactではキャメルケースで記述することでイベントが利用可能
 - onclick→onClick, onchange→onChange

```
1 export const App = () =>{  
2   const onclickButton = ()=>{  
3     alert('click it');  
4   }  
5  
6   return (  
7     <>  
8       <h1>Reactでの表示</h1>  
9       <p>ReactのApp.jsxから表示します</p>  
10      <button onClick={onclickButton}>クリック</button>  
11    </>  
12  );  
13 }
```

クリックされたときに実行
される関数を作成

onClick時に実行したいも
のを{}内に記述

イベント

- Reactの場合、addEventListenerを直接使うのではなく、SyntheticEventという仕組みを使っている
 - React独自のイベントラッパーでブラウザの標準イベントをラップし、どのブラウザでも統一的に利用できるようにしている
 - イベントの挙動やプロパティの差異を吸収している
- 仮想DOMとの整合性を取る
 - addEventListenerで直接DOMに付与すると、差分が生じる可能性が出てきてしまう
 - ゆえにaddEventListenerを使ってしまうと、Reactの動作に影響が出る

ありがとうございました。
また次回。