

React実習

01. React概要

株式会社ジードライブ

今回学ぶこと

- Reactの概要
- Reactアプリの開発環境
- Reactプロジェクトの構成
- Reactアプリの作成と実行の手順

Reactの概要

Reactとは

- Meta (Facebook)社が開発するUIライブラリ
 - <https://ja.react.dev/>
 - UIはユーザーインターフェースの略で、ここではWebアプリの画面を構成する部品(何かしらの機能をもつことができる)を指す
- SPA (Single Page Application) が作りやすい
 - 「シングルページ」という言葉が使われているが、複数のURLで構成されるWebアプリを作成することもできる
 - SPAでは、URLの書き換えをイベントの一つと見なし、JavaScriptでそれを検知して画面を書き換える
 - ⇒ サーバーに対して、新たにHTMLを要求するわけではない
 - ⇒ ユーザー視点では、複数ページのWebアプリに見える
- 宣言的なビュー、コンポーネントベース、Webアプリ開発以外にも利用できる、といった特徴がある

宣言的なビュー(画面構築)

- 通常のJavaScriptや旧来のライブラリでは、画面を書き換えるために、以下のような手順を踏む
 - ① 画面に表示させるデータを、何かしらの処理(合計金額の計算、Web APIからのデータ取得など)を通じて生成する
 - ② 書き換えるHTMLの要素を選択する
 - ③ 選択したHTML要素に対して、処理結果を反映させる
- Reactを使った開発では、開発者は①の処理だけ記述する
 - 開発者はどのようなデータを表示したいかを宣言するだけでよい
 - HTMLへの反映(DOM操作)は、Reactが行う
 - ⇒ 仮想DOMというしくみを使い、効率的にDOMを操作する

DOM操作

コンポーネントベース

- コンポーネントは、ページ(画面)を構成する部品
 - 小さなコンポーネント(見出し、画像、ボタンなど)が集まって、大きなコンポーネント(ヘッダ、フッタ、サイドバー、ナビゲーションなど)を形成し、大きなコンポーネントが集まって、一つのページを形成する
 - コンポーネントは再利用でき、その組み合わせによって、様々なページを構築することができる
- コンポーネントは、見た目と機能で構成される
 - 見た目は、JSXと呼ばれるHTMLに似た構文で記述する
 - 他のコンポーネントに影響を与えない、コンポーネントごとのスタイルを適用することができる
 - コンポーネントには、表示させるデータの状態を管理する機能やイベントに対応する機能などを持たせることができる

Webアプリ以外の利用場面

- React Native
 - iOSやAndroidで動くアプリの開発
- React 360
 - VR空間におけるUIの開発
- Electron + React
 - Reactとデスクトップアプリ用フレームワーク「Electron」を組み合わせて、開発を行うことができる

Learn once, write anywhere.
一度学習すれば、どこでも使える

Reactアプリの開発

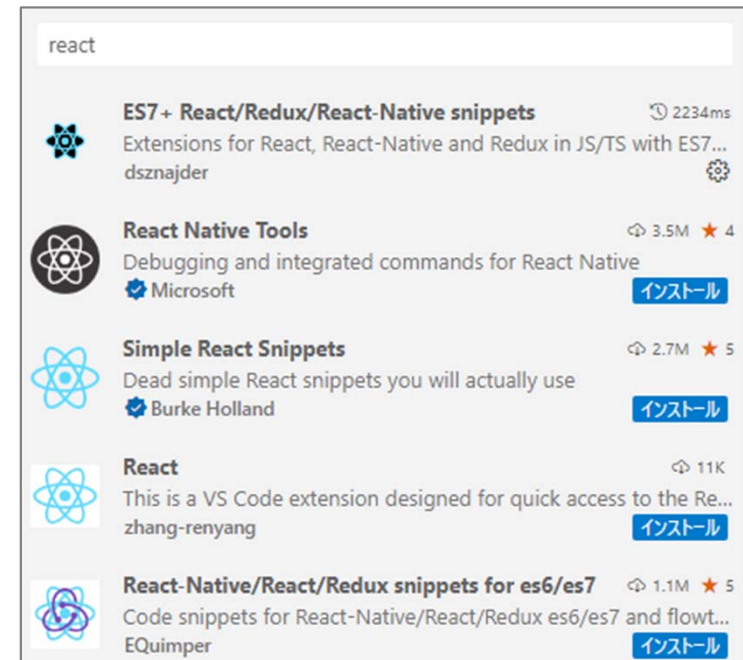
Reactアプリの開発に必要な物

- Node.js
 - 開発用サーバーの起動やビルドなどに必要
- Webブラウザ
 - React用の拡張機能を入れておくと便利
- テキストエディタ
 - VS Codeは、React用の拡張機能が充実している



React Developer Tools

おすすめ



Reactプロジェクトの作成

- Reactプロジェクトを作成する場合には、Viteを使用するとよい
 - <https://ja.vite.dev/>
- 以下のコマンドでベースとなるファイルやフォルダが生成され、必要なパッケージがインストールされる

```
npm create vite@latest プロジェクト名
```

例: プロジェクト用にmyappディレクトリを作成する

```
npm create vite@latest myapp
```

例: カレントディレクトリをプロジェクト用ディレクトリとする

```
npm create vite@latest .
```

Reactプロジェクトの作成

- 初回はcreate-viteのインストールを求められる
⇒ y を入力する

```
$ npm create vite@latest myapp
Need to install the following packages:
create-vite@6.1.1
Ok to proceed? (y) y
```

- 矢印キーを使い、ReactとJavaScriptを選択する

```
? Select a framework: » -
  Vanilla
  Vue
> React
  Preact
  Lit
  Svelte
  Solid
  Qwik
  Angular
  Others
```

```
✓ Select a framework: » React
? Select a variant: » - Use arrow keys
  TypeScript
  TypeScript + SWC
> JavaScript
  JavaScript + SWC
  React Router v7 ↑
```

Reactアプリの起動

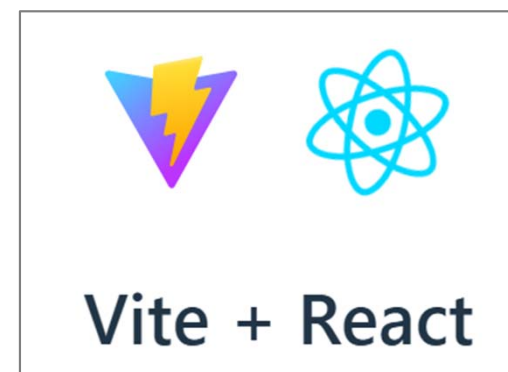
- 以下のコマンドを実行し、必要なパッケージのインストールを完了させる

```
cd myapp (必要に応じて、cdコマンドでディレクトリを変更)  
npm install
```

- 以下のコマンドでアプリを起動する

```
npm run dev
```

- アプリは以下のURLからアクセスできる
<http://localhost:5173>



表示される画面

Reactアプリの停止

- Reactアプリを実行したターミナル内で `Ctrl + C` を押すとReactアプリは停止する
 - [バッチジョブを終了しますか (Y/N) ?] と表示される場合は、[Y]キーを押す



```
問題 出力 デバッグ コンソール ターミナル

Compiled successfully!

You can now view create-first-app in the browser.

Local:      http://localhost:3000
On Your Network:  http://192.168.56.102:3000

Note that the development build is not optimized.
To create a production build, use npm run build.

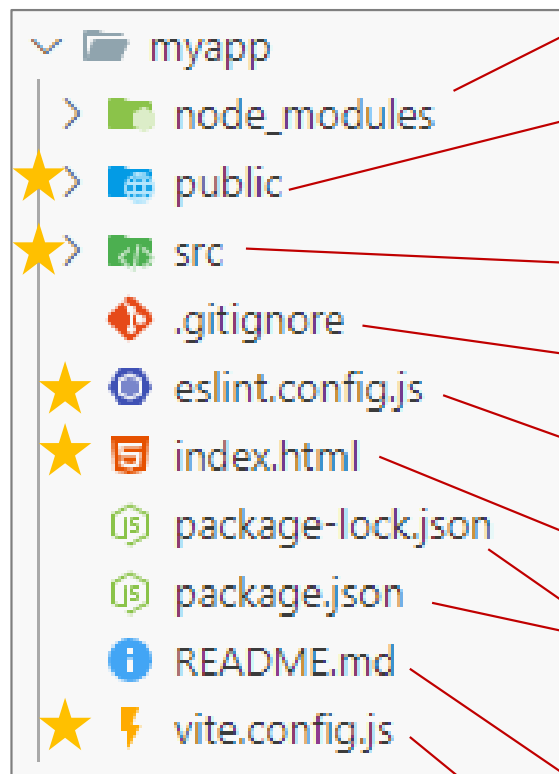
webpack compiled successfully
バッチ ジョブを終了しますか (Y/N)? Y
PS D:\Users\zdis56\study\React-Kyozaip10-react-outline\create-first-app> □
```

練習

- 練習01-1

Reactプロジェクトの構成

- ルートディレクトリの内容



ローカルインストールしたパッケージが格納される

ブラウザから直接アクセスできるフォルダ
⇒ CSSや画像などを配置

Reactアプリを構成するソースファイルを配置

Gitの管理から除外したいファイルを記す

Eslintの設定

ReactアプリをマウントするためのHTML

インストールしたパッケージの
情報などが記されている

アプリの概要を記述する

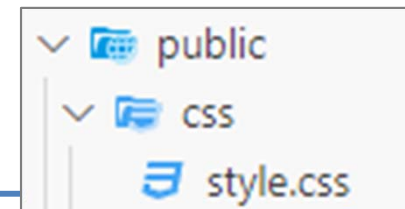
Viteの設定ファイル

publicフォルダ

- publicフォルダには、CSS、画像、フォントなどの静的ファイルを配置する
 - index.htmlから参照するファイルを配置

「/public」は記述不要

```
<link rel="stylesheet" href="/css/style.css" />
```



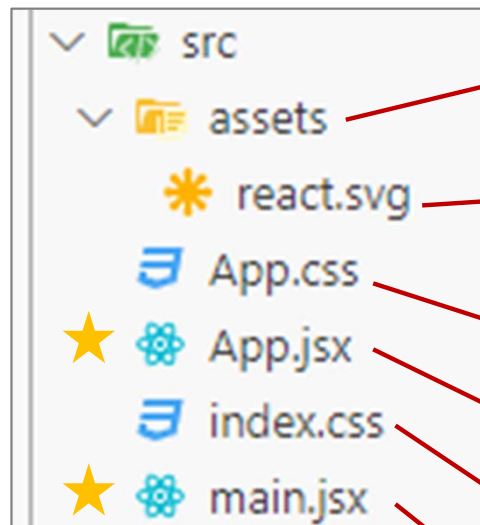
- public内のvite.svgは削除して構わない



削除可能

srcフォルダ

- srcフォルダの内容



src内のJavaScriptから参照するための静的ファイルを配置(削除可能)

App.jsから参照されているReactのアイコン画像(削除可能)

App.jsから参照されているCSS(削除可能)

全体の親となるコンポーネントファイル

main.jsから参照されているCSS(削除可能)

Reactアプリのベースとなるファイル

index.html / main.jsx

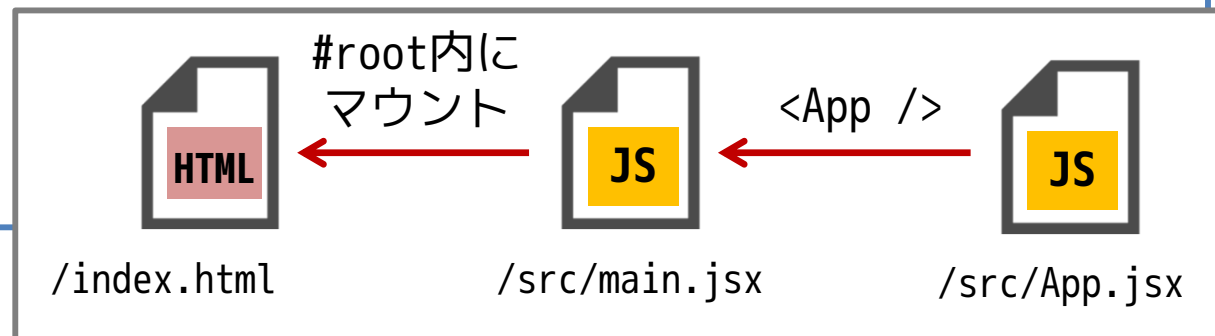
- main.jsx内では、index.htmlの<div id="root"></div>内にAppコンポーネントをマウントする記述がされている

index.html

```
<body>
  <div id="root"></div>
  <script type="module" src="/src/main.jsx"></script>
</body>
```

src/main.jsx

```
createRoot(document.getElementById('root')).render(
  <StrictMode>
    <App />
  </StrictMode>,
)
```

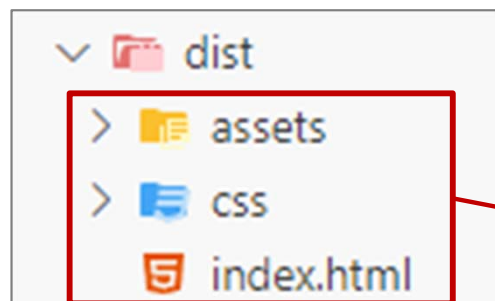


デプロイ用ファイルのビルド

- Webサーバーにデプロイするためのファイルを生成するには、以下のコマンドを実行する

```
npm run build
```

- distフォルダにデプロイ用のパッケージが出力されるので、このフォルダの中身をWebサーバーに配備する



Webサーバーに配備

vite.config.js

- vite.config.jsではアプリケーション起動時のポート番号やビルドファイルの出力先などの指定ができる
 - <https://vite.dev/config/>

vite.config.jsへの追記例

```
export default defineConfig({
  plugins: [react()],
  base: "/myapp",
  server: {
    port: 3000
  },
  build: {
    outDir: "app"
  }
})
```

URLが http://localhost:3000/myapp になる

ビルドファイルの出力先がappフォルダになる

練習

- 練習01-2

補足: create-react-app

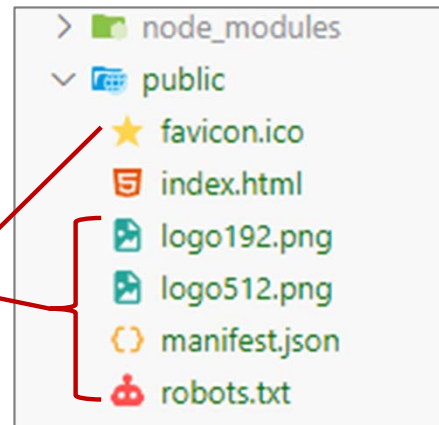
create-react-app

- Reactプロジェクト作成に際しては、以下のコマンドを使用することも可能 (ただし現在は非推奨)

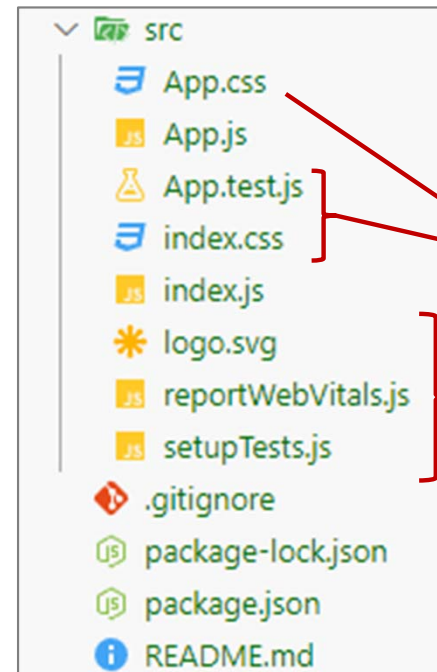
npx create-react-app プロジェクト名

- デフォルトで生成されるファイルが Vite よりも多い

削除可能



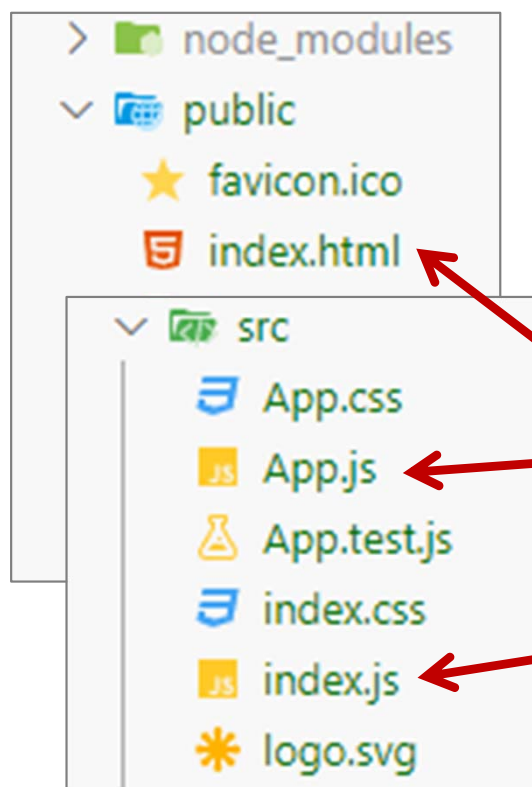
削除可能



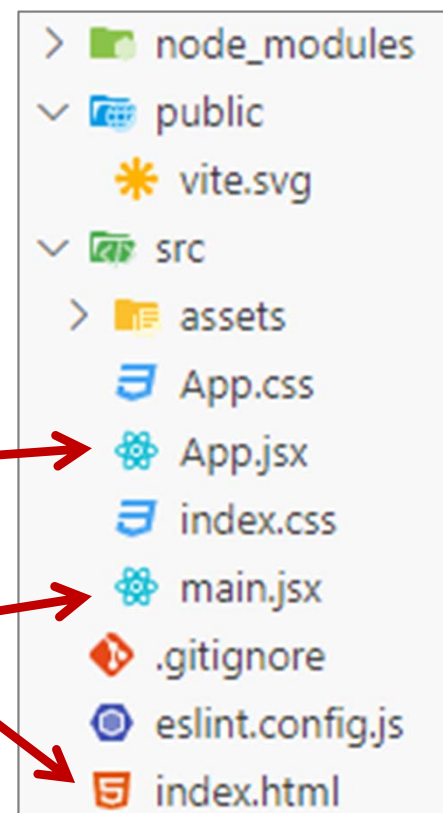
create-react-appとVite

- プロジェクト内のファイルは以下のように対応する
 - index.js が Vite の main.jsx に相当する

create-react-app



Vite

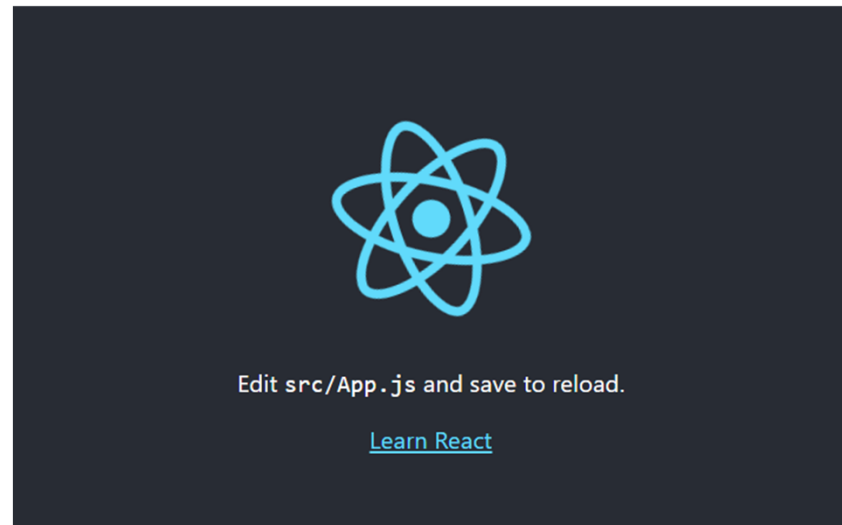


アプリケーションの起動

- アプリを起動する際は以下のコマンドを実行する

```
npm start
```

- エラーが発生する場合は、src/index.js内のreportWebVitalsの記述を削除する



表示される画面