Web アプリ開発の基礎：フロントエンド完全ガイド

開発学習ノート

2026 年 2 月 7 日

目次

はじめに

本ドキュメントは、React を用いたフロントエンド開発の全体像と、プロジェクトを構成する全ファイルの役割を詳細に解説したものである。特に「どのファイルがアプリのどの挙動を制御しているか」に重点を置いている。

第 1 部：フロントエンド製作の流れ

Step 1: 土台と道具の準備

まず、自分のパソコンでプログラミングができる状態を作る。

Node.js（ノード・ジェイエス）

パソコンの中で JavaScript を動かすためのエンジン。これがないと開発ツールが動かない。家づくりで言う「電力」。

npm（エヌ・ピー・エム）

世界中の開発者が作った便利な道具（ライブラリ）をカタログから選んでダウンロードしてくれる管理ツール。

Step 2: 骨組みを作る

Vite（ヴィート）

React などの開発環境を高速に構築するビルドツール。面倒な設定を自動化してくれる現場監督。

React（リアクト）

「UI（画面の部品）」を作るためのライブラリ。HTML を直接書くのではなく、「Component（部品）」を組み合わせて画面を作る。

■コマンド: `npm create vite@latest study-app - -template react`

Step 3: デザインの準備

Tailwind CSS

CSS ファイルを別に作らず、タグの中にクラス名を書くだけでデザインできるツール。

Step 4: **実装と確認**

Localhost（ローカルホスト）

自分のパソコンの中に一時的に立ち上げた Web サーバー。自分だけが見られるテスト環境。

第 2 部：ファイル構成図解（File Map）

プロジェクトフォルダ（study-app）内の全ファイルの役割を、「ソースコード」「設定ファイル」「システムファイル」の 3 つに分類して詳細解説する。

1. ソースコード（src フォルダ：開発の主戦場）

ここにあるファイルが、実際にブラウザに表示される画面や動きを作る。

- `src/App.jsx`
  【役割：メインステージ・脚本】
  このアプリの「本体」。今回編集した入力フォームやスライダーはここに記述されている。
  - **役割詳細**:
    - **見た目 (JSX)**: `return（...）`の中に書かれた HTML 風のコードが、実際の画面になる。
    - **記憶 (useState)**: ユーザーが入力した「偏差値」や「勉強時間」を一時保存する箱を作る。
    - **動き (Function)**:「ボタンが押されたら何をするか」というロジックを記述する。
- `src/main.jsx`
  【役割：電源スイッチ・架け橋】
  プログラムが最初に動き出す場所。
  - **役割詳細**: `index.html` にある「空っぽの箱（root）」を見つけ出し、そこに `App.jsx` の内容を流し込む（マウントする）処理を行っている。また、`index.css` を読み込んでアプリ全体にデザインを適用させるのもここの仕事。
- `src/index.css`
  【役割：デザインの源泉】
  アプリ全体のスタイルシート。
  - **役割詳細**: 通常の CSS を書く場所だが、今回は Tailwind CSS を使うための「3 つの呪文（`@tailwind ...`）」が書かれている。これが Tailwind の全機能を呼び出している。
- `src/assets/`
  【役割：素材置き場】
  画像ファイルやアイコンなどを置くフォルダ。ここに置いた画像は、プログラム内で `import` して使うことができる。

2. 設定ファイル（ルート階層：司令塔）

アプリの動かし方やルールを決めるファイル群。

- `package.json`
  【役割：プロジェクトの身分証・レシピ】
  - **詳細**: アプリ名、バージョン、そして「このアプリを作るのに必要な部品（React, Tailwind など）」のリストが書かれている。他人と開発するときは、このファイルを共有することで同じ環境を再現できる。
- `vite.config.js`
  【役割：現場監督への指示書】

Vite（ビルドツール）の設定。React を使うためのプラグイン設定などが書かれている。ポート番号を変えたい時などはここを編集する。

- **`tailwind.config.js`**
  【役割：デザインのルールブック】
  Tailwind CSS の設定。「どのファイルにデザインを適用するか（content）」を指定する重要ファイル。ここが間違っているとデザインが反映されない。
- **`postcss.config.js`**
  【役割：翻訳機の設定】
  Tailwind などの新しい CSS 記法を、ブラウザが理解できる標準的な CSS に変換するための設定。
- **`.eslintrc.cjs` / `eslint.config.js`**
  【役割：校正係】
  コードに書き間違いや、推奨されない書き方がないかをチェックするツール（ESLint）の設定ファイル。

3. システム・管理ファイル（触るな危険エリア）

自動生成されたり、ブラウザが直接読み込んだりするファイル群。

【注意】 node_modules/ （フォルダ）

【役割：巨大倉庫・部品の山】
`npm install` でダウンロードしてきた数千個のライブラリ（部品）の実物が保管されている場所。

- **注意**: 非常にサイズが大きい。中身を直接編集してはいけない。削除しても `package.json` があれば `npm install` で復元できる。

- **`package-lock.json`**
  【役割：納品書・確定したレシピ】
  `package.json` よりも詳しく、「実際にインストールされた部品の正確なバージョン」が自動で記録されるファイル。手動で編集してはいけない。
- **`.gitignore`**
  【役割：無視リスト】
  Git（保存ツール）に対して、「保存しなくていいファイル（node_modules など）」を教えるリスト。これがないと、巨大なファイルを誤ってアップロードしてしまう。
- **`index.html`**
  【役割：外箱・コンテナ】
  ブラウザが一番最初に読み込むファイル。
  - **詳細**: 中身はほぼ空っぽで、`<div id="root"></div>` という「React を表示するための場所」だけが用意されている。また、`src/main.jsx` を読み込むスクリプトタグがあり、そこから全てが始まる。
- **`public/`**
  【役割：そのまま公開する棚】
  ここに置いたファイル（favicon.ico など）は、加工されずにそのままインターネット上に公開される。