全4回で学ぶNext.js入門講座
1回目

2025/11/04

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

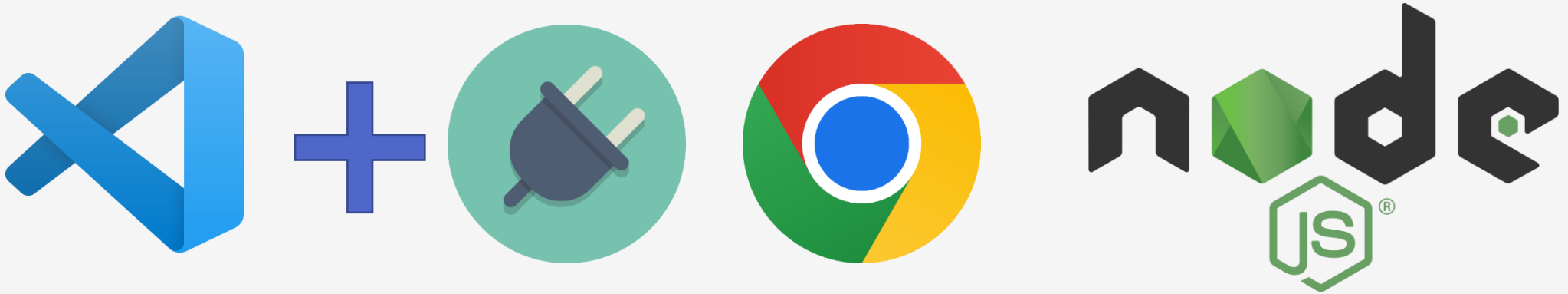
- Next.jsはどういうものであるかを知る
- Next.jsの基本的な構文を知る
- SSRの利用
- データアクセス
- Next.jsの基本的な構造を理解する

NEXT.JS

今回のアジェンダ

- Next.jsの概要
- プロジェクト作成
- JSXの基礎、コンポーネント作成、props
- ページ作成とルーティング
- フォームの基本
- useStateで入力管理

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
  - NativeJSで記述します

本講座における仕様

- Next.jsは現時点でのStable版の最新版である15系列を利用
  - 最新版である、16は利用しませんし、扱いません
- Reactについて多少は知っている前提で進めます

Next.jsとは何か

- Reactはクライアント側だけで動作するJSのUIライブラリー
  - 強力な機能を有しているため、利用者も多い
- Reactを使いつつ、サーバーサイドで動作するプログラムを作成することができるフレームワークとして登場したのがNext.js
  - Vercel社が中心となって開発、提供している
    - VercelはNext.jsのデプロイ環境を提供している

  - フロントエンド側の言語とサーバーサイド側の言語が同一にすることが可能
  - 開発効率の向上に繋がるとされる

Next.jsとは何か

- プリレンダリング機能があるのが特徴
  - サーバー側でReactを実行し、HTMLデータを生成することが可能
  - 方法としてSSRとSSGの2種類存在する
  - React単体ではプリレンダリングが無いため、ブラウザーで開いたときに初めて実行されるため、表示まで遅い
- サーバー側でHTMLデータを生成することで、クライアント側の負担が無く、表示が高速化することができる
  - SEOとして有利になる

Next.jsとは何か

- バージョンごとに機能や記述方法、概念が結構変わる
  - 例：Next.js13で新しいルーティング方法が登場（Appルーティング）
  - 旧来の書き方だと通用しなくなるケースがあるため、記述を参考にするときは、どのバージョン向きの記述か注意する必要がある
- 画像の最適化機能が存在する
- 各ページごとに必要なJSが自動的に分割される
  - Reactの場合は、1つの巨大なJSファイルが生成される

Next.jsとは何か

- CSS Modulesがサポートされているため、各コンポーネントに対してローカルなCSSクラスを定義可能
  - CSS同士が衝突することが少ない
  - 必要なCSSだけをロードするので読み込みが高速になる
- 他のCSSフレームワークも使える

Next.jsとは何か（デメリット）

- 学習コストは比較的高くなる
  - Reactのある程度の知識とSSR,SSGなどNext.js独自の機能への知識吸収が必要
  - SSRやSSGが比較的分かりづらい
- 小規模プロジェクトでの採用は向かない
  - Next.jsの機能は要らないことが多く、構成が複雑になるだけ
- SSRなどを使う場合はサーバー環境が必要
  - デプロイが比較的難しい
  - 普通のレンタルサーバーなどではまず動作しない

Next.jsでのプロジェクト作成

- Next.jsはReactと同様にNode.js環境及びNPMが必要
- SSR、SSGを動作させるためにもNode.jsが必要
  - 恐らくNode.js互換の実行環境でも動作するとは思う
- プロジェクトはNPXコマンドで作成

NPXを使ってNext.jsでのプロジェクト作成

- NPX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
- Windows環境ではWindows PowerShellは利用しないこと

NPXを使ってNext.jsをインストール

- フォルダーを移動する
  - cdと入力し半角スペースを空けて、フォルダーへのパスを記入して、Enter（Return）キー
    - フォルダーをドラッグ＆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

NPXを使ってNext.jsをインストール

- フォルダーを移動したらnpx create-next-appと入力

```
C:¥Users¥user¥desktop¥next>npx create-next-app@15 sample_next_app
```

- 最初の実行時に「create-next-app@15.5.6」Ok？的なことを聞かれるので、Enterキーで進める
  - この場合、Next.jsの15.5.6バージョンで進めることになる
- 後のsample_next_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）

NPXを使ってNext.jsをインストール

- ある程度進むと質問が出てくる

Would you like to use TypeScript? >> No / Yes

今回はNoを選択
カーソルキーで切替が可能

Which linter would you like to use? >> ESLint

今回はESLintを選択

Would you like to use Tailwind CSS? >> No / Yes

今回はYesを選択

Would you like to use 'src/' directory? >> No / Yes

今回はYesを選択
カーソルキーで切替が可能

Would you like to use App Router?(recommended) >> No / Yes

今回はYesを選択

Would you like to use Turbopack? (recommended) >> No / Yes

今回はYesを選択

Would you like to customize the default import･･･ >> No / Yes

今回はNoを選択

NPXを使ってNext.jsをインストール

- 質問が終わってからインストールと設定が行なわれる
- Success!という表示が出ればインストールは完了

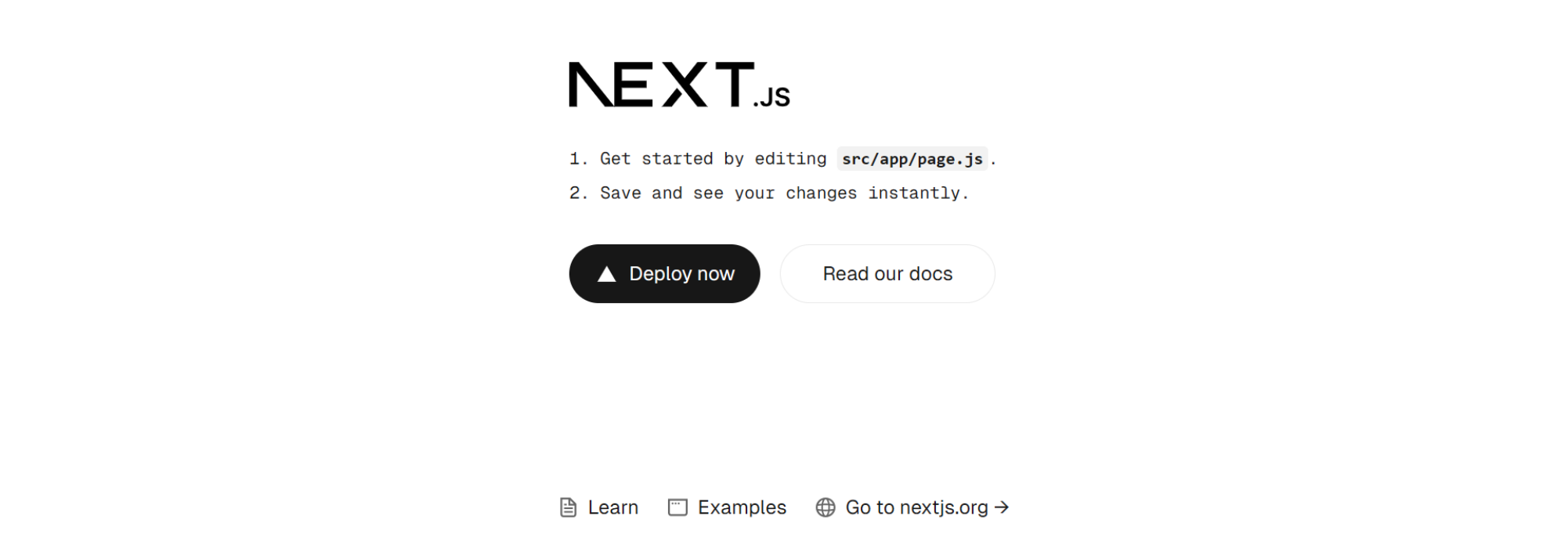
- インストールはsample_next_appフォルダー内に行なわれる
  - このフォルダーをVSCodeで開いて編集を行なっていく
- 一度表示を確認してみる

```
cd ./sample_next_app
npm run dev
```

ディレクトリを移動

開発サーバーを起動

開発サーバーでの表示



- http://localhost:3000の表示が出れば、クリックまたはCtrlキーを押しながらクリックすればブラウザーが起動する
- 上記のような表示が出れば、インストールは完了し、開発環境が起動している

どこが表示されているか

- 初期状態では、srcフォルダーにあるappフォルダー内のファイルが読み出されて表示されている
  - layout.js：1つだけ存在する全体的な共通するレイアウトを規定する
  - page.js：表示するページを定義するもの
- 初期状態では、layout.jsを利用し、page.jsを表示している
  - layout.jsではCSSファイルとフォントファイルを読み込み、metaタグを設定している

Tailwind CSS

- 今回はCSSでTailwind CSSを使えるようにしているので、Tailwind CSSで使えるクラスは全部指定することが可能
  - https://tailwindcss.com/docs/installation にクラス名が載っているが、数が多い
- 複数クラスも当然可能なので、組み合わせて見せたいスタイルにすればOK


```
<h1 class="text-2xl font-bold text-center text-blue-600">
  Hello Tailwind!
</h1>
```

ファイルベースルーティング

- Next.jsではファイルを作成することで、ルーティングが可能
  - ルーティング＝URLでどこで処理をさせるかを判断して、処理を依頼する役割を担う
- フォルダーを作成し、内部にファイルを作成する
  - page.jsファイル名でフォルダー名と同じ関数を作成する
  - src/app/hello/page.jsに記述

```
export default function Hello() {
  return <h1>Hello Page</h1>;
}
```

State(useState)

- Reactの場合、画面に表示するデータ、可変の状態を全てStateとして管理する
  - コンポーネントの状態を示す値
  - 状態を管理して、イベント実行時などに値を変更することでアプリケーションを実現する
- useState関数を利用して管理を行う
  - importする必要がある

```
import { useState } from "react";
```

useStateの利用

- useStateは配列の形で1つめにStateの変数、
  2つ目にその変数を更新するための関数を設定する

```
const [num , setNum] = useState();
```

- 変数名を付けて、暗黙的に更新側はset変数名とすることが多い
  - 変数の初期値はundefinedになるので、useState()の引数に値を指定すれば、初期化される

useStateの利用

src/app/counter/page.js
などに記述

```
"use client";
import { useState } from "react";
export default function Counter() {
  const [num, setNum] = useState(0);
  const onClickButton = () => {
    setNum((prev) => prev + 1);
  };
  return (
    <div className="p-6 text-center">
      <button onClick={onClickButton}>
        クリック
      </button>
      <p className="mt-4">クリックした回数:
{num}</p>
    </div>
  );
}
```

- ボタンをクリックする度にonclickButtonが動く
  - setNum()でnum値を増やしている
  - 動的に変更した値を表示

useState内の関数で更新

```
const onClickButton = () => {
    setNum((prev) => prev + 1);
};
```

- setNum内で関数を定義してしまう
  - 関数の引数に前のstate値が入ってくる
    - prevには実行される前の値が入ってくる
    - prevに1をプラスして戻す

フォームの作成

- useStateを使うことで、入力内容を保持できる
  - onChangeイベントを使うことで、入力内容が変化したら関数を呼び出すことが可能
  - app routerを使う場合、"use client";を記述する必要がある
    - イベントやstateを使うファイルには use client を付ける
    - この辺は次回詳しく

```
"use client";
import { useState } from "react";

export default function Form() {
  const [name, setName] = useState("");
  return (
    <>
      <input
onChange={(e)=>setName(e.target.value)} />
      <p>{name}</p>
    </>
  );
}
```

ありがとうございました。

また次回。