Next.Js v.15

→ It is a React FrameWork for building full-stack Web Application.

→ They have inbuild features like:

- Routing
- optimized Rendering
- data fetching
- Bundling, Compiling and more. ...

→ Opinions and Conventions should be followed to implement these features.

→ It can use Frontend components as well as backend API's with in the same nextJs application.

→ Rendering → It has both Server-Side-Rendering (SSR) and Client-Side-Rendering (CSR).

It helps for better Search Engine Optimization (SEO)

Installation (version 15).

Need:

NodeJs → v.18.18 or above. (Requirment minimum)

```
npx create-next-app@latest
```

It has eslint, ⊙ Customize import alias

- Turbopack,
- TypeScript
- Tailwind Css.

1) Folder Structure → Config files

- package.json (lock.json also)
- tailwind and postcss config.
- ts config
- next.config
- next-env.d.ts → typescript declaration for nextjs
- gitignore.

2) ~~folders~~ → folder

- .next → it will be create on npm run dev / npm build
- node_modules.
- public folder
- source / src.
- App → inside source for App Routing
- layout.tsx → have the RootLayout
- Page.tsx → UI for the page

React - Server - Component [RSC]

React Server Component is a new architecture that was introduced by React and quickly accepted by next.js.

This architecture introduce two new approach :

- → Server Component
- → Client Component.

* Server - Component

→ By default, Next.js treats all component as Server Component

→ These Component can perform server-side tasks like reading files or fetching data directly from database.

→ But they can't use React hooks or handle user interaction.

Client - Component

→ To create Client Component, we need to add "use client" directive at the top of the Component file.

→ They can't perform S-S tasks.

→ It is traditional React Component

Routing

- It uses a file System based Routing System
- URLs you can access in your browser are determined by how you organize your file and folder in your code.

Routing Convension

- All routes must live inside app folder
- Route file must be named either page.js or page.tsx
- Each folder represent a segment of the URL path.

Inside App folder, you need to create other folders for routing exam :

eg: Src
- App
  - About
    - page.tsx → it will be http://localhost/about

→ if the file name Changes route also change.

→ page.tsx will be the initial route.

* Nested Route

→ If you want nested route, create new folder with the route name like we did before.

Dynamic Routing

Routing

→ To make it dynamic we use [name

eg: App
↓
Products
↓
[ProductId]
↓
Page.tsx

It will Consider as a dynamic route

we Can also resolve the Promise with async await to access the dynamic Content

Code Snippet:

//src/app/product/[ProductId]/page.tsx → Path

```
export default function ProductDetails ({
    params,
}: { params : Promise< { productId : string } >;
}) {
    const {productId} = (await params).productId;
    return <h1> Details about product {productId} </h1>;
}
```

dynamic folder name.

To access the url parms.

Params:

Every Page in the App Router receives Route parameters through the params Props.