**Next.js Setup**

**Setting Up Your Next.js/React Development Environment**

This lesson walks you through the initial setup of a foundational Next.js application using React, TypeScript, and Tailwind CSS. We'll use pnpm as our package manager. The goal is to create a clean starting point, ts-tsender-ui-cu, which will serve as the basis for subsequent development.

While we use Next.js (a popular React framework) and React itself, this guide focuses purely on the setup process. The broader focus remains on blockchain technology. The JavaScript ecosystem, particularly frameworks like React and Next.js, evolves rapidly, and extensive learning resources already exist. We encourage you to leverage official documentation (like react.dev and nextjs.org) and AI tools (like ChatGPT or DeepSeek) if you encounter specific framework concepts that are unclear. Pause this lesson and consult those resources as needed. Our aim here is to establish the project structure efficiently.

**Prerequisites:**

* Ensure you have pnpm installed on your system.

**1. Initiating the Project**

First, open your terminal within your preferred code editor (like VS Code) in the directory where you want to create your project. Run the following command to start the Next.js application creation process:

pnpm create next-app@latest

**2. Configuration Choices**

The create-next-app command-line interface (CLI) will prompt you for several configuration options. Use the following settings:

* **What is your project named?** ts-tsender-ui-cu
* **Would you like to use TypeScript?** Yes (Provides type safety, beneficial for larger projects)
* **Would you like to use ESLint?** No (Skipped for simplicity in this initial setup)
* **Would you like to use Tailwind CSS?** Yes (A utility-first CSS framework for easier styling)
* **Would you like to use**src/**directory?** Yes (A common convention for organizing source code)
* **Would you like to use App Router? (recommended)** Yes (The modern, recommended routing system in Next.js)
* **Would you like to use Turbopack for builds? (alpha)** Yes (A faster Rust-based bundler for development speed)
* **Would you like to customize the default import alias (**@/\***)?** Yes (Press Enter to accept the default @/\* alias)

The CLI will now install the necessary dependencies (react, react-dom, next) and development dependencies (typescript, @types/react, @types/node, tailwindcss, postcss, autoprefixer).

**3. Correcting Folder Structure (If Necessary)**

Sometimes, create-next-app might create the project folder *inside* your current directory, resulting in a nested structure (e.g., your-folder/ts-tsender-ui-cu/ts-tsender-ui-cu). If this happens:

1. Use your file explorer (within VS Code or your operating system) to move all files and folders from the *inner* ts-tsender-ui-cu directory up one level into the *outer* ts-tsender-ui-cu directory.
2. Delete the now-empty inner ts-tsender-ui-cu folder.

Ensure your project root contains files like package.json, next.config.ts, and the src directory directly.

**4. Understanding Key Files and Concepts**

Let's briefly review some important files and concepts generated:

* package.json**:** Defines project metadata, scripts (like dev, build), and lists dependencies (react, next) and dev dependencies (typescript, tailwindcss).

// Example relevant sections (versions may differ)

{

"name": "ts-tsender-ui-cu",

"scripts": {

"dev": "next dev --turbopack",

// ... other scripts

},

"dependencies": {

"react": "...",

"react-dom": "...",

"next": "..."

},

"devDependencies": {

"typescript": "...",

"tailwindcss": "...",

// ... other dev dependencies

}

}

* pnpm-lock.yaml**:** Locks dependency versions for consistent installations across environments.
* tsconfig.json**:** Configures the TypeScript compiler.
* next.config.ts**:** Configures Next.js specific settings.
* tailwind.config.ts**&**postcss.config.mjs**:** Configuration files for Tailwind CSS.
* src/**Directory:** Contains your application's source code.
* src/app/**Directory:** Core of the Next.js App Router. Folders define routes.
* src/app/layout.tsx**:** Defines the root layout component wrapping all pages. Contains <html> and <body> tags. Uses .tsx because it includes JSX (HTML-like syntax in TypeScript/JavaScript) for the React component structure.
* src/app/page.tsx**:** Defines the React component for the application's root route (/). This is your homepage. Also uses .tsx.
* src/app/globals.css**:** Contains global styles and Tailwind CSS directives.
* **TypeScript (**.ts**,**.tsx**):** .ts files contain standard TypeScript code. .tsx files are TypeScript files that can also contain JSX, used for defining React components.
* **React Components:** Reusable UI pieces written as functions returning JSX (e.g., Home in page.tsx, RootLayout in layout.tsx).

**5. Running the Development Server**

To see your application in action, run the development script defined in package.json:

pnpm run dev

This command starts a local development server (usually at http://localhost:3000) with hot reloading enabled. Open this URL in your web browser. You should see the default Next.js starter page. Hot reloading means that when you save changes to your code files, the browser will automatically update without a full page refresh.

**6. Simplifying the Boilerplate Code**

The default Next.js starter page contains a lot of example content. Let's simplify page.tsx and layout.tsx to a bare minimum.

**a) Simplify**src/app/page.tsx**:**

Open src/app/page.tsx. Remove the import Image... line and replace the entire content within the return (...) statement of the Home component with a simple div.

* **Final Code (**src/app/page.tsx**):**

export default function Home() {

return (

<div>

Hi

</div>

);

}

**b) Simplify**src/app/layout.tsx**:**

Open src/app/layout.tsx. Make the following changes:

1. Remove the font imports (e.g., import { GeistSans }...).
2. Remove the const definitions for these fonts if present.
3. Update the metadata object: change title to "TSender" and description to "A simple and fast email sender".
4. Remove the className prop from the <body> tag.

* **Final Code (**src/app/layout.tsx**):**

import type { Metadata } from "next";

import "./globals.css";

​

export const metadata: Metadata = {

title: "TSender",

description: "A simple and fast email sender",

};

​

export default function RootLayout({

children,

}: Readonly<{

children: React.ReactNode;

}>) {

return (

<html lang="en">

<body>{children}</body>

</html>

);

}

Save both files. Your browser window showing localhost:3000 should automatically update (due to hot reloading) and now display only the word "Hi".

You have now successfully set up a minimal Next.js/React project with TypeScript and Tailwind CSS using pnpm. This provides a clean foundation for building the user interface components required for the subsequent steps of the project. Remember to consult external resources and AI tools for deeper dives into React or Next.js specifics as needed.