🚀 Day 2 of 60: My React Learning Journey! 🚀

1.Why NodeJs?

Node.js is needed for developing a React app because it helps in several ways:

a. It runs a local server so we can see our app as we build it.
b. It lets us install and manage libraries and tools needed for our app.
c. It helps compile and bundle our code to make our app run faster.
d. It allows React to run on the server for quicker page loads.
e. It can run scripts to automate tasks like testing and building our app.

In simple terms, Node.js makes it easier to develop, manage, and improve our React app.

2.What is package.json?

In simple terms, `package.json` is a file that keeps track of all the important information about our project. It lists the packages our project needs, the commands we can run, and other details like the project name and version. It helps in managing and automating various aspects of our Node.js and React projects.

3.Letter Conventions in React JS

In React.js, there are certain naming conventions for writing code to ensure consistency and readability. Here are some key conventions:

a. React components should be named using PascalCase, where each word starts with an uppercase letter. For eg:- " MyComponent() "
b. Component files should also use PascalCase. For eg:- " MyComponent.js "
c. Other files such as styles, utilities, etc., often use kebab-case. For eg:- " my-component-style.css "
d. Variables and prop names should be written in camelCase, where the first word is lowercase and subsequent words start with an uppercase letter. For eg:- "userName" , "userAge"
e. Event handler functions should use camelCase and typically start with 'handle'. For eg:- " handleClick "
f. CSS class names are usually written in kebab-case. For eg:- " .my-component{ background-color: blue;} "
g. Constants are often written in uppercase letters with underscores separating words. For eg:- " const API_URL = https://api.example.com "
h. Organize files based on features or components to keep the project structure modular and scalable.

4. What is jsx?

JSX (JavaScript XML) is a syntax extension for JavaScript used in React. It allows us to write HTML-like code inside JavaScript, making it easier to build and understand UI components. We can embed JavaScript expressions within JSX using curly braces `{}`.
For eg:- " const element = <h1> Hello, {userName}!</h1>; "
This line of code creates a JSX element. It looks like HTML but is written inside Javascript. The `{userName}` part is a Javascript expression that will be replaced with the value of `userName`.This allows us to dynamically insert Javascript values into our HTML structure.

5. Key Features of JSX

JSX (JavaScript XML) is a syntax extension for JavaScript used with React to describe what the UI should look like. Here are its key features:

a. HTML-Like Syntax: JSX syntax resembles HTML, making it easier for developers to write and understand the UI structure.
b. Embedding JavaScript: We can embed JavaScript expressions within JSX using curly braces {}. This allows us to include variables, functions, and expressions directly within your HTML-like code.
c. Component Integration:JSX makes it easy to define and use React components. Components can be nested, passed data via props, and managed with state.
d. Conditional Rendering: Using JavaScript's ternary operator or logical &&, we can conditionally render elements in a clean and readable manner.
e. Event Handling: JSX allows us to handle events like onClick, onChange, etc., directly within your JSX code.
f. Inline Styles: We can apply styles directly using the style attribute, which takes a JavaScript object with camelCased properties.
g. Fragments: JSX supports fragments, which allow us to group multiple elements without adding extra nodes to the DOM.

In short, JSX brings several powerful features to React development, including an HTML-like syntax, the ability to embed JavaScript expressions, component integration, conditional rendering, event handling, inline styles, security against injection attacks, and fragment support. These features make JSX a convenient and efficient way to describe UIs in a declarative manner.

6. Components in React Js

In React.js, components are like reusable building blocks for our UI. They can be either simple functions or classes that return HTML-like code (called JSX). Components can receive inputs (called props) and manage their own data (called state). By using

components, we can break down a complex user interface into smaller, manageable pieces, making it easier to develop and maintain our application.

7. Props(Short for 'Properties')

In simple terms, props(short for properties) in React are like parameters we pass to a function. They allow us to send data from a parent component to a child component, making it possible to customize and reuse components with different information. Props are read-only, meaning the child component can use them but cannot change them.

8. What is state?

In React.js, state is a built-in object that allows components to manage and store data that can change over time. Unlike props, which are read-only and passed from parent to child components, state is managed within the component itself. When the state of a component changes, React automatically re-renders the component to reflect the new state.

For eg:- state is a way to keep track of changing data, like a counter starting at 0. The component displays this initial value, and when a user clicks a button to increment the counter, the state updates with the new value (e.g., from 0 to 1). React then automatically re-renders the component to show the updated count. This allows the application to dynamically respond to user interactions, making it more interactive and engaging.