# Module-9

# React - Components, State, Props

# What is React Js?

# Ans: React JS is a popular open-source JavaScript library for building user interfaces, particularly single-page applications. It allows developers to build UI components that can efficiently update when data changes. React was developed by Facebook and is widely used for its ability to create dynamic, interactive web apps with a component-based architecture. React focuses on the view layer of an application Model-View-Controller and makes it easy to build reusable components.

# What is NPM in React Js?

# Ans: NPM (Node Package Manager) is a package manager for JavaScript. It is an essential tool for managing packages or libraries that are required for React applications. NPM allows you to easily install, update, and manage dependencies in your project. In React JS, you often use NPM to install React, React-DOM, and other third-party libraries required for your project.

# Example usage in React:-

# npm install react react-dom

# What is Role of Node Js in react Js?

# Ans: Node JS is a runtime environment that allows JavaScript to be run on the server side. In the context of React JS, Node JS plays a crucial role in:

# Running development servers (e.g., webpack-dev-server).

# Installing packages via NPM.

# Transpiling JSX code into plain JavaScript using tools like Babel.

# Bundling and optimizing the code for production deployment.

# Even though React itself is a client-side library, Node JS is commonly used in the development workflow for React apps.

# What is a CLI command in React JS?

# Ans: CLI (Command Line Interface) commands in React JS are used to interact with the React development environment through the terminal. One of the most common CLI tools for React is Create React App, which is used to quickly set up a new React project.

# Examples of CLI commands in React JS:

# Create a new React app:

# npx create-react-app my-app

# Start the development server:

# npm start

# Install dependencies:

# npm install

# Build the application for production:

# npm run build

# What is Components in React Js?

# Ans: In React JS, components are the building blocks of the user interface. A component is a self-contained, reusable piece of code that can define its own state and behaviour. Components can be written as:

# Functional Components: These are simpler components that are just JavaScript functions that return JSX.

# Class Components: These are ES6 classes that extend React.Component and have lifecycle methods.

# Example of a functional component:

# function MyComponent() {

# return <h1>Hello, React!</h1>;

# }

# What is Header and Content Components in React Js?

# Ans: In React JS, Header and Content components are examples of common UI components that can be used to structure a web page.

# Header Component: This might include elements like a navigation bar, logo, and other branding elements.

# Content Component: This can represent the main body of the page where the primary content is displayed.

# Example:

# function Header() {

# return <header><h1>Welcome to My Website</h1></header>;}

# function Content() {

# return <main><p>This is the content of the page.</p></main>;

# }

# How to install React Js on Windows, Linux Operating System? How to Install NPM and How to check version of NPM?

# Ans: To install React.js and npm (Node Package Manager) on Windows and Linux, you need to follow a few steps to get your environment ready for development. Here's a detailed guide for both operating systems:

# 1. Installing React.js and npm

# React requires Node.js and npm to be installed first, as they provide the necessary environment for package management and running React applications.

# A. Install Node.js and npm

# Node.js comes bundled with npm (Node Package Manager). So when you install Node.js, npm will be installed automatically.

# On Windows:

# Download Node.js:

# Go to the official Node.js website: <https://nodejs.org>.

# Download the LTS (Long Term Support) version for stability.

# Install Node.js:

# Run the downloaded installer and follow the on-screen instructions.

# During installation, make sure to check the option that says "Automatically install the necessary tools" (if available).

# This will install both Node.js and npm.

# Verify Installation: After the installation is complete, open the Command Prompt or PowerShell and run the following commands to verify the installation:

# node -v

# npm -v

# This will display the installed versions of Node.js and npm.

# On Linux (Ubuntu/Debian):

# Update System Packages:

# Open the terminal and update your package lists:

# sudo apt update

# Install Node.js and npm:

# Install Node.js and npm via the package manager. To install the latest stable LTS version, you can use the NodeSource repository:

# curl-sL https://deb.nodesource.com/setup\_18.x | sudo -E bash -sudo apt install nodejs

# This will install both Node.js and npm on your system.

# Verify Installation: After installation, check the versions of Node.js and npm using the commands:

# node -v

# npm -v

# This will display the installed versions.

# 2. Install React.js

# Once Node.js and npm are installed, you can use Create React App to easily set up a new React project.

# Using Create React App (Recommended)

# Install Create React App globally (Optional but recommended):

# In the terminal or command prompt, run the following command to install the Create React App tool globally:

# npm install -g create-react-app

# This tool simplifies setting up a new React project by creating a starter template for you.

# Create a New React Application:

# Navigate to the folder where you want to create your React project. Then run:

# npx create-react-app my-app

# Replace my-app with the name of your project. This will download the necessary files and set up a new React project in the my-app folder.

# Navigate to the Project Folder:

# cd my-app

# Start the React Development Server:

# Now that the app is set up, start the development server:

# npm start

# This will open the React app in your browser at http://localhost:3000.

# 3. Install NPM (if it's not installed)

# If you don't already have npm installed (or if it’s missing for any reason), you can install or update it separately after installing Node.js.

# Installing or Updating npm

# On Windows and Linux, after installing Node.js, npm should already be installed. If you need to update npm or install it separately, follow these steps:

# To Install npm (if missing):

# sudo apt install npm # For Linux (Ubuntu/Debian)

# On Windows, npm should come with Node.js, so you won’t need to install it separately.

# To Update npm:

# npm install -g npm

# 4. Check the Version of npm

# To verify the version of npm installed, use the following command:

# npm -v

# This will print the installed version of npm in the terminal or command prompt. For example, it might return something like:

# 8.5.5

# This command works on both Windows and Linux.

# Summary of Steps:

# Install Node.js (which includes npm).

# Verify the installation of Node.js and npm with node -v and npm -v.

# Use Create React App to set up a new React project.

# Run the development server using npm start.

# Optionally, install or update npm with npm install -g npm.

# How to check version of React Js?

# Ans: To check the version of React.js in your project, you can use several methods depending on the setup. Here are the most common ways to check the version of React:

# 1. Check package.json File

# In every React project, the package.json file contains the dependencies used in the project, including the version of React.

# Open the package.json file in the root directory of your React project.

# Look for the "react" and "react-dom" fields under dependencies. They should look something like this:

# Json:

# "dependencies": {

# "react": "^18.2.0",

# "react-dom": "^18.2.0"

# }

# The version number next to "react" shows the installed version.

# 2. Using npm Command

# If you are in your React project directory, you can run the following command in the terminal or command prompt to check the installed version of React:

# Bash:

# npm list react

# This will return the version of React installed in your project. Example output:

# Perl:

# my-app@0.1.0 /path/to/your/project

# └── [react@18.2.0](mailto:react@18.2.0)

# If you want to check the version of React DOM, you can run:

# Bash:

# npm list react-dom

# 3.Using npx Command

# You can use the npx command to check the version of React if you don’t want to look into the node\_modules folder manually:

# Bash:

# npx react –version

# This command will return the installed version of React.

# 4.Using yarn (if using Yarn)

# If you're using Yarn as your package manager instead of npm, you can check the installed version of React by running:

# Bash:

# yarn list react

# This will display the version of React in your project.

# Conclusion:

# To summarize, you can check the React version:

# In package.json: Look for the "react" entry in the dependencies.

# Using npm: Run npm list react.

# Using npx: Run npx react --version.

# If using Yarn: Run yarn list react.

# How to change in components of React Js?

# Ans: To modify or change components in React JS:

# Open the component file (e.g., App.js or any other component).

# Edit the JSX or the logic of the component as per your requirements.

# Save the file, and React will automatically update the UI in the development environment (thanks to hot-reloading).

# Example:

# function App() {

# return

# <div><h1>UpdatedHeading</h1></div>;

# }

# When you make changes, the browser will automatically reflect the changes without needing a manual refresh, assuming you're running the development server (npm start).

# How to Create a List View in React Js?

# Ans: To create a list view in React.js, you can use the best practices shown in the image. Here’s a detailed explanation and a practical example:

# Best Practices for Rendering Lists in React

# Use Array.map() to iterate over your array and render components or HTML for each item.

# Avoid traditional for loops; they are less declarative in React and not optimized for JSX rendering.

# Assign unique keys to list items using an identifier from the data (e.g., id).

# Do not use the array index as the key unless absolutely necessary. Using the index may cause rendering issues when the list changes dynamically.

# Example Code for List View in React

# Jsx:

# import React from 'react';

# const ListView = () => {

# // Sample data for the list

# const items = [

# { id: 1, name: 'React' },

# { id: 2, name: 'Angular' },

# { id: 3, name: 'Vue' },

# { id: 4, name: 'Svelte' },

# ];

# return (

# <div>

# <h1>Framework List</h1>

# <ul>

# {/\* Use Array.map() to render the list \*/}

# {items.map((item) => (

# <li key={item.id}>{item.name}</li> // Unique key using item.id

# ))}

# </ul>

# </div>

# );

# };

# export default ListView;

# Key Features in the Code:

# Array.map(): It transforms each object in the items array into a JSX element.

# Unique Key (item.id): Used for each list item to help React efficiently manage updates.

# Dynamic Rendering: This allows for a flexible and reusable list view.

# Rendering the Component:

# Save the above code in a file (e.g., ListView.js) and include it in your React application like this:

# Jsx:

# import React from 'react';

# import ReactDOM from 'react-dom';

# import ListView from './ListView';

# const App = () => (

# <div>

# <ListView />

# </div>

# );

# ReactDOM.render(<App/>,document.getElementById('root'));

# This will display a dynamic list of items rendered "the React way." Let me know if you'd like further clarification!

# Create Increment decrement state change by button click?

# Ans: To create an Increment-Decrement functionality using React, you can use the useState hook to manage the state of a counter. Here's a complete example of implementing the UI and logic for the buttons to increment, decrement, and reset the counter:

# Example Code for Increment, Decrement, and Reset Buttons :

# import React, { useState } from 'react';

# const CounterApp = () => {

# // Declare a state variable `count` with an initial value of 0

# const [count, setCount] = useState(0);

# return (

# <div style={{ textAlign: 'center', marginTop: '50px', backgroundColor: 'blue', color: 'white', height: '100vh' }}>

# <h1>React Web</h1>

# <h2>{count}</h2>

# {/\* Decrement Button \*/}

# <button

# style={{ margin: '10px', padding: '10px 20px', backgroundColor: 'black', color: 'white', border: 'none' }}

# onClick={() => setCount(count - 1)} >

# Decrement

# </button>

# {/\* Increment Button \*/}

# <button

# style={{ margin: '10px', padding: '10px 20px', backgroundColor: 'black', color: 'white', border: 'none' }}

# onClick={() => setCount(count + 1)} >

# Increment

# </button>

# {/\* Reset Button \*/}

# <button

# style={{ margin: '10px', padding: '10px 20px', backgroundColor: 'black', color: 'white', border: 'none' }}

# onClick={() => setCount(0)}

# >

# Reset

# </button>

# </div>

# );

# };

# export default CounterApp;

# Rendering the Component:

# Jsx:

# import React from 'react';

# import ReactDOM from 'react-dom';

# import CounterApp from './CounterApp';

# const App = () => (

# <div>

# <CounterApp />

# </div>

# );

# ReactDOM.render(<App />, document.getElementById('root'));