**Module – 3 React Js**

**1.What is React Js?**

React.js, often simply referred to as React, is an open-source JavaScript library developed and maintained by Facebook. It is used for building user interfaces, particularly for single-page applications where UI updates are frequent. React allows developers to create reusable UI components that can be composed together to build complex user interfaces.

One of the main features of React is its virtual DOM (Document Object Model), which is a lightweight representation of the actual DOM in memory. React uses this virtual DOM to efficiently update the actual DOM, minimizing the number of DOM manipulations needed, which leads to better performance.

React follows a component-based architecture, where UIs are divided into reusable components, each responsible for rendering a small part of the UI. These components can have their own state and lifecycle methods, making it easier to manage complex UI logic.

React has gained significant popularity among developers due to its simplicity, performance, and the vibrant ecosystem of libraries and tools that have emerged around it, such as Redux for state management, React Router for routing, and many others.

**2. What is NPM in React Js?**

The command npm is used to download JavaScript packages from Node Package Manager, and npx is used to execute JavaScript packages downloaded this way.

This command will download the NPM package create-react-app to a subdirectory of the current working directory named node\_modules.

**3. What is Role of Node Js in react Js?**

1. High server load: Using Nodejs with React makes sense when your web application needs handling of multiple requests and maintaining server load balance.

2. Real-time data: If your application’s core is based on Real-time Data-Intensive management or Data Streaming, using Nodejs is highly advisable for continued server connection.

3. JSON APIs: Building JSON APIs for your application is very efficient with Nodejs due to high code reusability and easy code sharing in Reactjs.

4. Single Page Applications (SPA): Developing Single Page Applications in React while using Node to build a lightweight backend model for asynchronous data loading through callback functions.

5. MERN stack: Nodejs can also be used with React with MERN (MongoDB, Express, React, and Nodejs) stack.

6. Scalability: Node and React enables developers to build multi-device, responsive, data-driven web apps with scalability for large projects, ensuring optimal website performance across various devices.

7. Fast Development: Using React and Node for web app development yields high ROI, saving time and money. These technologies excel in creating fast and easy-to-maintain websites.

8. Single Language for Front-end and Back-end: Using Node and React together, developers avoid the need to learn complex back-end languages like Python or Ruby. They can use Node for server-side development and React for front-end coding seamlessly, saving resources, money, and time.

9. Organized Process: The combo of React and Node streamlines web development. These scalable, effective, and fast technologies work together to create high-functioning websites.

**4. What is CLI command In React Js?**

In React.js, CLI (Command Line Interface) commands are commands that you run from your terminal or command prompt to perform various tasks related to creating, developing, building, and deploying React.js applications. These commands are provided by tools like Create React App, which is a popular CLI tool for setting up React.js projects with a predefined folder structure, development server, and build configurations.

Here are some common CLI commands used in React.js development with Create React App:

1.Creating a New React App: You can use the npx create-react-app command followed by the name of your project to create a new React.js project. For example:

npx create-react-app my-react-app

2. Starting the Development Server: Once you've created a React.js project, you can navigate into its directory and use the npm start command to start the development server. This command runs your React.js application locally and allows you to view it in your web browser:

cd my-react-app

npm start

3. Building the Production Version: When you're ready to deploy your React.js application to a production environment, you can use the npm run build command. This command creates an optimized production build of your application that is ready for deployment:

npm run build

4. Running Tests: React.js applications often include tests to ensure that the code behaves as expected. You can use the npm test command to run the tests included in your project:

npm test

5. Ejecting: Create React App provides a pre-configured development environment out of the box, but if you need more control over your project's configuration, you can "eject" from Create React App using the npm run eject command. This command exposes the project's configuration files so that you can customize them as needed:

npm run eject

These are just a few examples of CLI commands commonly used in React.js development. Depending on your project's requirements and the tools you're using, there may be additional commands available to you.

**5. What is Components in React Js?**

There are two main types of components in React.js:

1.Functional Components: Functional components are JavaScript functions that accept props (short for properties) as an argument and return JSX (JavaScript XML), which describes the UI that the component should render. Functional components are simple and concise, making them ideal for representing UI elements that don't have state or lifecycle methods.Example of a functional component:

import React from 'react';

function Greeting(props) {

return <h1>Hello, {props.name}!</h1>;

}

export default Greeting;

2. Class Components: Class components are ES6 classes that extend the React.Component class. They have a render() method that returns JSX to define the UI. Class components are used when you need to manage state or use lifecycle methods.Example of a class component:

import React, { Component } from 'react';

class Greeting extends Component {

render() {

return <h1>Hello, {this.props.name}!</h1>;

}

}

export default Greeting;

**6. What is Header and Content Components in React Js?**

1. Header Component: The Header component typically contains elements that are displayed at the top of the application, such as navigation menus, branding elements, search bars, user profile information, or any other content that is meant to be consistently visible across different pages or views of the application.Example of a Header component:

import React from 'react';

function Header() {

return (

<header>

<nav>

<ul>

<li><a href="/">Home</a></li>

<li><a href="/about">About</a></li>

<li><a href="/contact">Contact</a></li>

</ul>

</nav>

</header>

);

}

export default Header;

2. Content Component: The Content component typically contains the main content of the application, such as articles, posts, images, videos, or any other dynamic or static content that changes based on the user's interaction or navigation within the application. The Content component can vary significantly depending on the specific functionality and purpose of the application.Example of a Content component:

import React from 'react';

function Content() {

return (

<main>

<h1>Welcome to My React App</h1>

<p>This is the main content area of the application.</p>

</main>

);

}

export default Content;

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

To install React.js and NPM on Windows and Linux operating systems, you can follow these general steps:

Installing React.js:

1. Install Node.js: React.js development typically relies on Node.js, so the first step is to install Node.js, which includes NPM (Node Package Manager). You can download the installer from the official Node.js website: [Node.js Downloads](https://nodejs.org/en/download/)

2. Create a New React App: Once Node.js is installed, you can use Create React App, a CLI tool, to quickly set up a new React.js project. Open your terminal or command prompt and run the following command:

npx create-react-app my-react-app

Replace `my-react-app` with the desired name of your project.

3. Navigate to Project Directory: After the project is created, navigate into the project directory:

cd my-react-app

4. Start Development Server: Start the development server to see your React.js application running locally:

npm start

Installing NPM:

Since NPM is included with Node.js, you don't need to install it separately. When you install Node.js, NPM is automatically installed alongside it.

Checking NPM Version:

To check the version of NPM installed on your system, you can use the following command in your terminal or command prompt:

npm -v

This command will display the version number of NPM installed on your system.

Following these steps should help you get started with React.js development on both Windows and Linux operating systems.

**8. How to check version of React Js?**

To check the version of React.js installed in your project, you can follow these steps:

1. Navigate to your React.js project directory: Open your terminal or command prompt and navigate to the directory of your React.js project using the `cd` command.

2. Check the package.json file: Inside your project directory, you'll find a file named `package.json`. Open this file in a text editor or view its contents in the terminal using a command-line text editor like `cat` (Linux) or `type` (Windows).

3. Find the version of React.js: In the `package.json` file, look for the `"react"` entry under the `"dependencies"` section. The version number of React.js will be specified next to it.

Here's an example of how the `"dependencies"` section might look in a `package.json` file:

json

"dependencies": {

"react": "^17.0.2",

"react-dom": "^17.0.2",

}

In this example, `"react"` has a version number of `^17.0.2`.

Alternatively, you can use the following command in your terminal or command prompt within your project directory to display the version of React.js installed:

bash

npm list react

This command will output the version of React.js installed in your project.

By following these steps, you can easily check the version of React.js in your React.js project.

**9. How to change in components of React Js?**

setState() enqueues changes to the component state and tells React that this component and its children need to be re-rendered with the updated state. This is the primary method you use to update the user interface in response to event handlers and server responses.

setState(updater[, callback])

**10. How to Create a List View in React Js?**

Creating a list view in React.js involves rendering a list of items dynamically using JavaScript and JSX. Here's a step-by-step guide on how to create a simple list view in a React.js application:

1. Set Up Your React.js Project: If you haven't already, set up a new React.js project using Create React App or any other method of your choice.

2. Create a Component for the List View: Create a new component file for your list view. For example, you could create a file named `ListView.js` in the `src` directory of your project.

3. Define the List View Component: In the `ListView.js` file, define a functional component that will represent your list view. Here's an example of how you can define a simple list view component:

import React from 'react';

function ListView({ items }) {

return (

<ul>

{items.map(item => (

<li key={item.id}>{item.text}</li>

))}

</ul>

);

}

export default ListView;

4. Pass Data to the List View Component: In your main application component or any other parent component, define an array of items and pass it as a prop to the `ListView` component. For example:

import React from 'react';

import ListView from './ListView';

function App() {

const items = [

{ id: 1, text: 'Item 1' },

{ id: 2, text: 'Item 2' },

{ id: 3, text: 'Item 3' }

];

return (

<div>

<h1>List View Example</h1>

<ListView items={items} />

</div>

);

}

export default App;

5. Render the List View Component: In your main application component (`App.js` or equivalent), import and render the `ListView` component passing the array of items as a prop.

6. Run Your React.js Application: Run your React.js application locally using the development server (`npm start` command) to view the list view in your web browser.

By following these steps, you'll create a simple list view in your React.js application that dynamically renders a list of items. You can customize the appearance and behavior of the list view by modifying the JSX code and component logic as needed.