**ReactJs Assignments**

**[ UDAY RAJPUT ]**

**1) What is React Js?**

Ans :-

**React.js**, commonly referred to as React, is a JavaScript library used for building user interfaces (UIs) for web applications. It was developed by Facebook and is maintained by Facebook and a community of individual developers and companies. React allows developers to create reusable UI components.

Key points about React:

**1 ) Component-Based:** React uses a component-based architecture where UIs are divided into reusable components, each responsible for its own state and rendering logic.

**2 ) Virtual DOM:** React uses a virtual DOM (Document Object Model) to efficiently update and render UI components. Changes to the virtual DOM are batched and then applied to the real DOM, improving performance.

**3 ) Declarative:** React uses a declarative programming style, allowing developers to describe how the UI should look based on the application's state, rather than manually manipulating the DOM.

**4 ) JSX:** React introduces JSX (JavaScript XML), a syntax extension that allows developers to write HTML-like code within JavaScript, making UI components more intuitive and readable.

**5 ) One-Way Data Binding:** React follows a unidirectional data flow, where data flows only in one direction (from parent to child components), which helps maintain the predictability of the application state.

**2) What is NPM in ReactJs?**

**Ans :-**

NPM (Node Package Manager) in React.js is a tool used for managing and installing packages (libraries, tools, and dependencies) for React applications. It is primarily used to install and manage third-party packages that provide useful functionalities for building React applications.

Key points about NPM in React.js:

**1 ) Package Management:** NPM allows developers to easily install, update, and remove packages needed for React development.

2 ) **Dependency Resolution:** NPM automatically resolves and installs dependencies required by the packages you use in your React project.

**3 ) Scripts:** NPM provides a way to run custom scripts defined in your project's package.json file, making it easy to execute common tasks like starting a development server or building your application.

**5 ) Registry:** NPM hosts a vast repository of open-source packages that can be searched and utilized in React projects.

**3) What is Role of NodeJs in ReactJs?**

**Ans :-**

Node.js plays a key role in React.js development by providing a runtime environment for running JavaScript on the server side. It allows React applications to be served, built, and managed more efficiently. Node.js is commonly used with React for:

**1 ) Server-side Rendering:** Node.js can be used to render React components on the server side, enabling faster initial page loads and improved SEO.

**2 ) Build Tools:** Node.js is used with build tools like web pack and Babel to bundle and transpile React code into a format that browsers can understand.

**3 ) Development Server:** Node.js can serve as a development server for running and testing React applications locally.

**4 ) Backend Integration:** Node.js can be used to build the backend APIs and services that a React frontend communicates with, creating a full-stack JavaScript application.

**4) What is CLI command in ReactJs?**

**Ans :-**

In React.js, CLI (Command Line Interface) commands are used to perform various development tasks, such as creating, building, and running React applications. The primary CLI tool for React development is create-react-app.

Key CLI commands in React.js using create-react-app:

- Creating a New React Project:

* fnpx create-react-app my-app

- Starting the Development Server:

cd my-app

npm start

- Building the Production Build:

npm run build

- Running Tests:

npm test

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

**Ans :-**

In React.js, components are the building blocks of a user interface. They are reusable and independent pieces of code that encapsulate a part of the UI's functionality and can be composed together to create complex UIs. There are two main types of components in React:

1. Functional Components: These are simple JavaScript functions that accept props (short for properties) as input and return React elements to describe what should appear on the screen. They are also known as stateless components because they don't have their own internal state.

jsx

function Welcome(props) {

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

}

2. Class Components: These are ES6 classes that extend React.Component. They have more features than functional components, such as local state and lifecycle methods.

jsx

class Welcome extends React.Component {

render() {

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

}

}

Components can be composed together to form complex UIs. They can also accept inputs called props, which allow them to be customizable and reusable in different contexts. React applications are typically built by composing a tree of components, with each component responsible for rendering a part of the UI.

**6) What is Header and Content Components in ReactJs?**

**Ans :-** I

In a typical web application, the header and content components serve distinct purposes:

1. Header Component: This component typically contains elements that appear at the top of the page, such as the site logo, navigation menu, user authentication controls, etc. The header component often remains consistent across different pages of the application, providing users with a consistent navigation experience.

Example of a header component in React:

jsx

import React from 'react';

function Header() {

return (

<header>

<h1>My App</h1>

<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: This component represents the main content area of the web page, where the dynamic content or specific functionality of the page is displayed. The content component can vary significantly depending on the specific page or route within the application.

Example of a content component in React:

jsx

import React from 'react';

function Content() {

return (

<div>

<h2>Welcome to our website!</h2>

<p>This is the main content area where you can find information about our services.</p>

</div>

);

}

export default Content;

By separating the header and content components, you can achieve better code organization, reusability, and maintainability in your React application. Additionally, this separation allows for easier styling and customization of each component independently.

**7) 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 Node Package Manager (NPM) on Windows and Linux operating systems, follow these steps:

Installing React.js and NPM on Windows:

1. Install Node.js: React.js requires Node.js, which includes NPM. You can download the installer from the [Node.js website](https://nodejs.org/) and follow the installation instructions.

2. Create a React App: Once Node.js is installed, you can create a new React app using npx (which comes with NPM). Open a command prompt or PowerShell window and run the following command:

npx create-react-app my-react-app

This will create a new directory named my-react-app with a basic React project structure.

3. Navigate to the Project Directory: Move into the newly created project directory:

cd my-react-app

4. Start the Development Server: You can start the development server by running:

npm start

This will launch the React app in your default web browser.

Installing React.js and NPM on Linux:

1. Install Node.js: You can install Node.js on Linux using your package manager. For example, on Ubuntu, you can use apt:

sudo apt update

sudo apt install nodejs

sudo apt install npm

2. Create a React App: After Node.js and NPM are installed, you can create a new React app using npx, similar to the Windows instructions.

3. Navigate to the Project Directory: Move into the newly created project directory.

4. Start the Development Server: You can start the development server using npm start.

Checking the Version of NPM:

You can check the version of NPM installed on your system by running the following command in your terminal or command prompt:

npm -v

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

Once you have React.js and NPM installed, you can start building React applications and managing dependencies using NPM commands.

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

**Ans :-**

To check the version of React.js installed in your project, you can use either npm or yarn, depending on which package manager you're using. Here's how you can do it:

Using npm:

npm list react

Using yarn:

yarn list react

Both of these commands will display the version of React.js installed in your project's dependencies. If React.js is not installed, you won't see any output.

You can also check the React.js version directly within your code by importing React and outputting its version:

javascript

import React from 'react';

console.log(React.version);

This will log the React.js version to the console when your application runs.

These methods allow you to verify the version of React.js being used in your project and ensure compatibility with other libraries and frameworks.

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

**Ans :-**

To change components in React.js, you typically need to modify the code of the component itself. Here are the general steps to make changes to a React component:

1. Identify the Component to Change: Determine which component(s) you need to modify to achieve the desired behavior or appearance.

2. Make Changes to the Component Code: Open the file containing the component you want to change and modify the JSX code or JavaScript logic as needed. This might involve adding, removing, or updating elements, props, state, or event handlers within the component.

For example, if you want to change the text displayed by a component, you would locate the JSX code responsible for rendering the text and update it accordingly:

jsx

function MyComponent() {

return <h1>Hello, World!</h1>; // Change the text "Hello, World!" to your desired text

}

3. Save the Changes: Save the modified component file.

4. Test the Changes: Run your React application to see the effect of the changes you've made. You can do this by starting the development server using npm start or yarn start depending on your package manager.

5. Inspect and Debug: If the changes don't produce the desired result or cause errors, inspect the browser console for any error messages and review your changes to identify and fix any issues.

6. Commit Changes (Optional): If you're using version control (e.g., Git), you may want to commit your changes to your repository to track them and collaborate with others.

7. Deploy Changes (If Applicable): If you're making changes to a production application, follow your deployment process to push the changes to your production environment.

Repeat these steps as necessary to continue making changes and improvements to your React.js components. Additionally, it's essential to follow best practices such as keeping components small, modular, and reusable to maintain a clean and manageable codebase.