**ReactJS Assignment**

**Module 3**

**Qs 1 :- What is React Js?**

**Ans :**

* **ReactJS** is a **JavaScript library** for building **user interfaces**.
* It's used for making **web applications** that can **change** what they show without **reloading** the whole page.
* React helps developers make **parts** of the webpage that can be **reused**.
* It's good for making **complex** and **interactive** websites.
* React makes it easier to **keep track** of how things look and **update** them when needed.
* It's **declarative**, meaning you tell it **what you want**, and it figures out **how to do it**.

**Qs 2 :- What is NPM in React Js?**

**Ans :**

npm (Node Package Manager) is a critical tool in ReactJS development, serving as a central hub for managing project dependencies. It simplifies the process of installing, updating, and removing packages, ensuring that developers can easily maintain a consistent set of dependencies across different environments. npm's package.json file plays a key role in this by tracking project dependencies and their specific versions.

Moreover, npm offers powerful scripting capabilities, allowing developers to define custom scripts for various tasks such as building the project, starting a development server, running tests, or deploying the application. This scripting functionality enhances the development workflow by automating repetitive tasks and improving productivity.

Additionally, npm serves as a platform for sharing and discovering packages, contributing to the vast ecosystem of reusable components and libraries in the ReactJS community. Overall, npm's role in dependency management, scripting, and package distribution makes it an essential tool for ReactJS developers, enabling efficient and streamlined development processes.

**Qs 3 :- What is Role of Node Js in react Js?**

**Ans :**

Node.js plays several important roles in a React.js application:

Server-side Rendering: Node.js can be used to render React components on the server side. This is useful for improving initial load times and SEO, as the server can send a fully rendered page to the client instead of just the JavaScript bundle.

Development Server: Node.js can be used to run a development server for a React.js application. This server can automatically reload the application when changes are made, making the development process faster and more efficient.

Build Tools: Node.js is often used in conjunction with build tools like Webpack or Babel to compile React.js code into a format that can be run in the browser. Node.js provides the environment needed to run these build tools.

Backend API: Node.js can be used to create a backend API for a React.js application. This API can handle things like database interactions and authentication, allowing the React.js frontend to communicate with the backend.

Overall, Node.js is an essential part of the React.js ecosystem, providing the tools and environment needed to develop, build, and run React.js applications.

**Qs 4 :- What is CLI command In React Js?**

**Ans :**

In React.js, CLI (Command Line Interface) commands are used to perform various development tasks, such as creating a new React project, running a development server, building the project for production, and running tests. These commands are typically executed in the terminal or command prompt.

Some common CLI commands in React.js include:

Create a New Project: npx create-react-app my-app

This command creates a new React project named "my-app" using the Create React App toolchain.

Start Development Server: npm start

This command starts the development server and opens the React application in a web browser. It also enables hot module replacement, allowing changes to be reflected in the browser without a full page reload.

Build for Production: npm run build

This command builds the React application for production, optimizing the bundle for performance and creating static files that can be deployed to a web server.

Run Tests: npm test

This command runs the test suite for the React application, executing any test files or test suites that have been created.

Eject: npm run eject

This command ejects the Create React App configuration, exposing all configuration files and dependencies. Ejecting is irreversible and should be done with caution.

These CLI commands help streamline the development process and are essential for building and managing React.js applications.

**Qs 5 :- What is Components in React Js?**

**Ans :**

In React.js, components are the building blocks of a user interface. They are reusable, self-contained pieces of code that define how a part of the UI should appear and behave. Components can range from simple elements like buttons or input fields to complex elements like entire forms or even entire pages.

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

Functional Components: These are JavaScript functions that accept props (short for properties) as input and return React elements to describe what should be rendered. They are simple and used for presenting UI based on the input props.

Class Components: These are ES6 classes that extend React.Component. They have additional features such as local state and lifecycle methods, making them suitable for more complex UI logic and interactions.

Components can be composed together to create larger, more complex UIs. They promote reusability, modularity, and maintainability of code, making them a key concept in React.js development.

**Qs 6:- What is Header and Content Components in React Js?**

**Ans:**

In a React.js application, Header and Content components are examples of how you might organize your user interface into smaller, reusable pieces. Here's a brief overview:

Header Component:

The Header component typically contains elements that appear at the top of the application, such as a navigation bar, logo, or user profile information.

It can be a functional component if it's mostly presentational or a class component if it needs to manage its own state or use lifecycle methods.

Example usage: <Header />

Content Component:

The Content component represents the main content area of the application, where most of the dynamic content is displayed.

It can vary widely depending on the application, containing things like articles, forms, lists, or other components.

Like the Header component, it can be a functional or class component based on its complexity and requirements.

Example usage: <Content />

These components help to organize the UI into logical sections, making it easier to manage and maintain. They can also be reused across different parts of the application, improving code reusability and maintainability.

**Qs 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 on Windows, Linux, or any other operating system, you first need to have Node.js installed, as it includes npm (Node Package Manager). Here's a general guide:

Install Node.js and npm:

1. Windows:

- Download the Node.js installer from the [official Node.js website](https://nodejs.org/).

- Run the installer and follow the instructions to install Node.js and npm.

2. Linux:

You can install Node.js and npm using your distribution's package manager. For example, on Ubuntu, you can use:

**CODE**

sudo apt update

sudo apt install nodejs npm

Install React.js:

Once you have Node.js and npm installed, you can create a new React.js project using `create-react-app`:

1. Open a terminal or command prompt.

2. Run the following command to create a new React.js project:

**CODE**

npx create-react-app my-app

Replace `my-app` with the name you want to give your project.

3. Change into the newly created directory:

**CODE**

cd my-app

4. Start the development server:

**CODE**

npm start

Check npm version:

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

**CODE**

npm -v

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

That's it! You should now have React.js installed and running on your Windows or Linux system.

**Qs 8 :- How to check version of React Js?**

**Ans :**

To check the version of React.js that you have installed globally, you can use the following command in your terminal or command prompt:

**npm list -g react**

This command will display the version of React.js that is installed globally on your system.

**Qs 9 :- How to change in components of React Js?**

**Ans:**

To change components in React.js, you'll typically edit the JSX code of your components. Here's a general process for making changes:

Identify the Component: Locate the component you want to change in your project's file structure.

Edit the JSX Code: Open the component file (usually with a .jsx or .js extension) in a text editor. Modify the JSX code to make the desired changes. For example, you might change the text content, add new elements, or update component logic.

Save Changes: Save the file after making your modifications.

Check Changes: Run your React application to see the changes you've made. If you're using create-react-app, you can start the development server with npm start.

Review Changes: Open your web browser and navigate to your React application to see the changes reflected in the UI.

Debug and Iterate: If necessary, debug any issues that arise from your changes and iterate on the design or functionality until you're satisfied with the result.

Remember to follow best practices such as using descriptive variable names, organizing your code logically, and testing your changes thoroughly to ensure they work as intended.