# **React JS (Module – 9 React Components, State, Props**

**(Q.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.

**(Q.2) What is NPM in React Js?**

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

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

**(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.

1. **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.
2. **Development Server:** Node.js can serve as a development server for running and testing React applications locally.
3. **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.

**(Q.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, 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:

* npx create-react-app reacttask

- Starting the Development Server:

* cd reacttask
* npm start

- Building the Production Build:

* npm run build

- Running Tests:

npm test

**(Q.5) What is Components in React Js?**

**(ANS):**

In React.js, components are the building blocks used to create user interfaces. A component is a JavaScript class or function that encapsulates a part of the UI.

Key points about components in React.js:

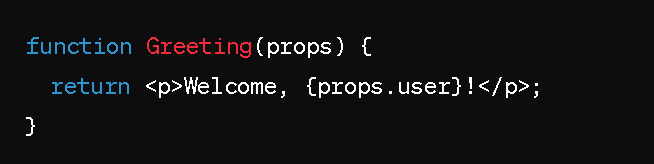
1. **Reusable:** Components can be reused throughout an application, promoting code reusability and maintainability.

1. **Modular:** Each component represents a specific piece of UI, making the application structure more modular and easier to manage.

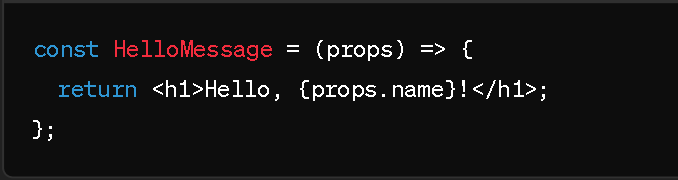
1. **Hierarchical:** Components can be nested within other components, forming a hierarchical structure that represents the UI's layout and functionality.
2. **State Management:** Components can manage their own state using `useState` hook (for functional components) or `this.state` (for class components), allowing them to update and re-render based on changes.
3. **Lifecycle Methods:** Class components have lifecycle methods (e.g., `componentDidMount`, `componentDidUpdate`) that allow developers to perform actions at specific points during a component's lifecycle.

Examples of React components:

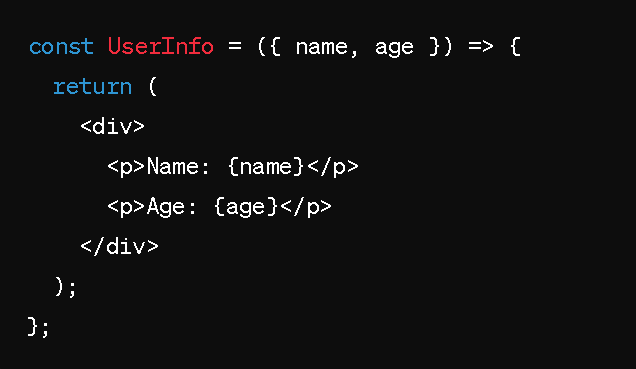
1) Functional Component (using Arrow Function):



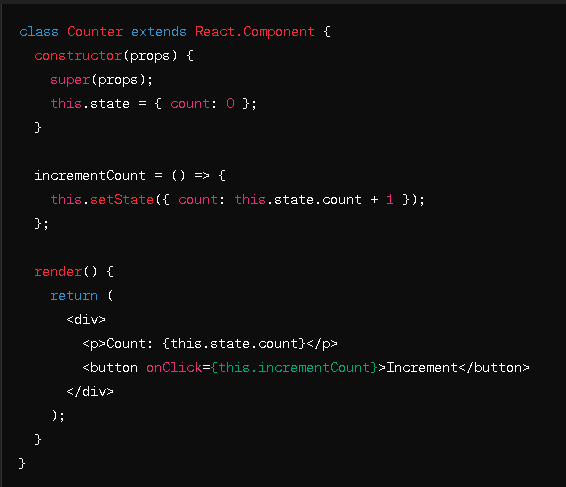
2) Functional Component (using Function Declaration):



3) Functional Component with Destructuring (ES6):



4) Class Component:



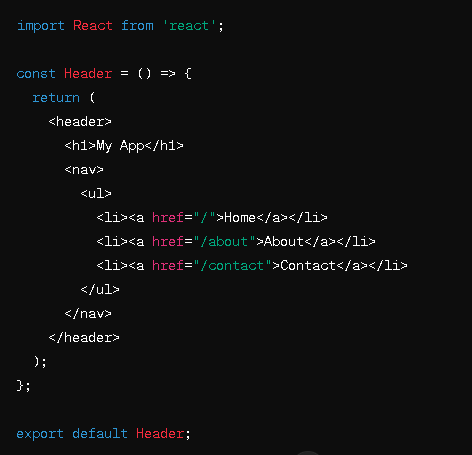
**(Q.6) What is Header and Content Components in React Js?**

**(ANS):**

In React.js, "Header" and "Content" components are common UI components used to structure and organize the layout of a web application.

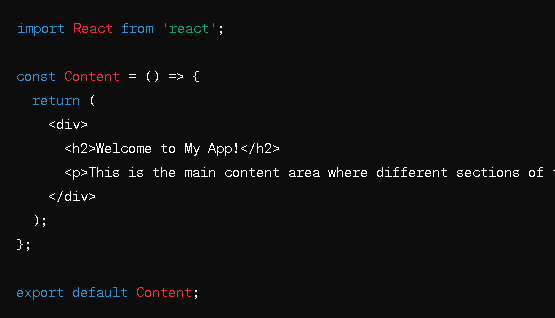
**1. Header Component:**

The Header component typically represents the top section of a webpage or application layout. It often includes elements such as navigation menus, branding (e.g., logo), and other UI elements that remain consistent across multiple pages or views within the application

 Example of a simple Header component:

**2. Content Component:**

The Content component represents the main content area of a webpage or application. It is responsible for rendering the primary content specific to each page or view. This could include text, images, forms, lists, or other components that make up the main body of the application's interface.

 Example of a simple Content component:

**(Q.7) How to install React Js on Windows, Linux Operating System? How to Install**

**NPM and How to check version of NPM?**

**(ANS):**

Installing React.js and npm (Node Package Manager) on both Windows and Linux operating systems is quite similar. Here are the steps:

**### Installing React.js and npm:**

**1. Install Node.js:** React.js depends on Node.js, so you need to install it first. You can download the installer for your operating system from the official Node.js website: [Node.js Download](https://nodejs.org/).

**2. Check Installation:** After installation, you can verify that Node.js and npm are installed correctly by opening a terminal (Command Prompt on Windows or Terminal on Linux) and typing the following commands:

**node -v**

**npm -v**

This will display the installed version of Node.js and npm respectively. If you see version numbers, it means Node.js and npm are successfully installed.

**3. Create a React App:** Once npm is installed, you can create a new React app using `create-react-app`, a command-line tool provided by Facebook for creating React applications.

npx create-react-app my-react-app

Replace `my-react-app` with the name you want for your React application.

4. **Navigate to the App Directory**: Change your current directory to the newly created React app directory.

**cd my-react-app**

**5. Start the Development Server:** Once inside your React app directory, start the development server using npm.

npm start

This command will start the development server and open your default web browser to view your React application running locally.

### Checking npm Version:

To check the version of npm, you can simply run the following command in your terminal:

npm -v

**(Q.8) How to check version of React Js?**

**(ANS):**

To check the version of React.js installed in your project, you can navigate to your project directory in the terminal and run the following command:

npm list react

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

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

**(ANS):**

To change components in React.js:

**1. Identify Component**: Locate the component you want to change within your React project. Components are typically stored in separate files.

**2. Edit Component File**: Open the file containing the component you want to change using a text editor or an IDE.

**3. Make Changes:** Modify the JSX code within the component file to reflect the desired changes. You can update text, styles, props, state, or any other aspect of the component's behavior.

**4. Save Changes:** Once you've made the necessary modifications, save the file.

**5. View Changes:** If your development server is running (you can start it using `npm start`), the changes will automatically be reflected in your React application. If not, start the server to see the updated component in action.

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

**(ANS):**

To create a list view in React.js:

1. Define a component for each item in the list.

2. Create a parent component to hold and manage the list.

3. Use the map function to iterate over your data and render each item component.

4. Pass data to each item component as props.

5. Style your list components as needed with CSS.

**Example**

// Item component

const Item = ({ text }) => {

return <li>{text}</li>;

};

// Parent component

const ListView = ({ data }) => {

return (

<ul>

{data.map((item, index) => (

<Item key={index} text={item} />

))}

</ul>

);

};

// Usage

const App = () => {

const myList = ["Item 1", "Item 2", "Item 3"];

return <ListView data={myList} />;

};