**React JS Assignments**

**What is React Js?**

Ans =

* An open-source javascript package called React.js, or just React, is mostly used for creating user interfaces for online applications. In 2013, Facebook developed it and made it available to the general public. With React, developers can design dynamic and reactive user interface components that quickly update and render modifications in response to changes in the application's data or state.

Key features and concepts of React.js include:

* Component-Based Architecture: Reusable user interface components are used in the construction of React apps. Components can be used to form sophisticated user interfaces, and each one maintains its own state.
* Document Object Model (or virtual DOM): React makes use of a virtual DOM to enhance the efficiency of updating the real DOM. React minimizes performance bottlenecks by determining the most effective way to update the virtual DOM when UI changes are made, then updating the actual DOM only when necessary.
* JSX (javascript XML): React leverages JSX, a syntax extension that lets programmers write javascript code that looks like HTML. This facilitates the articulation of UI component structure in a more declarative manner.
* React operates on a unidirectional data flow, in which information moves from parent components to child components. This keeps the state of the application predictable and facilitates debugging and comprehension of the data flow inside the application.
* React Hooks: Originally released in React version 16.8, hooks are methods that let developers access React capabilities, such as state, in functional components. Previously, these features were limited to class components. In React apps, hooks offer a more succinct and practical way to manage state and side effects.
* Lifecycle Methods for Components: React components include lifecycle methods that let developers hook into various lifecycle stages, like mounting, updating, and unmounting. This makes it possible for developers to carry out operations like data fetching, UI updates, and resource cleanup when needed.
* React.js's simplicity, speed, and robust community support have contributed to its considerable rise in popularity among developers. It frequently works in tandem with other frameworks and tools, such Redux for state management and React Router for single-page application routing.

**What is NPM in React Js?**

* NPM is an acronym for Node Package Manager in React.js. It is the largest software registry in the world and a package management for javascript. Installing, sharing, and managing dependencies is done in javascript projects—including React.js apps—using NPM.
* When working with React.js, developers frequently utilize NPM to install several packages and libraries, such Axios for HTTP requests, Redux for state management, react itself, and React Router for routing, Helmet, Bootstrap, among many others, that are necessary for creating React apps.
* By offering a command-line interface for installing, updating, and managing versions of packages, NPM streamlines the process of managing dependencies. A file called package.json, which contains a list of all the dependencies and their versions needed for the project, also aids in the efficient definition and management of dependencies for the project.

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

* Server-side Development: javascript may be executed on the server side by developers using the Node.js runtime environment. While Node.js is used to develop server-side functionality, apis, and manage server-side rendering (SSR) of React components, React.js is largely used to build client-side user interfaces.
* API Development: restful apis and graphql endpoints that supply data to React.js front-end applications are frequently made with Node.js. These apis are capable of processing requests made by the client-side React application, retrieving information from databases or outside sources, and responding to the client.
* Integration with Build Tools: The development of React.js makes extensive use of the vast ecosystem of npm packages and build tools offered by Node.js. Node.js is usually used to run tools like webpack, Babel, and eslint, which are frequently used in React.js projects for code bundling, transpilation, and linting.
* Server-side Rendering (SSR): React components can be rendered server-side using Node.js. Instead of depending solely on client-side rendering, SSR generates the first HTML on the server and sends it to the client, improving both the speed and search engine optimization (SEO) of React apps.
* Development Environment: Local development servers and build scripts in React.js applications frequently use Node.js as their runtime environment. Using the aid of tools like npm or yarn scripts, developers may automate a variety of development operations, operate development servers, and compile JSX and javascript code using Node.js.

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

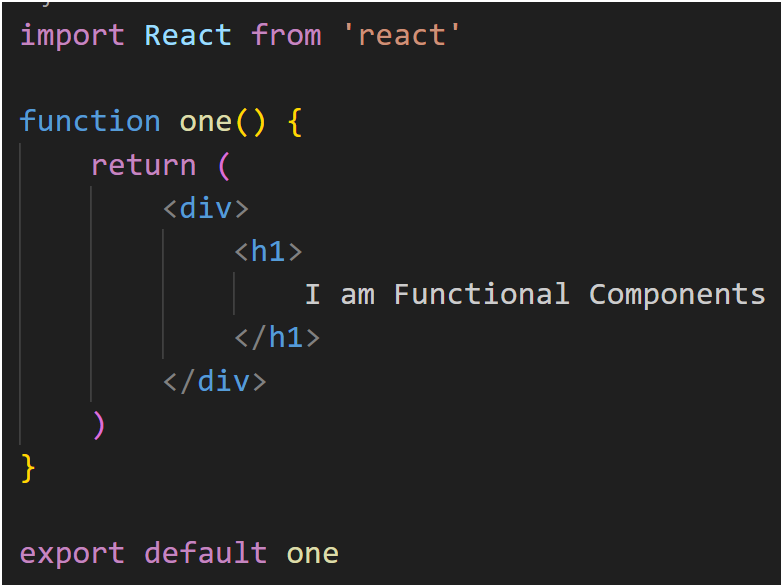
* Create React App is one of the most widely used aCLI tools for React.js development. A command-line program called Create React App creates a new React.js project with an already-configured development environment. It gives you a folder structure, configuration files, and all the dependencies you need to get started with React development right away.
* The following are some often used CLI commands for creating React.js apps using Create React App:

Npx create-react-app my-app

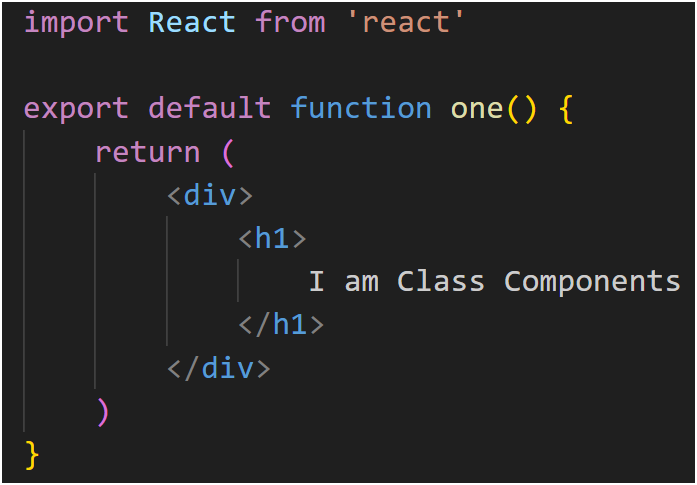
* Npm start: This command opens the React application in the browser and starts the development server. The application can then be viewed and tested locally by developers.
* Npm run build: This command bundles and optimizes the code to create a production-ready build of the React application. This command's output is usually utilized for deployment.
* Npm test: This command uses testing frameworks like Jest to run the test suites related to the React application.
* Npm run eject: This command allows developers to modify the build configuration by removing the Create React App setup and exposing all configuration files.
* Npm SASS : npm -g i sass This command use for install sass dependency.
* Npm Bootstrap : npm install react-bootstrap bootstrap This command use for install bootstrap dependency. Which Allow to Create web page with bootstrap class it’s very easy to use for developers.
* Build-react-app: You can start a new React.js project using this command. As an illustration:

**What is Components in React Js?**

* There are two main types of components in React.js:
* Functional Components: JSX (JavaScript XML), which describes the user interface, is returned by these JavaScript functions. Simpler components that don't need to maintain state or lifecycle methods are usually classified as functional components. They also go by the names presentational or stateless components.   
    
  A example of a Functional components is:



* Class Components: These are classes in ES6 that extend the React.Component class. More sophisticated components that must manage state, undertake lifecycle operations, or carry out other tasks like obtaining data from APIs are classified as class components.   
  An example of a class component is:



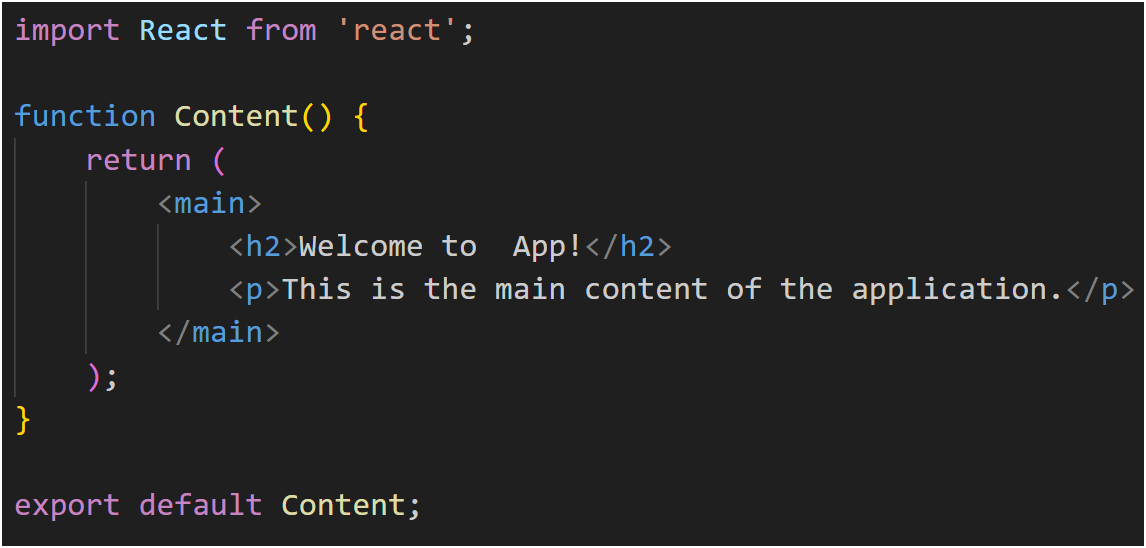
**What is Header and Content Components in React Js?**

* Part of the header: The top portion of an application's user interface is usually represented by the Header component, which frequently has items like a search bar, navigation menu, logo, and other material that is the same on all pages or views of the program.
* To offer a consistent user experience across many pages or views in the application, the reusable Header component can be added at the top.   
  An example of React code for a simple Header component would be as follows:



**Component of Content:**

* The main area of a user interface where the majority of the application's content or functionality is seen is represented by the Content component. The content of the Content component can alter dynamically in response to user interactions, route modifications, and other circumstances, in contrast to the Header component, which frequently stays static across many views.   
  An example of React code for a simple Content component may be as follows:



**How to install React Js on Windows, linux Operating System? How to install NPM and How to check version of NPM?**

**Installing React.js and npm on Windows:**

1. **Install Node.js:**

* Go to the [Node.js website](https://nodejs.org/).
* Download the latest version for Windows.
* Run the installer and follow the instructions.

1. **Install React.js using npm:**

* Once Node.js is installed, open Command Prompt or PowerShell.
* To install React.js globally, run:

npm install -g create-react-app

1. Create a new React app:

* To create a new React app, run:

npx create-react-app my-app

* Replace "my-app" with your desired project name.

1. **Navigate to your project directory:**

Go to your project directory:

cd my-app

Installing React.js and npm on Linux:

Install Node.js and npm:

Open a terminal.

Update the package index:

sudo apt update (in terminal)

sudo apt install nodejs npm (in terminal)

Once Node.js and npm are installed, follow the same steps as in the Windows section.

**Checking npm Version:**

To check the version of npm installed on your system, open a terminal or command prompt and run:

npm -v

This command will display the version of npm installed.

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

* Using npm = Open your terminal or command prompt, navigate to your project directory, and run one of the following commands. Code I terminal = npm list react
* Using Developer Tools in a Browser: You can also use the Developer Tools to determine the version of React.js while you're using a web browser to execute a React application. Launch the application through a web browser. To access Developer Tools, either right-click on the page and choose "Inspect" or press Ctrl + Shift + I (or Cmd + Option + I on Mac).   
  Navigate to the "Console" menu. Input React.version and hit Return. This will show you the version of React.js is being used by your application.
* Also check package.jason file for check version of React Js

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

**State Administration:**   
  
Use state if the changes made to the component are internal and don't need to be shared with other components.  
If you're using functional components, you can use the useState hook or initialize state in the constructor.   
  
Use the function returned by useState or the setState method to update the state.

import React, { useState } from 'react';

function MyComponent() {

const [counter, setCounter] = useState(0);

const incrementCounter = () => {

setCounter(counter + 1);

};

return (

<div>

<p>Counter: {counter}</p>

<button onClick={incrementCounter}>Increment</button>

</div>

);

}

export default MyComponent;

Props:

Props should be used if the modifications made to the component rely on outside information or properties that are passed down from its parent.  
Usually, you modify the data passed by the parent component in order to modify the props.

***Lists and Hooks***

**Explain Life cycle in Class Component and functional component with Hooks.**

**Class Components:**

* **Phase of Mounting:**  
  constructor(): This function is triggered during the creation of a component instance.  
  componentWillMount() is no longer supported and should not be used. carried out immediately prior to the component mounting to the DOM.   
    
  Render the component with render().   
  After the component is mounted to the DOM, componentDidMount() is called. Perfect for creating subscriptions or making API calls.
* **Update Stage:**  
  componentWillReceiveProps(nextProps): This function is deprecated and should not be used. called whenever a new set of props is given to the component.  
  ought toComponentUpdate(nextProps, nextState): This function decides whether or not to render the component again.   
    
  componentWillUpdate(nextProps, nextState): This function is deprecated and should not be used. called immediately before rendering in the event that new props or states arrive.   
  render(): Renders the element once more.   
  componentDidUpdate(prevProps, prevState): Called right away following an update. Excellent for responding to prop or state changes by updating the DOM.
* **Phase of Unmounting:**  
  componentWillUnmount(): This function is called just before a component is destroyed and unmounted. Perfect for clearing out network requests and getting rid of event listeners.

**Functional Components with Hooks:**

* **Phase of Mounting:**  
  hook for adding state to functional components is useState().  
  useEffect(() => {}, []): Hook that combines the functionality of componentDidUpdate, componentDidMount, and componentWillUnmount. replaces componentDidMount and componentDidUpdate and runs after each render. To ensure that the effect only occurs when those dependencies change, dependencies can be specified using the second argument.
* **Update Stage:**  
  Updates in the functional component are brought about by re-renders due to updates in the props or state.  
  Similar to the updating stage in class components, useState() and useEffect() can be used to handle state updates and side effects, respectively.
* **Phase of Unmounting:**   
    
  The cleanup function that is supplied to useEffect() can be used to carry out cleanup operations. Prior to the component being taken out of the user interface, this function executes.

***Module-5) React - Styling & Advance React***

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

**&**

**Create Increment decrement state change by button click?**

**&**

**Create Shopping site home page**

ANS = <https://github.com/kbu09/REACT_TASK/tree/main/ALL_TASK>