**MODULE – REACTJS**

**Q.1 What is ReactJS?**

ReactJS is a JavaScript library for building user interfaces, particularly for single-page applications where UI updates are frequent. Developed and maintained by Facebook, React allows developers to create reusable UI components that update efficiently and automatically when data changes. Its key feature is the virtual DOM (Document Object Model), which enables efficient rendering by updating only the parts of the actual DOM that need to change, rather than re-rendering the entire page.

React follows a component-based architecture, making it easier to manage and scale complex UIs. It's widely used for building interactive and dynamic web applications, providing a declarative syntax for defining how the UI should look based on the application state.

**Q.2 What is NPM in ReactJS?**

In React JS, NPM serves as a package manager that simplifies the process of handling external libraries and tools. It's used to define and manage project dependencies, such as React itself or other packages your project relies on. The `package.json` file is where you list these dependencies, and then using the `npm install` command, you can easily fetch and install them. This ensures that different developers working on the same project have a consistent set of dependencies, making collaboration smoother.

**Q.3 What is role of Node JS in React JS?**

Got it, let me simplify that. Node.js is often used with React for:

**1. Server-Side Rendering (SSR):** Makes React apps load faster by rendering on the server.

**2. Build Tools and Package Management:** Node.js powers tools like npm and yarn, managing dependencies and scripts.

**3. Development Server:** Sets up a server for easy React development, used in tools like Create React App.

**Q.4 What is CLI command in React JS?**

The main CLI command in React JS is **`create-react-app`**. It sets up a new React project with a default configuration. Use it like this:

npx create-react-app your-app-name

Then navigate into the project folder and start the development server:

cd your-app-name

npm start

**Q.5 What is components in React JS?**

In React JS, components are reusable and self-contained pieces of code that define specific parts of a user interface. They can be either functional or class-based, encapsulating functionality and structure. Components make it easier to manage and update different parts of a React application.

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

In React JS:

- **Header Component:** Contains elements like logos and navigation menus at the top of the page.

function HeaderComponent() {

return (

<header>

<h1>My App</h1>

<nav>...</nav>

</header>

);

}

- **Content Component:** Represents the main content area of a page.

function ContentComponent() {

return (

<main>

<p>Welcome to my app!</p>

{/\* Other dynamic content \*/}

</main>

);

}

These components organize and structure the UI in a React application.

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

Install Node.js and NPM:

**Windows:**

- Download Node.js installer from [Node.js Downloads] (https://nodejs.org/en/download/).

- Run the installer and follow instructions.

**Linux:**

- Use package manager (e.g., `apt` on Ubuntu):

sudo apt-get update

sudo apt-get install nodejs npm

- Alternatively, use ‘nvm’:

curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.39.1/install.sh

nvm install node

**Check Versions:**

In terminal or command prompt:

node -v

npm -v

**Install React:**

npx create-react-app my-react-app

cd my-react-app

npm start

Access React app at ‘http://localhost:3000’.

**Q.8 How to check version of React JS?**

To check the version of React JS in project:

1. Open a terminal.

2. Navigate to React project.

3. Run:

npm list react

Or check the "dependencies" in **‘package.json’** file.

**Q.9 How to change in components of React JS?**

To make changes to components in React JS:

**1. Edit Component Code:**

- Open the file containing React component.

- Modify the JSX code to make the desired changes.

**2. Save the File:**

- Save the changes in code editor.

**3. Check the Result:**

- If React app is running, the changes should automatically reflect in the browser.

- If not, start or restart the development server using **‘npm start’**.

For example, we want to change text in a component:

// Before

function MyComponent () {

return <div>Hello, React! </div>;

}

// After

function MyComponent () {

return <div>Hello, Updated React! </div>;

}

These are general steps, and the specific process might depend on the development environment or tooling.

**Q. 10 How to create a list view in React JS?**

It's a good practice to use a unique identifier for each item instead of the array index. If your data items have a unique identifier (like an `id`), you can use that to ensure stability in the rendering. Here's an updated example:

**ListView.js:**

import React from 'react';

function ListView({ items }) {

return (

<ul>

{items.map (item => (

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

))}

</ul>

);

}

export default ListView;

**App.js:**

import React from 'react';

import ListView from './ListView';

function App () {

const data = [

{id: 1, name: 'Item 1’},

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

{id: 3, name: 'Item 3’},

];

return (

<div>

<h1>List View Example</h1>

<ListView items={data} />

</div>

);

}

export default App;

In this example, each item in the ‘data’ array has an **‘id’**, which is used as the **‘key’** in the **‘map’ function**. Using a unique identifier helps React efficiently update and render the list when changes occur.

**MODULE – Lists and Hooks**

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

In React, components have a lifecycle that consists of various phases during which you can execute code. The lifecycle methods differ between class components and functional components with hooks. Here's a brief overview:

**Class Component Lifecycle:**

1. Mounting Phase:

- `constructor ()`: Initializes the component state and binds event handlers.

- `static getDerivedStateFromProps()`: Used to update the state based on changes in props.

- `render ()`: Renders the component.

- `componentDidMount()`: Invoked after the component is mounted to the DOM. Useful for making network requests or initializing subscriptions.

2. Updating Phase:

- `static getDerivedStateFromProps ()`: Similar to the mounting phase, used to update state based on changes in props.

- `shouldComponentUpdate()`: Determines if the component should re-render. Defaults to `true`.

- `render()`: Renders the component.

- `getSnapshotBeforeUpdate()`: Captures information from the DOM before an update.

- `componentDidUpdate()`: Invoked after the component is updated in the DOM.

3. Unmounting Phase:

- `componentWillUnmount()`: Invoked just before a component is unmounted and destroyed. Used for cleanup.

**Functional Component with Hooks Lifecycle:**

1. Mounting Phase:

- `useState()`: Initializes state.

- `useEffect(() => {}, [])`: Equivalent to `componentDidMount`. Runs after the first render.

2. Updating Phase:

- `useState()`: Updates state.

- `useEffect(() => {})`: Equivalent to `componentDidUpdate`. Runs after every render.

3. Unmounting Phase:

- `useEffect(() => { return () => {} }, [])`: Equivalent to `componentWillUnmount`. Cleans up effects from the component.

In functional components with hooks, the `useEffect` hook is a versatile replacement for various lifecycle methods in class components. The second argument to `useEffect` allows you to specify dependencies that trigger the effect, similar to how `shouldComponentUpdate` works.

Functional components with hooks are the modern approach in React, providing a more concise and readable way to manage component lifecycles and state.

**MODULE – React – Applying Redux**

**Q.1 What is Redux?**

Redux is a state management library for React that centralizes and manages the state of an application in a predictable way. It uses actions, reducers, and a single store to handle state changes, making it easier to manage and debug application state.

**Q.2 What is Redux Thunk used for?**

Redux Thunk is middleware for Redux that enables action creators to return functions instead of plain objects. This is handy for handling asynchronous operations, like API calls, within Redux actions.

**Q.3 What is Pure Component? When to use it over Component?**

It is a feature that we can import in React. These components are only used with class based components as compare to “useMemo” state which is used with functional components.

The main advantage of using it over components is that it is used to stop/block the re-rendering of a component. Pure component basically does a comparison check of same states if both the state initial values are same then it won’t re-render and if both times the values are different then it will allow the re-rendering process of a component.

**Q.4 What is the second argument that can optionally be passed tosetState and what is its purpose?**

The second argument that can optionally be passed to `setState` in React is a callback function. This callback is executed after the state has been updated and the component has been re-rendered.

The purpose of this callback is to perform actions or trigger code that relies on the updated state. Since `setState` is asynchronous, using the callback ensures that you are working with the most recent state.

example:

this.setState({ count: this.state.count + 1 }, () => {

console.log('State updated:', this.state.count);

});

In this example, the callback function will be executed after the `setState` operation is completed, allowing to safely access the updated state.