Module 18) Reactjs

1. Introduction to React.js

Q1) What is React.js? How is it different from other JavaScript frameworks and libraries?

Rect js is an libraries of javascript

React js is a product of facebook

React js libraries are managed and developed by facebook engineers

React js is fast load data on web browsers.

React is used to load data on web browsers without page refresh or reload.

React js provides many features and libraries.

React js is an libraries of frontend developments.

React js is used a life cycle to create web applications of website

Life cycle of React js

a)initializations

b)mounting

c)updating

d)unmounting

libraries: Gives you freedom to choose tools and architecture. Lightweight and focuses on specific functionality.

Frameworks: Provides pre-defined structures and conventions. Larger and more feature-complete.

Q2) Explain the core principles of React such as the virtual DOM and componentbased architecture.

React is a popular JavaScript library for building user interfaces,

particularly for single-page applications.

Its core principles include the Virtual DOM, Component-Based Architecture

Virtual Dom:

The Virtual DOM is a lightweight, its representation of the actual DOM.

Instead of directly manipulating the DOM,

React uses the Virtual DOM to make updates more efficient.

Component-Based Architecture:

React encourages breaking down the user interface into small, reusable, and self-contained pieces called components.

Each component represents a specific part of the UI, such as a button, a form, or a navigation bar.

Q3) What are the advantages of using React.js in web development?

It is composable.

It is declarative.

Write once, and learn anywhere.

It is simple.

SEO friendly.

Fast, efficient, and easy to learn.

It guarantees stable code.

It is backed by a strong community.

2. JSX (JavaScript XML)

Q1) What is JSX in React.js? Why is it used?
JSX :JSX is a syntax extension for JavaScript that allows you to write HTML-like code in your JavaScript files

Use:

readability: Makes React code more readable and easier to understand. developer Experience: Improves the way UI structures are defined. error Handling: Provides better error messages. Integration: Seamlessly embeds JavaScript expressions within HTML-like code.

Q2) How is JSX different from regular JavaScript? Can you write JavaScript inside

JSX?

JSX :HTML like syntax within JavaScript.

Regular JS : Uses JavaScript objects to create and manipulate DOM. so it's diffrent.

Yes, you can write JavaScript inside JSX using curly braces {}.

Q3) Discuss the importance of using curly braces {} in JSX expressions

JSX lets you write HTML-like markup inside a JavaScript file, keeping rendering logic and content in the same place.

Sometimes you will want to add a little JavaScript logic or reference a dynamic property inside that markup.

In this situation, you can use curly braces in your JSX to open a window to JavaScript.

```
function Greeting(props) {
 return <h1>Hello, {props.name}!</h1>;
}
```

3. Components (Functional & Class Components)

Q1) What are components in React? Explain the difference between functional components and class components.

Components :Reusable pieces of code that represent UI elements or parts of an application.

- functional components :

A simple Javascript function that accepts props as input and returns JSX

initially stateless, but can now manage state using React Hooks.

generally simpler and more concise.

limited lifecycle hooks available, but can now use Hooks for more complex lifecycle management (e.g., useEffect).

- class components :

a JavaScript class that extends React.Component.

manages internal state using the this.state object.

can be more complex due to the use of this, constructors, and lifecycle methods.

access to a full range of lifecycle methods (e.g., componentDidMount, componentDidUpdate).

Q2) How do you pass data to a component using props?

- Define the Prop in the Child Component :

EXAMPLE :

```
function Welcome(props) {
  return <h1>HI!, {props.name}</h1>;
}
```

- Pass the Prop from the Parent Component :

EXAMPLE :

```
function App() {
  return (
    <div>
      <Welcome name="RAJ" />
      <Welcome name="PATEL" />
    </div>
  );
}
```

- Access the Prop in the Child Component :

EXAMPLE

```
function Welcome(props) {
  return <h1>HI!, {props.name}</h1>;
}
```

Q3) What is the role of render() in class components?

Defining the UI : It describes what the component should look like on the screen by returning JSX

Accessing data: It accesses and uses props and state to determine the component's output.

Controlling output: It controls the structure, content, and appearance of the component's UI.

4. Props and State

Q1) What are props in React.js? How are props different from state?

props stands for properties

props are arguments passed into react js components

props are passed as arguments into react function components

props are passed to components via html attributes

props to pass data into a component.

state to manage data inside a component.

props is a Immutable.

state is a mutable.

Q2) Explain the concept of state in React and how it is used to manage component data.

state is an inbuild object in react js

state pass data one components to another components

state is pass a data in a unidirectional in components

state provides immutable objects inside of components

state is used to update data one components to another components

we intialised state using hook or using in function based components only.

state can not used inside of class based components because of hooks in no more support class based components .

Q3) Why is this.setState() used in class components, and how does it work?

use the setState() method to change the state object

it will ensure that the component knows it’s been updated

and calls the render() method (and all the other lifecycle methods).

In class components, this.setState is used to update the component's state.

React schedules a re-render of the component whenever the state is updated.

this.setState({ key: newValue });