1.Introduction to React.js

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

React js in a library of javascript.

React is built around the concept of components, where each component is a self-contained unit with its own logic and view.

It uses state approach. Like for different UIs different states. If state changes UI will change.

React uses virtual DOM which is a lightweight copy of the real DOM. Changes are first made to the virtual DOM and then efficiently reconciled with the actual DOM

• Question 2: Explain the core principles of React such as the virtual DOM and component based architecture.

Virtual DOM: It is much light weight copy of actual DOM. When a component’s state changes React first updates changes. Then comparison is done with previous version of virtual DOM and current version of virtual DOM And then applies minimum number of changes to the virtual DOM.

Component based architecture:

In React, everything is a component. Components are self-contained units that manage their own state and render the UI. Components can be either class-based or function-based

• Question 3: What are the advantages of using React.js in web development?

React’s component-based structure allows you to reuse code across different parts of your application, making development faster and more maintainable.

The virtual DOM ensures that only the necessary parts of the UI are re-rendered when the state changes, leading to improved performance.

React has a huge and active community of developers, which means you have access to a wealth of open-source libraries, tools, and resources.

React provides a set of developer tools that allow you to inspect the component tree, monitor state and props, and track performance issues.

2.JSX

Question 1: What is JSX in React.js? Why is it used?

JSX (JavaScript XML) is a syntax extension for JavaScript used in React.js

It makes the code more readable and easier to understand, since the UI structure is written in a syntax similar to HTML.

It is allowing developers to describe what the UI should look like based on the current state or data.

It combines the power of JavaScript and HTML-like structure.

• Question 2: How is JSX different from regular JavaScript? Can you write JavaScript inside JSX?

JSX is similar to regular JavaScript, but it allows you to write HTML-like syntax in JavaScript code.

In JSX, anything inside curly braces {} is evaluated as JavaScript. So, if you want to use JavaScript expressions (like variables, functions, or even simple calculations), you can do that inside JSX.

const name = "Pinjal";

const element = <h1>Hello, {name}!</h1>;

• Question 3: Discuss the importance of using curly braces {} in JSX expressions.

Curly braces {} in JSX are used to embed JavaScript expressions inside the JSX.User can ingect variables, call functions, conditional logic, loop through array and different elements inside {}.

const age = 25; const element = <p>You are {age} years old.</p>

3.Components

Question1:What are components in React? Explain the difference between functional components and class components

Components are the building blocks of a React application. They are reusable, self-contained units that define how a UI should appear.

Difference between Functional Components and Class Components:

Functional Components:

* + Definition: A functional component is a simple JavaScript function that accepts props as an argument and returns a React element (JSX).
  +  Syntax: Shorter and simpler to write.

Class Components:

* Definition: A class component is a class that extends React.Component and must have a render() method that returns a React element (JSX).

Syntax: Requires more boilerplate code (like constructor, render() method).

.

Question2: How do you pass data to a component using props?

Propes are used to pass the data from one component to another.

function Parent() {

return <Child name="John" age={30} />;

}

function Child(props) {

return (

<div>

<h1>Hello, {props.name}!</h1>

<p>Age: {props.age}</p>

</div>

);

}

Question 3: What is the role of render() in class components?

Every class component must have a render() method. It returns JSX that defines the UI of the component.

Whenever there is a change in state or props, React will call the render() method to re-render the component and reflect the changes in the UI.

4.Props and state

Question 1: What are props in React.js? How are props different from state?

Propes are used to pass the data from one component to another

Props are immutable, meaning they cannot be changed by the component receiving them.

State is used to manage data that changes within a component.

State is mutable and can be updated within the component.

State is owned by the component itself.

Question 2: Explain the concept of state in React and how it is used to manage component

data.

In React, state refers to an object that holds data or information that can change over the lifetime of a component. State is used to manage dynamic data that may change based on user interaction, server responses, or other factors.

const [stateVariable, setState] = useState(initialValue);

import React, { useState } from 'react';

function Counter() {

const [count, setCount] = useState(0); // Declare state variable

const increment = () => {

setCount(count + 1); // Update the state

};

return (

<div>

<p>Current count: {count}</p>

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

</div>

);

}

export default Counter;

Question 3:Why is this.setState() used in class components, and how does it work?

In React, when working with class components, the this.setState() method is used to update the state of the component.

import React, { Component } from 'react';

class Counter extends Component {

constructor(props) {

super(props);

this.state = {

count: 0

};

}

increment = () => {

this.setState((prevState) => ({

count: prevState.count + 1

}));

};

decrement = () => {

this.setState({

count: this.state.count - 1

});

};

render() {

return (

<div>

<p>Current Count: {this.state.count}</p>

<button onClick={this.increment}>Increment</button>

<button onClick={this.decrement}>Decrement</button>

</div>

);

}

}

export default Counter;