Module 7

MERNstack – React js

Question 1: What is React.js? How is it different from other JavaScript frameworks and libraries ?

Ans.

React.js is a **JavaScript library** for building user interfaces, especially single-page applications. It allows developers to create reusable UI components and manage the state efficiently, ensuring fast and responsive updates to the DOM.

1. **Library, not Framework:** Unlike full-fledged frameworks like Angular or Vue.js, React focuses only on the **view layer** (UI) in MVC.
2. **Virtual DOM:** React uses a **Virtual DOM**, which improves performance by minimizing direct DOM manipulation and updating only what's necessary.
3. **Component-Based Architecture:** React emphasizes **reusable and declarative components**, making development more modular and maintainable.
4. **Unidirectional Data Flow:** React ensures a clear **one-way data flow**, simplifying debugging and improving predictability.

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

Ans.

1. **Virtual DOM:**

- The Virtual DOM is a lightweight copy of the real DOM.
- React uses it to efficiently update the UI.
- Instead of directly updating the real DOM, React updates the Virtual DOM first, calculates the difference (diffing algorithm), and applies minimal changes to the real DOM.
- This process improves performance, especially for large applications.

How to explain to the interviewer:

"React uses a Virtual DOM to optimize updates. It identifies changes in the UI and applies only the necessary updates to the real DOM, which makes it faster and more efficient."

2. **Component-Based Architecture:**

- React divides the UI into **independent, reusable components**.
- Each component is like a small, isolated piece of the UI, making the application modular and maintainable.

- Components can have their own state and logic, and they are combined to build complex interfaces.

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

Ans.

1. **Component-Based Architecture:**
   React allows you to build reusable, modular UI components, making development faster and easier to maintain.

2. **Fast Performance with Virtual DOM:**
   React updates only the necessary parts of the real DOM using its Virtual DOM, ensuring smooth and efficient UI rendering.

3. **Declarative Syntax:**
   React's declarative approach makes it easy to describe how the UI should look and behave, improving readability and debugging.

4. **Rich Ecosystem and Community Support:**
   With a large developer community and numerous third-party libraries, React has solutions for almost every need (e.g., routing, state management).

5. **Unidirectional Data Flow:**
   React's one-way data binding makes the application state predictable and easier to debug.

6. **Cross-Platform Development:**
   React Native, built on React.js, allows developers to build mobile applications for iOS and Android using the same concepts.

7. **SEO-Friendly:**
   React supports server-side rendering (SSR), which helps improve the SEO of single-page applications.

JSX (JavaScript XML)

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

Ans.

JSX (JavaScript XML) is a syntax extension for JavaScript that allows you to write HTML-like code directly within JavaScript. It is used in React to describe what the UI should look like.

```
const element = <h1>Hello, World!</h1>;
```

This code looks like HTML but is converted to JavaScript by React during the build process.

Why is it used ?

1. **Improves Readability:**

   JSX makes it easier to write and understand the structure of components by combining HTML-like syntax with JavaScript.

2. **Integration with JavaScript:**

You can embed JavaScript expressions directly into JSX using curly braces { } for dynamic content.

3. **React-Specific Features:**

   JSX works seamlessly with React's component-based architecture, allowing developers to pass props and manage states easily.

4. **Babel Compilation:**

   JSX is transpiled by tools like Babel into plain JavaScript, making it browser-compatible.

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

Ans.

1. **HTML-Like Syntax:**
   JSX allows writing HTML-like syntax directly in JavaScript. Regular JavaScript doesn't have this capability.

2. **Requires Compilation:**
   JSX needs to be transpiled (e.g., by Babel) into plain JavaScript before browsers can understand it, while regular JavaScript runs directly in the browser.

3. **Integration with UI:**
   JSX is specifically designed for building React components, making UI code more declarative and intuitive, whereas regular JavaScript is general-purpose.

4. **Attributes and Syntax Differences:**
   JSX uses **camelCase** for HTML attributes (e.g., `className` instead of `class`). Additionally, JSX expressions are wrapped in curly braces `{}`.

Can you write JavaScript inside JSX?

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

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

Here, `{name}` is a JavaScript expression embedded inside JSX.

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

Ans.

In JSX, curly braces `{}` are used to embed **JavaScript expressions** directly into the markup. They are essential for creating dynamic content, making JSX more powerful and interactive.

Why Curly Braces {} Are Important:

1. **Dynamic Content Rendering:**
   Curly braces allow you to insert dynamic values (e.g., variables, function results) into the UI.

2. **Conditional Rendering:**
   They enable conditional rendering using expressions like ternary operators.

3. **Embedding Functions:**
   You can use curly braces to call JavaScript functions and display their return values.

4. **Dynamic Styling:**
   {} is used to pass JavaScript objects for inline styles.

Components (Functional & Class Components)

Question 1: What are components in React? Explain the difference between functionalcomponents and class components.

Ans.

React lets you create components, reusable UI elements for your app. In a React app, every piece of UI is a component. React components are regular JavaScript functions except: Their names always begin with a capital letter. They return JSX markup.

difference between functionalcomponents and class components.

Functional Component	Class Component
• Used for presenting static data	• Used for dynamic sources of data
• Can't handle fetching data	• Handles any data that might change (fetching data, user events, etc)

• Easy to write	• Knows when it gets erendered to the device(useful for data fetching)
•	• More code to write

`const Header = () => {`
`  return <Text>Hi there!</Text>`
`}`
	`class Header extends Component{`
	`  render(){`
	`  return<Text>Hi There</Text>`
	`  }`
	`}`

Question 2*: *How do you pass data to a component using props?

Ans.

To pass data to a component using **props** in React, follow these steps:

1. **Define the parent component** where you want to pass the data from.
2. **Pass data to the child component** by including it as an attribute in the child component tag.
3. **Access the props** inside the child component to use the passed data.

example:

Parent Component (App.js):

```
import React from 'react';
import ChildComponent from './ChildComponent';

function App() {
  const message = "Hello from Parent!";

  return (
    <div>
      <ChildComponent message={message} />
    </div>
  );
}

export default App;
```

Child Component (ChildComponent.js):

```
import React from 'react';

function ChildComponent(props) {
  return (
    <div>
      <h1>{props.message}</h1>
    </div>
  );
}

export default ChildComponent;
```

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

Ans.

Render in React JS is a fundamental part of class components. It is used to display the component on the UI returned as HTML or JSX components. The ReactDOM. render() function takes two arguments, HTML code and an HTML element.

Props and State

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

Ans.

- Props are arguments passed into React components.
- Props are passed to components via HTML attributes.

  props stands for properties.

- React Props are like function arguments in JavaScript *and* attributes in HTML.

- To send props into a component, use the same syntax as HTML attributes

State	Props
State is managed within the component	Props gets passed to the component
State can be changed(mutable)	Props are read only and cannot be changed(immutable)
State changes can be asynchronous	Props are read only
State is controlled by react components	Props are controlled by whoever by whoever renders the components
State can used to display changes with the component	Props are used to communicate between components

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

Ans.

State in React is a powerful feature for managing component-specific, dynamic data. It enables components to track and react to changes, providing a responsive and interactive user experience.

the `useState` Hook is used to manage state.

```
import React, { useState } from 'react';
```

```jsx
function Counter() {
  const [count, setCount] = useState(0); // Initialize state

  return (
    <div>
      <h1>Count: {count}</h1>
      <button onClick={() => setCount(count + 1)}>Increment</button> {/* Update state */}
    </div>
  );
}

export default Counter;
```

How State Helps Manage Component Data:

1. **Dynamic UI Updates :** State allows components to re-render when the data changes.

2. **Event Handling :** State is often updated in response to user actions, such as clicks, input changes, or form submissions.

3. **Conditional Rendering :** Based on the state, components can conditionally render parts of the UI.

4. **Data Sharing :** While state is local, it can be passed down as props to child components for broader use within an application.

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

Ans.

In **class components** in React, this.setState() is used to update the component's state. React's state management relies on immutability, meaning the state object cannot be modified directly. Instead, this.setState() is the method provided by React to.

1. **Update State :** Change the values in the state object.
2. **Trigger a Re-render** : Inform React that the component's state has changed, prompting React to re-render the component and update the UI accordingly.

How Does this.setState() Work ?

1. **Merge State Updates:**
   - this.setState() only updates the specified properties in the state object. It performs a **shallow merge** with the existing state, leaving other properties unchanged.

2. **Asynchronous Nature**:
   - this.setState() is asynchronous for performance optimization. React batches state updates to minimize re-renders.
   - Because of this, relying on this.state immediately after calling this.setState() might not give the updated value.

3. **State Update with Functions**:
   - When the new state depends on the previous state, it's recommended to use a **function updater** to ensure the correct value.

Syntax:

```
this.setState(updater, [callback]);
```

- **updater**: An object or function used to update the state.
- **callback** (optional): A function executed after the state update and re-render are complete.