Module 9 - Introduction to React.js

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

Answer :

React.js is a JavaScript library developed by Meta for building dynamic, interactive user interfaces. It uses a component-based architecture and Virtual DOM to enhance performance and simplify UI development.

1. Library vs. Framework: Focuses on the view layer, unlike full frameworks like Angular.
2. Virtual DOM: Updates only changed parts of the DOM for better performance.
3. Component-Based: UI is built with reusable, independent components.
4. Unidirectional Data Flow: Easier to debug compared to two-way binding in Angular.
5. JSX Syntax: HTML-like syntax in JavaScript for easier UI structure.
6. Ecosystem Flexibility: Allows custom tool selection; not opinionated.
7. Community Support: Vast resources, third-party tools, and tutorials.

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

Answer :

1. Virtual DOM

- A lightweight copy of the real DOM.
- React updates only the changed parts instead of the whole DOM, improving performance and ensuring smooth UI updates.

2. Component-Based Architecture

- UI is divided into reusable, self-contained components that manage their own logic and state.
- Promotes reusability, maintainability, and scalability.

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

Answer :

High Performance: Uses the Virtual DOM for faster and efficient updates.
Reusable Components: Simplifies development and enhances maintainability.
Unidirectional Data Flow: Easy to trace and debug changes.
JSX Syntax: Combines HTML and JavaScript for intuitive UI coding.
Large Ecosystem: Extensive libraries, tools, and community support.
Flexibility: Can be integrated with other tools and frameworks.
SEO-Friendly: Server-side rendering improves search engine visibility.

JSX (JavaScript XML)

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

Answer :

JSX (JavaScript XML) is a syntax extension for JavaScript that allows writing HTML-like code directly in JavaScript.

1. Readable and Intuitive: Combines HTML structure with JavaScript logic for better readability.
2. Ease of Component Creation: Simplifies building and visualizing React components.
3. Enhanced Developer Experience: Offers syntax highlighting and compile-time error checking.
4. React Integration: Translates JSX into `React.createElement()` calls for rendering.

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

Answer :

JSX (JavaScript XML) allows you to write HTML-like syntax directly within JavaScript. It is a syntax extension of JavaScript used in React to describe UI elements. Unlike regular JavaScript, JSX needs to be compiled (e.g., by Babel) into standard JavaScript before it can be executed by the browser.

Yes, you can write JavaScript expressions inside JSX by enclosing them in curly braces `{}`. For example:

```
const name = "Shakshi";
return <h1>Hello, {name}!</h1>;
```

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

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

Answer :

Curly braces `{}` in JSX allow embedding JavaScript expressions inside the JSX code. They are essential for dynamic content, enabling you to:

1. Inject Variables: Display variable values in the UI.
   Example: `<p>{userName}</p>`
2. Perform Calculations: Use expressions directly.
   Example: `<p>{2 + 3}</p>`
3. Call Functions: Invoke functions for dynamic rendering.
   Example: `<p>{getGreeting()}</p>`
4. Conditional Rendering: Display content conditionally.
   Example: `{isLoggedIn ? <p>Welcome</p> : <p>Please log in</p>}`

Without `{}`, JSX treats content as plain text, making it static.

Components (Functional & Class Components)

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

Answer:

Components are the building blocks of a React application. They are reusable pieces of code that define how a portion of the user interface (UI) should look and behave. Components can be functional or class-based.

Difference Between Functional and Class Components

Aspect	Functional Components	Class Components
Definition	Functions that return JSX.	ES6 classes that extend `React.Component`.
State Handling	Use hooks (e.g., `useState`, `useEffect`).	Manage state using `this.state`.
Lifecycle Methods	Achieved with hooks.	Use built-in lifecycle methods.
Code Simplicity	Simpler and more concise.	More verbose.
Performance	Slightly better performance due to no `this`.	Slightly slower due to `this` binding.

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

Answer:

In React, props (short for properties) are used to pass data from a parent component to a child component.

1. In the Parent Component: You pass data to a child component by adding attributes to the child component like HTML attributes.
2. In the Child Component: The child component accesses the passed data via the `props` object.

Example:

Parent Component:

```
function Parent() {
  const name = "Shakshi";
  return <Child name={name} />;
}
```

Child Component:

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

In this example:

- The `Parent` component passes the `name` variable as a prop to the `Child` component.
- The `Child` component receives the `name` via `props` and displays it in an `h1` element.

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

Answer:

Role of `render()` in Class Components:

- The `render()` method is a required method in class components.
- It returns JSX, which gets converted into HTML that the browser can render.
- It is called automatically whenever the component's state or props change to update the UI.

Example:

```
class Greeting extends React.Component {
  render() {
    return <h1>Hello, {this.props.name}!</h1>;
  }
}
```

In this example:

- The `render()` method returns JSX (`<h1>Hello, {this.props.name}!</h1>`) that React uses to update the UI.
- It can also include conditional rendering, loops, and other logic to decide what to render based on the component's state or props.

Key Points:

- `render()` must always return JSX or `null`.
- It doesn't modify the state; it simply renders the output based on the current state and props.

Props and State

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

Answer:

Props (short for "properties") are used to pass data from a parent component to a child component. They allow components to be dynamic and reusable by providing them with external values.

Difference Between Props and State

Aspect	Props	State
Purpose	Used to pass data from parent to child.	Used to store and manage data within a component.
Mutability	**Read-only**; cannot be modified by the child component.	**Mutable**; can be changed by the component itself using `setState()`.
Change Trigger	Changes when the parent component updates.	Changes can be triggered within the component itself.
Management	Managed by the parent component.	Managed within the component.

Example:

```
// Parent Component
function Parent() {
  return <Child message="Hello, World!" />;
}

// Child Component
function Child(props) {
  return <h1>{props.message}</h1>;
}
```

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

State is a data object used to store and manage dynamic information in a component. It determines how the component renders and behaves.

- **Initialization**: Set using `useState` (functional components) or `this.state` (class components).
- **Updating State**: Use `setState()` in class components or the setter from `useState()` in functional components.
- **Re-rendering**: React re-renders the component whenever state changes.

Functional Component:

```
const [count, setCount] = useState(0);
setCount(count + 1);  // Update state
```

Class Component:

```
this.setState({ count: this.state.count + 1 }); // Update state
```

State is mutable, and updates cause re-renders.

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

`this.setState()` is used in class components to update the component's state. It triggers a re-render of the component with the updated state, ensuring the UI reflects the changes.

How Does `this.setState()` Work?

1. State Update: It updates the state object, merging the new data with the current state.
2. Re-render: After updating, React re-renders the component to reflect the changes on the UI.
3. Asynchronous: The update is not immediate but scheduled, allowing React to optimize rendering.

Example:

```
this.setState({ count: this.state.count + 1 });  // Updates state
```