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

React.js is a popular open-source JavaScript library developed by Facebook for building user interfaces, especially single-page applications.

Differences from other frameworks/libraries:

React focuses exclusively on the view layer (UI) of applications (unlike Angular, which is a full MVC framework).

It uses a component-based architecture, allowing reusable, modular UI components.

React employs the virtual DOM for efficient UI updates.

2. Core Principles of React

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

Virtual DOM:

A lightweight in-memory representation of the actual DOM.

React updates the virtual DOM first, calculates the difference (diffing), and updates only the necessary parts of the real DOM, improving performance.

Component-Based Architecture:

React applications are built using reusable, encapsulated components.

Each component manages its own state and logic, leading to better code maintainability and scalability.

3. Advantages of Using React.js

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

Performance: The virtual DOM optimizes rendering.

Reusability: Components can be reused, reducing redundancy and improving development speed.

Declarative UI: Developers describe "what" the UI should look like, and React handles "how" to render it.

Large Ecosystem: Rich ecosystem with tools like React Router, Redux, and a strong community.

Cross-Platform: React Native extends React for mobile app development.

4. JSX in React.js

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

JSX (JavaScript XML): A syntax extension that allows writing HTML-like code in JavaScript.

Usage:

Makes it easier to visualize the component structure.

Transpiled to regular JavaScript during build time by tools like Babel.

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

JSX is not a string or HTML; it's syntactic sugar for React.createElement() calls.

You can embed JavaScript in JSX using curly braces {}.

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

Curly braces allow embedding dynamic JavaScript expressions (e.g., variables, functions, or expressions like {user.name}).

JSX treats everything inside curly braces as JavaScript.

5. React Components

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

Components: Independent, reusable building blocks of a React app.

Functional Components:

Stateless (until React hooks introduced state management).

Defined as plain JavaScript functions.

Example:

jsx

Copy code

function Welcome(props) {

return <h1>Hello, {props.name}</h1>;

}

Class Components:

Stateful, require extending React.Component.

Have lifecycle methods and the render() function.

Example:

jsx

Copy code

class Welcome extends React.Component {

render() {

return <h1>Hello, {this.props.name}</h1>;

}

}

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

Props are passed to components as attributes in JSX.

jsx

Copy code

<Welcome name="John" />

// Accessed as `this.props.name` (class) or `props.name` (function).

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

render() is a lifecycle method in class components.

It returns JSX (or null) to describe what should be rendered on the screen.

6. Props vs State

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

Props:

Read-only.

Used to pass data from parent to child components.

State:

Managed within a component.

Represents mutable data that affects component behavior or rendering.

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

State is an object in class components or a value in function components (via hooks like useState) that holds dynamic data.

Updating state triggers re-rendering of the component.

const [count, setCount] = useState(0);

setCount(count + 1);

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

this.setState() updates the component's state and triggers a re-render.

It ensures state updates are merged and batched for efficiency.

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