1. What is React.js?

Answer: React.js is a JavaScript library for building user interfaces, primarily used for developing single-page applications with complex UIs. It allows developers to build reusable UI components and efficiently update the view using a virtual DOM.

2. What are the main features of React?

Answer:

JSX (JavaScript XML): A syntax extension that allows HTML to be written in JavaScript.

Components: React applications are built using components, which can be functional or class-based.

Virtual DOM: React uses a virtual DOM to efficiently update the UI.

Unidirectional Data Flow: Data flows in one direction, making it predictable.

Hooks: A set of functions that allow state and lifecycle methods in functional components.

3. What is JSX in React?

Answer: JSX is a syntax extension that allows you to write HTML structures within JavaScript code. It is then compiled into regular JavaScript that browsers can understand. It helps in writing cleaner and more readable React code.

4. What are components in React?

Answer: Components are reusable building blocks in React. They can be either functional components (written as functions) or class components (written as ES6 classes). Components accept inputs called "props" and return a React element (UI).

5. What is the difference between class components and functional components?

Answer:

Class Components: They are ES6 classes that extend React.Component and require a render() method.

Functional Components: These are simpler components written as functions. With the introduction of hooks in React 16.8, functional components can now use state and lifecycle methods, making them more powerful.

6. What are props in React?

Answer: Props (short for properties) are read-only inputs that are passed into React components. They allow data to be passed from a parent component to a child component.

7. What is state in React?

Answer: State is a JavaScript object that holds data that can change over time and affect the rendering of a component. It is mutable and is used to manage local component data.

8. What are hooks in React?

Answer: Hooks are functions that allow functional components to have state and

side effects. Some of the commonly used hooks are:

useState: For managing state in functional components.

useEffect: For handling side effects like data fetching, subscriptions, etc.

useContext: For accessing context in a functional component.

9. What is the virtual DOM in React?

Answer: The virtual DOM is a lightweight copy of the actual DOM in memory. React uses it to optimize the rendering process. When the state of a component changes, React updates the virtual DOM first, then compares it with the real DOM to efficiently update only the changed parts.

10. What is the purpose of useEffect in React?

Answer: The useEffect hook allows you to perform side effects in function components, such as fetching data, directly manipulating the DOM, and subscribing to external data sources. It runs after every render, but you can control its execution with dependency arrays.

11. What is context in React?

Answer: Context provides a way to share values between components without explicitly passing props through every level of the component tree. It is useful for global data like theme, language, and authentication status.

12. What is React Router?

Answer: React Router is a library for routing in React applications. It allows you to navigate between different components based on the URL. It provides components like Route, Link, and BrowserRouter to manage the routing.

13. What are controlled components in React?

Answer: Controlled components are form elements whose value is controlled by React state. The form data is managed by the React component and updated through event handlers like onChange.

14. What are uncontrolled components in React?

Answer: Uncontrolled components are form elements whose value is handled by the DOM itself, not by React state. You can use refs to get the current value of the input field.

15. What is the difference between state and props in React?

Answer:

State: Local to a component and mutable. It can be changed by the component itself.

Props: Passed from a parent to a child component and are immutable within the child component.

16. What is the significance of the key prop in React?

Answer: The key prop is used to identify each element in a list and help React efficiently update and re-render only the changed elements when the list is updated.

17. What is the useRef hook in React?

Answer: The useRef hook is used to persist values across renders without causing a re-render. It can also be used to directly access DOM elements.

18. What is the purpose of shouldComponentUpdate in React?

Answer: shouldComponentUpdate is a lifecycle method in class components that allows you to prevent unnecessary re-renders by returning false if the component's state or props haven't changed.

19. What is Redux?

Answer: Redux is a state management library for JavaScript applications. It provides a single store for the entire application's state, and the state can only be updated by dispatching actions to reducers. It helps manage complex state logic in large applications.

20. What is the difference between Redux and Context API?

Answer:

Redux: More suitable for larger applications where state management needs to be centralized and more complex. It uses actions and reducers to manage state.

Context API: A simpler solution that provides a way to pass data through the component tree without having to pass props manually at every level. It is often used for small to medium-sized applications.

21. What is a higher-order component (HOC) in React?

Answer: A higher-order component (HOC) is a function that takes a component and returns a new component with additional props or behavior. It is used for code reuse and composability.

22. What are React lifecycle methods?

Answer: Lifecycle methods are functions in class components that are invoked at different stages of a component's life cycle (e.g., when it's created, updated, or destroyed). Some common lifecycle methods are componentDidMount, componentDidUpdate, and componentWillUnmount.

23. What is the difference between componentDidMount and useEffect?

Answer:

componentDidMount is a lifecycle method in class components that runs after the component is mounted.

useEffect is a hook that can be used in functional components to perform side effects. It can run after every render, or you can control when it runs using a dependency array.

24. What is React's ReactDOM.render() method?

Answer: ReactDOM.render() is a method used to render a React component or elements into the DOM. It takes two arguments: the React component and the DOM element where the component should be rendered.

25. What are the advantages of using React?

Answer:

Declarative syntax: React's declarative syntax makes it easier to understand and maintain UIs.

Component-based architecture: Allows for reusability and modularity.

Virtual DOM: React's virtual DOM improves performance by reducing unnecessary DOM manipulations.

Unidirectional data flow: Makes the flow of data predictable and easier to debug.

26. What is React.StrictMode?

Answer: React.StrictMode is a wrapper component that helps in identifying potential problems in the application by running checks in development mode. It does not render any visible UI but highlights unsafe lifecycle methods, deprecated APIs, and other potential issues.

27. What is the difference between React.createElement() and JSX?

Answer:

React.createElement() is the method used by React to create virtual DOM elements. It is the underlying mechanism for JSX.

JSX is a syntax extension that looks like HTML and is transformed into React.createElement() calls.

28. What is the useCallback hook in React?

Answer: useCallback is a hook that returns a memoized version of a function, preventing it from being re-created on every render. It is useful for optimizing performance in components that depend on functions passed down as props.