ReactJS Interview Questionary (60 Questions & Answers)

React is a popular JavaScript library for building user interfaces, especially single-page applications. Below is a comprehensive list of 60 interview questions and answers covering various aspects of ReactJS, from basics to advanced topics. It's a great resource for anyone preparing for a ReactJS interview.

Table of Contents

Basics of React

1.- **Question:** What is ReactJS? a.- It's a framework for develop Backend components. b.- It’s a library of JavaScript and TypeScript components created to help developers build fast UIs with a consistent and flexible architecture, and provides strong mechanisms to beautify the UI with ease. c.- It is a Java-based component set with strong UI features developed on backend for the frontend. d.- It’s a modern NoSQL database.

Answer: b **Explanation:** ReactJS is a frontend JavaScript library developed by Facebook, designed to build dynamic and interactive user interfaces efficiently, especially for single-page applications.

2.- **Question:** Who developed ReactJS? a.- Google b.- Microsoft c.- Facebook (now Meta) d.- Twitter

Answer: c **Explanation:** ReactJS was created by Jordan Walke, a software engineer at Facebook, and released in 2013.

3.- **Question:** Is ReactJS a framework or a library? a.- A full-stack framework b.- A frontend framework c.- A library for building UIs d.- A database driver

Answer: c **Explanation:** ReactJS is technically a **library**, not a full framework. It focuses mainly on the view layer (UI) in the MVC architecture.

4.- **Question:** What are the main features of React? a.- Virtual DOM, Component-Based, Declarative UI b.- Routing, Database, ORM c.- Security, Authentication, Sessions d.- File System, CLI, Debugging

Answer: a **Explanation:** ReactJS uses a **Virtual DOM** to improve performance, has a **component-based architecture**, and promotes **declarative programming**.

5.- **Question:** What is JSX? a.- A special syntax extension that allows writing HTML-like code in JavaScript. b.- A backend templating engine for NodeJS. c.- A JSON parser for React. d.- A styling language for UI components.

Answer: a **Explanation:** JSX (JavaScript XML) allows developers to write HTML elements directly in JavaScript, making UI code more readable and expressive.

6.- **Question:** Why use JSX? a.- It improves debugging and UI readability. b.- It runs faster than plain JavaScript. c.- It allows backend API calls. d.- It compiles directly to C++.

Answer: a **Explanation:** JSX improves **developer productivity** by making code more intuitive and closer to HTML. JSX compiles into `React.createElement()` calls.

7.- **Question:** Which company maintains ReactJS? a.- Microsoft b.- Facebook (Meta) c.- Oracle d.- IBM

Answer: b **Explanation:** ReactJS is maintained by Meta and a large open-source community.

8.- **Question:** What does Virtual DOM mean? a.- A real DOM stored in the cloud. b.- A lightweight copy of the real DOM used to optimize updates. c.- A relational database for UI. d.- An API for styling.

Answer: b **Explanation:** Virtual DOM is a lightweight representation of the real DOM. React updates the virtual DOM first and then efficiently updates only the changed parts in the real DOM.

9.- **Question:** What is the difference between Real DOM and Virtual DOM? a.- Real DOM is faster than Virtual DOM. b.- Virtual DOM updates the entire page each time. c.- Virtual DOM is a lightweight copy of Real DOM, allowing optimized updates. d.- They are the same thing.

Answer: c **Explanation:** Updating the Real DOM is slow, but the Virtual DOM optimizes re-renders by calculating diffs and updating only changed nodes.

10.- **Question:** Which of the following is true about ReactJS? a.- React is only for mobile apps. b.- React is unidirectional in data flow. c.- React is only for desktop applications. d.- React requires Java backend.

Answer: b **Explanation:** React enforces **one-way data binding**, meaning data flows from parent to child components, making the UI predictable.

Components

11.- **Question:** What are React components? a.- Database queries. b.- Independent, reusable pieces of UI. c.- Backend services. d.- Network requests.

Answer: b **Explanation:** React components are reusable UI blocks that define how a part of the interface should look and behave.

12.- **Question:** What are the two types of components in React? a.- Functional and Class components. b.- Static and Dynamic components. c.- Local and Global components. d.- Synchronous and Asynchronous components.

Answer: a **Explanation:** React supports **Functional components** (simpler, use hooks) and **Class components** (older, use lifecycle methods).

13.- **Question:** What is a functional component? a.- A class-based UI structure. b.- A function that returns JSX. c.- A middleware function in Node.js. d.- A styling method.

Answer: b **Explanation:** Functional components are JavaScript functions that return JSX, representing UI.

14.- **Question:** What is a class component? a.- A component defined as a class extending React.Component. b.- A database schema. c.- A CSS class. d.- An HTML form.

Answer: a **Explanation:** Class components extend `React.Component` and use methods like `render()` to return JSX.

15.- **Question:** Which type of component is preferred in modern React? a.- Class components b.- Functional components with Hooks c.- Database components d.- XML components

Answer: b **Explanation:** Functional components with **Hooks** are now the preferred way, as they are simpler and more efficient.

16.- **Question:** How do you create a functional component? a.- `function MyComponent() { return <h1>Hello</h1>; }` b.- `class MyComponent {}` c.- `def MyComponent():` d.- `component MyComponent()`

Answer: a **Explanation:** Functional components are simple functions returning JSX.

17.- **Question:** How do you create a class component? a.- `class MyComponent extends React.Component { render() { return <h1>Hello</h1>; } }` b.- `def MyComponent():` c.- `component MyComponent()` d.- `React.create(MyComponent)`

Answer: a **Explanation:** Class components extend `React.Component` and implement the `render()` method.

18.- **Question:** What is props in React? a.- Properties passed to components. b.- A backend API. c.- A type of hook. d.- A CSS class.

Answer: a **Explanation:** Props are inputs passed from a parent component to a child, making components reusable.

19.- **Question:** What are default props? a.- Global settings for all apps. b.- Fallback values for props if not provided. c.- Props stored in Redux. d.- Props from backend.

Answer: b **Explanation:** Default props define fallback values if no prop is passed.

20.- **Question:** What are propTypes in React? a.- A database schema for props. b.- A type-checking feature for component props. c.- A CSS property. d.- A backend type system.

Answer: b **Explanation:** `propTypes` are used to enforce type checking for props, helping developers catch errors.

State & Lifecycle

21.- **Question:** What is state in React? a.- A database. b.- An object that holds dynamic data in a component. c.- A CSS style. d.- A server session.

Answer: b **Explanation:** State is a local, mutable data storage inside a component.

22.- **Question:** How do you initialize state in a class component? a.- `this.state = { count: 0 }` inside the constructor. b.- `this.setState = {}` c.- `useState(0)` d.- `state.init()`

Answer: a **Explanation:** Class components initialize state in the constructor with `this.state = {}`.

23.- **Question:** How do you update state in a class component? a.- `this.state.count = 1` b.- `this.setState({ count: 1 })` c.- `setState.count(1)` d.- `update.state(1)`

Answer: b **Explanation:** State must be updated using `this.setState()` to trigger re-renders.

24.- **Question:** How do you create state in a functional component? a.- `this.state = {}` b.- `useState(initialValue)` c.- `setState()` d.- `state.init()`

Answer: b **Explanation:** Functional components use the `useState` hook to manage state.

25.- **Question:** What are React lifecycle methods? a.- Methods for API calls. b.- Methods that define different phases of a component's life. c.- CSS lifecycle rules. d.- Redux actions.

Answer: b **Explanation:** Lifecycle methods are special methods in class components that run at specific phases (mounting, updating, unmounting).

26.- **Question:** Which lifecycle method is used to fetch data after component mounts? a.- componentDidMount() b.- render() c.- componentWillUnmount() d.- shouldComponentUpdate()

Answer: a **Explanation:** `componentDidMount()` is called once after the component mounts, ideal for API calls.

27.- **Question:** Which lifecycle method is used before a component is removed? a.- componentWillUnmount() b.- componentDidMount() c.- render() d.- shouldComponentUpdate()

Answer: a **Explanation:** `componentWillUnmount()` is called before the component is destroyed, often used for cleanup.

28.- **Question:** Which method is required in a class component? a.- render() b.- componentDidMount() c.- constructor() d.- setState()

Answer: a **Explanation:** Every class component must implement the `render()` method.

29.- **Question:** How do hooks relate to lifecycle methods? a.- They replace lifecycle methods in functional components. b.- They are only for Redux. c.- They work only in class components. d.- They are not related.

Answer: a **Explanation:** Hooks like `useEffect` replicate lifecycle behavior (mounting, updating, unmounting) in functional components.

30.- **Question:** Which hook replaces componentDidMount in functional components? a.- useEffect with empty dependency array b.- useState c.- useReducer d.- useRef

Answer: a **Explanation:** `useEffect(() => {...}, [])` runs only once, mimicking `componentDidMount`.

Hooks

31.- **Question:** What are React hooks? a.- Functions that let functional components use state and lifecycle features. b.- CSS features. c.- Redux actions. d.- Database calls.

Answer: a **Explanation:** Hooks allow functional components to use features previously only available in class components.

32.- **Question:** When were hooks introduced? a.- React 12 b.- React 16.8 c.- React 15 d.- React Native release

Answer: b **Explanation:** Hooks were introduced in React 16.8.

33.- **Question:** Which hook is used for state management? a.- useState b.- useEffect c.- useRef d.- useReducer

Answer: a **Explanation:** `useState` provides state variables in functional components.

34.- **Question:** Which hook is used for side effects? a.- useState b.- useEffect c.- useRef d.- useContext

Answer: b **Explanation:** `useEffect` handles side effects like data fetching, subscriptions, and DOM manipulations.

35.- **Question:** Which hook is used for references? a.- useRef b.- useState c.- useEffect d.- useMemo

Answer: a **Explanation:** `useRef` creates mutable references that persist across renders.

36.- **Question:** Which hook is used for memoization? a.- useMemo b.- useReducer c.- useEffect d.- useRef

Answer: a **Explanation:** `useMemo` caches the result of a computation to avoid unnecessary recalculations.

37.- **Question:** Which hook is used for context? a.- useContext b.- useRef c.- useMemo d.- useEffect

Answer: a **Explanation:** `useContext` allows functional components to consume values from React Context.

38.- **Question:** Which hook is used for reducers? a.- useReducer b.- useContext c.- useEffect d.- useRef

Answer: a **Explanation:** `useReducer` is an alternative to `useState` for complex state logic.

39.- **Question:** What rules must hooks follow? a.- Only call hooks at the top level of functional components. b.- Only call hooks inside React components or custom hooks. c.- Never call hooks in loops or conditions. d.- All of the above.

Answer: d **Explanation:** Hooks have strict rules to ensure consistent execution order.

40.- **Question:** Can hooks be used in class components? a.- Yes b.- No c.- Only with plugins d.- Only in Next.js

Answer: b **Explanation:** Hooks are designed only for functional components.

Advanced React

41.- **Question:** What is React Context? a.- A database. b.- A way to share data across components without props drilling. c.- A lifecycle method. d.- A CSS class.

Answer: b **Explanation:** Context allows passing data globally without prop drilling.

42.- **Question:** What is Redux? a.- A CSS library. b.- A state management library often used with React. c.- A lifecycle hook. d.- A routing system.

Answer: b **Explanation:** Redux is a predictable state container often used for global state in React.

43.- **Question:** What are React Fragments? a.- A way to group elements without extra DOM nodes. b.- A state hook. c.- A lifecycle method. d.- A CSS framework.

Answer: a **Explanation:** Fragments let you group children without adding extra `<div>` elements.

44.- **Question:** What are Higher-Order Components (HOCs)? a.- Components wrapped with CSS. b.- Functions that take a component and return a new component. c.- Redux actions. d.- Database queries.

Answer: b **Explanation:** HOCs add additional functionality to existing components.

45.- **Question:** What is React Router used for? a.- Styling components. b.- Handling navigation in single-page applications. c.- Fetching APIs. d.- Database connections.

Answer: b **Explanation:** React Router manages navigation without reloading the page.

46.- **Question:** What are controlled components? a.- Components controlled by CSS. b.- Form components whose values are controlled by React state. c.- Components from Redux. d.- Database-driven components.

Answer: b **Explanation:** Controlled components store input values in state, making them React-controlled.

47.- **Question:** What are uncontrolled components? a.- Components not used in production. b.- Form elements that handle their own state internally. c.- Redux components. d.- Stateless components.

Answer: b **Explanation:** Uncontrolled components manage their own state via the DOM, not React.

48.- **Question:** What is React reconciliation? a.- Merging databases. b.- The process of updating the DOM by comparing Virtual DOM and Real DOM. c.- A lifecycle method. d.- A CSS update.

Answer: b **Explanation:** Reconciliation is React's diffing algorithm that updates only changed elements.

49.- **Question:** What is React Fiber? a.- A CSS framework. b.- The new React reconciliation algorithm introduced in React 16. c.- A database. d.- A lifecycle method.

Answer: b **Explanation:** Fiber is the new core algorithm introduced in React 16 to handle rendering more efficiently.

50.- **Question:** What is code splitting in React? a.- Dividing CSS files. b.- Splitting code into smaller bundles for performance. c.- Dividing database queries. d.- Lifecycle splitting.

Answer: b **Explanation:** Code splitting optimizes loading by delivering only the code needed for a given route/component.

Testing & Performance

51.- **Question:** How do you test React components? a.- Using Jest and React Testing Library. b.- Using SQL queries. c.- Using NodeJS only. d.- Manual inspection only.

Answer: a **Explanation:** Jest and React Testing Library are popular testing tools for React.

52.- **Question:** What is shallow rendering? a.- Rendering the entire component tree. b.- Rendering a component without its child components. c.- Rendering database. d.- Rendering CSS.

Answer: b **Explanation:** Shallow rendering tests a component in isolation, ignoring its children.

53.- **Question:** What is React.StrictMode? a.- A debugging tool for highlighting potential problems. b.- A CSS library. c.- A lifecycle method. d.- A state hook.

Answer: a **Explanation:** StrictMode helps detect unsafe lifecycle methods and potential issues.

54.- **Question:** What is lazy loading in React? a.- Loading data slowly. b.- Loading components only when needed. c.- Loading Redux state. d.- Loading CSS.

Answer: b **Explanation:** Lazy loading improves performance by loading components only when required.

55.- **Question:** What is memoization in React? a.- Storing CSS values. b.- Optimizing performance by caching function results. c.- Redux state management. d.- Lifecycle optimization.

Answer: b **Explanation:** Memoization avoids unnecessary re-renders by caching computations or components.

56.- **Question:** What is React.memo? a.- A hook. b.- A higher-order component that

memoizes functional components. c.- A lifecycle method. d.- A Redux action.

Answer: b **Explanation:** `React.memo` optimizes functional components by re-rendering only if props change.

57.- **Question:** What is useCallback used for? a.- Memoizing functions. b.- Memoizing values. c.- Handling refs. d.- Redux actions.

Answer: a **Explanation:** `useCallback` memoizes callback functions to avoid unnecessary re-creations.

58.- **Question:** What is useMemo used for? a.- Memoizing values from expensive calculations. b.- Memoizing functions. c.- Redux actions. d.- Lifecycle hooks.

Answer: a **Explanation:** `useMemo` caches computed values, recalculating only when dependencies change.

59.- **Question:** What is server-side rendering (SSR) in React? a.- Rendering components in the database. b.- Rendering components on the server before sending HTML to the browser. c.- Rendering only on the client. d.- Rendering CSS on backend.

Answer: b **Explanation:** SSR improves performance and SEO by pre-rendering React components on the server.

60.- **Question:** What is hydration in React? a.- Adding water to servers. b.- Attaching React event listeners to server-rendered HTML. c.- Adding CSS to components. d.- Loading Redux state.

Answer: b **Explanation:** Hydration attaches React's event system to HTML rendered by SSR, making it interactive.