**Senior React + Redux Interview Questions**

## **React (30 Questions)**

### **1. useEffect vs useLayoutEffect – Give a scenario where useLayoutEffect prevents UI flicker.**

### **2. What is the output of the following?**

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

useEffect(() => {

setCount(count + 1);

}, [count]);

Why does this cause an infinite loop? Fix it.

### **3. Write a useCounter custom hook that returns a count and increment(), decrement() functions.**

### **4. Optimize the component:**

const handleClick = () => {

console.log(item.name);

};

return <button onClick={handleClick}>Click</button>;

It’s being passed to 100 items in a list. Prevent unnecessary re-renders.

### **5. How do you prevent stale closures in useEffect when accessing latest props/state?**

### **6. Implement a reusable useDebounce(value, delay) hook.**

### **7. What’s wrong with this? Fix it:**

const [items, setItems] = useState([]);

useEffect(() => {

fetch("/api/items").then(res => res.json()).then(setItems);

}, [items]);

**8. Convert this class component to functional using hooks:**

componentDidMount() {

this.fetchData();

}

**9. Refactor to use controlled component:**

<input />

### **10. What is the difference between useMemo and useCallback? When would you use each?**

### **11. Build a <List> component that accepts a renderItem prop and renders any array generically.**

### **12. What is the output of this code? Explain why.**

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

const double = useMemo(() => count \* 2, []);

### **13. Create a basic error boundary component using class components.**

### **14. Explain why lifting state up is often necessary. Refactor a child-to-parent communication.**

### **15. How can you lazy load a route in React Router v6?**

### **16. What are Suspense and lazy? Show usage.**

### **17. Write a usePrevious custom hook to access the previous value of a prop.**

### **18. Implement a TextInput component that accepts only digits (0–9).**

### **19. Refactor a list rendering to use a key properly. What problems does using index cause?**

### **20. What’s a good way to cancel a fetch inside useEffect to prevent memory leaks?**

### **21. What is the difference between useEffect(() => ..., []) and useLayoutEffect(() => ..., [])? When would you use one over the other?**

*Rendering lifecycle and visual side effects.*

### **22. Why is state considered asynchronous in React? How do you reliably update state based on the previous value?**

*Functional updates: setState(prev => prev + 1).*

### **23. How would you avoid unnecessary re-renders in a component tree with heavy props and nested children?**

*Mention React.memo, useMemo, useCallback, and selectors in state management.*

### **24. When would you use Context API over Redux, and when is Redux the better choice?**

*Shows architectural thinking — global static config (Context) vs dynamic state (Redux).*

### **25. What are the pitfalls of using index as a key in a list? What problems can it cause?**

*Answer should mention reordering issues and stale component state.*

### **26. Explain how you’d design a reusable Modal or Dropdown component in React.**

*Component isolation, props, children, onClose, portals.*

### **27. How do you manage side-effects like data fetching in React? Can you walk through a pattern you use for loading/error/success states?**

*Use of useEffect, loading flags, and error handling.*

### **28. What are controlled vs uncontrolled components? Which one is preferred and why?**

*Controlled: React manages the state. Preferred for predictability.*

### **29. Describe the Virtual DOM and reconciliation. Why is React performant despite re-renders?**

*Understanding of React’s rendering engine.*

### **30. If a parent component re-renders, do all child components re-render too? How can this be prevented?**

*Look for React.memo, prop comparison, and lifting state .*

## **Redux / Redux Toolkit (30 Questions)**

### **1. Write a slice using createSlice to manage a list of todos.**

### **2. Add async fetch to your slice using createAsyncThunk. Handle loading, success, error.**

### **3. How do you access and update state in a React component using useSelector and useDispatch?**

### **4. What is a selector? How would you memoize it using createSelector?**

### **5. Given this state, write a selector to get total currentValue of selected assets:**

{

portfolio: {

items: [{ id, name, selected, currentValue }],

filter: "Stock"

}

}

### **6. Normalize this array into a dictionary format in Redux slice:**

[

{ id: "1", name: "AAPL" },

{ id: "2", name: "TSLA" }

]

### **7. What problem does createEntityAdapter solve? How is it different from manual normalization?**

### **8. What middleware would you use to log actions and state after every dispatch?**

### **9. How do you structure Redux slices for a multi-feature app (e.g., auth, dashboard, settings)?**

### **10. Write a middleware that blocks any action of type "auth/LOGIN" if already logged in.**

### **11. What happens if two reducers try to update the same slice of state? How would you manage conflicts?**

### **12. Write a test for a Redux reducer using Jest. Mock initial state and assert updates.**

### **13. How do you avoid re-rendering all components when one slice of Redux state changes?**

### **14. What is the difference between dispatching plain actions vs thunks?**

### **15. Explain Redux DevTools and how they help debugging. How do you integrate them?**

### **16. What’s the difference between global Redux store and local component state? When to use which?**

### **17. Implement undo/redo functionality for a Redux-powered canvas app.**

### **18. How would you persist Redux state across refreshes? What libraries can help?**

### **19. Convert the below legacy Redux code to Redux Toolkit:**

function counterReducer(state = 0, action) {

switch (action.type) {

case "INC": return state + 1;

default: return state;

}

}

### **20. Can you implement optimistic updates in Redux? Describe the pattern.**

### **21. Explain the core concepts of Redux: Store, Action, Reducer. How does data flow through the system?**

*Classic question — answer should explain unidirectional flow clearly.*

### **22. What problems does Redux Toolkit solve over traditional Redux?**

*Mention boilerplate reduction, createSlice, immer, built-in middleware.*

### **23. What is createAsyncThunk? How does it work and when would you use it?**

*Understand async flow handling and integration with slices.*

### **24. How do you manage loading, error, and success states using Redux Toolkit when performing async operations?**

*Describe pending, fulfilled, rejected action types and slice state structure.*

### **25. How would you normalize data in Redux? Why is it important in large apps?**

*Describe createEntityAdapter or manual normalization with IDs.*

### **26. What are some common performance issues in Redux and how can you optimize them?**

*Answer should include: selectors, memoization, avoiding state bloat, limiting re-renders.*

### **27. What are selectors and why are they important in Redux? How do you create a memoized selector?**

*Describe use of reselect or createSelector.*

### **28. What’s the difference between useSelector and connect()? When would you use one over the other?**

*Function components use useSelector; class components use connect().*

### **29. How would you structure Redux in a large app with multiple domains (auth, dashboard, settings, etc.)?**

*Describe feature-based folders, modular slices, lazy loading reducers.*

### **30. What is a Redux middleware? Can you give an example of a custom middleware you’ve written or used?**

*Describe logging, analytics, async handling, token injection, etc.*

==================================================================================================