React Hooks & Context (Intro, Rules, State, Effect Hook, Custom Hooks) (15)

31. What are the two main rules of hooks in React?

Two main rules of hooks are:

1. We can only call hooks at the top level and never inside loops or conditions

2. And we can only call them inside React functions.

32. How does useState differ from setting state in class components?

The useState gives you state in functional components directly and there is no need for this.setState or classes.

33. How do you update state based on the previous value using useState?

We should pass a callback to setState

setCount(prev => prev + 1);

34. What are some common use cases for the useEffect hook?

Use cases are:Fetching data, running timers, subscribing to events, or syncing with local storage.

35. How do you clean up side effects in useEffect?

To clean up side effects in useEffect we have to return a cleanup function from useEffect

ex:

useEffect(() => {

const id = setInterval(...);

return () => clearInterval(id);

}, []);

36. What happens if you forget to provide a dependency array in useEffect?

The effect runs after every render and it can cause infinite loops.

37. What is the difference between useContext and prop drilling?

Prop drilling passes data manually through each component layer.

useContext is used to provide data directly to deeply nested components without using intermediate props.

38. How do you create a React Context provider and consumer using hooks?

We can create a reach context using React.createContext,

wrap the app with a Provider and use useContext.

Ex:

const MyContext = React.createContext();

const App = () => (

<MyContext.Provider value={value}>

<Child />

</MyContext.Provider>

);

const Child = () => {

const value = React.useContext(MyContext);

return <div>{value}</div>;

};

39. How do you avoid re-renders when passing context values?

To avoid re-renders we have to memoize the value which is passed to the Provider using useMemo.

const value = useMemo(() => ({ user }), [user]);

40. Give an example of a custom hook for form input handling.

function useInput(initial) {

const [value, setValue] = useState(initial);

return { value, onChange: e => setValue(e.target.value) };

}

41. What is the difference between useEffect and useLayoutEffect?

useEffect runs after painting and is asynchronous and non-blocking

useLayoutEffect runs before painting and is synchronous and it blocks browser paint

42. How can you create a custom hook for API fetching?

function useFetch(url) {

const [data, setData] = useState(null);

useEffect(() => { fetch(url).then(r => r.json()).then(setData); }, [url]);

return data;

}

43. What is the difference between multiple useEffect hooks vs a single one with multiple logics?

Multiple hooks allow better separation of concerns and are easier to maintain.

One single big hook can become harder to read and debug

44. Why can’t hooks be used inside conditional statements?

We can’t hooks be used inside conditional statements because hooks rely on call order between renders.

Conditional usage breaks this order and it causes bugs.

45. How would you share logic between multiple components using hooks?

We have to extract the logic into a custom hook and call that hook in any component.

API Integration with Fetch & Axios (GET, POST, PUT, DELETE) (10)

46. What is the difference between Fetch API and Axios in React?

Axios automatically transforms JSON and supports request cancelation and also has shorter syntax.

Fetch requires manual res.json() and additional code for features like cancelation.

47. How do you make a GET request using Axios in useEffect?

useEffect(() => {

axios.get('/api/data').then(res => setData(res.data));

}, []);

48. How do you handle errors in Axios requests?

axios.get('/api').catch(error => console.error(error));

49. How do you send POST requests with JSON body using Axios?

axios.post('/api', { name: 'Abhi' });

50. What are the differences in default headers between Fetch and Axios?

Axios sets the Content-Type: application/json automatically.

Fetch does not set the Content-Type: application/json automatically— must specify headers manually.

51. How do you send a PUT request with Axios to update existing data?

axios.put('/api/1', { name: 'Updated' });

52. How do you delete data from an API using Axios?

Using:

axios.delete('/api/1');

53. How do you cancel an Axios request in progress?

const controller = new AbortController();

axios.get('/api', { signal: controller.signal });

controller.abort();

54. What is an Axios interceptor and why would you use it?

Interceptors modify requests/responses globally like to attach tokens or log errors.

55. How do you handle loading states during API requests in React?

We have to use a state variable:

loading = true before request,

loading = false after success/error.

Pure Components (5)

56. What is a Pure Component in React?

a Pure Component in React is a component that only re-renders when its props change.

57. How do Pure Components improve performance?

They improve performance by skipping unnecessary renders, making apps faster.

58. How is React.memo related to Pure Components in function components?

React.memo makes functional components act like pure components.

59. What kind of props changes will cause a Pure Component to re-render?

Any change to props like a new value or reference, even if the content looks the same will cause a Pure Component to re-render.

60. What are the limitations of Pure Components?

They only do a shallow prop comparison — deep objects or arrays can still cause unwanted re-renders.