Interview FeedBack – React Developer

Natarajan

Interviewed Date: 13 May 2026

React Developer

Technical SoftSkill Question Upskill Plan

Technical

Where You Excel Technically—and Where You Can Grow

Core Concepts

Strengths

- Described props as a way to pass data from a parent component to a child component.
- Characterized props as read-only and associated them with reusable/configurable components.
- Explained state as data managed inside a component that can change over time and affect UI updates.

Improvements

- Clarify the lifecycle and ownership boundaries by contrasting who controls updates for props versus state.
- Demonstrate applied usage by providing a concrete component example showing props flowing in and state updating from user interaction.
- Strengthen conceptual precision around immutability and update patterns by describing how state changes trigger re-renders.

Forms

Strengths

- Explained controlled components by tying input values to React state.
- Used useState as the storage mechanism for input values and described updating state on change events.
- Referenced submit handling with preventDefault to avoid page refresh behavior.

Improvements

- Expand validation detail by specifying how validation results are represented and surfaced in the UI.
- Clarify the form flow by describing how input-level state, error state, and submit state interact.
- Strengthen the example quality by outlining a simple field set and showing how validity gates submission.

Git Version Control and CI/CD Pipelines

Strengths

- Identified running lint and test commands as merge-time quality gates.
- Referenced CI tooling options such as GitHub Actions and GitLab.
- Explained the idea of blocking merges when checks fail.

Improvements

- Clarify enforcement mechanisms by describing how branch rules connect to CI status checks to gate merges.
- Strengthen specificity by naming how lint/test jobs are organized and reported as required checks.
- Improve explanation structure by presenting a clear sequence from pipeline run to merge blocking outcome.

Hooks

Strengths

- Associated useEffect with handling side effects in functional components.
- Provided relevant side-effect examples including API calls, timeouts, event listeners, document title updates, and subscriptions.
- Described dependency array patterns: none (runs every render), empty array (runs once), and specified dependencies (runs on change).

Improvements

- Correct hook identification by consistently distinguishing useEffect from other hooks referenced in the explanation.
- Deepen reasoning by explaining why each dependency array pattern is chosen for different side-effect scenarios.
- Strengthen correctness around render timing by describing how effects relate to render cycles in practical terms.

JavaScript Testing with Jest and React Testing Library

Strengths

- Engaged with the question and attempted to connect act() to unit testing contexts.
- Referenced the idea of testing across different units such as functions and components.
- Maintained a testing-focused framing by keeping the answer within the scope of unit testing concepts.

Improvements

- Define act() precisely by articulating its role in flushing updates triggered by React state/effects during tests.
- Differentiate usage contexts by describing when act() is required versus when tooling handles it implicitly.
- Strengthen practical understanding by connecting act() to typical async or interaction-driven update scenarios in component tests.

Soft Skills

How You Connect, Collaborate, and Communicate

Strengths

- Maintained a cooperative tone and made a clear effort to answer each prompt.
- Used recognizable technical keywords to anchor responses (useState, preventDefault, dependency array, CI checks).
- Kept answers generally aligned to the asked topics without diverging into unrelated areas.

Improvements

- Refine delivery structure by answering with a clear definition followed by a concrete example and a brief rationale.
- Enhance clarity by using shorter, more direct sentences that reduce ambiguity in key technical statements.
- Strengthen completeness by explicitly concluding each answer with the requested example or decision criteria.

Questions

Questions

Interview feedback – Here's How You Did

Q1: What is the role of act() in testing, and when do you need to use it?

Answer: What is the role of act in testing? What is the act testing? When it, I would, I would say the functions, method, components, and services, these are the things for the act of the unit testing.

Suggestion: Response does not explain that act() is used to wrap updates in tests so React flushes effects and state updates before assertions, and it instead gives a vague statement about functions, methods, components, and services without addressing when or why act() is needed.

Q2: Explain the difference between props and state in React, and give an example of when you would use each in a component.

Answer: Explain the difference between props and state and react, and give an example of when you were using a component. So props are used for the pass the data from the parent component to the child component. State is like used to start the blanks to the component and can change about time. The crops he can do only the read only then pass from the outside is from the configurable and reusable component. We go with the state the sum mutable manager inside the component and operating the straight free run under the UI.

Suggestion: Answer correctly distinguishes props as data passed from parent to child and generally read-only, and state as internal, mutable data that changes over time and drives UI updates, but it lacks a clear concrete example for each and includes some unclear phrasing that reduces depth.

Q3: Describe how you would handle a simple form in React (input fields, validation, and submit) using controlled components.

Answer: Well, so I would go with the useState stored from the value. On change, like update the state when the user types. As validation checks input before submit. If you say the handle submit, prevent the page refreshing. Just example, there is a method called prevent default. Since the input value is from the React state, this is called the controlled component. Attention, you have switched to another tab.

Suggestion: Answer correctly describes the core controlled-component approach by storing input values in React state, updating state on change, validating before submit, and preventing default form submission, but it lacks detail on structuring handlers, validation feedback, and error state management.

Upskill Plan

A step-by-step plan to level up

1. Step 1: React data flow fundamentals (props vs state)
2. Step 2: Controlled forms with validation behavior
3. Step 3: Effects and dependencies in hooks
4. Step 4: Quality gates: CI enforcement and React test correctness

React data flow fundamentals (props vs state)

Focus areas

- Ownership and update control for props versus state
- Read-only props versus mutable state as a concept
- Concrete component examples that combine props input with state updates
- UI re-render implications of state changes

Outcome

Communicate and apply props/state distinctions with clear, example-driven explanations.