React Interview Questions PDF

React interviews cover components, props and state, hooks such as useState and useEffect, how rendering and the virtual DOM work, keys in lists, controlled forms, context, performance with memoisation, routing and testing. This page gives short model answers with small code examples, written for practice and not taken from any company.

Key takeaways

- A component is a function that returns UI; props go in, state lives inside.
- Hooks such as useState and useEffect add state and side effects to function components.
- React re-renders a component when its state or props change; keys help it track list items.
- Optimise only after measuring: memoisation helps in specific cases.
- Know controlled forms, context, routing basics and how to test components.

What is React interview questions?

React is a JavaScript library for building user interfaces from reusable components, where the screen is described as a function of state and React updates the page efficiently when state changes.

What is React and why is it used?

React is a library for building user interfaces. You write components that describe what the UI should look like for the current data, and React updates the page when the data changes. Teams use it for reusable components, a large ecosystem and a clear data flow. React is a library for the view layer; routing and data fetching come from additional tools.

What are components, props and state?

A **component** is a function that returns markup (JSX). **Props** are inputs passed from a parent and are read-only. **State** is data owned by a component that can change over time and triggers a re-render.

```
function Counter({ start }) {
  const [count, setCount] = useState(start);
  return <button onClick={() => setCount(count + 1)}>Clicked {count}</button>;
}
```

What is JSX?

JSX is a syntax that looks like HTML inside JavaScript. It is compiled to function calls that create elements. Attributes use camelCase such as `className`, and JavaScript expressions go inside curly braces.

What is the virtual DOM and how does rendering work?

React keeps a lightweight description of the UI in memory. When state changes, React renders the component again, compares the new description with the previous one (reconciliation) and updates only the parts of the real DOM that changed. This makes updates predictable. A re-render is not the same as a DOM update.

Why do list items need keys?

Keys give each item a stable identity so React can match items between renders. Use a unique, stable ID from your data, not the array index when items can be reordered, added or removed. Wrong keys can cause bugs such as input values jumping between rows.

What are the main hooks?

useState stores state. **useEffect** runs side effects such as fetching data or subscriptions after render, with a dependency array that controls when it runs. **useRef** holds a mutable value or a DOM reference without causing re-renders. **useMemo** caches a calculated value and **useCallback** caches a function between renders. **useContext** reads shared data. **useReducer** manages complex state with actions.

```
useEffect(() => {
  const id = setInterval(tick, 1000);
  return () => clearInterval(id);   // cleanup
}, []);
```

What are the rules of hooks?

Call hooks at the top level of a component or custom hook, not inside loops, conditions or nested functions, and only from React functions. The rules ensure React can match hook calls to state between renders.

What are controlled and uncontrolled components?

In a controlled input, React state holds the value and each change updates it through `onChange`. In an uncontrolled input, the DOM keeps the value and you read it with a ref when needed. Controlled inputs make validation and conditional logic easier.

How do I share data between components?

Lift state up to a common parent and pass it down as props. For data needed by many levels, use context to avoid passing props through every layer. For larger apps, teams use state libraries; choose them when the need is clear.

How do I improve performance?

Measure first with the profiler. Common steps are splitting large components, using `React.memo` for components that re-render with the same props, using `useMemo` and `useCallback` where they prevent expensive work, virtualising long lists and loading code lazily. Unneeded memoisation adds complexity without benefit.

How does routing and data fetching work?

Routing libraries map URLs to components. Data is fetched in effects or with data libraries that handle caching, loading and error states. Always handle loading and failure states in the UI and avoid setting state after a component has unmounted.

How do I test React components?

Use a test runner with a testing library that renders the component, interacts as a user would and asserts on visible output. Prefer testing behaviour (what the user sees) to implementation details such as internal state.

What common mistakes do candidates make?

Mutating state directly instead of creating a new object or array, forgetting the dependency array in an effect so that it runs on every render, using the array index as a key in a changing list, putting derived values in state when they can be calculated during render and fetching data without handling loading and error states. Naming these mistakes, with the fix for each, shows real experience.

What small coding tasks are common?

Build a counter, a to-do list with add and remove, a search box that filters a list, a form with validation or a component that fetches and shows data. Talk through the state you need, where it lives and how events update it. Keep components small, name them clearly and handle empty and error cases.

Frequently asked questions

What is React?

React is a JavaScript library for building user interfaces from reusable components that update efficiently when state changes.

What is the difference between props and state?

Props are read-only inputs passed from a parent, while state is data owned by a component that can change and trigger a re-render.

Why are keys needed in lists?

Keys give list items a stable identity so React can match them between renders and avoid bugs when items change order.

When does useEffect run?

It runs after render, and the dependency array controls when it runs again. An empty array means it runs once after the first render.

Should I memoise everything?

No. Measure first with the profiler, because unneeded memoisation adds complexity without any real benefit.

Sources and further reading

- React documentation
- MDN JavaScript reference

About this PDF

These PDFs are study material for learning and revision. They are not a certificate, a degree or a promise of a job, and technology changes, so check official documentation for the latest details. Spot an error? Write to info@uncodemy.com and the trainer team will review it.

More free resources and courses: uncodemy.com | info@uncodemy.com | +91 98183 66550