MIKE SALARI

TOP 300 REACT NATIVE INTERVIEW QUESTIONS

300 questions from Junior to Expert
with verified code and primary sources

SECOND EDITION · 2026

salari.dev

FREE SAMPLE · TOP 300 REACT NATIVE INTERVIEW QUESTIONS

Most candidates know the answer. Strong candidates know what comes next.

300 React Native interview questions that teach you to explain trade-offs, handle follow-ups, and demonstrate production-level engineering judgment.

You know React Native. Can you explain your decisions under pressure? Knowing an API is the starting point. Strong interview answers explain what happens in production, where the design breaks, how you would measure success, and what you would do when your first approach fails.

What you will be able to do

- ✓ Say the answer first, in plain words, then add the production detail.
- ✓ Explain a decision: the constraint, the trade-off and what you would measure.
- ✓ Name the limits and the alternative before the interviewer asks.
- ✓ Handle architecture, performance and reliability rounds.
- ✓ Know what the interviewer is scoring and which mistakes cost you the point.
- ✓ Tell mid, senior and staff depth apart, and write to the level you are interviewing for.

WHAT MAKES THIS BOOK DIFFERENT

Five parts for every answer

Every one of the 300 answers has the same five parts, so you can rehearse it out loud and notice which part you skip.

PART	WHAT IT GIVES YOU
The answer	What you say first, in plain words.
Production depth	What changes in a shipped app, and what you would measure.
Common pitfall	What candidates say or do, and what it costs them.
What the interviewer is testing	The signal behind the question, in one line.
Further reading	One to three primary sources: React Native, React, Expo, Apple, Android, Google Play.

Depth grows with the level

LEVEL	THE ANSWER MUST CONTAIN
Junior	Definition, small example, one rule of thumb, and basic error handling.
Mid-level	Implementation boundaries, production failure, testing, and a justified trade-off.
Senior	The constraint, the trade-off chosen, a metric with a threshold, rollout, and recovery.
Staff and Expert	System and organizational constraints, a named artifact, a decision rule, and the mechanism behind it.

The book is a preparation system, not a guarantee. It shows what an interviewer listens for. It is not a tutorial.

SEE THE DIFFERENCE

The same question, two answers

Question 87. How do you prevent memory leaks in effects?

A FIRST ANSWER

"Return a cleanup function from the effect and remove the listener."

Correct. Written for this comparison, not quoted from a candidate. It is also where the answer stops.

Then comes the follow-up: a profile screen fetches a user when `userId` changes, the user taps through three profiles, and the response for the first one arrives last. What does the screen show? The late response for the first profile can overwrite the data of the profile the user is now looking at.

THE ANSWER IN THE BOOK

Anything an effect starts that outlives its render can leak: a subscription, a timer, a listener or a request. The fix is a cleanup that undoes exactly what the setup did: remove the listener, clear the timer, abort the request.

The warning "can't perform a state update on an unmounted component" belonged to React 17 and earlier. React 18 removed it because it mostly fired for harmless cases. The real problems remain: wasted work, and stale writes, where a slow response for an old dependency overwrites newer state. Abort the request, or ignore its result after cleanup, and check `res.ok` before parsing, because `fetch` resolves on a 404 or 500.

The difference. The first answer is correct. The second shows experience: it names the stale write, the fix and the check, and it rejects the flag that only silences a retired warning. This is Question 87, on page 109 of the full book.

MIGRATION
SENIOR · 170 · FULL BOOK PAGE 213

How do you plan a migration from the old architecture?

THE ANSWER

On React Native 0.82 and later there is no opt-in: `newArchEnabled=false` is ignored, so the plan is about readiness, not a switch. The 0.82 post recommends moving first to 0.81 or Expo SDK 54, the last versions that still allow the Legacy Architecture, enabling the New Architecture there, and only then upgrading to 0.82. Spend one release on that stepping stone so breakage shows up in isolation.

PRODUCTION DEPTH

Inventory every native dependency. React Native Directory stores a New Architecture status per library (supported, new-architecture-only, unsupported, or untested when nobody has reported), and `npx expo-doctor` checks your native dependencies against that data, listing packages that are unsupported, untested or unmaintained. It needs network access, so run it in CI too. For the rest, know what the interop layer covers: legacy native modules and view components run without migration (on by default since 0.74), but it does not support views that define a custom `ShadowNode`, and code that spreads or enumerates `NativeModules` objects breaks. Interop stays for the foreseeable future, so treat it as a bridge. Roll out with the gates in Question 169; binaries cannot be recalled, so recovery is a halted rollout plus a fix, with a server flag on risky features.

COMMON PITFALL

Trusting a green `expo-doctor` run as readiness. It reports metadata, not a test of your app: an untested library may work and a supported one may break on your usage, so run each against the stepping-stone release.

WHAT THE INTERVIEWER IS TESTING

Inventory from Directory data first, then sequence with known interop limits.

FURTHER READING

React Native 0.82: A New Era
reactnative.dev/blog/2025/10/08/react-native-0.82

Enable libraries: ensure compatibility with the Interop Layer
github.com/reactwg/react-native-new-architecture/blob/main/docs/enable-libraries.md

expo-doctor React Native Directory check (source)
github.com/expo/expo/blob/main/packages/expo-doctor/src/checks/ReactNativeDirectoryCheck.ts

What the full book covers

Six levels, one method. Each level trains a different part of the answer.

Junior 1 to 50

Build a precise foundation. These questions test whether you can write and reason about ordinary React Native code without hiding rendering, state, or lifecycle behavior.

Junior to Mid 51 to 100

Move from recognizing an API to explaining cause and effect: why a component renders, where data belongs, how failures are represented, and what should be tested.

Mid 101 to 160

Explain the current architecture accurately, cross the JavaScript and native boundary on purpose, and reason about startup, security, persistence, lifecycle, and platform APIs.

Senior 161 to 210

Answer with the constraint first, choose an explicit trade-off, define the measurement, and include rollout and recovery. The implementation is only one part of the system.

Staff 211 to 260

Connect technical direction to product outcomes, team topology, governance, developer experience, and measurable platform health, with a named artifact and a decision rule.

Expert 261 to 300

Name the mechanisms: module boundaries, offline correctness, typed configuration, observability, security, and how a platform evolves without a rewrite.

And in every answer

- 99 TypeScript examples across 95 questions, each type-checked with TypeScript 6.0.3, React Native 0.88.0, React 19.3.0, and Expo SDK 58.
- 62 decision tables, a night-before chapter, and a study route for 7 days, 30 days, or a staff loop.
- Every one of the 300 questions has five parts: the answer, production depth, a common pitfall, what the interviewer is testing, and further reading from primary sources.

TOP 300 REACT NATIVE INTERVIEW QUESTIONS · SECOND EDITION

300 questions from junior to expert, with verified code, decision tables, and a study route. Updated for React Native 0.88, React 19.3, and Expo SDK 58.

Buy the full version

The depth interviewers expect. PDF and EPUB on Gumroad. Current price on the store.

★★★★★

"Great books, great author!"

Joshua Elden, Gumroad

A general review of the author's books, posted on this book's Gumroad page.

Get the full book

salari.dev/books/top-react-native-interview-questions

Staff mobile engineer · 500+ technical interviews from the hiring side. 15+ years at Apple, Adobe, Cisco, Visa, Mastercard and Deloitte.

React Native and the React logo are trademarks of their respective owners. This independent publication is not affiliated with or endorsed by Meta.