React Native Interview Q&As

Section 1: Basics & Framework Overview (Q1–Q5)

Q1. What is React Native?
Answer:

- 🔹 Open-source **framework by Meta** for mobile apps.
- Uses **JavaScript + React** and compiles to native code.
- Runs on **iOS & Android** from a single codebase.
  💡 **Tip:** Emphasize **cross-platform reusability** and faster development.

Q2. Difference between React and React Native?
Answer:

- **React** → Web UIs with HTML + CSS.
- **React Native** → Mobile UIs with **native components**.
- React runs in browsers; React Native uses a **JS bridge**.
  💡 **Tip:** Mention **shared logic + platform-specific UI**.

Q3. What are core components in React Native?
Answer:

- **View, Text, Image, ScrollView, FlatList, TextInput, Button**.
- **TouchableOpacity** → For taps & clicks.
  💡 **Tip:** Familiarity with these is key for **UI development**.

Q4. What is JSX?
Answer:

- JavaScript XML syntax extension.
- Write **HTML-like code** in JS.
- Maps directly to **native components** in RN.
  💡 **Tip:** Essential for building **reusable components**.

Q5. What is Flexbox in React Native?
Answer:

- Layout system for **responsive UI**.
- Key props: **flexDirection, justifyContent, alignItems**.
- Works across **all screen sizes**.
  💡 **Tip:** Master Flexbox for **consistent mobile layouts**.

Section 2: UI Components & Layout (Q6–Q12)

Q6. Difference between ScrollView and FlatList?
Answer:

- **ScrollView** → Renders all items at once (small lists).
- **FlatList** → Lazy-loads items (better for large lists).
- Prevents **performance issues** on long lists.
  💡 **Pro Tip:** Always prefer FlatList for **large datasets**.

Q7. What is KeyboardAvoidingView?
Answer:

- Adjusts UI when **keyboard appears**.
- Prevents **overlap with input fields**.
- Enhances **form usability**.
  💡 **Tip:** Always wrap **input-heavy screens**.

Q8. What is SafeAreaView?
Answer:

- Ensures content stays within **safe areas**.
- Avoids notches and interactive areas.
- Improves **UI consistency** on modern devices.
  💡 **Tip:** Use for **root-level screens**.

Q9. What is TouchableOpacity?
Answer:

- Makes **views clickable**.
- Reduces **opacity** on press.
- Used for **buttons or touchable items**.
  💡 **Tip:** Combine with **accessibility labels**.

Q10. What is the Platform module?
Answer:

- Detects **OS type** (iOS/Android).
- Enables **platform-specific code**.
- Useful for **conditional rendering/styling**.
  💡 **Tip:** Always test **both platforms**.

Q11. Dimensions API?
Answer:

- Provides **screen width & height**.
- Enables **responsive layouts**.
- Useful for **dynamic styling**.
  💡 **Tip:** Combine with Flexbox for **adaptive designs**.

Q12. PixelRatio API?
Answer:

- Returns **device pixel density**.
- Helps **scale images & UI elements**.
- Ensures **consistent look** across devices.
  💡 **Tip:** Use for **high-resolution graphics**.

Section 3: State & Props Management (Q13–Q19)

Q13. What are props?
Answer:

- Short for **properties**.
- Passed **from parent → child** component.

- **Immutable**, used for **configurations & dynamic content**.
  💡 **Tip:** Use props to **customize reusable components**.

Q14. What are state and setState?
Answer:

- **State** → Local data of a component.
- **setState/useState** → Updates state & triggers render.
- Stores **dynamic UI data**.
  💡 **Tip:** Manage **form inputs, counters, or toggles** with state.

Q15. What is Redux?
Answer:

- Predictable **state container**.
- Centralizes **app state** across components.
- Works seamlessly with **React & RN**.
  💡 **Tip:** Ideal for **complex state management** in large apps.

Q16. useEffect hook?
Answer:

- Handles **side effects** in functional components.
- Replaces componentDidMount & componentDidUpdate.
- Commonly used for **API calls or subscriptions**.
  💡 **Tip:** Always set **dependency arrays** to control execution.

Q17. useRef hook?
Answer:

- Stores **mutable values** across renders.
- Access **DOM or child component refs**.
- Does **not trigger re-renders**.
  💡 **Tip:** Use for **animations or focus management**.

Q18. useCallback hook?
Answer:

- Memoizes **functions** to prevent recreation.

- Optimizes **performance in child components**.
- Useful for **handlers & callbacks**.
  💡 **Tip:** Use with **expensive re-rendering components**.

Q19. useMemo hook?
Answer:

- Memoizes **calculated values**.
- Recomputes only when **dependencies change**.
- Optimizes **expensive computations**.
  💡 **Tip:** Prevent unnecessary **re-render calculations**.

Section 4: APIs & Device Features (Q20–Q30)

Q20. PureComponent & memo?
Answer:

- **PureComponent** → Optimizes class components.
- **memo** → Optimizes functional components.
- Prevents **unnecessary re-renders**.
  💡 **Tip:** Use in **performance-critical screens**.

Q21. Linking API?
Answer:

- Opens **external apps or URLs**.
- Supports **deep linking**.
- Enables **cross-app navigation**.
  💡 **Tip:** Use for **email, phone, or social app links**.

Q22. Alert API?
Answer:

- Displays **native alerts**.
- Supports **buttons & callbacks**.
- For **notifications & confirmations**.
  💡 **Tip:** Keep messages **short & actionable**.

Q23. Share API?
Answer:

- Shares content with **other apps**.
- Supports **text, URLs, images**.
- Opens **native share dialog**.
  💡 **Tip:** Use for **social sharing features**.

Q24. PushNotificationIOS?
Answer:

- Handles **push notifications on iOS**.
- Schedule or display notifications in real-time.
- Requires **native setup**.
  💡 **Tip:** Integrate with **APNs or third-party services**.

Q25. PermissionsAndroid?
Answer:

- Requests **runtime permissions** on Android.
- Needed for **camera, location, storage** access.
- Enhances **app security & UX**.
  💡 **Tip:** Always handle **denied permissions gracefully**.

Q26. AsyncStorage?
Answer:

- Key-value storage for **lightweight data**.
- Stores **tokens, preferences, flags**.
- Use @react-native-async-storage/async-storage.
  💡 **Tip:** Avoid for large datasets.

Q27. JS Bridge?
Answer:

- Connects **JS code to native APIs**.
- Converts calls **both ways**.
- Enables access to **device features**.
  💡 **Tip:** Minimize bridge calls for **better performance**.

Q28. Hot Reloading?
Answer:

- Reflects code changes **instantly** without rebuild.
- Preserves **component state**.
- Speeds up **development workflow**.
  💡 **Tip:** Use for **rapid prototyping**.

Q29. Metro Bundler?
Answer:

- Bundles & serves **JS code**.
- Supports **live reloading**.
- Optimized for **mobile performance**.
  💡 **Tip:** Know config for **custom builds**.

Q30. Expo?
Answer:

- Platform + tools for **easy RN development**.
- Provides **prebuilt libraries & deployment tools**.
- Speeds up **prototyping & testing**.
  💡 **Tip:** Great for **MVPs**, but may limit native modules.