Next.js Interview Q&As (Part 2)

(Routing, Data Fetching & App Router)

🔹 Section 1 Routing Basics

Q1. What is file-based routing in Next.js?
Answer: Pages inside /pages or /app become routes automatically.
Example: pages/about.js → /about.
No need for external routing libraries.
💡 Tip: Simpler than React Router setup.

Q2. How do you create dynamic routes?
Answer: Use square brackets → [id].js.
Example: pages/posts/[id].js → /posts/1.
Combine with getStaticPaths for SSG.
💡 Tip: Makes SEO-friendly URLs.

Q3. How do you handle nested routes?
Answer: Create folders inside /pages or /app.
Example: pages/dashboard/settings.js → /dashboard/settings.
💡 Tip: Folder structure = route structure.

Q4. Difference between Pages Router and App Router?
Answer: Pages Router → Old system (Next.js 12 and below).
App Router → Next.js 13+, uses server components and layouts.
💡 Tip: App Router is the future of Next.js.

Q5. How do layouts work in App Router?
Answer: Place layout.js inside /app.
Wraps all pages inside a section.
Supports nested layouts.
💡 Tip: Great for dashboards with shared sidebars.

🔹 Section 2 Data Fetching

Q6. What is getStaticProps used for?
Answer: Fetch data at build time (SSG).
Runs only on server, never client.

Props passed to page before render.
💡 Tip: Use for static blogs/docs.

Q7. What is getServerSideProps used for?
Answer: Runs at request time (SSR).
Fetches fresh data per request.
Returns props before rendering.
💡 Tip: Use for dashboards, dynamic data.

Q8. What is getStaticPaths used for?
Answer: Defines dynamic paths for SSG.
Works with [id].js.
Pre-renders pages at build time.
💡 Tip: Common for blogs/products.

Q9. How do you fetch data in App Router?
Answer: Use async server components with fetch().
Data fetching happens server-side.
No need for getStaticProps/getServerSideProps.
💡 Tip: Simpler and more efficient.

Q10. Difference between Server and Client Components?
Answer: Server Components → Default, run on server.
Client Components → "use client", run in browser.
💡 Tip: Use client only when interaction is needed.

Q11. What is ISR (Incremental Static Regeneration)?
Answer: Updates static pages after deployment.
Use revalidate in getStaticProps.
💡 Tip: Mix of SSG + live updates.

Q12. How do you revalidate pages in Next.js 13?
Answer: Use fetch(url, { next: { revalidate: 10 } }).
Rebuilds page every 10 seconds.
💡 Tip: Cleaner than older ISR method.

Q13. Can you use SWR/React Query in Next.js?
Answer: Yes, for client-side fetching.
SWR = caching, revalidation hooks.
Works well with App Router.
💡 Tip: Use when data updates on client frequently.

Q14. What is the difference between SSR and SSG?
Answer: SSR → Data fetched at request time (slower).

SSG → Data fetched at build time (faster).
💡 Tip: SSR for live data, SSG for static pages.

Q15. What is the difference between SSG and ISR?
Answer: SSG → Built once at build time.
ISR → Rebuilds periodically after deployment.
💡 Tip: ISR is “SSG with updates”.

🔹 Section 3 Middleware & API Routes

Q16. What is Middleware in Next.js?
Answer: Code that runs before request completes.
Used for auth, redirects, logging.
Defined in middleware.js.
💡 Tip: Runs at the Edge.

Q17. How do you handle redirects with Middleware?
Answer: Use NextResponse.redirect("/login").
Triggered before page loads.
💡 Tip: Perfect for authentication.

Q18. What are API routes in Next.js?
Answer: Functions inside /pages/api/.
Run on server only.
Return JSON or handle requests.
💡 Tip: Use for backend logic.

Q19. How do you fetch data from API routes?
Answer: Call /api/endpoint using fetch or Axios.
Handled on server, no CORS issue.
💡 Tip: Small apps don’t need external backend.

Q20. What is the difference between API Routes and Middleware?
Answer: API Routes → Handle requests, return responses.
Middleware → Runs before routes, used for checks.
💡 Tip: Middleware ≠ API replacement.

🔹 Section 4 Routing & Data Utilities

Q21. What is useRouter hook?
Answer: Provides route info (path, query, params).

Example: const router = useRouter().
Used for navigation and accessing query params.
💡 Tip: Works only in Client Components.

Q22. How do you navigate programmatically in Next.js?
Answer: Use useRouter().push('/path').
Alternative → router.replace() for no history.
💡 Tip: For conditional redirects.

Q23. What is shallow routing?
Answer: Updates URL without refetching data.
Enabled with router.push(url, undefined, { shallow: true }).
💡 Tip: Faster page updates.

Q24. How do you handle 404 pages in Next.js?
Answer: Create pages/404.js.
Automatically used when no route matches.
💡 Tip: Can customize with own design.

Q25. How do you handle custom error pages?
Answer: Create pages/_error.js.
Handles server errors.
💡 Tip: In App Router, use error.js.

🔹 Section 5 Advanced Routing

Q26. What are Catch-All routes?
Answer: Use [...slug].js.
Example: /pages/docs/[...slug].js → /docs/a/b/c.
💡 Tip: Use for nested docs/blog routes.

Q27. What are Optional Catch-All routes?
Answer: Use [[...slug]].js.
Matches /docs, /docs/a, /docs/a/b.
💡 Tip: Flexible dynamic routing.

Q28. How do you create API route dynamic parameters?
Answer: Define [id].js in /pages/api/.
Access via req.query.id.
💡 Tip: Works same as page routing.

Q29. How do you pass query parameters in Next.js?
Answer: /about?name=John.

Access via router.query in Client Components.
💡 Tip: Always check for undefined during SSR.

Q30. What is the role of next.config.js in routing/data?
Answer: Configure redirects, rewrites, headers.
Also used for image domains & env variables.
💡 Tip: Important for custom configs