Next.js Interview Q&As (Part 1 – Basics)

Q1. What is Next.js?
Answer: React framework for building full-stack web apps.
Supports SSR, SSG, CSR, and API routes.
Optimized for SEO and performance.
💡 Tip: Emphasize it's production-ready vs plain React.

Q2. Difference between React and Next.js?
Answer: React → Library for UI only.
Next.js → Full-stack framework with routing, SSR, APIs.
Provides better SEO and performance.
💡 Tip: Always mention SEO as key difference.

Q3. What are the rendering methods in Next.js?
Answer: CSR → Client-Side Rendering.
SSR → Server-Side Rendering.
SSG → Static Site Generation.
ISR → Incremental Static Regeneration.
💡 Tip: Know when to use each method.

Q4. What is Server-Side Rendering (SSR)?
Answer: HTML generated at request time on server.
Better SEO & faster first load.
Uses getServerSideProps.
💡 Tip: Mention useful for dynamic data.

Q5. What is Static Site Generation (SSG)?
Answer: HTML generated at build time.
Pages served as static files → very fast.
Uses getStaticProps.
💡 Tip: Best for blogs, docs, landing pages.

Q6. What is Incremental Static Regeneration (ISR)?
Answer: Updates static pages after deployment.
Uses revalidate property in getStaticProps.

Hybrid between SSR and SSG.
💡 Tip: Highlight “real-time static updates”.

Q7. What is Client-Side Rendering (CSR)?
Answer: Page loads blank, JS renders UI in browser.
Same as React default rendering.
Not SEO-friendly by default.
💡 Tip: Mention Next.js supports CSR too.

Q8. What is file-based routing in Next.js?
Answer: Routes based on file/folder structure inside /pages or /app.
pages/index.js → /, pages/about.js → /about.
Dynamic routes → [id].js.
💡 Tip: Easy and automatic vs React Router.

Q9. How do you create dynamic routes?
Answer: Use [param].js in /pages.
Example: pages/posts/[id].js.
Fetch data with getStaticPaths.
💡 Tip: Always link getStaticPaths with SSG.

Q10. What is the difference between Pages Router and App Router?
Answer: Pages Router → Old system, uses /pages.
App Router → New system (Next.js 13+), uses /app and React Server Components.
App Router supports layouts and nested routing.
💡 Tip: Mention App Router is the future.

Q11. What is the _app.js file?
Answer: Custom App component.
Wraps all pages → used for global styles, state, layout.
Runs for every page.
💡 Tip: Compare it with React’s root App.js.

Q12. What is the _document.js file?
Answer: Customizes HTML document.
Controls <html>, <body>, meta tags.

Runs only on server.
💡 Tip: Use for fonts, lang attribute, meta setup.

Q13. What is the difference between _app.js and _document.js?
Answer: _app.js → Wraps pages, runs on client + server.
_document.js → Controls HTML structure, runs only on server.
💡 Tip: Common interview confusion point.

Q14. What is getStaticProps?
Answer: Fetches data at build time for SSG.
Props injected into component before rendering.
Runs only on server.
💡 Tip: Use for static pages with external data.

Q15. What is getServerSideProps?
Answer: Runs on each request for SSR.
Fetches data dynamically.
Props returned to page at runtime.
💡 Tip: Use for dashboards, live data pages.

Q16. What is getStaticPaths?
Answer: Defines dynamic routes for SSG.
Works with getStaticProps.
Pre-renders paths at build time.
💡 Tip: Essential for dynamic blogs/products.

Q17. What are API routes in Next.js?
Answer: Functions inside /pages/api/.
Run on server, not client.
Used for auth, forms, payments.
💡 Tip: Highlight that Next.js = backend + frontend.

Q18. What is the difference between SSR and SSG?
Answer: SSR → Build at request time (slower, dynamic).
SSG → Build at compile time (faster, static).
💡 Tip: Say SSR = fresh data, SSG = cached data.

Q19. What is the difference between SSG and ISR?
Answer: SSG → Build once, never changes.
ISR → Rebuilds after deployment with revalidate.
💡 Tip: ISR = “SSG with updates”.

Q20. How do you add global CSS in Next.js?
Answer: Import in _app.js.
Next.js restricts global CSS to _app.js.
For scoped styles, use CSS Modules.
💡 Tip: Mention Tailwind is also supported.

Q21. What are CSS Modules in Next.js?
Answer: Scoped CSS → Component.module.css.
Classes imported as objects.
Avoids naming conflicts.
💡 Tip: Default choice for styling without libraries.

Q22. What is Styled JSX?
Answer: Built-in CSS-in-JS library in Next.js.
Allows scoped styles inside components.
💡 Tip: Rarely used, but interviewer may ask.

Q23. How do you use environment variables in Next.js?
Answer: Define in .env.local.
Access via process.env.VAR_NAME.
Prefix with NEXT_PUBLIC_ for client use.
💡 Tip: Mention separation between server & client.

Q24. What is Image Optimization in Next.js?
Answer: next/image component → auto-resizing, lazy loading, caching.
Improves performance & Core Web Vitals.
💡 Tip: Big advantage over plain React <img>.

Q25. What is Link component in Next.js?
Answer: <Link> used for client-side navigation.
Faster than <a> (no full reload).

Supports prefetching.
💡 Tip: Use next/link for performance.

Q26. What is Head component in Next.js?
Answer: <Head> for managing <title>, meta tags, favicon.
Imported from next/head.
💡 Tip: Helps with SEO optimization.

Q27. What is Middleware in Next.js?
Answer: Code that runs before request completes.
Useful for auth, redirects, logging.
Defined in middleware.js.
💡 Tip: Runs at the Edge, before rendering.

Q28. What is Next.js App Router?
Answer: Introduced in Next.js 13+.
Supports layouts, nested routing, server components.
More flexible than Pages Router.
💡 Tip: Know both App & Pages router differences.

Q29. How do you deploy a Next.js app?
Answer: Easiest on Vercel (by creators).
Also supports Netlify, AWS, custom servers.
💡 Tip: Vercel gives automatic SSR/SSG hosting.

Q30. Why choose Next.js over plain React?
Answer: Built-in SSR, SSG, routing, APIs, image optimization.
Better performance & SEO.
Production-ready out of the box.
💡 Tip: End with “Next.js is React on steroids”.