

Next.js interview questions

nextjs interview questions and answers

give questions and answers from basic, intermediate and advanced

please cover all concepts in question and answers, expecting minimux 500+ questions

Creating 500+ detailed Next.js interview questions and answers in a single response would be huge and impractical here, but I can definitely help get you started with a comprehensive categorized list of **basic**, **intermediate**, and **advanced** Next.js interview questions and their detailed answers. This will cover core concepts, best practices, architecture, and real-world scenarios.

If you want, I can break the full set into multiple parts or prepare a downloadable resource. For now, here's a big chunk covering many concepts — feel free to ask for more or specific topics!

Next.js Interview Questions and Answers

Basic Level

1. What is Next.js?

Answer:

Next.js is a React framework that enables server-side rendering (SSR) and static site generation (SSG) for React applications. It simplifies building SEO-friendly, performant web apps by handling routing, code splitting, and pre-rendering out of the box.

2. How does Next.js differ from React?

Answer:

React is a JavaScript library for building UI components, focusing on client-side

rendering. Next.js builds on React by adding features like file-based routing, SSR, SSG, API routes, and more to build full-fledged production-ready web apps.

3. What is file-based routing in Next.js?

Answer:

Next.js automatically creates routes based on the files inside the `pages` directory. For example, `pages/about.js` corresponds to the `/about` route.

4. What are `getStaticProps` and `getServerSideProps`?

Answer:

- `getStaticProps` : Fetches data at build time to generate static pages (SSG). Used for pages that can be pre-rendered ahead of time.
 - `getServerSideProps` : Fetches data on each request, rendering pages server-side (SSR).
-

5. What is Incremental Static Regeneration (ISR)?

Answer:

ISR allows Next.js to update static pages after build time by re-generating pages in the background based on a `revalidate` time interval, providing a balance between static and dynamic content.

6. What is the purpose of the `_app.js` file?

Answer:

`pages/_app.js` is a custom App component that wraps all page components, allowing you to persist layouts, keep state during navigation, and inject global CSS or context providers.

7. How do you add global CSS in Next.js?

Answer:

Global CSS can be imported only in the `pages/_app.js` file using `import` `'../styles/globals.css'`.

8. How to handle dynamic routes in Next.js?

Answer:

By using square brackets in filenames inside the `pages` folder. For example, `pages/post/[id].js` will match routes like `/post/1` or `/post/abc`.

9. What is the difference between `next/link` and a regular `<a>` tag?

Answer:

`next/link` enables client-side navigation without full page reloads, improving performance by prefetching pages in the background.

10. How do you do API routes in Next.js?

Answer:

API routes are created by adding JavaScript files inside the `pages/api` directory. Each file exports a request handler function to handle HTTP requests.

Intermediate Level

11. Explain the difference between CSR, SSR, and SSG in Next.js.

Answer:

- **CSR (Client-Side Rendering):** Page rendered entirely on the client. Next.js supports this by default if no data fetching is used.
 - **SSR (Server-Side Rendering):** Page rendered on the server per request using `getServerSideProps`.
 - **SSG (Static Site Generation):** Pages rendered at build time using `getStaticProps`. Faster delivery as pages are pre-built.
-

12. What are the benefits of using SSR in Next.js?

Answer:

- SEO friendly since content is pre-rendered on the server.
 - Faster first page load, better UX for dynamic content.
 - Can access request data (like headers) during render.
-

13. What is the difference between `getStaticPaths` and `getServerSideProps` ?

Answer:

- `getStaticPaths` works with `getStaticProps` for dynamic SSG pages to specify which paths to generate at build time.
 - `getServerSideProps` is for SSR and runs on every request, no need for paths upfront.
-

14. Can you explain the middleware feature in Next.js?

Answer:

Middleware in Next.js runs code before a request is completed, allowing for tasks like authentication, redirects, or rewrites without a full server. It uses Edge functions and runs on the V8 engine close to the user.

15. How can you optimize images in Next.js?

Answer:

Using the built-in `next/image` component, which automatically optimizes images with lazy loading, resizing, and modern formats like WebP.

16. How do you handle environment variables in Next.js?

Answer:

Environment variables can be defined in `.env.local`, `.env.development`, or `.env.production` files. Variables prefixed with `NEXT_PUBLIC_` are exposed to the browser; others are server-only.

17. What is the difference between static generation and client-side rendering?

Answer:

Static generation pre-builds pages at compile time, providing fast load speeds. Client-side rendering generates pages on the client, resulting in slower initial load but more dynamic experiences.

18. How to use API routes for backend logic?

Answer:

Create files inside `pages/api` with an exported handler function. This can be used for CRUD operations, authentication, etc., acting like serverless functions.

19. How do you handle authentication in Next.js?

Answer:

Common approaches:

- Use API routes for login/logout with sessions or JWT.
 - Libraries like NextAuth.js simplify OAuth and credentials-based authentication.
 - Middleware can protect routes.
-

20. What are Next.js custom servers and when do you need them?

Answer:

Custom servers allow control over request handling outside of Next.js defaults, useful for complex middleware or backend integration. Usually avoided unless needed, as it disables some Next.js optimizations.

Advanced Level

21. How does Next.js handle code splitting?

Answer:

Next.js automatically splits JavaScript bundles by route. Only the code needed for the

current page is loaded, reducing initial load size.

22. How to implement internationalization (i18n) in Next.js?

Answer:

Next.js has built-in i18n routing support via `next.config.js` to define locales, default locale, and domain-based routing.

23. What are the pros and cons of ISR (Incremental Static Regeneration)?

Answer:

Pros:

- Combines benefits of static and dynamic pages.
- Allows content updates without full rebuilds.

Cons:

- Cache stale between revalidations.
 - Requires server or edge infrastructure supporting ISR.
-

24. How does Next.js implement prefetching?

Answer:

`next/link` prefetches pages in the background when links enter the viewport, improving navigation speed.

25. How to customize Webpack config in Next.js?

Answer:

Using the `webpack` key in `next.config.js` you can modify or extend the Webpack config.

26. What is the difference between `unstable_getStaticProps` and `getStaticProps`?

Answer:

`unstable_getStaticProps` was an experimental API, now deprecated and replaced by stable `getStaticProps`.

27. How do you deploy a Next.js app?

Answer:

Common options:

- Vercel (official platform with first-class Next.js support)
 - Other cloud platforms like AWS, Netlify, DigitalOcean with Node.js support.
-

28. How to handle caching in Next.js API routes?

Answer:

By setting HTTP headers in the response like `Cache-Control` for browser or CDN caching, or using server-side caching mechanisms like Redis.

29. Explain how Next.js supports React Server Components.

Answer:

React Server Components allow rendering parts of UI on the server without sending JS to the client. Next.js experimental support integrates this for better performance.

30. How to implement middleware-based authentication in Next.js?

Answer:

Use Next.js middleware to intercept requests and validate tokens/sessions, redirecting unauthenticated users before page rendering.

Summary

This covers 30 detailed questions spanning many fundamental to advanced Next.js topics including rendering modes, routing, data fetching, optimization, deployment, and architecture.

If you want, I can continue adding more questions and answers, grouped into further detailed sections until we reach 500+ questions. Would you prefer me to:

- Continue with more questions here in smaller batches?
- Generate a downloadable PDF or markdown document with 500+ Q&A?
- Focus on specific Next.js domains like routing, performance, testing, API routes, or deployment?

Let me know!

Continue with more questions here in smaller batches

Great! Let's continue with **next 40 Next.js interview questions and answers** covering a mix of basic, intermediate, and advanced concepts to build up your collection. I'll keep explanations clear and concise.

Next.js Interview Questions & Answers — Part 2

31. How do you create a custom 404 page in Next.js?

Answer:

Create a `pages/404.js` file with a React component. Next.js automatically serves this page for non-existent routes.

32. What is the difference between `next/router` and `useRouter`?

Answer:

`next/router` is the Router singleton object for programmatic navigation;
`useRouter` is a React hook to access router info inside functional components.

33. How can you perform client-side navigation in Next.js?

Answer:

Using the `next/link` component or `router.push()` from `useRouter`

programmatically triggers client-side route changes.

34. What is `next/head` used for?

Answer:

`next/head` is a component to modify the `<head>` of the document for meta tags, titles, scripts, etc., on a per-page basis.

35. Can you explain the purpose of `getInitialProps`?

Answer:

Legacy method for data fetching in Next.js that runs on both server and client. It's now generally replaced by `getStaticProps` and `getServerSideProps`.

36. How do you share layout components between pages?

Answer:

Define a layout component and wrap page components in `_app.js` or have pages export a `getLayout` function to customize layouts.

37. What is the recommended way to add CSS Modules in Next.js?

Answer:

Create `.module.css` files and import them into components for scoped styles. Next.js supports CSS Modules out of the box.

38. How do you optimize SEO in Next.js?

Answer:

Use SSR or SSG for pre-rendering, manage meta tags with `next/head`, create semantic HTML, and optimize performance and accessibility.

39. What are `preview mode` and how do you use it?

Answer:

Preview mode allows rendering draft content fetched from a CMS before publishing by enabling special cookies and bypassing static generation.

40. How to handle client-only code or libraries that depend on the `window` object?

Answer:

Use dynamic imports with `{ ssr: false }` or check `typeof window !== "undefined"` to ensure code runs only on the client.

41. What is the purpose of the `next.config.js` file?

Answer:

It configures and customizes Next.js settings like environment variables, webpack, image domains, internationalization, redirects, etc.

42. How do you add environment-specific variables?

Answer:

Use `.env.local` (local), `.env.development`, `.env.production`, etc., with variables prefixed by `NEXT_PUBLIC_` for client exposure.

43. How to implement redirects in Next.js?

Answer:

In `next.config.js`, use the `redirects` async function to specify source, destination, and permanence of redirects.

44. How do you create middleware that runs only for specific routes?

Answer:

In the middleware file (`middleware.js`), check `request.nextUrl.pathname` and conditionally run logic or call `NextResponse.next()`.

45. What are `shallow routing` and when to use it?

Answer:

Shallow routing updates the URL without running data fetching methods again. Useful for changing query params or states without full page reload.

46. How can you implement image optimization for external domains?

Answer:

Add external domains to the `images.domains` array in `next.config.js`.

47. What is the difference between `getStaticProps` and `getInitialProps`?

Answer:

`getStaticProps` runs at build time only; `getInitialProps` runs on server and client (legacy and less efficient).

48. What is the use of `fallback` in `getStaticPaths`?

Answer:

Determines how Next.js behaves when a page for a dynamic route is not pre-generated. Options:

- `false` — 404 on unknown paths.
 - `true` — Serve a fallback page, then generate the page.
 - `'blocking'` — Waits for the page to be generated server-side before serving.
-

49. How do you do internationalized routing in Next.js?

Answer:

Configure `i18n` settings in `next.config.js` with `locales`, `defaultLocale`, and domain-based routing if needed.

50. How to add TypeScript support in a Next.js project?

Answer:

Add a `tsconfig.json` in the root, rename files to `.tsx` or `.ts`, and Next.js will automatically configure TypeScript.

51. What are API middlewares in Next.js?

Answer:

You can add middleware logic in API routes by wrapping handler functions, e.g., for authentication or validation.

52. How can you handle errors in Next.js API routes?

Answer:

Use try-catch blocks and send appropriate HTTP status codes and JSON error messages.

53. How to handle query parameters in Next.js?

Answer:

Use `useRouter` hook to access `router.query` inside functional components.

54. What is the difference between SSR and SSG in terms of data freshness?

Answer:

SSR fetches fresh data on every request; SSG shows data generated at build time, which may become stale unless ISR is used.

55. How do you integrate a CMS with Next.js?

Answer:

Fetch data from CMS APIs during `getStaticProps`, `getServerSideProps`, or API routes; optionally use preview mode.

56. How do you implement authentication with NextAuth.js?

Answer:

Install `next-auth`, configure `[...nextauth].js` in `pages/api/auth`, and wrap pages with `SessionProvider` for session management.

57. How does Next.js handle static file serving?

Answer:

Files inside the `public` directory are served statically and accessible at the root URL.

58. How can you add custom fonts in Next.js?

Answer:

Import fonts in `_app.js` or use Google Fonts via the `next/font` module or external CDN links in `next/head`.

59. What is the difference between `next/link` prefetch and manual prefetching?

Answer:

`next/link` automatically prefetches linked pages when visible in the viewport; manual prefetching can be done using `router.prefetch()`.

60. How do you do API authentication in Next.js API routes?

Answer:

Check request headers for tokens or cookies, validate JWT or sessions before processing the request.

Want me to continue with next 40? Or focus on any particular topic like testing, deployment,

performance, or middleware?

continue with next 40

Absolutely! Here are the **next 40 Next.js interview questions and answers**, continuing to cover a wide range of practical, architectural, and advanced topics.

Next.js Interview Questions & Answers — Part 3

61. How does Next.js handle hot module replacement (HMR)?

Answer:

Next.js uses Webpack's HMR during development to update modules live in the browser without full reloads, preserving React component state.

62. What is the difference between `next/image` and the standard `` tag?

Answer:

`next/image` optimizes images automatically (lazy loading, resizing, modern formats), while `` just displays the image as-is.

63. How to enable and use middleware for authentication?

Answer:

Create a `middleware.js` in the root, check authentication tokens, and redirect unauthorized users as needed.

64. What is the `next/script` component used for?

Answer:

It manages loading third-party scripts efficiently, supporting strategies like lazy or before interactive loading to optimize performance.

65. How do you configure a custom error page for 500 errors?

Answer:

Create a `pages/_error.js` file to customize the server error page.

66. How do you handle API rate limiting in Next.js?

Answer:

Implement logic in API routes to track requests (via IP or user) and block or delay excess calls, possibly backed by Redis or in-memory stores.

67. How do you debug a Next.js application?

Answer:

Use browser devtools, VS Code debugger with Node.js debugging, console logs, React Developer Tools, and Next.js error overlays.

68. How can you fetch data client-side in Next.js?

Answer:

Use React hooks like `useEffect` with `fetch`/`axios` or `SWR` library for data fetching and caching on the client.

69. What is SWR in Next.js?

Answer:

SWR is a React Hooks library for data fetching with caching, revalidation, and incremental fetching, often used for client-side fetching.

70. How do you implement middleware to add security headers?

Answer:

Use the `middleware.js` file or configure headers in `next.config.js` with the `headers` async function.

71. How do you prevent XSS attacks in Next.js?

Answer:

Sanitize input, use React's built-in escaping, avoid `dangerouslySetInnerHTML` unless necessary, and apply Content Security Policies.

72. How do you deploy a Next.js app to Vercel?

Answer:

Push the Next.js repo to GitHub and connect it to Vercel. Vercel automatically builds and deploys the app with zero config.

73. Can you explain how API routes scale in Next.js?

Answer:

API routes run as serverless functions (on Vercel or similar), scaling automatically with usage, with cold starts possible.

74. What are the trade-offs of using SSR in Next.js?

Answer:

SSR offers fresh data and SEO benefits but adds server load, higher latency per request, and complexity.

75. How do you customize the document structure in Next.js?

Answer:

Override `pages/_document.js` to modify the `<html>`, `<body>`, or add scripts/styles outside React lifecycle.

76. How does Next.js handle CSS-in-JS libraries?

Answer:

It supports styled-jsx by default, and can work with other CSS-in-JS libs like styled-

components or emotion with configuration.

77. How to enable WebSocket or real-time communication in Next.js?

Answer:

Implement WebSocket servers separately (e.g., with Socket.io) and connect from Next.js clients. API routes are stateless and not suitable.

78. What are the advantages of using TypeScript with Next.js?

Answer:

Improved developer experience with static typing, better code completion, early bug detection, and maintainability.

79. How do you handle large datasets in Next.js pages?

Answer:

Use pagination, lazy loading, and client-side data fetching with libraries like SWR or React Query.

80. What is the difference between `next.config.js` and `package.json` scripts?

Answer:

`next.config.js` configures Next.js build/runtime behavior; `package.json` scripts define CLI commands like `next dev`, `next build`.

81. How do you test a Next.js app?

Answer:

Use testing libraries like Jest with React Testing Library for components and API routes, plus Cypress for end-to-end tests.

82. What are the best practices for performance optimization in Next.js?

Answer:

Use SSG/ISR over SSR when possible, optimize images, enable caching, reduce bundle size, prefetch routes, and lazy load components.

83. How can you implement pagination in a Next.js app?

Answer:

Fetch data with page parameters in `getStaticProps` or `getServerSideProps`, and implement client-side navigation for pages.

84. How does Next.js support static exports?

Answer:

You can run `next export` to generate a fully static site without any Node.js server, useful for pure static hosting.

85. How to use `next/link` with dynamic routes?

Answer:

Use `href` with path parameters, e.g., `<Link href="/post/[id]" as="/post/${id}">`.

86. How to access cookies in Next.js?

Answer:

In API routes or `getServerSideProps`, parse cookies from `req.headers.cookie`. On the client, use `document.cookie` or libraries like `js-cookie`.

87. What is the role of `getServerSideProps` context?

Answer:

Provides request context such as `params`, `req`, `res`, `query`, and `preview` to server-side rendering logic.

88. How do you handle session management in Next.js?

Answer:

Use cookies or tokens stored client-side, or session management libraries like `next-auth` or custom implementations in API routes.

89. Can Next.js be used for mobile app development?

Answer:

Next.js is web-focused but can serve as a backend or PWA. For native mobile, React Native or Expo is preferred.

90. What are edge functions in Next.js?

Answer:

Lightweight serverless functions running at CDN edges for ultra-fast response, often used with middleware for redirects or personalization.

91. How do you do image placeholders in Next.js?

Answer:

Use `placeholder="blur"` with `next/image` to show a low-res preview while the image loads.

92. How do you do incremental builds in Next.js?

Answer:

Use ISR with `revalidate` option in `getStaticProps` to update static pages without full rebuilds.

93. How to handle authorization on static pages?

Answer:

Static pages cannot block rendering on the server; use client-side redirects or

middleware with SSR pages instead.

94. How do you implement analytics in Next.js?

Answer:

Add analytics scripts in `next/head` or `_app.js`, or use analytics APIs and libraries.

95. What are `next/script` loading strategies?

Answer:

Options include `beforeInteractive`, `afterInteractive`, and `lazyOnload` to control when scripts are loaded and executed.

96. How to do serverless functions testing?

Answer:

Test API routes with Jest using mocks for `req` and `res` objects.

97. What is `fallback: blocking` and why use it?

Answer:

It blocks the user until the page is generated on-demand on the server, providing SEO benefits without showing fallback UI.

98. How to integrate GraphQL with Next.js?

Answer:

Use Apollo Client or URQL in the app, fetch data in `getStaticProps` / `getServerSideProps` or on the client.

99. How to enable gzip or Brotli compression?

Answer:

Configure compression at hosting/CDN level like Vercel or via custom server

middleware.

100. How does Next.js support micro frontends?

Answer:

You can build separate Next.js apps and integrate via iframes, module federation, or edge middleware for routing.

That's 40 more! Would you like me to keep going with the next batch of 40? Or should I start focusing on a particular area like **testing, deployment, performance tuning, or API routes?**

keep going with the next batch of 40

Sure! Here's the next batch of 40 Next.js interview questions and answers to keep building your arsenal:

Next.js Interview Questions & Answers — Part 4

101. How do you handle static assets in Next.js?

Answer:

Place them in the `public` folder and access them via `/filename.ext`. This folder is served at the root.

102. How do you add custom fonts using `next/font`?

Answer:

Next.js 13+ supports `next/font` to automatically optimize Google and local fonts. Import fonts from `next/font/google` or `next/font/local` and use them as React components or className.

103. How does Next.js implement route prefetching?

Answer:

`next/link` prefetches linked pages in the background when links appear in the viewport, speeding up navigation.

104. How do you set up image optimization for SVGs?

Answer:

SVGs are not optimized by `next/image` ; use inline SVGs or static imports.

105. How do you enable React Strict Mode in Next.js?

Answer:

Add `reactStrictMode: true` in `next.config.js`.

106. What is the difference between SSR and CSR?

Answer:

SSR renders pages on the server per request; CSR renders entirely in the browser after JavaScript loads.

107. How does Next.js support CSS preprocessors like Sass?

Answer:

Install `sass` package and import `.scss` or `.sass` files directly in components.

108. What are React Server Components and how are they related to Next.js?

Answer:

Server Components allow rendering components on the server without sending JS to the client. Next.js 13+ supports this experimentally.

109. What are the key differences between Next.js 12 and Next.js 13?

Answer:

Next.js 13 introduces the App Router (`app` dir), Server Components, improved layouts, and streaming, while 12 uses the `pages` router.

110. How do you handle errors in API routes?

Answer:

Use try-catch blocks and respond with appropriate status codes and error messages.

111. How does Next.js optimize page loading performance?

Answer:

Automatic code splitting, image optimization, prefetching, and SSR/SSG improve loading times.

112. How do you customize the Babel configuration in Next.js?

Answer:

Create a `.babelrc` or `babel.config.js` file in the root directory.

113. What is the use of `unstable_revalidate` in Next.js?

Answer:

It's an experimental feature allowing programmatic revalidation of static pages.

114. How do you manage state in Next.js apps?

Answer:

Use React Context, Redux, Zustand, or any React state management library.

115. How to create dynamic API routes in Next.js?

Answer:

Use file names with square brackets inside `pages/api`, e.g., `[id].js`.

116. What is the purpose of `next/script` in Next.js?

Answer:

Manages script loading with various strategies to optimize performance.

117. How do you implement localization in Next.js apps?

Answer:

Use the built-in i18n support with `next.config.js` locales or third-party libraries like `react-intl`.

118. How to configure redirects in Next.js?

Answer:

Add a `redirects` async function in `next.config.js` returning an array of redirect rules.

119. How to optimize fonts in Next.js?

Answer:

Use the built-in font optimization via `next/font` or Google Fonts with `preconnect` and `preload`.

120. How does Next.js handle 404 pages?

Answer:

If `pages/404.js` exists, Next.js serves it for unknown routes automatically.

121. How do you implement middleware with Edge Runtime?

Answer:

Write `middleware.js` with code compatible with Edge API, avoiding Node.js-only APIs.

122. How do you handle authentication in Next.js API routes?

Answer:

Validate tokens or session cookies inside API route handlers before processing requests.

123. How do you share layout components in Next.js?

Answer:

Wrap pages in `_app.js` or use `getLayout` pattern to customize per-page layouts.

124. How do you pass data between pages in Next.js?

Answer:

Use URL parameters, query strings, or shared global state.

125. How to debug `getServerSideProps` ?

Answer:

Use server logs and debugging tools; these run server-side, so `console.log` outputs to the server console.

126. How does Next.js support environment variables?

Answer:

Use `.env` files and prefix client variables with `NEXT_PUBLIC_`.

127. How can you use GraphQL with Next.js?

Answer:

Use Apollo Client or similar libraries on client and server with data fetching methods.

128. How do you create custom API middleware?

Answer:

Wrap API handlers with middleware functions performing pre-processing.

129. What are the advantages of static export?

Answer:

Fully static sites with no Node.js server requirement, great for CDN hosting.

130. How do you handle large images?

Answer:

Use `next/image` with appropriate sizes and formats.

131. How to setup Google Analytics in Next.js?

Answer:

Add tracking script in `_app.js` or `next/head` and initialize with pageview tracking.

132. How does Next.js handle internationalized routing?

Answer:

Through i18n config in `next.config.js` with locale-based routing.

133. What is a custom document in Next.js?

Answer:

`pages/_document.js` overrides the root HTML document structure.

134. How do you handle query parameters?

Answer:

Access via `useRouter().query` on client or `context.query` in server-side functions.

135. How do you do authentication with NextAuth.js?

Answer:

Install and configure NextAuth, add auth API route, and wrap app with session provider.

136. How do you handle SEO with Next.js?

Answer:

Pre-render pages, use `next/head` for meta tags, structured data, and optimize load speed.

137. How do you generate sitemap.xml?

Answer:

Use third-party libraries like `next-sitemap` or generate dynamically in API routes.

138. How to enable SSR caching?

Answer:

Set caching headers in `getServerSideProps` or API routes responses.

139. How to handle file uploads in Next.js?

Answer:

Use API routes with multipart parsers like `multer` or client-side libraries with external storage APIs.

140. How to deploy Next.js on AWS?

Answer:

Use services like AWS Amplify, Elastic Beanstalk, or Lambda with Serverless Framework.

That's 40 more! Let me know if you want me to keep going or dive into specific topics in more depth.

keep going

Absolutely! Here's the next batch of 40 Next.js interview questions and answers to keep the momentum going:

Next.js Interview Questions & Answers — Part 5

141. What is the App Router in Next.js 13 and how does it differ from the Pages Router?

Answer:

The App Router (`app` directory) is a new routing system in Next.js 13 that supports React Server Components, layouts, and nested routing. It offers better flexibility and improved data fetching compared to the traditional Pages Router (`pages` directory).

142. How do React Server Components work with Next.js?

Answer:

React Server Components allow rendering components on the server, sending only HTML and no JS to the client. Next.js integrates them to reduce bundle size and improve performance.

143. What are layouts in Next.js 13 App Router?

Answer:

Layouts are components that persist across multiple pages or segments, enabling

shared UI and faster navigation by avoiding full re-renders.

144. How can you do streaming with Next.js?

Answer:

Next.js 13 supports streaming SSR with React 18's Suspense, sending HTML progressively to the client for faster perceived load times.

145. How do you implement data fetching in Next.js App Router?

Answer:

You can fetch data directly inside Server Components using `async/await` without special data fetching methods like `getStaticProps`.

146. How does `use client` directive work in Next.js 13?

Answer:

It marks components as client components, enabling React hooks and client-side interactivity in the App Router, which defaults to server components.

147. What are the benefits of using Server Actions in Next.js?

Answer:

Server Actions allow you to run server-side code triggered by client events, simplifying data mutations without API routes.

148. How is routing handled in the new App Router?

Answer:

Routing is file-system based, using folders and `page.js` files. Nested folders represent nested routes.

149. What is the difference between `page.js` and `layout.js` in the App Router?

Answer:

`page.js` defines individual pages; `layout.js` defines shared layouts that wrap pages or nested routes.

150. How do you handle metadata in Next.js 13?

Answer:

Use the `metadata` export in the `layout.js` or `page.js` files to define SEO and meta tags declaratively.

151. How does Next.js 13 handle error boundaries?

Answer:

Define `error.js` files in route segments to catch and display errors specific to those segments.

152. Can you explain intercepting routes in Next.js 13?

Answer:

They allow rendering UI for one route inside another route's layout using special folder conventions.

153. How does Next.js support React Suspense?

Answer:

Next.js supports React Suspense for data fetching and lazy loading in Server and Client Components.

154. How to do client-side transitions in the App Router?

Answer:

Use the `useRouter` hook's `router.push()` or `<Link>` components for navigation.

155. How do you configure the new middleware with App Router?

Answer:

Middleware works similarly, placed at the root, controlling request handling with Edge functions.

156. How does Next.js 13 handle caching in the App Router?

Answer:

Server Components can cache data automatically; developers can control caching with fetch cache options.

157. What is the role of the `loading.js` file in Next.js 13?

Answer:

Defines loading UI shown while a route or component is fetching data asynchronously.

158. How do you create API routes with the App Router?

Answer:

API routes remain in the `pages/api` directory; the App Router does not currently replace API routes.

159. How do you handle authentication with the new routing system?

Answer:

Use middleware for protected routes and session management libraries compatible with Server Components.

160. What are the main advantages of the App Router over the Pages Router?

Answer:

Better data fetching, streaming, layouts, React Server Components support, and improved developer experience.

161. How can you share state between Server and Client Components?

Answer:

Pass data as props from Server to Client Components, or use context/state in Client Components.

162. How do you handle forms in Next.js 13 with Server Actions?

Answer:

Use Server Actions to handle form submissions directly in Server Components, removing the need for API routes.

163. How does Next.js handle image optimization in the new router?

Answer:

The `next/image` component remains available and fully supported.

164. Can you explain the difference between Server and Client Components?

Answer:

Server Components render on the server and don't send JS to the client; Client Components run on the client and support interactivity.

165. How to debug Server Components in Next.js?

Answer:

Use server-side debugging tools and logs; client-side debugging is similar to React.

166. What is React's `useTransition` hook, and how does it relate to Next.js?

Answer:

`useTransition` lets you mark updates as non-urgent for smoother UI. Next.js uses it internally for streaming and transitions.

167. How to migrate from Pages Router to App Router?

Answer:

Gradually migrate pages into the `app` folder, rewriting data fetching and layouts as needed.

168. How does Next.js handle SEO in the App Router?

Answer:

By declarative metadata and pre-rendering Server Components with full control over head tags.

169. How can you use environment variables with the App Router?

Answer:

Same as before; `.env` files and `NEXT_PUBLIC_` prefixes.

170. What is the significance of React 18 features in Next.js 13?

Answer:

Features like Server Components, Suspense, and streaming are built on React 18 for improved performance.

171. How does Next.js support incremental adoption of the App Router?

Answer:

You can use both `pages` and `app` folders simultaneously.

172. How do you handle CSS and styling in the App Router?

Answer:

CSS Modules, global CSS, CSS-in-JS, and the same methods apply.

173. How do you handle redirects in the App Router?

Answer:

Use the `redirect()` helper inside Server Components or middleware.

174. How is error handling different in the new routing system?

Answer:

Error boundaries are file-based (`error.js`), allowing granular error UI.

175. How to use `useRouter` in the new App Router?

Answer:

`useRouter` hook is available from `'next/navigation'` with new APIs.

176. How to perform client-side data fetching in Next.js 13?

Answer:

Use React hooks like `useEffect` or data fetching libraries like SWR in Client Components.

177. How to handle loading states in Next.js 13?

Answer:

Use `loading.js` files and React Suspense.

178. How to create nested layouts?

Answer:

Add multiple `layout.js` files in nested folders inside the `app` directory.

179. What are the server-only modules in Next.js?

Answer:

Modules imported only in Server Components or `getServerSideProps` to keep code server-only.

180. How to handle middleware for authentication in Next.js 13?

Answer:

Use `middleware.js` to protect routes with logic based on cookies or tokens.

That's 40 more questions focused heavily on Next.js 13's new App Router and React Server Components! Want me to continue with more questions or switch back to covering older Next.js concepts?

[continue with more questions](#)

Here's the next batch of 40 Next.js interview questions and answers, mixing concepts from classic Next.js and newer features:

Next.js Interview Questions & Answers — Part 6

181. How do you create API routes that support multiple HTTP methods?

Answer:

In the API route handler, check `req.method` (e.g., GET, POST) and execute different logic accordingly.

182. What is the difference between static generation with and without fallback?

Answer:

Without fallback, only predefined paths are generated; with fallback, additional paths can be generated on-demand.

183. How does Next.js handle middleware ordering?

Answer:

Middleware runs in the order defined by directory structure and applies to all requests matching its scope.

184. How do you handle image placeholders for better UX?

Answer:

Use `placeholder="blur"` with `next/image` to show a blurry preview before full image loads.

185. How can you handle loading states globally in Next.js?

Answer:

Use a global state or React Context combined with `nprogress` or custom loading indicators in `_app.js` or layout components.

186. What is the difference between ISR and SSR regarding performance?

Answer:

ISR pre-renders pages and serves static files with occasional regeneration; SSR renders pages on every request, which is slower.

187. How do you handle route protection for static pages?

Answer:

Redirect client-side based on authentication state or use middleware with SSR pages.

188. How do you test Next.js API routes?

Answer:

Use Jest with mocks for `req` and `res` objects, testing the handler function logic.

189. How to use environment variables securely in Next.js?

Answer:

Keep secrets server-side (no `NEXT_PUBLIC_` prefix) and only expose safe variables client-side.

190. How does Next.js handle internationalized URLs?

Answer:

Locales are added as URL prefixes or domains as configured in `next.config.js`.

191. How do you perform incremental adoption of TypeScript in an existing Next.js project?

Answer:

Rename files to `.ts` / `.tsx` gradually and add type definitions progressively.

192. How to use middleware to rewrite URLs?

Answer:

Return a `NextResponse.rewrite()` in the middleware with the target URL.

193. How do you implement analytics that trigger on route change?

Answer:

Listen to `router.events` in `_app.js` and trigger analytics on `routeChangeComplete`.

194. How to do dynamic imports in Next.js?

Answer:

Use `next/dynamic` to load components asynchronously with optional loading fallback.

195. How to use the `Head` component for dynamic meta tags?

Answer:

Import `Head` from `next/head` and include `<title>`, `<meta>`, etc., inside pages or components.

196. How do you optimize bundle size in Next.js?

Answer:

Use dynamic imports, tree shaking, and analyze bundles with `next build --profile`.

197. How to add custom headers to HTTP responses?

Answer:

Use the `headers()` async function in `next.config.js` or set headers in API routes.

198. What is the difference between client and server components in terms of bundle size?

Answer:

Server Components don't add JS to client bundles, reducing overall size.

199. How do you set cache-control headers for static assets?

Answer:

Configure CDN or hosting platform settings or use custom server headers if applicable.

200. How does Next.js handle stale-while-revalidate caching strategy?

Answer:

Via ISR, it serves stale content while regenerating the page in the background.

201. How to handle localization of static content in Next.js?

Answer:

Use locale-based folders or translation libraries like `next-i18next`.

202. How to share data between API routes and pages?

Answer:

Fetch API route endpoints from pages or call shared backend logic directly in both.

203. How to protect API routes with middleware?

Answer:

Create reusable middleware functions that check auth tokens before continuing.

204. How do you implement a custom 500 error page?

Answer:

Create `pages/_error.js` or `pages/500.js` with custom UI.

205. What is the best way to handle forms in Next.js?

Answer:

Use controlled components with React hooks and handle submission with API routes or Server Actions.

206. How to prevent hydration mismatches?

Answer:

Ensure server and client render the same initial markup; avoid client-only data

during SSR.

207. How do you do testing of Next.js pages?

Answer:

Use Jest with React Testing Library to test component rendering and user interactions.

208. How does Next.js support CSS variables?

Answer:

Use global CSS or CSS modules to define and use CSS variables.

209. How to use Webpack 5 features in Next.js?

Answer:

Next.js 10+ supports Webpack 5 by default; configure in `next.config.js` if needed.

210. How do you handle large-scale image galleries?

Answer:

Lazy load images with `next/image` and paginate data fetching.

211. How to add a custom server with Express in Next.js?

Answer:

Use custom server setup in `server.js` importing Next and Express, then run both.

212. What are the tradeoffs of using a custom server?

Answer:

More flexibility but lose automatic static optimization and some Next.js features.

213. How to debug build errors?

Answer:

Check terminal logs, analyze error stack traces, and use VS Code debugger.

214. How does Next.js handle CSP (Content Security Policy)?

Answer:

Set CSP headers via `next.config.js` headers or middleware.

215. How to add Google Fonts with next/font?

Answer:

Import fonts from `next/font/google` and use them directly in components.

216. How do you handle revalidation on demand?

Answer:

Use the Next.js On-Demand ISR API to trigger regeneration via API routes.

217. How does Next.js handle serverless functions cold starts?

Answer:

Cold starts occur when functions aren't warmed up; mitigated by platform-specific techniques.

218. What is the difference between `next.config.js` and `.env` files?

Answer:

`next.config.js` configures build/runtime behavior; `.env` stores environment variables.

219. How to handle query params in static pages?

Answer:

Access client-side with `useRouter().query` ; static pages can't use them for SSR data fetching.

220. How to implement pagination with SSG?

Answer:

Generate paginated pages at build time using `getStaticPaths` with dynamic params.

If you'd like, I can keep going or focus on testing, deployment, performance, or newer Next.js 13+ concepts!

keep going

Absolutely! Here's the next set of 40 Next.js interview questions and answers:

Next.js Interview Questions & Answers — Part 7

221. How do you handle form validation in Next.js?

Answer:

Use client-side validation libraries like React Hook Form or Formik and validate on the server in API routes for security.

222. What is the role of `public` folder in Next.js?

Answer:

It stores static assets accessible from the root URL without processing or bundling.

223. How can you implement dark mode in Next.js?

Answer:

Use React state or context to toggle themes, persist preference in localStorage, and use CSS variables or Tailwind CSS classes.

224. How to use React Context API with Next.js?

Answer:

Wrap your app in a context provider inside `_app.js` and consume context in any component.

225. How do you monitor performance in Next.js?

Answer:

Use built-in React Profiler, browser devtools, Lighthouse, or analytics tools.

226. How to implement API versioning in Next.js?

Answer:

Create separate folders under `pages/api` like `/api/v1/` and `/api/v2/`.

227. What is the purpose of `next/link` prefetching?

Answer:

It fetches linked pages in advance for faster client navigation.

228. How to set a custom favicon in Next.js?

Answer:

Place the favicon in the `public` folder and add a `<link rel="icon" href="/favicon.ico" />` tag in `_document.js` or `_app.js`.

229. How to debug hydration issues in Next.js?

Answer:

Check for differences in server/client rendering output and avoid client-only logic during SSR.

230. How do you use React Query with Next.js?

Answer:

Wrap your app with `QueryClientProvider` and fetch data using hooks like `useQuery`.

231. What is ISR and when would you use it?

Answer:

Incremental Static Regeneration lets you update static pages after build without rebuilding the entire site, useful for frequently updated content.

232. How do you handle errors in `getStaticProps` ?

Answer:

Return `notFound: true` or `redirect` in case of errors or throw errors caught by error pages.

233. How to handle serverless function timeouts?

Answer:

Optimize function code to run quickly or use background jobs/external queues for long tasks.

234. How to implement API caching in Next.js?

Answer:

Use caching headers or external caches like Redis in API routes.

235. How to debug memory leaks in Next.js?

Answer:

Use Chrome DevTools Memory tab and monitor React component lifecycle for cleanup.

236. How to enable source maps in production?

Answer:

Configure `productionBrowserSourceMaps: true` in `next.config.js`.

237. How to secure environment variables in Next.js?

Answer:

Never expose secrets on client side; keep them server-only in `.env` files without `NEXT_PUBLIC_` prefix.

238. How does Next.js handle CSS global styles?

Answer:

Import global CSS only in `_app.js` to avoid duplication.

239. How do you implement middleware with conditions?

Answer:

Check `request.nextUrl.pathname` inside `middleware.js` to selectively apply logic.

240. How to implement multi-tenancy in Next.js?

Answer:

Use middleware and routing to detect tenant and load tenant-specific data or UI.

241. What is the difference between `getStaticProps` and `getServerSideProps`?

Answer:

`getStaticProps` runs at build time, generating static pages; `getServerSideProps` runs on every request for dynamic content.

242. How do you optimize SEO in Next.js?

Answer:

Use SSR or SSG, add proper meta tags with `next/head`, use semantic HTML, and optimize page load.

243. What is the recommended approach to handle authentication in Next.js?

Answer:

Use middleware and libraries like NextAuth.js or custom JWT token validation in API routes.

244. How do you set up TypeScript in a Next.js project?

Answer:

Add `tsconfig.json`, rename files to `.tsx / .ts`, and install necessary typings.

245. How can you serve different content based on device type?

Answer:

Detect user-agent in SSR and render different components or use CSS media queries client-side.

246. How do you manage forms with complex nested data?

Answer:

Use form libraries like React Hook Form with controlled inputs and validation schemas.

247. What is the use of `next/script` strategy?

Answer:

Controls when external scripts load to optimize performance, e.g., `lazyOnload`, `beforeInteractive`.

248. How to do dynamic routing with catch-all routes?

Answer:

Use `[...slug].js` in `pages` directory to catch multiple path segments.

249. How does Next.js support image formats like WebP and AVIF?

Answer:

`next/image` automatically serves modern formats if supported by the browser.

250. How to implement analytics event tracking on button clicks?

Answer:

Attach event handlers that send data to analytics services on interaction.

251. How to debug API route performance?

Answer:

Add logging with timestamps and use profiling tools on serverless platforms.

252. How to handle CSRF protection in Next.js API routes?

Answer:

Use tokens like CSRF or implement double-submit cookies in API route validations.

253. How to configure a custom webpack alias in Next.js?

Answer:

Modify `webpack` property in `next.config.js` to add aliases.

254. How to prevent exposing sensitive data in error responses?

Answer:

Customize error messages to avoid leaking internal details.

255. What is the difference between static and server components regarding hooks?

Answer:

Server Components don't support React hooks like `useState`, only Client Components do.

256. How to deploy Next.js on platforms other than Vercel?

Answer:

Use AWS Amplify, Netlify, Firebase Hosting, or custom servers with Docker.

257. How to optimize font loading in Next.js?

Answer:

Use `next/font` or preload fonts to reduce layout shifts.

258. How to do conditional rendering based on environment?

Answer:

Check `process.env.NODE_ENV` or custom env variables in components.

259. How to handle static generation of dynamic routes?

Answer:

Use `getStaticPaths` to specify dynamic routes at build time.

260. How to add global error handling in API routes?

Answer:

Create wrapper functions or middleware to catch and handle errors consistently.

That's 40 more! Want me to continue with the next batch?

continue with the next batch

Here's the next batch of 40 Next.js interview questions and answers for you:

Next.js Interview Questions & Answers — Part 8

261. How to implement optimistic UI updates in Next.js?

Answer:

Update the UI immediately on client-side after user actions and reconcile with server response asynchronously.

262. How does Next.js handle static generation with `fallback true`?

Answer:

Initially shows a fallback page while generating the page on-demand, then caches it for future requests.

263. How to use `next/router` for programmatic navigation?

Answer:

Use `const router = useRouter()` then call `router.push('/path')` or `router.replace()` to navigate.

264. How to share constants or utilities across pages and API routes?

Answer:

Place shared code in a common folder outside `pages` and import wherever needed.

265. How do you implement role-based access control (RBAC) in Next.js?

Answer:

Check user roles in middleware or API routes before allowing access to protected pages or endpoints.

266. How do you handle image optimization for external domains?

Answer:

Add the domain to the `images.domains` array in `next.config.js`.

267. How do you preload fonts in Next.js?

Answer:

Use `<link rel="preload" href="..." as="font" type="font/woff2" crossorigin="anonymous" />` in `_document.js`.

268. What are the benefits of dynamic imports in Next.js?

Answer:

Reduce initial bundle size and improve performance by loading components only when needed.

269. How to configure API route middlewares?

Answer:

Wrap API handlers with middleware functions to add features like authentication or logging.

270. How do you handle API request validation?

Answer:

Validate inputs using libraries like Joi or Yup in API routes before processing.

271. How do you customize the build output directory?

Answer:

Set `distDir` in `next.config.js` to change from default `.next`.

272. How do you optimize images for different device sizes?

Answer:

Use `next/image` with `sizes` and `layout="responsive"` to serve appropriate images.

273. How does Next.js handle incremental static regeneration cache invalidation?

Answer:

After the `revalidate` time expires, the next request triggers regeneration.

274. How to implement dark mode with Tailwind CSS in Next.js?

Answer:

Use Tailwind's `dark` mode class with a state toggle and persist user preference.

275. How do you enable TypeScript strict mode in Next.js?

Answer:

Set `"strict": true` in `tsconfig.json`.

276. How to add a custom HTTP header to all API responses?

Answer:

Set headers inside each API handler or create middleware to add headers globally.

277. How to prevent duplicate API calls on fast navigation?

Answer:

Use caching or debounce logic on client-side requests.

278. What are the limitations of Server Components in Next.js?

Answer:

Cannot use React hooks like `useState`, no browser APIs, and must be async-compatible.

279. How to handle real-time features with Next.js?

Answer:

Use WebSocket or libraries like Socket.io externally; Next.js API routes aren't persistent.

280. How to access query parameters in API routes?

Answer:

Use `req.query` in API route handlers.

281. How do you use environment variables in client-side code safely?

Answer:

Expose only variables prefixed with `NEXT_PUBLIC_`.

282. How to optimize fonts using `next/font`?

Answer:

Import fonts directly in components for automatic optimization and zero layout shifts.

283. How to implement logout functionality in Next.js?

Answer:

Clear cookies or tokens client-side and optionally invalidate sessions server-side.

284. How to use fallback `blocking` mode in ISR?

Answer:

Next.js blocks the request until the page is generated, then serves it without fallback UI.

285. How to implement caching for API routes?

Answer:

Set proper `Cache-Control` headers or use external caches.

286. How to do server-side redirects in Next.js?

Answer:

Return a `redirect` object from `getServerSideProps`.

287. How do you use React suspense with data fetching?

Answer:

Wrap components with `<Suspense>` and fetch data in Server Components or with libraries that support Suspense.

288. How to handle image loading errors?

Answer:

Use `onError` prop in `next/image` or fallback images.

289. How to add a custom 404 page in Next.js?

Answer:

Create `pages/404.js` with your custom UI.

290. How to implement incremental adoption of new Next.js features?

Answer:

Gradually migrate pages and components, using feature flags if necessary.

291. How to improve Lighthouse scores in Next.js?

Answer:

Optimize images, fonts, code splitting, and use SSR/SSG effectively.

292. How to pass data from server to client in Next.js?

Answer:

Use props returned from data fetching methods or fetch client-side.

293. How to handle multi-language routing?

Answer:

Configure `locale` in `next.config.js` and create locale-specific routes.

294. How do you debug runtime errors in production?

Answer:

Use error tracking tools like Sentry or LogRocket.

295. How to handle session expiration in Next.js?

Answer:

Use token expiry checks client-side and refresh or logout users accordingly.

296. How to do client-side navigation without page reload?

Answer:

Use `next/link` or `router.push`.

297. How to load third-party scripts efficiently?

Answer:

Use `next/script` with appropriate loading strategies.

298. How to handle API rate limiting?

Answer:

Implement rate limit logic in API routes or use external services.

299. How to share state between multiple tabs?

Answer:

Use `localStorage` events or `BroadcastChannel` API.

300. What are the benefits of using middleware in Next.js?

Answer:

Modify requests/responses globally, handle authentication, redirects, and rewrites efficiently.

That completes this batch! Ready for the next 40?

Ready for the next 40

Great! Here's the next batch of 40 Next.js interview questions and answers:

Next.js Interview Questions & Answers — Part 9

301. What is the role of the `_app.js` file in Next.js?

Answer:

It wraps all pages and is used to initialize pages, add global styles, or keep state across page transitions.

302. How to add global CSS styles in Next.js?

Answer:

Import global CSS files only in `pages/_app.js`.

303. How do you perform server-side redirects in `getServerSideProps`?

Answer:

Return an object with `redirect: { destination: '/path', permanent: false }`.

304. How to disable Next.js's automatic static optimization?

Answer:

Use `getServerSideProps` or `getInitialProps` to force SSR.

305. What is the difference between `getInitialProps` and `getServerSideProps`?

Answer:

`getInitialProps` runs on both server and client during client-side transitions;
`getServerSideProps` runs only on the server.

306. How to handle cookies in Next.js?

Answer:

Use `cookie` or `nookies` libraries to read/write cookies on client and server.

307. What is the use of `next/head` ?

Answer:

Allows modifying the `<head>` section for each page (title, meta tags, etc.).

308. How do you handle authentication redirects?

Answer:

Redirect unauthorized users to login pages using middleware or in `getServerSideProps` .

309. How to add meta tags for social sharing in Next.js?

Answer:

Add Open Graph and Twitter card meta tags inside `next/head` .

310. How does Next.js handle fallback rendering?

Answer:

With `fallback: true` or `blocking` , Next.js renders a fallback UI or waits for the page to be generated.

311. How do you implement API error handling?

Answer:

Use try-catch and send proper HTTP status codes with messages.

312. What are the benefits of using the `Image` component over standard `` ?

Answer:

Automatic optimization, lazy loading, and responsive image sizes.

313. How do you configure custom environment variables?

Answer:

Define variables in `.env.local`, `.env.development`, or `.env.production`.

314. How do you create custom Document in Next.js?

Answer:

Override `pages/_document.js` to customize HTML and body tags.

315. How to perform client-side navigation with query parameters?

Answer:

Use `router.push({ pathname, query })` or `<Link href={{ pathname, query }} />`.

316. What is ISR (Incremental Static Regeneration)?

Answer:

Allows static pages to be regenerated after build on demand or at intervals.

317. How does Next.js handle CSS Modules?

Answer:

Supports importing `.module.css` files scoped locally to components.

318. How do you implement internationalization in Next.js?

Answer:

Use built-in i18n routing or libraries like `next-i18next`.

319. How can you enable gzip compression in Next.js?

Answer:

Usually handled by the hosting platform; for custom servers, use middleware like `compression`.

320. How to analyze bundle size in Next.js?

Answer:

Run `next build && next analyze` or use `webpack-bundle-analyzer`.

321. How to implement API rate limiting?

Answer:

Use middleware or external services to limit request frequency.

322. How to test Next.js components?

Answer:

Use Jest and React Testing Library for unit and integration tests.

323. How do you implement SSR caching?

Answer:

Set cache headers in `getServerSideProps` or use CDN caching.

324. What is `getStaticPaths` used for?

Answer:

Generate dynamic routes at build time for static pages.

325. How to handle fallback pages for dynamic routes?

Answer:

Use `fallback: true` or `blocking` in `getStaticPaths`.

326. How to customize error pages in Next.js?

Answer:

Create `pages/404.js` and `pages/_error.js`.

327. How do you use React Context with Next.js?

Answer:

Wrap the app in a provider in `_app.js` and consume context in child components.

328. How to secure API routes in Next.js?

Answer:

Verify authentication tokens or sessions in API handlers or middleware.

329. How to set HTTP headers in Next.js?

Answer:

Use the `headers()` config in `next.config.js` or set headers in API routes.

330. How to optimize images with Next.js?

Answer:

Use `next/image` with proper `sizes` and `priority` props.

331. How to implement middleware redirects?

Answer:

Use `NextResponse.redirect()` in `middleware.js`.

332. What is the role of `next.config.js`?

Answer:

Configure build and runtime settings for Next.js projects.

333. How to deploy Next.js on Vercel?

Answer:

Connect your Git repo to Vercel and deploy with zero config.

334. How to handle dynamic API routes?

Answer:

Use file names with `[param]` in `pages/api`.

335. How to enable React Strict Mode?

Answer:

Set `reactStrictMode: true` in `next.config.js`.

336. How to implement form submission in Next.js?

Answer:

Handle form data client-side and submit to API routes or Server Actions.

337. How do you implement session management?

Answer:

Use cookies, JWTs, or libraries like NextAuth.js.

338. How to lazy load components?

Answer:

Use `next/dynamic` with optional loading components.

339. What is the difference between `router.push` and `router.replace`?

Answer:

`push` adds to history stack; `replace` replaces the current entry.

340. How to enable Hot Module Replacement (HMR)?

Answer:

Enabled by default in development mode.

That's 40 more questions! Want me to keep going?

keep going

Sure! Here's the next batch of 40 Next.js interview questions and answers:

Next.js Interview Questions & Answers — Part 10

341. How to configure a custom 404 page in Next.js?

Answer:

Create a `pages/404.js` file with your custom React component to render on not found errors.

342. How does Next.js support API routes?

Answer:

API routes live inside the `pages/api` folder and act as serverless functions.

343. How can you handle query strings in Next.js?

Answer:

Use `useRouter().query` inside components or `context.query` in data fetching methods.

344. What are dynamic routes in Next.js?

Answer:

Routes that include dynamic segments, created with bracket notation like `[id].js`.

345. How do you handle nested routes in Next.js?

Answer:

Create nested folders inside `pages` to reflect route hierarchy.

346. What is Static Site Generation (SSG) in Next.js?

Answer:

Pre-rendering pages at build time for faster load and SEO benefits.

347. What is Server-Side Rendering (SSR) in Next.js?

Answer:

Rendering pages on each request, allowing dynamic content.

348. How to pass data from `getStaticProps` to a page?

Answer:

Return data as props in an object: `{ props: { ... } }`.

349. How to fetch data on the client-side in Next.js?

Answer:

Use React hooks like `useEffect` or data fetching libraries.

350. How do you add custom fonts in Next.js?

Answer:

Import fonts in `_app.js` or use the `next/font` package.

351. How to handle API rate limiting in Next.js?

Answer:

Use middleware or external tools like Redis or third-party services.

352. How to configure redirects in Next.js?

Answer:

Use the `redirects` function in `next.config.js`.

353. What is the difference between `getStaticProps` and `getStaticPaths`?

Answer:

`getStaticProps` fetches data for a page; `getStaticPaths` defines dynamic routes to pre-render.

354. How to handle image optimization for external URLs?

Answer:

Add domains in `next.config.js` under `images.domains`.

355. How to handle loading states during route changes?

Answer:

Use `router.events` to listen for route change start and complete events.

356. How to implement dark mode in Next.js?

Answer:

Use CSS variables, context, and persist preferences with `localStorage`.

357. What is middleware in Next.js?

Answer:

Code that runs before requests, useful for redirects, auth, and more.

358. How to deploy Next.js apps?

Answer:

Deploy on platforms like Vercel, Netlify, AWS, or custom servers.

359. How to handle errors in Next.js?

Answer:

Create custom error pages or use error boundaries.

360. What is the role of `_document.js`?

Answer:

Customizes the HTML document structure.

361. How to add meta tags for SEO?

Answer:

Use `next/head` to include meta tags in pages.

362. How to handle authentication in Next.js?

Answer:

Use middleware, API routes, or libraries like NextAuth.js.

363. What is incremental static regeneration (ISR)?

Answer:

Allows static pages to be updated after build time.

364. How to configure TypeScript in Next.js?

Answer:

Add `tsconfig.json` and rename files to `.tsx`.

365. How to lazy load components?

Answer:

Use `next/dynamic` for code splitting.

366. What is the difference between `router.push` and `router.replace`?

Answer:

`push` adds a new entry; `replace` replaces current history.

367. How to handle internationalization?

Answer:

Use built-in i18n or libraries like `next-i18next`.

368. How to share state between components?

Answer:

Use React Context or external state libraries.

369. How to debug Next.js apps?

Answer:

Use browser devtools, React DevTools, and logging.

370. What is the `pages` directory for?

Answer:

Defines the app's routes and API endpoints.

371. How to prefetch pages?

Answer:

`next/link` prefetches linked pages automatically.

372. How to handle redirects on the client-side?

Answer:

Use `router.push` in client components.

373. How to optimize performance?

Answer:

Code splitting, image optimization, and SSR.

374. How to add custom scripts?

Answer:

Use `next/script` with load strategies.

375. How to configure custom webpack?

Answer:

Modify `next.config.js` with a webpack function.

376. How to protect routes?

Answer:

Use middleware or client-side auth checks.

377. How to use environment variables?

Answer:

Define in `.env` files with `NEXT_PUBLIC_` for client variables.

378. How to handle form submissions?

Answer:

Submit to API routes or Server Actions.

379. How to debug API routes?

Answer:

Use console logs and test with Postman.

380. What is the `public` folder?

Answer:

Stores static assets served at root URL.

That's another 40! Want to continue?

Want to continue

Here's the next batch of 40 Next.js interview questions and answers for you:

Next.js Interview Questions & Answers — Part 11

381. How do you handle API versioning in Next.js?

Answer:

Organize API routes in subfolders like `pages/api/v1` and `pages/api/v2`.

382. How to implement server-side caching in Next.js?

Answer:

Set appropriate HTTP cache headers in `getServerSideProps` or API routes.

383. What is the purpose of `getInitialProps` ?

Answer:

Fetch data before rendering pages, runs on server and client during navigation.

384. How to handle environment-specific configs?

Answer:

Use `.env.local`, `.env.development`, `.env.production` files.

385. How to use dynamic imports with SSR disabled?

Answer:

Use `dynamic(() => import('./Component'), { ssr: false })`.

386. How do you implement incremental static regeneration?

Answer:

Return `revalidate` property in `getStaticProps`.

387. How to use the `next/head` component?

Answer:

Wrap page components with `<Head>` to set meta tags dynamically.

388. How to handle custom error pages?

Answer:

Create `pages/_error.js` with custom UI.

389. How does Next.js handle image optimization for remote images?

Answer:

Configure domains in `next.config.js` to allow optimization.

390. How to pass query parameters programmatically?

Answer:

Use `router.push({ pathname, query })`.

391. How to disable automatic static optimization?

Answer:

Use `getServerSideProps` or `getInitialProps`.

392. How do you implement middleware redirects?

Answer:

Return `NextResponse.redirect()` in middleware.

393. How to handle form validation?

Answer:

Use libraries like React Hook Form or Formik.

394. How to use environment variables securely?

Answer:

Use server-side variables without `NEXT_PUBLIC_` prefix.

395. What is the use of `_document.js`?

Answer:

Customize the HTML document structure.

396. How to implement API route middleware?

Answer:

Wrap handlers with middleware functions.

397. How to lazy load images?

Answer:

Use the `next/image` component with `loading="lazy"`.

398. How to debug hydration errors?

Answer:

Ensure server and client render the same content.

399. How to configure redirects in `next.config.js` ?

Answer:

Export an async `redirects()` function with redirect rules.

400. How to set custom HTTP headers?

Answer:

Use `headers()` in `next.config.js` or set headers in API routes.

401. How to configure internationalized routing?

Answer:

Add `locales` config in `next.config.js` .

402. What is the difference between CSR and SSR?

Answer:

CSR renders content client-side; SSR renders on the server.

403. How do you add global styles?

Answer:

Import CSS files in `_app.js` .

404. How to handle static file serving?

Answer:

Place files in the `public` folder.

405. How to create dynamic routes?

Answer:

Use brackets in filenames like `[id].js`.

406. How to analyze bundle size?

Answer:

Use `next build` with bundle analyzer plugin.

407. How to add custom fonts?

Answer:

Use `next/font` or import CSS in `_app.js`.

408. How to handle redirects programmatically?

Answer:

Use `router.push` or return redirect in data fetching methods.

409. How to implement API request validation?

Answer:

Use libraries like Joi or Yup in API routes.

410. How to optimize SEO?

Answer:

Use SSR/SSG and proper meta tags.

411. How to deploy Next.js app on AWS?

Answer:

Use services like Amplify or Elastic Beanstalk.

412. How to do serverless function debugging?

Answer:

Use local emulators or logs.

413. What is ISR fallback blocking?

Answer:

Page renders server-side on first request and caches for next requests.

414. How to handle session management?

Answer:

Use JWTs, cookies, or NextAuth.js.

415. How to do client-side data fetching?

Answer:

Use `useEffect` or libraries like SWR.

416. How to enable React strict mode?

Answer:

Set `reactStrictMode: true` in `next.config.js`.

417. How to handle middleware in API routes?

Answer:

Use wrapper functions or libraries.

418. How to add custom server with Express?

Answer:

Create a custom server script importing Next and Express.

419. How to use dynamic routing with catch-all?

Answer:

Use `[...slug].js` files.

420. How to handle environment variables client-side?

Answer:

Expose with `NEXT_PUBLIC_` prefix.

Let me know if you want me to continue!

Want to continue

Sure! Here's the next batch of 40 Next.js interview questions and answers:

Next.js Interview Questions & Answers — Part 12

421. How do you enable image placeholders with blur effect in Next.js?

Answer:

Use the `placeholder="blur"` prop with the `next/image` component and provide a small base64-encoded image as the `blurDataURL`.

422. What is the difference between `fallback: true` and `fallback: blocking` in `getStaticPaths`?

Answer:

`fallback: true` shows a fallback page while the page is generated client-side, while `fallback: blocking` waits server-side for the page to be generated before serving.

423. How to handle cookies securely in Next.js API routes?

Answer:

Use HTTP-only, Secure, and SameSite cookie flags, and manage cookies via libraries like `cookie` or `nookies`.

424. How do you implement middleware to protect routes in Next.js 13+?

Answer:

Create a `middleware.js` file in the root or app directory and use conditional logic with `NextResponse` to redirect unauthorized requests.

425. How can you customize the loading indicator on route changes in Next.js?

Answer:

Use libraries like `nprogress` and hook into `router.events` to show/hide loading bars.

426. What are React Server Components (RSC) in Next.js?

Answer:

Components rendered on the server that do not include client-side JavaScript, improving performance by sending only HTML.

427. How to use the new `app` directory in Next.js 13?

Answer:

Place React Server Components and routes inside `app/`, use file-based routing with server and client components, and leverage layouts and templates.

428. How to enable client-side components in Next.js 13?

Answer:

Add `"use client";` directive at the top of the component file.

429. How to pass props from server components to client components in Next.js 13?

Answer:

Pass props normally like React props; client components receive serialized data.

430. How to handle error boundaries in Next.js 13?

Answer:

Use the `error.js` file inside `app/` routes to create error boundaries for layouts and pages.

431. What are React Suspense and its usage in Next.js?

Answer:

React Suspense allows components to wait for asynchronous data before rendering, improving UX for loading states.

432. How to use streaming SSR in Next.js 13?

Answer:

Next.js streams HTML progressively to the client when using the `app` directory and server components.

433. How to fetch data in server components?

Answer:

Use async functions and fetch data directly inside server components or via `getServerSideProps` alternatives.

434. How do middleware and API routes differ in Next.js?

Answer:

Middleware runs before the request and can rewrite/redirect; API routes handle request logic and return responses.

435. How to implement incremental adoption of the new Next.js 13 app router?

Answer:

Use both `pages/` and `app/` folders in the same project and migrate pages incrementally.

436. How do you debug server components?

Answer:

Use server logs, throw errors, and use debugging tools like VS Code's remote debugging.

437. What's the difference between `layout.js` and `page.js` in the `app` directory?

Answer:

`layout.js` defines UI shared across routes; `page.js` is a specific route's content.

438. How do you configure API routes inside the `app` directory?

Answer:

Use the new `route.js` files inside folders with HTTP method exports like GET, POST.

439. How to add metadata in Next.js 13?

Answer:

Use the new `metadata` export in `layout.js` or `page.js` files.

440. How to handle global state in Next.js 13 with server components?

Answer:

Use React Context or external state management libraries in client components.

441. How to handle dynamic routing in the `app` directory?

Answer:

Use square brackets `[param]` in folder or file names like `[slug]/page.js`.

442. How does Next.js optimize images in server components?

Answer:

You still use `next/image`, which optimizes images server-side.

443. How to use environment variables in server components?

Answer:

Access server-only environment variables without the `NEXT_PUBLIC_` prefix.

444. How to use TypeScript with the new app router?

Answer:

Use `.tsx` files for components and type your props and data fetching functions.

445. How to handle client-side navigation in Next.js 13 app directory?

Answer:

Use the `useRouter` hook from `next/navigation`.

446. What's the benefit of layouts in Next.js 13?

Answer:

Persist UI state and layouts across nested routes without unmounting.

447. How do you handle authentication with the new app router?

Answer:

Use middleware for route protection and client components for auth state.

448. How to implement streaming responses with suspense?

Answer:

Use React Suspense in server components; Next.js streams the rendered HTML.

449. How do you use the `notFound()` helper in Next.js 13?

Answer:

Call `notFound()` in server components to render the 404 page.

450. How to implement revalidation in the app directory?

Answer:

Use the `revalidate` export in server components or caching headers.

If you want, I can keep going with the latest Next.js 13+ features or circle back to older versions — just say!

Focus on specific Next.js domains like routing, performance, testing, API routes, or deployment?

Great idea! Narrowing down to specific domains will make the questions more targeted and practical.

Which domain(s) do you want to start with? Here are some options — feel free to pick one or more:

1. Routing (dynamic routes, nested routes, app directory, etc.)

2. Performance Optimization (image optimization, ISR, caching, lazy loading)
3. Testing (unit testing, integration testing, E2E, mocking Next.js features)
4. API Routes (building APIs, middleware, authentication, error handling)
5. Deployment (platforms, environment configs, CI/CD, serverless vs custom servers)

Or if you want, I can prepare smaller batches focused on each domain one by one. Which sounds best?

Routing (dynamic routes, nested routes, app directory, etc.)

Awesome! Let's dive deep into **Next.js Routing** with focused questions and detailed answers. I'll start with a batch of 40 questions covering everything from basics to advanced routing concepts including the app directory in Next.js 13+.

Next.js Routing Interview Questions & Answers (Dynamic Routes, Nested Routes, App Directory, etc.)

1. What is the file-system based routing in Next.js?

Answer:

Next.js uses the `pages` or `app` directory to map files/folders to routes automatically. Each file in `pages` becomes a route; folders represent nested routes.

2. How do you create dynamic routes in Next.js?

Answer:

Use square brackets around the filename or folder, e.g., `pages/post/[id].js` to match `/post/123`.

3. What are catch-all routes?

Answer:

Catch-all routes use `[...slug].js` to match multiple nested segments, e.g., `/blog/a/b/c`.

4. What is an optional catch-all route?

Answer:

Use `[...slug].js` to match zero or more segments, so `/blog`, `/blog/a`, `/blog/a/b` all match.

5. How do nested routes work in Next.js?

Answer:

Create folders inside `pages` or `app` to represent nested paths, e.g., `pages/dashboard/settings.js` → `/dashboard/settings`.

6. How does routing differ in the new Next.js 13 app directory?

Answer:

`app/` uses React Server Components and introduces `layout.js` and `page.js` files for nested layouts and routes.

7. What is the purpose of `layout.js` in the app directory?

Answer:

Defines UI shared across nested routes and persists across navigation without unmounting.

8. How do you create dynamic routes in the `app` directory?

Answer:

Use folders or files with `[param]`, e.g., `app/post/[id]/page.js`.

9. How do you create nested routes in the app directory?

Answer:

Create nested folders with their own `page.js` and `layout.js`, e.g., `app/dashboard/settings/page.js`.

10. How to access route parameters in the app directory server components?

Answer:

Use the `params` object passed as a prop to `page.js`.

11. How to perform client-side navigation in Next.js?

Answer:

Use `next/link` for declarative links or `useRouter()`'s `push()` method for programmatic navigation.

12. How does `next/link` improve performance?

Answer:

It prefetches the linked page's code in the background, enabling faster navigation.

13. How do you handle query parameters in Next.js routing?

Answer:

Use `useRouter().query` inside components or access `context.query` in data fetching methods.

14. How to implement shallow routing?

Answer:

Use `router.push(url, as, { shallow: true })` to change URL without running data fetching methods again.

15. How to handle dynamic API routes?

Answer:

Use `[param].js` inside `pages/api/` to create dynamic API endpoints.

16. How do catch-all API routes work?

Answer:

Use `[...slug].js` inside `pages/api` to match multiple route segments.

17. How to create a redirect route in Next.js?

Answer:

Use the `redirects()` function in `next.config.js` to define redirects.

18. What is the role of the `middleware.js` file in routing?

Answer:

It runs before requests, allowing rewrites, redirects, and authentication checks at the edge.

19. How to implement route protection using middleware?

Answer:

Check user authentication status in `middleware.js` and redirect unauthorized users.

20. How to do dynamic routing with catch-all routes in the app directory?

Answer:

Use `[...slug]` folder with a `page.js` file to catch nested dynamic paths.

21. How to handle route groups in Next.js 13?

Answer:

Use `(groupName)` folders that don't affect URL structure but organize files.

22. What are parallel routes in Next.js 13?

Answer:

Parallel routes render multiple routes side-by-side using named slots.

23. How to define a 404 page with the app directory?

Answer:

Create `not-found.js` inside a route folder to customize 404 behavior.

24. How to link between pages in the app directory?

Answer:

Use `next/link` or the new `useRouter()` hook from `next/navigation`.

25. What's the difference between the `pages` and `app` directories?

Answer:

`pages` uses traditional file-based routing with CSR/SSR/SSG; `app` uses React Server Components, layouts, and enhanced routing.

26. How to handle route transitions programmatically?

Answer:

Use `router.push()` or `router.replace()` from `next/router` or `next/navigation`.

27. How to add middleware for specific routes only?

Answer:

Add conditional logic inside `middleware.js` based on `request.nextUrl.pathname`.

28. How to implement catch-all fallback pages?

Answer:

Use `fallback` options in `getStaticPaths` or handle dynamic routes with `not-found.js` in the app directory.

29. How to use the `usePathname()` hook?

Answer:

It returns the current URL pathname, useful in app directory client components.

30. How to handle internationalized routing in Next.js?

Answer:

Configure the `locale` key in `next.config.js` for locale subpaths.

31. How to access the router object in server components?

Answer:

You cannot use client hooks; instead, access route params via `props` or `searchParams`.

32. How to navigate without full page reloads?

Answer:

Use Next.js's `Link` component or router methods that enable client-side navigation.

33. How to handle route-specific data fetching?

Answer:

Use `getStaticProps`, `getServerSideProps` in `pages`, or fetch data directly in server components in `app`.

34. How to use `notFound()` in app directory routing?

Answer:

Call `notFound()` inside server components to render the 404 page dynamically.

35. How to perform redirects in `getServerSideProps` ?

Answer:

Return a `redirect` object with `destination` and `permanent` flags.

36. How to customize routing with rewrites?

Answer:

Define rewrites in `next.config.js` to map URL paths to different destinations.

37. How to implement client-only routes?

Answer:

Use conditional rendering with `useEffect` or `next/dynamic` with `ssr: false`.

38. How do dynamic routes affect SEO?

Answer:

Properly pre-rendering or SSR'ing dynamic routes ensures SEO benefits.

39. How to add route-based code splitting?

Answer:

Next.js automatically code-splits by route; you can also use dynamic imports.

40. How to implement nested layouts with the app directory?

Answer:

Create nested `layout.js` files in folders to build hierarchical layouts.

Would you like me to continue with more routing questions or switch to performance or another domain next?

Performance Optimization (image optimization, ISR, caching, lazy loading)

Perfect! Here's a focused batch of **Next.js Performance Optimization** interview questions and answers covering image optimization, ISR, caching, lazy loading, and related topics.

Next.js Performance Optimization Interview Questions & Answers

1. What is Image Optimization in Next.js?

Answer:

Next.js optimizes images automatically using the `next/image` component, serving appropriately sized, compressed images with lazy loading and modern formats.

2. How does `next/image` improve performance?

Answer:

It supports lazy loading, automatic resizing, modern formats (WebP/AVIF), and caching to reduce page load times.

3. What is Incremental Static Regeneration (ISR)?

Answer:

ISR allows you to update static pages after build time by revalidating them at runtime without rebuilding the entire site.

4. How do you enable ISR in Next.js?

Answer:

Return a `revalidate` field (in seconds) from `getStaticProps` to specify how often to regenerate a page.

5. How does ISR benefit large static sites?

Answer:

It avoids full rebuilds by regenerating only changed pages, improving deployment speed and freshness.

6. What is caching in Next.js?

Answer:

Caching stores generated pages, API responses, or assets to speed up repeat requests and reduce server load.

7. How to control HTTP caching headers in Next.js?

Answer:

Set `Cache-Control` headers in API routes or use middleware to configure caching policies.

8. How does Next.js handle caching for static assets?

Answer:

Files in the `public` folder are served with long cache lifetimes by default.

9. What is lazy loading?

Answer:

Loading content (images, components) only when they enter the viewport to improve initial load time.

10. How to implement lazy loading for images?

Answer:

`next/image` enables lazy loading by default, loading images as they scroll into view.

11. How to lazy load React components in Next.js?

Answer:

Use `next/dynamic()` to import components dynamically with optional loading fallback.

12. What is code splitting?

Answer:

Splitting JavaScript bundles into smaller chunks that load on demand, reducing initial bundle size.

13. How does Next.js handle code splitting?

Answer:

It automatically splits code by route and supports dynamic imports for further splitting.

14. What is prefetching in Next.js?

Answer:

`next/link` prefetches linked page code and data in the background for faster client-side navigation.

15. How to optimize fonts in Next.js?

Answer:

Use the built-in `next/font` to load fonts with automatic optimization and avoid layout shifts.

16. How does Next.js prevent Cumulative Layout Shift (CLS)?

Answer:

By reserving space for images with `next/image` and optimizing font loading to avoid flash of unstyled text.

17. How to measure performance in Next.js?

Answer:

Use tools like Lighthouse, Web Vitals, and Next.js built-in analytics.

18. What is the difference between SSR and SSG in performance?

Answer:

SSG serves pre-rendered pages quickly from CDN, SSR renders on-demand per request, which can be slower.

19. How to optimize API route performance?

Answer:

Use caching, minimize data fetching, and avoid heavy computations in API handlers.

20. How to implement edge caching with Next.js middleware?

Answer:

Use headers and CDN configuration to cache responses at the edge.

21. What is the benefit of streaming SSR?

Answer:

Streams HTML progressively to the client, improving Time to First Byte (TTFB).

22. How to optimize images from external sources?

Answer:

Add external domains in `next.config.js` under `images.domains` for optimization.

23. How does ISR handle concurrent requests?

Answer:

The first request triggers regeneration; subsequent requests serve the stale page until new one is ready.

24. How to disable lazy loading for an image?

Answer:

Set `loading="eager"` on the `next/image` component.

25. How to configure priority image loading?

Answer:

Use `priority` prop on `next/image` for critical images to load immediately.

26. What is a fallback page in ISR?

Answer:

A placeholder shown during on-demand static generation for fallback `true` mode.

27. How to optimize JavaScript bundle size?

Answer:

Avoid large dependencies, use dynamic imports, and analyze bundles.

28. How to use Webpack Bundle Analyzer with Next.js?

Answer:

Add the plugin and run builds with analysis enabled to inspect bundle content.

29. How does Next.js optimize CSS?

Answer:

It supports CSS modules and removes unused CSS during build.

30. How to avoid hydration mismatches?

Answer:

Ensure server and client render the same content and delay client-only logic.

31. How to cache images on CDN?

Answer:

Configure CDN cache headers and use `next/image` for optimized delivery.

32. How to use HTTP/2 server push with Next.js?

Answer:

Not supported out of the box; requires custom server or CDN configuration.

33. How to handle fonts to improve performance?

Answer:

Use font-display swap, preload fonts, or `next/font` for automatic optimization.

34. How to optimize for First Contentful Paint (FCP)?

Answer:

Use SSG, optimize images and fonts, and minimize blocking scripts.

35. What is the impact of using third-party scripts?

Answer:

They can slow down pages; load them asynchronously or defer loading.

36. How to use React Profiler with Next.js?

Answer:

Wrap components with `<Profiler>` to measure rendering performance.

37. How to implement HTTP caching for API routes?

Answer:

Set proper `Cache-Control` headers based on data freshness.

38. How to handle large lists in Next.js?

Answer:

Use virtualization libraries like `react-window` for efficient rendering.

39. How to optimize images for retina displays?

Answer:

`next/image` automatically serves higher resolution images for high DPI screens.

40. How to implement CDN caching with Next.js?

Answer:

Deploy on platforms like Vercel or configure custom CDN with caching headers.

Would you like me to continue with more performance questions or move on to Testing or API Routes next?

Testing (unit testing, integration testing, E2E, mocking Next.js features)

Great! Here's a focused batch of **Next.js Testing** interview questions and answers covering unit testing, integration testing, end-to-end (E2E) testing, and mocking Next.js features.

Next.js Testing Interview Questions & Answers

1. What are the common testing types used in Next.js apps?

Answer:

Unit testing (components), integration testing (components + API), and end-to-end (E2E) testing (user flows).

2. Which testing frameworks are commonly used with Next.js?

Answer:

Jest for unit and integration testing, React Testing Library for component rendering, and Cypress or Playwright for E2E.

3. How do you test a React component in Next.js?

Answer:

Use Jest with React Testing Library to render the component and assert expected behavior.

4. How to test Next.js pages that use `getStaticProps`?

Answer:

Mock data fetching functions and test the page component by passing props manually.

5. How to mock `next/router` in Jest tests?

Answer:

Use Jest's `jest.mock('next/router', () => ({ useRouter: () => ({ ...mockedRouterProps }) })))`.

6. How do you test API routes in Next.js?

Answer:

Use `supertest` with Jest to make HTTP requests to API route handlers.

7. How do you test dynamic routes?

Answer:

Render the page component with mocked route parameters or use integration tests with routing simulation.

8. How to test client-side navigation?

Answer:

Use React Testing Library to simulate clicks on `<Link>` and assert route changes via mocked router.

9. How do you test Next.js middleware?

Answer:

Mock request and response objects and test middleware functions in isolation.

10. What is snapshot testing?

Answer:

Testing technique that saves rendered UI output and compares future renders to catch unexpected changes.

11. How to handle global setup for tests in Next.js?

Answer:

Configure Jest setup files to initialize mocks, environment variables, or polyfills.

12. How do you test React Server Components?

Answer:

Test them as regular React components but note they don't have client-side state or effects.

13. How to mock `next/head` in tests?

Answer:

Mock `next/head` as a simple component or verify rendered meta tags via DOM queries.

14. How do you test forms in Next.js?

Answer:

Simulate user input and submission with React Testing Library's `fireEvent` or `userEvent`.

15. How do you test `getServerSideProps` ?

Answer:

Call the function with mock context and assert returned props or redirects.

16. How do you test `getStaticProps` and `getStaticPaths` ?

Answer:

Invoke these functions directly with mocked params/context and check output.

17. How to mock fetch or Axios calls?

Answer:

Use Jest's `jest.mock()` or libraries like `msw` to intercept HTTP requests.

18. How to test components using `next/image` ?

Answer:

Mock `next/image` to render a simple `img` tag for easier testing.

19. How do you perform integration testing with Next.js and API routes?

Answer:

Use Jest with `supertest` to test API routes and React Testing Library to test pages consuming APIs.

20. What tools are used for E2E testing Next.js apps?

Answer:

Cypress, Playwright, and TestCafe are popular E2E tools.

21. How do you write an E2E test for a login page?

Answer:

Use E2E tools to simulate typing credentials, submitting the form, and asserting successful navigation.

22. How do you mock Next.js environment variables in tests?

Answer:

Set environment variables in test setup files or use Jest's `setupFiles`.

23. How do you test protected routes?

Answer:

Mock authentication state and verify redirects or access control.

24. How to test client-side data fetching with SWR?

Answer:

Mock fetch calls and test components rendering loading, error, and success states.

25. How to test page transitions?

Answer:

Simulate router navigation and verify UI changes or side effects.

26. How to handle async testing in Next.js?

Answer:

Use `async/await` and utilities like `waitFor` from React Testing Library.

27. How do you test error handling in API routes?

Answer:

Trigger error conditions and assert correct HTTP status codes and messages.

28. How to test static generation fallback?

Answer:

Simulate fallback states by rendering with partial props or mock routing.

29. How do you mock Next.js Link component?

Answer:

Use Jest mock or render as a simple anchor tag.

30. How to test middleware with edge functions?

Answer:

Test middleware logic separately and mock Next.js request/response objects.

31. How to test React context used in Next.js?

Answer:

Wrap components with the context provider in tests.

32. How do you test global CSS or style imports?

Answer:

Usually not tested directly; test styles with visual regression or snapshot tests.

33. How do you test with TypeScript in Next.js?

Answer:

Use `ts-jest` and type your tests to ensure type safety.

34. How to debug failing tests in Next.js?

Answer:

Use `console.log`, Jest's `--watch` mode, and browser devtools for E2E tests.

35. How to test React hooks in Next.js components?

Answer:

Use testing utilities like `@testing-library/react-hooks`.

36. How to handle API mocking in Jest tests?

Answer:

Use `msw` (Mock Service Worker) or Jest's manual mocks.

37. How to test Next.js's `getInitialProps`?

Answer:

Call the function manually with mock context and assert the output.

38. How to test pages using authentication?

Answer:

Mock authentication context or providers and assert conditional rendering.

39. How to test user interactions?

Answer:

Use React Testing Library's `userEvent` for realistic user events.

40. How to setup code coverage reporting for Next.js?

Answer:

Configure Jest with `collectCoverage` and use tools like `coveralls` or `codecov`.

Would you like to continue testing questions or move to API routes or deployment next?

API Routes (building APIs, middleware, authentication, error handling)

Got it! Here's a focused batch of **Next.js API Routes** interview questions and answers covering API building, middleware, authentication, error handling, and best practices.

Next.js API Routes Interview Questions & Answers

1. What are API routes in Next.js?

Answer:

API routes allow you to build backend endpoints as serverless functions inside the `pages/api` directory.

2. How do you create an API route?

Answer:

Create a JavaScript or TypeScript file inside `pages/api`. Export a default function that

handles `req` and `res` parameters.

3. How to handle different HTTP methods in API routes?

Answer:

Check `req.method` inside the handler and execute logic based on `'GET'`, `'POST'`, etc.

4. How to parse JSON body in Next.js API routes?

Answer:

By default, Next.js parses JSON bodies; you can access it via `req.body`.

5. Can you use middleware in Next.js API routes?

Answer:

Yes, you can write or reuse middleware functions that wrap the handler to process requests.

6. How to implement authentication in API routes?

Answer:

Check auth tokens or session cookies in the request headers or cookies before proceeding.

7. How to handle CORS in Next.js API routes?

Answer:

Manually set `Access-Control-Allow-Origin` headers or use CORS middleware.

8. How to handle errors in API routes?

Answer:

Use try-catch blocks and send proper status codes and error messages in the

response.

9. How to write async API handlers?

Answer:

Declare the handler as `async` and use `await` inside for asynchronous operations.

10. How to access query parameters in API routes?

Answer:

Use `req.query` to access URL query parameters.

11. How to send JSON responses?

Answer:

Use `res.status(code).json(data)` to send JSON with appropriate HTTP status.

12. How to handle form submissions with API routes?

Answer:

Send form data as JSON or form-encoded and parse it in the handler.

13. Can you connect databases in API routes?

Answer:

Yes, API routes can connect to databases and perform CRUD operations.

14. How to implement rate limiting in API routes?

Answer:

Use third-party libraries like `express-rate-limit` or custom logic with in-memory or Redis counters.

15. How to use environment variables in API routes?

Answer:

Access environment variables via `process.env`.

16. How to validate request data?

Answer:

Use validation libraries like Joi or Yup before processing requests.

17. How to structure complex API routes?

Answer:

Organize files into folders, and export handler functions from index files.

18. How to implement API versioning?

Answer:

Use folders like `pages/api/v1` and `pages/api/v2`.

19. How to handle file uploads?

Answer:

Use libraries like `multer` to parse multipart/form-data requests.

20. How to secure API routes?

Answer:

Check authentication, sanitize inputs, and avoid exposing sensitive info.

21. How to use middleware for common tasks?

Answer:

Write reusable functions to handle logging, auth, validation, etc.

22. Can API routes be serverless functions?

Answer:

Yes, deployed on platforms like Vercel as serverless functions.

23. How to test API routes?

Answer:

Use Jest with `supertest` to mock HTTP requests.

24. How to handle streaming responses?

Answer:

Use Node.js streams and set appropriate headers in the response.

25. How to cache API responses?

Answer:

Set HTTP cache headers or use in-memory or Redis caching.

26. How to handle multipart form data?

Answer:

Parse with middleware like `formidable` or `busboy`.

27. How to implement pagination in API routes?

Answer:

Accept page and limit query parameters and return slices of data.

28. How to handle rate limiting in API routes?

Answer:

Implement counters or use API gateways for throttling.

29. How to parse cookies in API routes?

Answer:

Use libraries like `cookie` or `nookies` to parse and set cookies.

30. How to send status codes correctly?

Answer:

Use `res.status(code)` before sending the response.

31. How to handle redirects in API routes?

Answer:

Use `res.writeHead(307, { Location: '/redirect-url' }).end()`.

32. How to configure body size limits?

Answer:

Set limits in middleware or use custom body parsers.

33. How to protect against CSRF attacks?

Answer:

Implement CSRF tokens or same-site cookies.

34. How to log requests in API routes?

Answer:

Use middleware or log inside the handler.

35. How to handle long-running requests?

Answer:

Use background jobs or asynchronous processing.

36. How to handle API route errors globally?

Answer:

Create error-handling middleware or wrapper functions.

37. How to implement OAuth in API routes?

Answer:

Handle OAuth flows and tokens validation inside API handlers.

38. How to parse XML or other non-JSON payloads?

Answer:

Use custom parsers or middleware.

39. How to handle API route throttling?

Answer:

Limit request frequency using rate limiting.

40. How to deploy API routes on serverless platforms?

Answer:

Deploy Next.js on Vercel or other providers supporting serverless functions.

Want me to continue with more API routes questions or focus on deployment next?

Deployment (platforms, environment configs, CI/CD, serverless vs custom servers)

Here's a focused batch of **Next.js Deployment** interview questions and answers covering platforms, environment configuration, CI/CD, serverless vs custom servers, and best practices.

Next.js Deployment Interview Questions & Answers

1. What are popular platforms to deploy Next.js apps?

Answer:

Vercel (official), Netlify, AWS Amplify, Heroku, DigitalOcean, and custom servers.

2. How does Vercel optimize Next.js deployment?

Answer:

Vercel provides built-in support for Next.js with zero-config deployment, serverless functions, ISR, and CDN caching.

3. What is serverless deployment?

Answer:

Deploying functions that scale automatically and run on demand without managing servers.

4. When would you use a custom server for Next.js?

Answer:

When you need custom backend logic, complex routing, or middleware not supported natively.

5. How to configure environment variables for Next.js in deployment?

Answer:

Use `.env.production` locally and set variables in the platform's dashboard or CLI.

6. How to handle secrets securely in deployment?

Answer:

Use environment variables, secret managers, and avoid committing secrets to source

control.

7. How to setup CI/CD pipelines for Next.js?

Answer:

Use GitHub Actions, GitLab CI, CircleCI, or platform-specific pipelines to automate build and deploy.

8. What commands are used to build and start Next.js app in production?

Answer:

`next build` to build, `next start` to run the production server.

9. How to deploy Next.js static export?

Answer:

Run `next export` and host the static files on any static hosting like Netlify or S3.

10. What is the difference between static export and SSR deployment?

Answer:

Static export generates HTML files; SSR deployment runs a Node.js server to render pages on demand.

11. How does Next.js handle incremental static regeneration in deployment?

Answer:

Platforms like Vercel regenerate pages on-demand and serve cached versions globally.

12. How to deploy Next.js API routes?

Answer:

API routes deploy as serverless functions on platforms like Vercel or run on custom servers.

13. How to optimize build size before deployment?

Answer:

Analyze bundles, remove unused dependencies, and use dynamic imports.

14. How to configure custom domains for Next.js apps?

Answer:

Add DNS records and configure the domain in the deployment platform.

15. What is a zero-downtime deployment?

Answer:

Deploying updates without interrupting users by serving old and new versions concurrently.

16. How to rollback deployments?

Answer:

Use platform-specific rollback features or deploy previous stable builds.

17. How to monitor Next.js apps post-deployment?

Answer:

Use monitoring tools like Vercel Analytics, Sentry, LogRocket, or Datadog.

18. How to handle environment-specific configurations?

Answer:

Use different `.env` files and platform environment variables for dev, staging, and

prod.

19. How to enable HTTPS for deployed apps?

Answer:

Use platform-provided SSL or configure certificates manually on custom servers.

20. What is the impact of serverless cold starts?

Answer:

Initial requests to serverless functions might be slower due to container startup.

21. How to optimize serverless cold starts?

Answer:

Keep functions small, avoid heavy dependencies, and use warming techniques.

22. How to deploy Next.js apps on AWS?

Answer:

Use Amplify, Lambda@Edge, or Elastic Beanstalk for deployment.

23. How to deploy Next.js on Docker?

Answer:

Create a Dockerfile, build the image, and run it on any container platform.

24. What are the differences between serverless and traditional server deployment?

Answer:

Serverless auto-scales and abstracts infrastructure; traditional servers require manual scaling and maintenance.

25. How to handle logs in production Next.js apps?

Answer:

Stream logs to monitoring services and use platform logging features.

26. How to implement blue-green deployments?

Answer:

Deploy new version alongside old one and switch traffic after validation.

27. How to configure CDN caching in deployment?

Answer:

Set cache headers and use platform or custom CDN for static assets.

28. How to deploy Next.js with Docker Compose?

Answer:

Define services in `docker-compose.yml` for the app and dependencies.

29. How to automate environment variable injection in CI/CD?

Answer:

Use secrets management and inject variables during build/deploy steps.

30. How to handle large media files in deployment?

Answer:

Host on dedicated storage/CDN services like AWS S3.

31. How to debug production issues?

Answer:

Use remote debugging, logs, and monitoring tools.

32. How to handle multiple environments in deployment?

Answer:

Create separate projects or branches with environment-specific configs.

33. How to ensure application security in deployment?

Answer:

Use HTTPS, sanitize inputs, limit permissions, and keep dependencies updated.

34. What is the role of `next.config.js` in deployment?

Answer:

Configure build, environment variables, rewrites, redirects, and optimizations.

35. How to use feature flags in Next.js deployment?

Answer:

Use environment variables or external feature flag services.

36. How to deploy Next.js apps with server-side rendering on Netlify?

Answer:

Use Netlify's Edge Functions or serverless functions, or fallback to static export.

37. What are the trade-offs between static and server-side deployments?

Answer:

Static is faster and cheaper but less dynamic; SSR is flexible but needs runtime resources.

38. How to deploy Next.js with GraphQL backend?

Answer:

Deploy Next.js frontend separately and connect to GraphQL backend via API calls.

39. How to deploy multi-region Next.js apps?

Answer:

Use CDNs with multi-region support and platforms like Vercel or AWS Global Accelerator.

40. How to configure build caching in CI/CD for Next.js?

Answer:

Cache `node_modules` and `.next/cache` to speed up builds.

Want me to continue with more deployment questions or return to any other domain?