

30 Frontend Interview Questions (Next.js, TypeScript, Tailwind)

Part 1: Basics (10 Questions)

Focus: Foundational knowledge of HTML, CSS, JS, and the tools in use.

1. What is the distinction between `interface` and `type` in TypeScript, and which should you use for React component props?

- **Answer:** Both can define the shape of an object. `interface` is extendable (via `extends`) and better for defining public API shapes. `type` is more flexible for unions (`|`) and intersections (`&`).
`structure`.
- **Best Practice:** Use `type` for Props for simple unions/compositions, or `interface` if you prefer object-oriented inheritance. In modern React, they are largely interchangeable for props, but consistency is key.

2. How does Tailwind CSS differ from traditional CSS frameworks like Bootstrap?

- **Answer:** Tailwind is a utilitarian framework. Instead of pre-built components (like a `.btn` class), it gives you low-level utility classes (like `px-4`, `bg-blue-500`) to build custom designs without leaving your HTML.
- **Benefit:** Reduces file size (via tree-shaking unused classes) and solves the problem of "fighting" against framework overrides.

3. What is the purpose of the `key` prop in React lists?

- **Answer:** The `key` prop gives elements a stable identity. React uses it to identify which items have changed, are added, or are removed.
- **Critical Note:** Never use `index` as a key if the list order can change, as it causes performance issues and state bugs. Use unique IDs instead.

4. Explain the "Box Model" in the context of Tailwind CSS.

- **Answer:** The Box Model consists of Content, Padding, Border, and Margin.
- **Tailwind Mapping:** `w/h` (Content), `p-{n}` (Padding), `border-{n}` (Border), `m-{n}` (Margin). Tailwind uses `box-border` (border-box) by default, meaning padding and border are included in the element's total width/height.

5. How do you access Environment Variables in a Next.js application on the client-side vs. server-side?

- **Answer:**
 - **Server-side:** `process.env.VARIABLE_NAME` (e.g., database secrets).
 - **Client-side:** Variables meant for the browser **must** be prefixed with `NEXT_PUBLIC_` (e.g., `NEXT_PUBLIC_API_URL`). Next.js inlines these at build time.

6. What is the difference between `null` and `undefined` in JavaScript/TypeScript?

- **Answer:** `undefined` means a variable has been declared but not assigned a value. `null` is an assignment value that represents "no value" or "intentional object absence."
- **TypeScript:** You often need to check for both, e.g., `if (value == null)` checks both.

7. What is the App Router file structure convention for defining a route in Next.js?

- **Answer:** Routes are defined by folders. A folder named `about` creates the `/about` route. The UI for that route must be defined in a `page.tsx` file within that folder (e.g., `app/about/page.tsx`).

8. How do you center a div horizontally and vertically using Tailwind CSS?

- **Answer:** The most common way is Flexbox:

```
<div className="flex justify-center items-center h-screen">
  {/* Content */}
</div>
```

9. What is the purpose of `.env.local` ?

- **Answer:** It stores local environment variables. It is typically added to `.gitignore` to prevent secrets (API keys, DB passwords) from being pushed to the code repository.

10. Explain Destructuring in JavaScript and give an example with React props.

- **Answer:** It allows unpacking values from arrays or properties from objects into distinct variables.
 - **Example:**

```
// Instead of props.title
const Component = ({ title, isActive }: Props) => { ... }
```

Part 2: Core Concepts (10 Questions)

Focus: Deep dive into React hooks, Next.js routing, and component architecture.

11. Explain the difference between Server Components and Client Components in Next.js 14+.

- **Answer:**
 - **Server Components (Default):** Render on the server, have access to backend resources (DB, FS), reduced bundle size, cannot use hooks (`useState`, `useEffect`).
 - **Client Components:** Render on the client (and pre-rendered on server), used for interactivity (clicks, state). Marked with 'use client' directive at the top of the file.

12. When would you use the `useEffect` hook? What does the dependency array do?

- **Answer:** `useEffect` is for handling side effects (data fetching, subscriptions, DOM manipulation).
 - **Dependency Array:**
 - `[]` : Runs only once on mount.
 - `[prop]` : Runs on mount and whenever `prop` changes.
 - *Missing array*: Runs on every render (bad for performance).

13. What is the role of `layout.tsx` in the Next.js App Router?

- **Answer:** It acts as a wrapper for all pages in its subtree. It preserves state, remains interactive, and does not re-render when navigating between sibling routes. Root `layout.tsx` is required to define `<html>` and `<body>`.

14. In TypeScript, what is a "Union Type" and how is it useful for Props?

- **Answer:** A union type allows a value to be one of several types.
 - **Example:** `status: 'loading' | 'success' | 'error'`. This is safer than making `status` a `string`, as it prevents invalid values.

15. How does Tailwind's `@apply` directive work, and why should you be cautious using it?

- **Answer:** `@apply` allows you to inline standard Tailwind utility classes into custom CSS classes (e.g., `.btn { @apply bg-blue-500; }`).
- **Caution:** Overusing it defeats the purpose of utility-first CSS, increases CSS bundle size, and reintroduces the maintenance burden of semantic naming.

16. How do you handle dynamic routes in Next.js (e.g., a product page for different IDs)?

- **Answer:** Create a folder with brackets, e.g., `app/products/[id]/page.tsx`.
- **Access:** The `id` is accessed via the `params` prop passed to the page component (server components) or `useParams` hook (client components).

17. What is "Prop Drilling" and how can you avoid it?

- **Answer:** Passing data through multiple layers of components that don't need it just to reach a deep child.
- **Solutions:** Use React Context API, a state management library (Zustand/Redux), or Component Composition (passing children).

18. Explain the concept of "Hydration" in Next.js.

- **Answer:** Hydration is the process where JavaScript attaches to the HTML generated by the server. The server sends static HTML (fast First Contentful Paint), and then React attaches event listeners to make it interactive. If HTML doesn't match the JS expectation, a "Hydration Mismatch" error occurs.

19. How do you implement a "Not Found" (404) page in App Router?

- **Answer:** Create a `not-found.tsx` file in the route segment. You can also programmatically trigger it by importing `notFound()` from `next/navigation` and calling it within a page or component logic.

20. Why might `console.log` inside a Server Component not appear in the browser dev tools?

- **Answer:** Server Components run entirely on the server. Their logs appear in the terminal where `npm run dev` is running, not the browser console.

Part 3: Intermediate / Practical (10 Questions)

Focus: Performance, Optimization, Best Practices, and Security.

21. How does the `next/image` component optimize performance compared to a standard `` tag?

- **Answer:**
 - **Automatic Resizing:** Serves correctly sized images for each device.
 - **Format Conversion:** Automatically serves modern web formats like WebP/AVIF.
 - **Lazy Loading:** Images are lazy-loaded by default (only loaded when entering the viewport).
 - **Prevent CLS:** Enforces width/height to prevent Cumulative Layout Shift.

22. You need to improve the SEO of a specific page. How do you set metadata in the App Router?

- **Answer:** Export a `metadata` object or a `generateMetadata` function from your `page.tsx`.

```
export const metadata: Metadata = {
  title: "Product Name",
  description: "Product details...",
};
```

23. How would you type a React Functional Component (`FC`) that accepts children components?

- **Answer:**

```
import { ReactNode } from 'react';
type Props = { children: ReactNode };
export default function Layout({ children }: Props) { ... }
```

- **Note:** Avoid `React.FC` in modern React; typing props directly is preferred.

24. What are TypeScript Generics, and give a practical Frontend example.

- **Answer:** Generics allow creating reusable components/functions that work with a variety of types.
 - **Example:** A reusable API response type:

```
type ApiResponse<T> = { data: T; status: number };
```

25. How can you optimize a search input to prevent API calls on every keystroke?

- **Answer: Debouncing.** Implement a delay (e.g., 300ms) so the API call only fires after the user stops typing.
 - **Implementation:** Use a `setTimeout` inside a `useEffect` that clears on cleanup, or a library like `lodash.debounce`.

26. Explain "Route Groups" in Next.js (folders defined with parenthesis `(folder)`).

- **Answer:** Route groups allow you to organize code logically without affecting the URL structure.
 - **Use Case:** `(auth)/login` and `(auth)/register` both resolve to `/login` and `/register`, but they can share a specific `layout.tsx` inside the `(auth)` folder that defines a layout unique to authentication pages.

27. What is the difference between `build` time and `runtime` data fetching in Next.js?

- **Answer:**
 - **Build time (Static):** Data fetched via `fetch` (with default caching) generates HTML during `next build`. Good for blogs/marketing pages.
 - **Runtime (Dynamic):** Data fetched with `cache: 'no-store'` or dynamic functions generates HTML on every request. Good for dashboards/user data.

28. How do you prevent CSS bloat when using Tailwind in a production build?

- **Answer:** PostCSS and Tailwind's compiler scan your content files (configured in `tailwind.config.ts`) and strip out any class names that are not actually valid or used in the code (Tree Shaking/Purging).

29. What is a "Race Condition" in data fetching, and how might you solve it in a `useEffect`?

- **Answer:** If a user changes a filter rapidly, an earlier request might finish *after* a later request, showing old data.
 - **Fix:** Use an `active` boolean flag in `useEffect` cleanup to ignore results from stale effects, or use a library like TanStack Query which handles cancellation automatically.

30. Why is `dangerouslySetInnerHTML` named that way, and when is it safe to use?

- **Answer:** It reminds developers that setting HTML directly opens up Cross-Site Scripting (XSS) attacks.
 - **Usage:** Only use it when you trust the source completely (e.g., a static CMS) or if the content has been sanitized using a library like DOMPurify.