

React 19 & Next.js App Router: RSC, Streaming & Caching Cheat Sheet

React Server Components wire format, Suspense streaming boundaries, Server Actions, and CSP nonce propagation.

EXECUTIVE SUMMARY

Architectural guide to React 19 and Next.js App Router: RSC serialization rules, selective hydration with ``, optimistic UI updates, and Core Web Vitals (LCP/INP/CLS) tuning.

KEY EQUATIONS & INVARIANTS

* Core Web Vitals Good Thresholds:

LaTeX: $\text{LCP} \leq 2.5\text{s}$, $\text{INP} \leq 200\text{ms}$, $\text{CLS} \leq 0.10$

Note: Evaluated at the 75th percentile (p75) of real-user field loads across mobile and desktop.

1. Server Components vs. Client Components Decision Matrix

Server Components execute exclusively on the server and contribute **0 KB** to the client JavaScript bundle. Pass Server Components as `children` props into Client Components (the Donut Pattern) to avoid pulling server-only trees into the client bundle.

Capability | Server Component (Default) | Client Component ('use client')

Direct DB / Secret / Filesystem Access | Yes (Safe) | No (Leaks to bundle)
 Client Bundle Size Impact | 0 KB JS | Included in client chunks
 React Hooks (useState, useEffect) | No | Yes
 Browser APIs (window, EventSource) | No | Yes
 Streaming via `<Suspense>` | Yes (Progressive HTML + Flight) | Renders fallback until chunk resolves

[TSX] The 'Donut Pattern': Composing Server Data Inside Interactive Client Shells

```
// ClientSidebarShell.tsx ('use client')
export function ClientSidebarShell({ children }: { children: React.ReactNode }) {
  const [open, setOpen] = useState(true);
  return <aside data-open={open}>{children}</aside>;
}

// page.tsx (Server Component)
export default async function CoursePage() {
  return (
    <ClientSidebarShell>
      <Suspense fallback={<CurriculumSkeleton />}>
        <ServerCurriculumTree /> { /* Stays 100% on server! */ }
      </Suspense>
    </ClientSidebarShell>
  );
}
```

COMMON PRODUCTION & INTERVIEW PITFALLS

- Importing a Server Component directly inside a `use client` file silently converts that component and its imports into Client Components.
- Awaiting slow sequential `fetch()` calls at the top of `page.tsx` instead of `Promise.all()` or `` blocks TTFB and LCP.

FREQUENTLY ASKED QUESTIONS

Q: Does 'use client' mean the component is skipped during Server-Side Rendering (SSR)?

A: No! Client Components are still pre-rendered to initial HTML on the server during SSR for fast First Contentful Paint, then hydrated on the client.