In the **App Router**, the paradigm shifts from "The whole page is CSR" to **"The page is Server-First by default."**

In the Pages Router, you decided the rendering strategy at the **page level**. In the App Router, you decide it at the **component level**.

---

1. The 'use client' Directive

In the App Router, every file is a **Server Component** by default.[1] To use Client-Side Rendering, you must explicitly add the 'use client' directive at the top of the file.

Crucial Interview Point: 'use client' does **not** mean the component only renders on the client. It means the component is part of the "Client Bundle" and will be **hydrated**. It still gets pre-rendered into HTML on the server first.

---

2. Code Example: CSR in App Router

Here is how you perform CSR data fetching in the App Router.

```
// app/profile/page.tsx
'use client'; // This marks the component for the client bundle

import { useState, useEffect } from 'react';

export default function ProfilePage() {
  const [data, setData] = useState(null);

  useEffect(() => {
    // This fetch runs in the browser after the initial shell is loaded
    fetch('/api/user')
      .then(res => res.json())
      .then(setData);
  }, []);

  if (!data) return <p>Loading...</p>;

  return <div><h1>{data.name}</h1></div>;
}
```

---

3. How it differs from the Pages Router

A. The "Leaf Component" Strategy

In the Pages Router, if you needed a useState, the whole page became a CSR page.
In the App Router, you are encouraged to move 'use client' as far down the component tree as possible (to the "leaves").2

- **Layout (Server):** Fetches the heavy data.
- **Navbar (Server):** Static links.
- **Search Input (Client):** Only this small component has 'use client' because it needs an onChange listener.

B. The RSC Payload

When you use a Client Component in the App Router, the server sends:

1. **HTML:** For the initial fast paint.
2. **RSC Payload:** A special condensed format that describes the component tree and tells React how to "slot in" the Client Components.

---

4. Key Differences: App Router vs. Pages Router CSR

Feature	Pages Router CSR	App Router CSR
Default State	CSR/Static	**Server Component (RSC)**
Granularity	Page-level	**Component-level**
Hydration	Hydrates the whole page	Hydrates only "Client" sub-trees
Bundle Size	Includes JS for the whole page	Includes JS **only** for 'use client' parts
Data Fetching	useEffect or useSWR	useEffect OR Server Actions

---

5. Interview "Gotchas" for the App Router

If you are asked about App Router CSR in an interview, mention these two things to sound like an expert:

1. **Interleaving:** You can nest a Client Component inside a Server Component, but you cannot import a Server Component into a Client Component. You must pass the Server Component as children or props.
2. **Static Prerendering:** Even with 'use client', Next.js will try to prerender that component to static HTML during the build unless you use dynamic(() => ..., { ssr: false }) or it depends on dynamic headers.

Summary of CSR Evolution

- **Plain React:** Browser does everything. Server sends an empty div.
- **Pages Router:** Server sends a static HTML shell; Browser "takes over" the whole page.
- **App Router:** Server sends a static HTML shell; Browser "takes over" only the specific interactive components you marked.