In the **App Router**, Static Site Generation (SSG) is no longer a "mode" you opt into with special functions. Instead, **Static Rendering is the default.** If you do not use "Dynamic Functions" (like cookies or headers) and you do not explicitly opt out of caching, Next.js will automatically generate your page as static HTML during the build process.

---

1. How SSG Works in App Router

Next.js evaluates your code at build time. If it sees a standard fetch or data access that doesn't depend on the specific user request, it pre-renders the result.

The Mechanism: generateStaticParams

This replaces the old getStaticPaths. It is used for dynamic routes (like [id]) to tell Next.js which IDs to bake into HTML files ahead of time.

Code Implementation

```
// app/products/[id]/page.tsx

// 1. Define which IDs to pre-render at build time
export async function generateStaticParams() {
  const products = await fetch('https://api.example.com/products').then(res => res.json());

  return products.map((product) => ({
    id: product.id,
  }));
}

// 2. Fetch data (Next.js caches this by default in App Router)
export default async function ProductPage({ params }: { params: { id: string } }) {
  const res = await fetch(`https://api.example.com/products/${params.id}`);
  const product = await res.json();

  return (
    <main>
      <h1>{product.name}</h1>
      <p>{product.description}</p>
    </main>
  );
}
```

---

2. Hydration in App Router SSG

Hydration in the App Router is significantly different and more efficient than the Pages Router because of **React Server Components (RSC)**.

1. **Static HTML Delivery:** The CDN sends the pre-rendered HTML.
2. **No Hydration for Server Components:** Because the ProductPage above is a Server Component, it **never hydrates**. React doesn't need to "wake up" the text or the layout. The HTML is sent, and that's it—no JS for that component is sent to the browser.
3. **Selective Hydration:** If you add a "Buy Now" button component with 'use client', **only that button** will hydrate.
4. **RSC Payload:** Next.js also sends a small "blueprint" file (the RSC payload). If the user navigates between pages, React uses this payload to update the UI without doing a full page refresh.

---

3. Comparing App Router SSG vs. Pages Router SSG

Feature	Pages Router (getStaticProps)	App Router (RSC)
Default Setting	Manual (must export function)	**Automatic / Default**
Hydration	Full Page (Entire tree)	**Selective** (Only Client parts)
Bundle Size	Includes JS for the whole page	**Zero JS** for Server Components
Data Fetching	Limited to page-level	Can happen in **any** component
Dynamic Routes	getStaticPaths	generateStaticParams

---

4. Interview Strategy: The "Full Route Cache"

In a System Design interview, mention that the App Router uses a **Full Route Cache**.

- **The Concept:** Next.js doesn't just cache the data; it caches the rendered HTML and the RSC payload on the server/CDN.
- **The Benefit:** This eliminates the need for the server to "think" at all. It just serves a file.
- **Invalidation:** You can break this cache (making it dynamic) by using revalidatePath or revalidateTag.

---

5. Interview Q&A: App Router SSG

Q: If everything is static by default, how do I make a page dynamic (SSR)?

A: You can use fetch(url, { cache: 'no-store' }), or use dynamic functions like cookies() or headers(). If Next.js detects these, it automatically switches that specific route from SSG to SSR.

Q: What is "Static Export"?

A: If your entire app uses SSG, you can run next export (or set output: 'export'). This produces a folder of pure HTML/CSS/JS that can be hosted on any provider (like GitHub Pages or an S3 bucket) without needing a Node.js server.

Q: Does generateStaticParams run in the browser?

A: No. It only runs at build time. If you use the "revalidate" feature (ISR), it might run on the server in the background, but never on the client.

Q: How does the App Router handle many pages?

A: Use dynamicParams = true. This tells Next.js: "Pre-render these specific IDs at build time, but if a user requests an ID I didn't pre-render, generate it on-demand (SSR) and then cache it (ISR)."
