Series 2: Data Fetching & Rendering

> **Goal**: Master how Next.js handles data and rendering — a *highly asked area* in interviews!

🔥 Must-Know Topics (Frequent in Interviews)

- `getStaticProps` (SSG)
- `getServerSideProps` (SSR)
- `getStaticPaths` (Dynamic Routes)
- **CSR vs SSR vs SSG** comparison
- **ISR** (Incremental Static Regeneration)
- **When to use what** in real-world projects
- **Client-side data fetching**

---

1. What is `getStaticProps` in Next.js? (Static Site Generation)

✅ **Interview Answer:**
`getStaticProps` is used to fetch data at build time. It's ideal for pages that don't change often (e.g. blog homepage, pricing page). The HTML is pre-generated during build and served fast via CDN.

🧪 **Code Example:**

```
export async function getStaticProps() {
  const res = await fetch('https://api.example.com/posts');
  const posts = await res.json();
  return { props: { posts } };
}
```

🧠 **Analogy:**
Imagine printing magazines in advance and delivering them. Everyone gets the same copy — fast and ready.

---

2. What is `getServerSideProps` in Next.js? (Server-Side Rendering)

✅ **Interview Answer:**
`getServerSideProps` fetches data on each request. It's great for dynamic content that updates frequently — like dashboards or user-specific pages.

🧪 **Code Example:**

```
export async function getServerSideProps(context) {
  const res = await fetch(`https://api.example.com/user/${context.params.id}`);
  const user = await res.json();
  return { props: { user } };
}
```

🧠 **Analogy:**
This is like cooking a fresh meal every time someone orders — always hot and current, but slower.

---

3. What is `getStaticPaths` and when is it used?

✅ **Interview Answer:**
`getStaticPaths` is used with `getStaticProps` for dynamic routes (e.g., `[id].js`). It tells Next.js which paths to pre-render at build time.

🧪 **Code Example:**

```
export async function getStaticPaths() {
  const res = await fetch('https://api.example.com/posts');
  const posts = await res.json();
  const paths = posts.map(post => ({ params: { id: post.id.toString() } }));
  return { paths, fallback: false };
}
```

🧠 **Analogy:**
You pre-print personalized birthday cards (paths) for known friends. Unknown ones? You don't make them yet.

---

4. What is Incremental Static Regeneration (ISR)?

✅ **Interview Answer:**
ISR lets you **update static content after build** using `revalidate`. It blends the speed of SSG with the freshness of SSR — a huge Next.js feature.

🧪 **Code Example:**

```
export async function getStaticProps() {
  const data = await fetchData();
  return {
    props: { data },
    revalidate: 60, // Rebuild page every 60 seconds
  };
}
```

🧠 **Analogy:**
Like auto-refreshing a printed flyer every hour. New info gets printed and delivered, but not for every single visitor.

---

5. What's the difference between CSR, SSR, SSG, and ISR?

✅ **Interview Answer:**

Rendering Type	When It Happens	Good For	SEO-Friendly
CSR	On client after load	Dashboards, Auth pages	❌ No
SSR	On every request	Dynamic, user-based content	✅ Yes
SSG	At build time	Blog, docs, marketing pages	✅ Yes
ISR	After build (cached)	Semi-frequent updates	✅ Yes

🧠 **Analogy Summary:**

- **CSR:** Cook your own food
- **SSR:** Freshly cooked each time
- **SSG:** Pre-packed meal
- **ISR:** Pre-packed but refreshed hourly

---

6. How does client-side fetching work in Next.js?

✅ **Interview Answer:**
Client-side fetching happens inside `useEffect()` using tools like `fetch`, **React Query**, or **SWR**. It's used when data is user-specific or doesn't affect SEO.

🧪 **Code Example:**

```
import { useEffect, useState } from 'react';

export default function Profile() {
  const [user, setUser] = useState(null);
  useEffect(() => {
    fetch('/api/user').then(res => res.json()).then(setUser);
  }, []);
  return <p>{user?.name}</p>;
}
```

🧠 **Analogy:**
Like ordering coffee after sitting down — slower, but tailored to you.

---

7. When to use SSG vs SSR vs CSR?

✅ **Interview Answer:**

- **SSG:** For public, stable content (blogs, docs)
- **SSR:** For personalized or real-time content (profile, dashboard)
- **CSR:** For non-SEO-critical pages (admin panels, auth)

📌 **Follow-up Interview Tip:** Be ready to *justify your choice* with an example (e.g., “For a product page with frequently changing prices, I’d choose SSR”).
