In a Frontend System Design interview, **Static Site Generation (SSG)** is the benchmark for speed. If a page can be static, it should be static.

Here is the deep-dive theoretical breakdown of SSG, how it handles hydration, and how it compares to the patterns we've already discussed.

What is Static Site Generation (SSG)?

Definition: SSG is a rendering pattern where the HTML for every page is generated **exactly once** at **Build Time** (when you run `npm run build`). These static files are then stored on a **CDN** (Content Delivery Network). When a user requests a page, the CDN serves the pre-made file instantly.

---

1. The Theoretical Concept: Build Time vs. Request Time

The most important distinction in SSG is **when** the rendering happens.

- **CSR & SSR:** Happen at **Request Time**. The work starts when the user clicks a link.
- **SSG:** Happens at **Build Time**. The work is finished before the user even thinks about visiting your site.

The Analogy:

- **SSR** is like a restaurant where the chef starts cooking only after you order.
- **SSG** is like a buffet or a vending machine. The food is prepared in advance, packaged, and waiting for you. You pick it up and eat it instantly.

When you "build" your application, the generator (like Next.js) fetches all the data from your CMS or Database and generates a unique, physical HTML file for every single route. These files are then pushed to a **CDN (Content Delivery Network)**.

---

2. Hydration in SSG

Hydration in SSG works exactly like it does in SSR, with one major advantage: the "server-side" part of the work is already done.

1. **Fastest Delivery:** Because the file is a static HTML document sitting on a CDN "at the edge" (physically close to the user), the browser receives the HTML almost instantly.
2. **Immediate Paint:** The user sees the full content of the page immediately.
3. **The "Dry" Period:** Just like SSR, the buttons don't work yet. The browser is still downloading the JavaScript bundle.
4. **The "Soak":** Once the JS arrives, React performs hydration. It walks through the static HTML and attaches event listeners.

Why is it better than SSR hydration? In SSR, the user has to wait for the server to fetch data before they even see the HTML. In SSG, the HTML is already there. The user can start reading while the "interactivity" (the JS) is still loading in the background.

---

3. Detailed Comparison: CSR vs. SSR vs. SSG

Feature	CSR	SSR	SSG
When is HTML made?	In the browser (Client)	On the server per request	**At Build Time**
Where is it hosted?	Simple File Server/CDN	**Running Node.js Server**	Simple File Server/CDN
Data Freshness	Real-time	Real-time	**Stale** (until next build)
TTFB (First Byte)	Fast (Empty HTML)	Slow (Server is busy)	**Fastest** (Static file)
SEO	Hardest	Excellent	**Excellent**
Scaling	Easy (CDN)	Hard (Server CPU load)	**Easiest** (CDN)

---

4. When to Use What? (Decision Framework)

In an interview, use this hierarchy:

1. **Can this page be static?** (e.g., Blog, Docs, About Us, Marketing Page).
   - **Yes:** Use **SSG**. It is the cheapest, fastest, and most secure.
2. **Is the data dynamic but not user-specific?** (e.g., E-commerce product list, News site).
   - **Yes:** Use **SSG + ISR** (Incremental Static Regeneration).
3. **Is the data highly personalized or real-time?** (e.g., Social Media Feed, Stock Tickers).
   - **Yes:** Use **SSR**.
4. **Is it a private tool with zero SEO requirement?** (e.g., Admin Panel, SaaS Dashboard).
   - **Yes:** Use **CSR**.

---

5. Critical Interview Q&A: SSG

Q1: What is the "Build Time" Bottleneck?

Answer: If you have an e-commerce site with 100,000 products, generating 100,000 HTML files during your build process will take hours. This slows down development and deployment.

- **The Fix:** Use **ISR (Incremental Static Regeneration)**. You build the top 1,000 popular products at build time and generate the other 99,000 "on-demand" as users request them.

Q2: How do you handle "Stale Data" in SSG?

Answer: Since SSG files are generated at build time, if a price changes in the database, the static HTML won't show it.

- **The Fix:** You can either trigger a **manual webhook** to rebuild the site when data changes, or use a **revalidation timer** (ISR) so the site checks for updates every X minutes.

Q3: Is SSG secure?

Answer: Yes, it's actually the *most* secure. Since there is no database connection or server-side code running when a user visits an SSG page, there is no "surface area" for many common attacks (like SQL injection) on the frontend delivery side.

Q4: How does SSG impact the "Largest Contentful Paint" (LCP)?

Answer: SSG usually results in the best LCP scores. Because the main image and text are already in the HTML file served by a CDN, the browser doesn't have to wait for a database query or a heavy JS execution to find and display the largest element on the screen.