Static Site Generation(SSG)

Understanding SSG

Static Site Generation (SSG) is a method of pre-rendering web pages at build time. This means that HTML pages are generated during the build process, not at runtime, and are then served to users as static files. This results in faster load times since the HTML is ready to be delivered without waiting for the server to process a request or run any code.

Benefits of SSG:

- **Performance:** Since pages are pre-rendered and served as static files, they load quickly.
- **SEO:** Pre-rendered content is more easily indexed by search engines, improving SEO.
- **Scalability:** Static files can be served from a CDN, making it easy to handle high traffic.

Building Static Sites with Next.js

Next.js simplifies the process of building static sites with SSG through the use of the getStaticProps and getStaticPaths functions:

- **getStaticProps:** This function allows you to fetch data at build time and pass it as props to your page component. It runs during the build process and can fetch data from APIs, databases, or the file system.

javascript Copy code

```javascript
export async function getStaticProps() {
  // Fetch data from an API
  const res = await fetch('https://api.example.com/data');
  const data = await res.json();

  return {
    props: {
      data,
    },
  };
}
```

getStaticPaths: When you have dynamic routes (e.g., /posts/[id]), this function lets you specify which paths should be pre-rendered. It's often used alongside getStaticProps to pre-render dynamic pages based on data fetched during build time.

javascript Copy code

```javascript
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 };
}
```

Incremental Static Regeneration (ISR)

Incremental Static Regeneration (ISR) allows you to update existing static pages by re-rendering them in the background as traffic comes in. This is powerful for pages that need to show the latest data without requiring a full rebuild of the entire site.

How ISR Works:

- With ISR, you can specify a revalidate interval in seconds for pages generated with getStaticProps. After the interval passes, the next request to that page will trigger a regeneration of the static content.
- During the regeneration process, the user receives the existing static content. Once the new content is ready, it's served to subsequent visitors.

Advantages of ISR:

- **Real-time updates:** Allows static pages to stay up-to-date without a full site rebuild.
- **Performance:** Retains the benefits of static pages while allowing dynamic content updates.

Use Cases for ISR:

- Blog posts that get occasional updates.
- Product pages where stock or pricing may change.
- News articles that might receive minor updates after publication.

These tools and strategies make Next.js a powerful framework for building modern, performance-optimized web applications with both static and dynamic content.