**Best SEO Practices for Next.js Apps**

[[](https://medium.com/@rajreetesh7?source=post_page-----6cad24c0fb0e--------------------------------)](https://medium.com/@rajreetesh7?source=post_page-----6cad24c0fb0e--------------------------------)

[Reetesh Kumar](https://medium.com/@rajreetesh7?source=post_page-----6cad24c0fb0e--------------------------------)

·

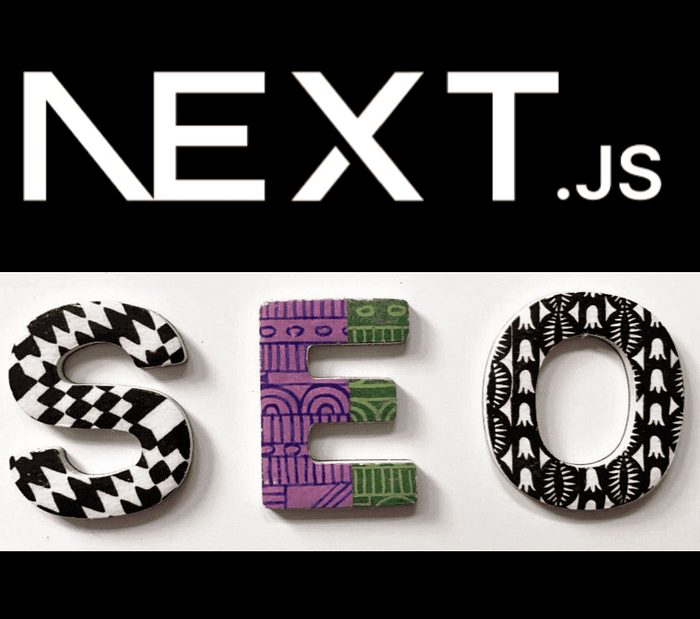
Follow

5 min read

·

Jan 9

73



Next.js and SEO

[Next.js](https://nextjs.org/) is a [React](https://react.dev/) framework that enables server-side rendering and generating static websites. It is a great framework for building SEO-friendly websites.

SEO is an important aspect of any website. It helps your website rank higher in search engines and get more traffic. In this article, we will discuss some best practices for Next.js SEO.

**What is SEO?**

SEO stands for Search Engine Optimization. It is the process of improving your website’s visibility in search engines like Google, Bing, and Yahoo. SEO helps you rank higher in search results and get more traffic to your website.

SEO is a complex process that requires a lot of time and effort. It is not something you can do overnight, but it is worth the effort if you want to get more traffic to your website.

Next.js has some built-in features that make it easier to build SEO-friendly websites. In this article, we will discuss some best practices for Next.js SEO.

**1) Robots.txt**

The first thing you should do is create a robots.txt file in the root directory of your website. This file tells search engines which pages they should crawl and which pages they should not crawl.

You can use the following code to create a robots.txt file:

// app/robots.ts (App Router)  
   
import { MetadataRoute } from "next";  
   
const robots: MetadataRoute.Robots = () => {  
 return {  
 rules: {  
 userAgent: "\*",  
 allow: "/",  
 disallow: "/private/", // Disallow private routes  
 },  
 sitemap: "https://example.com/sitemap.xml",  
 };  
};  
   
export default robots;

**2) Sitemap.xml**

The next thing you should do is create a sitemap.xml file in the root directory of your website. This file tells search engines which pages they should crawl and which pages they should not crawl.

Next.js has a built-in feature that generates a sitemap.xml file for you. You can use the following code to generate a sitemap.xml file:

// app/sitemap.ts (App Router)  
   
import { allPosts } from "/example.ts";  
import { MetadataRoute } from "next";  
   
export default async function sitemap(): Promise<MetadataRoute.Sitemap> {  
 const blogs = allPosts;  
 const URL = "https://example.com";  
   
 const restUrls = [  
 {  
 url: `${URL}/`, // Home Page  
 lastModified: new Date(),  
 priority: 1.0,  
 changeFrequency: "weekly",  
 },  
 {  
 url: `${URL}/blogs`, // Blogs Page  
 lastModified: new Date(),  
 priority: 0.8,  
 changeFrequency: "daily",  
 },  
 {  
 url: `${URL}/projects`, // Projects Page  
 lastModified: new Date(),  
 priority: 0.8,  
 changeFrequency: "weekly",  
 },  
 {  
 url: `${URL}/skills`, // Skills Page  
 lastModified: new Date(),  
 priority: 0.8,  
 changeFrequency: "weekly",  
 },  
 ];  
   
 const allBlogs = blogs.map(({ slugAsParams, date }) => {  
 return {  
 url: `${URL}/blog/${slugAsParams}`,  
 lastModified: new Date(date),  
 priority: 0.64,  
 changeFrequency: "daily",  
 };  
 });  
   
 return [...restUrls, ...allBlogs];  
}

Here, we are generating a sitemap.xml. We are adding or static pages and since we have allPosts we are make a dynamic route for each blog post.

We can even use the generateSitemaps function to generate dynamic routes for each blog post.

// app/sitemap.ts (App Router)  
   
export async function generateSitemaps() {  
 // the number of blog posts you have  
 return [{ id: "one" }, { id: "two" }, { id: "three" }, { id: "four" }];  
}  
   
export default async function sitemap({  
 id,  
}: {  
 id: string;  
}): Promise<MetadataRoute.Sitemap> {  
 // we can now fetch the blog from our DB using id.  
}

**3) Meta Tags**

Meta tags are HTML tags that provide metadata about your website. They are used by search engines to determine what your website is about and how it should be ranked in search results.

Next.js has a built-in feature that generates meta tags for you. You can use the following code to generate meta tags:

// app/layout.tsx (App Router)  
// app/page.tsx (App Router)  
   
import { Metadata } from "next";  
   
export async function generateMetadata(): Promise<Metadata> {  
 return {  
 metadataBase: new URL("https://example.com"),  
 title:"Name,  
 description: "description",  
 authors: [  
 {  
 name: "Site Name",  
 url: "https://example.com",  
 },  
 ],  
 twitter: {  
 card: "summary\_large\_image",  
 creator: "@example",  
 images: "some-image",  
 },  
 robots: "index, follow",  
 alternates: {  
 canonical: `https://example.com`,  
 languages: {  
 "en-US": "/",  
 },  
 },  
 openGraph: {  
 type: "website",  
 url: `https://example.com`,  
 title: "name",  
 description: "description",  
 siteName: "Site Name",  
 images: [  
 {  
 url: "some-image",  
 },  
 ],  
 },  
 assets: "some-image",  
 keywords: [  
 "keywords"  
 ],  
 };  
}

We can use generateMetadata function in our app/layout.tsx or app/page.tsx files. Since in Next.js routing is file based we can use this function in our app/layout.tsx file and it will be used in all our pages.

If we want a separate meta tag for each page we can use the same generateMetadata function in different pages.

You can create a utility function where to make it easier to generate meta tags. and just pass the parameters to the function. and call the function in your app/layout.tsx or app/page.tsx files.

// app/layout.tsx (App Router)  
// app/page.tsx (App Router)  
   
export async function generateMetadata() {  
 //commonMetaData is a utility function  
 const metaData = commonMetaData({  
 name: "",  
 desc: "",  
 image: "https:example.com/images/hero.png",  
 url: "/",  
 keywords: [],  
 });  
 return {  
 ...metaData,  
 };  
}

**4) Optimize Images and Fonts**

Next.js has built in Image component which is used to optimize images. I highly advises to use Image component instead of img tag.

Next.js next/font give us a way to optimize fonts. it will automatically optimize your fonts (including custom fonts) and remove external network requests for improved privacy and performance.

// app/layout.tsx (App Router)  
   
import { Roboto } from "next/font/google";  
   
const roboto = Roboto({  
 weight: "400",  
 subsets: ["latin"],  
 display: "swap",  
});  
   
export default function RootLayout({  
 children,  
}: {  
 children: React.ReactNode;  
}) {  
 return (  
 <html lang="en" className={roboto.className}>  
 <body>{children}</body>  
 </html>  
 );  
}

Images and fonts are important for SEO. They help search engines understand what your website is about and how it should be ranked in search results.

It also help to improve your website’s performance by reducing the number of network requests. and google also consider website performance as a ranking factor.

**5) Optimizing third-party Scripts**

Next.js has a built in Script component which is used to optimize third-party scripts.

The third-party script is fetched when the routes that use it are visited. This means that the script is only fetched when it is needed, which reduces the number of network requests and improves performance.

// app/layout.tsx (App Router)  
   
import Script from "next/script";  
   
const RootLayout = ({ children }: { children: React.ReactNode }) => {  
 return (  
 <>  
 {children}  
 <Script src="https://example.com/script.js" />  
 </>  
 );  
};  
   
export default RootLayout;

**Conclusion**

Next.js is a great framework for building SEO-friendly websites. It has some built-in features that make it easier to build SEO-friendly websites.

We have discussed some best practices for Next.js SEO in this article. I hope you found this article helpful. If you have any questions or suggestions, please leave them in the comments below.

[Nextjs](https://medium.com/tag/nextjs?source=post_page-----6cad24c0fb0e---------------nextjs-----------------)

[SEO](https://medium.com/tag/seo?source=post_page-----6cad24c0fb0e---------------seo-----------------)

[React](https://medium.com/tag/react?source=post_page-----6cad24c0fb0e---------------react-----------------)

[JavaScript](https://medium.com/tag/javascript?source=post_page-----6cad24c0fb0e---------------javascript-----------------)

[Web Development](https://medium.com/tag/web-development?source=post_page-----6cad24c0fb0e---------------web_development-----------------)