


[사용자] Next.js SEO(검색 엔진 최적화) 적용하기, 색인 차단 오류

≡ 구현 어려움 정도	☆☆
☑ 구현 완료	☑
📅 기간	@December 28, 2024
☑ 정리 완료	☑

 이전 글: [\[사용자\] Next.js 15에서 metadata 설정하기](#)

검색엔진에 잘 노출되고 상단에 위치한 웹 사이트가 많은 트래픽을 확보할 수 있습니다. 이는 회사 매출과 연관이 있어 기업에서는 꼭 필요한 기본적인 설정이라고 생각하여 진행 중인 프로젝트에 적용해보려 합니다.

SEO란?

SEO(검색 엔진 최적화, Search Engine Optimization)는 웹사이트의 검색 엔진 결과 페이지(SERP)에서 높은 순위를 차지하도록 웹사이트를 최적화하는 프로세스입니다.

즉, 웹사이트를 네이버, 구글과 같은 검색 엔진의 동작에 적합하게 맞추어 관련 정보를 검색할 때 웹사이트 노출 시, 상위에 위치하게 하는 작업입니다.

검색엔진최적화하는 방법에 대해 알아보도록 하겠습니다.

1. meta 태그 사용하기

메타 태그를 사용해 최적화를 하는 방법은 SEO 설정을 검색하면 가장 많이 나오는 방법 중 하나입니다.

메타 태그는 웹페이지가 담고 있는 콘텐츠가 아닌 웹 페이지 자체의 정보를 명시하기 위한 목적으로 사용되는 HTML 태그를 의미합니다.

이러한 웹 페이지의 메타 정보로는 주로 HTML의 meta 요소를 통해 마크업하며, HTML 문서 내에 `<head>` 요소 하위에 배치합니다.

때문에 유저가 보게되는 웹페이지의 콘텐츠에는 아무런 영향을 주지 않지만, 검색엔진과 같은 기계들이 웹페이지를 읽어야 할 때는 메타 태그의 내용들이 해당 서비스에서 어떻게 표시될지를 결정하는 핵심적인 요소가 됩니다.

[사용자] Next.js 15에서 metadata 설정하기

정적, 동적 메타데이터를 위에서 정리(링크)한 것 처럼 설정하거나 next.js의 `<Head>` 태그를 통해 설정 가능합니다.

```
// head tag 사용 예시
import Head from "next/head";

interface SEOProps {
  title: string;
}

const SEO = (props: SEOProps) => {
  <Head>
    <title>{title} | Coin Market</title>
    <meta
      name="description"
      content="Coin Market에 오신 것을 환영합니다."
    />
    ...
  </Head>
}
```

```
</Head>
}
```

아래에서 SEO 메타 태그의 종류에 대해 알아보겠습니다!

1) SEO 태그 - Title

```
<title>{title} | Coin Market</title>
```

- title 요소는 가장 대표적인 SEO 태그로 웹사이트의 제목을 명시합니다.
- title 태그를 사용할 때는 제목이 너무 길어지지 않도록 유의해야 합니다.
 - title이 너무 길어진다면 가독성이 떨어지므로, 영문 기준으로 40자, 한글 기준으로 20자가 넘지 않는 것을 권장합니다.

2) SEO 태그 - meta

```
<meta
  name="description"
  content="Coin Market에서 코인 시세를 확인해보세요."
/>
```

- 웹 사이트의 title 요소를 제외하고 다른 meta 정보는 모두 메타 태그를 사용해 표시합니다.
- 메타 태그를 사용할 때는 name 속성을 통해 메타 정보의 이름을 명시하고, content 속성을 통해 메타 정보의 내용을 표시합니다.
- description은 title에서 담지 못했던 상세한 내용들을 담을 수 있습니다.
 - description은 영문 160자, 한글 80자 이내로 작성하는 것을 권장합니다.

3) OG(Open Graph) 태그

OG(Open Graph) 태그는 웹사이트가 소셜 미디어에 공유될 때, 미리보기에 사용되는 메타 데이터를 정의하기 위한 태그입니다.

지인과 메시저로 어떤 URL을 공유 했을 때, 해당 웹사이트의 콘텐츠 미리보기를 카드 형태로 나타내는 것을 본 적이 있을 것입니다.

이렇게 SNS나 메시저를 통해 웹사이트의 링크를 공유하는 일이 잦아지면서 메타 태그가 좀 더 넓은 분야에서 활용되고 있는데 이러한 서비스들은 Open Graph 프로토콜이라고 불리는 업계 표준을 따라 웹사이트 콘텐츠 미리보기를 지원하고 있습니다.

따라서 Open Graph 프로토콜에서 정의하고 있는 메타 태그를 적절히 사용하면, 소셜 미디어에서 공유되는 콘텐츠의 가시성을 높일 수 있고 더 많은 트래픽을 유치할 수 있습니다.

- OG 태그는 `<head>` 태그 안에 정의되며, property 속성을 사용해 메타 데이터의 이름을 지정합니다.
- OG 태그의 이름은 `og:` 로 시작합니다.

(1) 웹사이트의 유형을 나타내는 og:type

```
<meta property="og:type" content="website" />
```

(2) 웹사이트의 제목을 나타내는 og:title

```
<meta property="og:title" content={` ${title} | Coin Market`} />
```

(3) 웹페이지의 상세 설명을 나타내는 og:description

```
<meta property="og:description" content="Coin Market에서 코인 시세를 확인해보세요." />
```

(4) 웹페이지의 주소를 나타는 og:url

```
<meta property="og:url" content="https://... url" />
```

(5) 웹페이지의 언어 선택을 나타내는 og:locale

```
<meta property="og:locale" content="ko_KR" />
```

(6) 웹페이지 썸네일(thumbnail) 이미지 주소를 나타내는 og:image (권장 크기: 1200 x 630)

```
<meta property="og:url" content="https://raw.githubusercontent.com/wwowww/crypto/9630be17f62e7ebf89d447f09a
```

메타 태그 설정하기

저의 경우, 정적 메타 데이터, 동적 메타데이터를 통해 title과 description을 설정해주었으므로, og 태그만 추가로 next.js의 <Head> 태그를 사용하는 방법도 있지만, [공식문서](#)를 참고해 `import { Metadata } from 'next';` 를 사용해 적용하도록 하겠습니다.

저의 경우 상세페이지만 동적 메타데이터를 적용했고, 나머지 페이지는 정적 메타데이터입니다.

정적 메타 데이터 적용 (OG 추가)

```
import { Metadata } from 'next';

export const metadata: Metadata = {
  title: "가상자산 플랫폼 | Coin Market",
  description: "Coin Market에서 코인 시세를 확인해보세요.",
  openGraph: {
    type: "website",
    url: "https://example.com",
    title: "가상자산 플랫폼 | Coin Market",
    description: "Coin Market에서 코인 시세를 확인해보세요.",
    siteName: "Coin Market",
    images: ['https://raw.githubusercontent.com/wwowww/crypto/b2e6f9c9ac0308d62f6b3c22a427347d5391e388/publ
  ],
};
```

동적 메타 데이터 적용 (OG 추가)

```
// 상세페이지 동적 메타데이터에 og 추가 적용
import { ReactNode } from 'react';
import { Metadata } from 'next';
import axios from 'axios';

type LayoutProps = {
  children: ReactNode;
  params: Promise<{ id: string }>;
};

export async function generateMetadata({ params }: LayoutProps): Promise<Metadata> {
```

```

const { id } = await params;

const coinListAll = await axios.get(`${process.env.NEXT_PUBLIC_BASE_URL}/markets?vs_currency=KWD&order=ma
const product = coinListAll.data.find((item: any) => item.symbol === id);

if (!product) {
  return {
    ...
  };
}

return {
  title: `${product?.name} | ₩ ${product?.total_volume.toLocaleString('ko-KR')}` || 'Coin Market',
  description: `${product?.name}은 현재 ₩ ${product?.total_volume.toLocaleString('ko-KR')}입니다. Coin Mark
  openGraph: {
    type: "website",
    url: "https://example.com",
    title: `${product?.name} | ₩ ${product?.total_volume.toLocaleString('ko-KR')}` || 'Coin Market을 통해
    description: `${product?.name}은 현재 ₩ ${product?.total_volume.toLocaleString('ko-KR')}입니다. Coin Ma
    siteName: "Coin Market",
    images: [product?.image || 'https://raw.githubusercontent.com/wwowww/crypto/9630be17f62e7ebf89d447f09
  },
};
}

export default function Layout({ children }: LayoutProps) {
  return (
    <div className="py-10 w-[1200px] mx-auto">
      {children}
    </div>
  );
}

```

2. 정적 사이트 생성 (Static Site Generation, SSG)

Next.js는 정적사이트 생성(SSG)을 통해 SEO 최적화에 유리한 정적 HTML을 생성할 수 있습니다.

`getStaticProps` 이나, `getServerSideProps` 를 사용해 페이지가 빌드 시점에 미리 렌더링되도록 하면 SEO에 좋습니다.

- `getStaticProps`를 사용해 빌드 시점에 데이터를 미리 가져와 HTML을 렌더링합니다. 이를 통 해 페이지 로드가 빠르고 검색 엔진이 페이지를 잘 인덱싱 할 수 있습니다.

```

// GetStaticProps 예시

import { GetStaticProps } from 'next';

export const getStaticProps: GetStaticProps = async () => {
  const res = await fetch('https://api.example.com/posts');
  const posts = await res.json();

  return {
    props: {
      posts,
    },
  };
};

```

```
export default function BlogPage({ posts }) {
  return (
    <div>
      <h1>블로그 포스트</h1>
      <ul>
        {posts.map((post) => (
          <li key={post.id}>{post.title}</li>
        ))}
      </ul>
    </div>
  );
}
```

3. 웹 접근성 (Accessibility) 지키기

검색엔진은 웹사이트의 구조적 요소를 바르게 이해할 수 있어야 하는데 그 과정에서 웹접근성을 지키는 것이 중요합니다.

- **Semantic HTML 태그:** `header`, `footer`, `article`, `section`, `main`, `nav` 등의 의미 있는 HTML 요소를 사용하여 페이지의 구조를 명확하게 정의
- **Alt 텍스트:** 이미지에 `alt` 속성을 추가하여 이미지가 무엇을 나타내는지 설명
- **ARIA 태그:** 시각적/청각적 장애를 가진 사용자들을 위한 **ARIA** 속성을 사용하여 접근성을 높이기

예전에 웹접근성 QA 업무를 진행하며, 웹 접근성에 대해 정리한 링크입니다. 링크를 참고해주세요. → [🌐 웹 접근성 정리 링크 참고하기](#)

4. 동적 라우팅과 301 리디렉션

- SEO 최적화를 위해 불필요한 URL 리디렉션을 피하고 깨진 링크가 없도록 해야합니다.
- 특정 페이지가 URL 변경이나 이동이 필요할 때, 301 리디렉션을 설정하여 검색엔진이 새 URL을 인식하도록 합니다.

next.config.js 파일에서 리디렉션을 설정할 수 있습니다.

```
module.exports = {
  async redirects() {
    return [
      {
        source: '/old-page',
        destination: '/new-page',
        permanent: true, // 301 리디렉션
      },
    ];
  },
};
```

5. 사이트 맵 (Sitemap) 사용

사이트 맵(Sitemap)은 검색 엔진에게 웹사이트의 페이지들을 알려주는 파일입니다.

사이트 맵을 작성하면 검색 엔진이 웹사이트를 더 잘 크롤링하고 인덱싱할 수 있습니다.

Next.js에서 사이트 맵을 자동으로 생성하려면 `next-sitemap` 같은 패키지를 사용할 수 있습니다.

1. 설치

```
pnpm add next-sitemap
```

2. next-sitemap.js 파일 설정하기

```
module.exports = {
  siteUrl: 'https://www.example.com',
  generateRobotsTxt: true, // robots.txt 파일 생성
};
```

3. 배포 후 사이트 맵 생성: 빌드 후 자동으로 사이트 맵을 생성합니다.

그런데 설정하고 Lighthouse를 실행해보니 아래와 같은 오류가 떴습니다.

▲ 페이지의 색인 생성이 차단됨

▲ robots.txt가 유효하지 않음 — 오류 17개 발견

오류가 발생한 이유는 robots 메타태그에서 noindex가 감지되어 구글에서 색인 요청을 거부한 것이라고 합니다.

robots 메타 태그는 검색 엔진 로봇을 제어하는 태그로 이 속성의 값을 변경함으로써 특정 검색 엔진에 나의 글 혹은 웹사이트 전체가 크롤링되는 것을 막거나 허용할 수 있습니다.

기본적으로 index, follow 값을 가지므로 따로 지정할 필요가 없고 all이 유효하게 대체가 가능하다고 합니다.

여기서 index는 페이지가 색인되어 검색 결과에 표시될 수 있다는 것이고 follow는 페이지를 포함해 링크된 페이지를 검색할 수 있음을 뜻합니다.

해결 방법

1. 사용하는 metadata에 아래 코드를 넣어줍니다.

```
robots: {
  index: true,
  googleBot: {
    index: true,
  },
},
```

```
// layout.tsx
import type { Metadata } from "next";

export const metadata: Metadata = {
  title: "가상자산 플랫폼 | Coin Market",
  description: "Coin Market에서 코인 시세를 확인해보세요.",
  openGraph: {
    type: "website",
    ...
  },
  robots: {
    index: true,
    googleBot: {
      index: true,
    },
  },
};
```

```
export default function RootLayout({
  children,
}: Readonly<{
  children: React.ReactNode;
}>) {
  return (
    <html lang="en">
      ...
    </html>
  );
}
```

2. `next-sitemap.js` 파일을 만들어 아래 코드를 넣어줍니다.

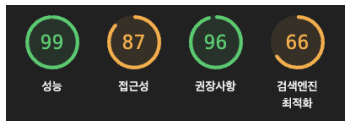
```
module.exports = {
  siteUrl: process.env.SITE_URL || '내 사이트 url',
  generateRobotsTxt: true,
};
```

3. `next.config.ts`에서 `headers`를 넣어줍니다. 이 때, `value: 'index, follow'` 값을 추가하여 색인을 허용해줍니다.

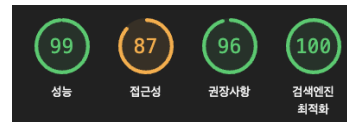
```
const nextConfig = {
  webpack: (config: any, { dev }: { dev: boolean }) => {
    ...
  },
  experimental: {
    ...
  },
  env: {
    ...
  },
  reactStrictMode: false,
  swcMinify: true,
  async headers() {
    return [
      {
        source: '/(.*)',
        headers: [
          {
            key: 'x-robots-tag',
            value: 'index, follow',
          },
        ],
      },
    ],
  },
};

export default nextConfig;
```

- 서버 헤더에서 `x-robots-tag` 확인: 이를 `index, follow` 로 수정
- HTML 메타 태그 수정: `noindex` 를 `index` 로 변경
- `robots.txt` 파일 수정: 크롤링을 차단하지 않도록 설정



검색엔진 최적화 전



검색엔진 최적화 후

참고

google analytics noindex issue nextjs

I've successfully added Google Analytics to my Next.js 13.5 project, and it's working well, showing me the pages users are visiting. However, I've encountered an issue where the <meta name="

<https://stackoverflow.com/questions/77663582/google-analytics-noindex-issue-nextjs>



<https://github.com/medusajs/nextjs-starter-medusa/issues/223>

<https://github.com/vercel/next.js/pull/59531>

Page Indexing report - Search Console Help

See which pages Google can find and index on your site, and learn about any indexing problems encountered. Open Page Indexing report

https://support.google.com/webmasters/answer/7440203#blocked_by_noindex_tag