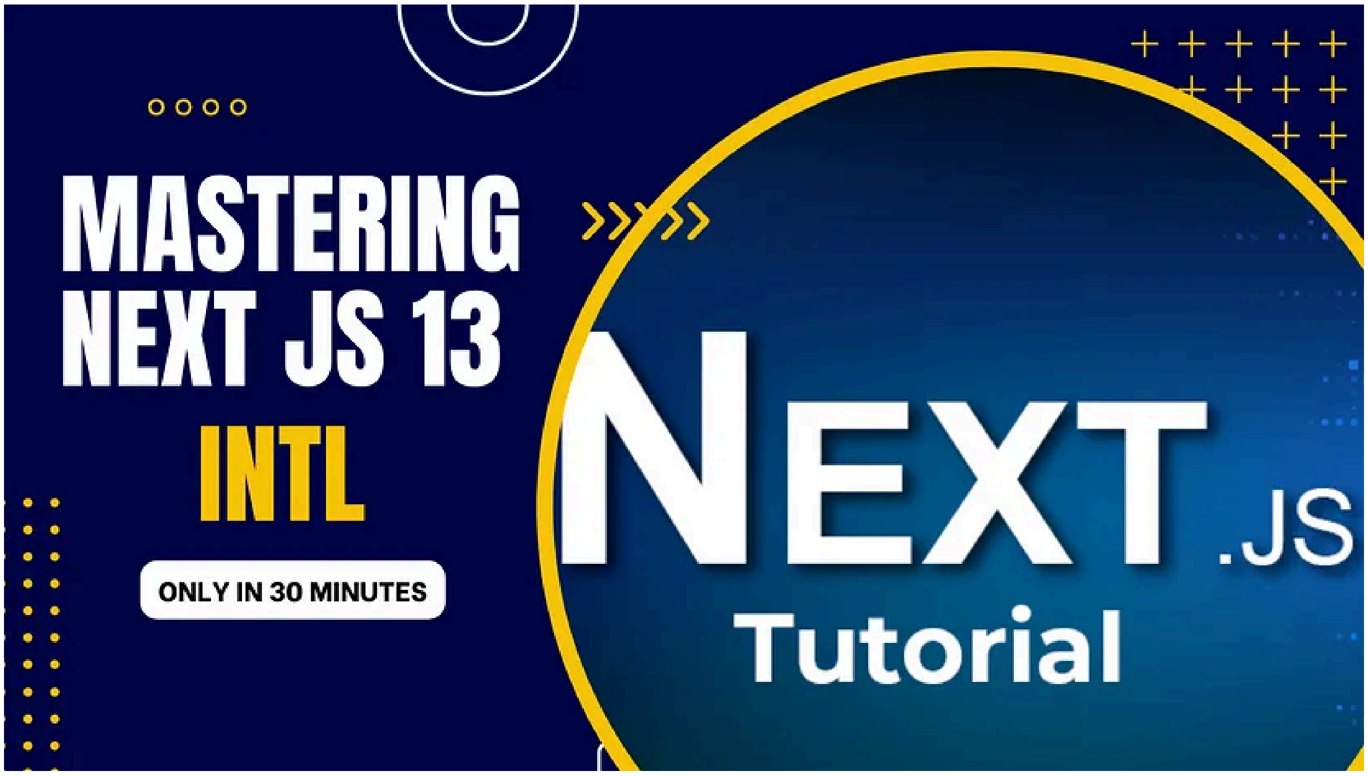
A Comprehensive Guide to Internationalization in Next.js

Darwin Keeley · Follow
5 min read · Oct 26, 2023

111 1



Next.js, a popular React framework, provides robust support for building applications with multiple language support. This guide will walk you through the steps to implement internationalization (i18n) in your Next.js project. We'll cover setting up routing, handling localization, and managing translated content.

Understanding Internationalization Terminology

Locale

A locale is an identifier for a set of language and formatting preferences. It includes the user's preferred language and may also indicate their geographic region. For example:

- `'en-US'` : English as spoken in the United States
- `'nl-NL'` : Dutch as spoken in the Netherlands
- `'en'` : English language without a specific region

Setting Up Next.js for Internationalization

Create a New Next.js Application

If you haven't already, create a new Next.js project using the following command:

```
$ npx create-next-app@latest my-i18n-app
```

Setting Up Routing for Internationalization

To provide a seamless experience for users, it's crucial to adapt your application based on their preferred language. This can be achieved by adjusting the routing mechanism.

Utilizing User's Language Preferences

By using libraries like `@formatjs/intl-localematcher` and `negotiator`, you can determine the user's preferred locale based on their headers and your supported locales. This helps you ensure that users are directed to the correct language version of your site.

```
$ npm install @formatjs/intl-localematcher negotiator
```

Implementing Locale-Based Routing

Next.js allows you to internationalize routing either through sub-paths `(/fr/products)` or domains `(my-site.fr/products)`. This information enables you to redirect users based on their preferred locale within the middleware.

and inside `src/` directory create `middleware.ts`

```
// middleware.ts
import { match } from '@formatjs/intl-localematcher'
import Negotiator from 'negotiator'
import { NextRequest, NextResponse } from 'next/server'

let defaultLocale = 'en'
let locales = ['cn', 'de', 'en']

// Get the preferred locale, similar to above or using a library
function getLocale(request: NextRequest) {
  const acceptedLanguage = request.headers.get('accept-language') ?? undefined
```

```
    let headers = { 'accept-language': acceptedLanguage }
    let languages = new Negotiator({ headers }).languages()

    return match(languages, locales, defaultLocale) // -> 'en-US'
  }

  export function middleware(request: NextRequest) {
    // Check if there is any supported locale in the pathname
    const pathname = request.nextUrl.pathname
    const pathnameIsMissingLocale = locales.every(
      (locale) => !pathname.startsWith(`/${locale}/`) && pathname !== `/${locale}`
    )

    // Redirect if there is no locale
    if (pathnameIsMissingLocale) {
      const locale = getLocale(request)

      // e.g. incoming request is /products
      // The new URL is now /en-US/products
      return NextResponse.redirect(new URL(`/${locale}/${pathname}`, request.url))
    }
  }

  export const config = {
    matcher: [
      // Skip all internal paths (_next, assets, api)
      '/((?!api|assets|.*\\..*|_next).*)',
      // Optional: only run on root (/) URL
      // '/'
    ],
  }
```

Organizing Files for Locale-Based Handling

To enable Next.js to dynamically manage different locales in the route, nest all special files within `app/[lang]`.

For example:

- `app/[lang]/page.tsx`

```
export default async function Page({ params: { lang } }) {
  return ...
}
```

Implementing Localization

Localization involves adapting the displayed content based on the user's preferred locale. This can be achieved by maintaining separate dictionaries for each supported language.

Creating Dictionaries

For example, let's consider English, Dutch, and Chinese translations for Next js main page:

`app/[lang]/dictionaries`

- `dictionaries/en.json`

```
{
  "get_started": "Get started by editing",
  "doc": "Find in-depth information about Next.js features and API.",
  "learn": "Learn about Next.js in an interactive course with&nbsp;quizzes!",
  "template": "Explore the Next.js 13 playground.",
  "deploy": "Instantly deploy your Next.js site to a shareable URL with Vercel."
}
```

- `dictionaries/de.json`

```
{
  "get_started": "Beginnen Sie, indem Sie bearbeiten",
  "doc": "Finden Sie ausführliche Informationen zu den Funktionen und der API vo
  "learn": "Erfahren Sie mehr über Next.js in einem interaktiven Kurs mit Quizfr
  "template": "Erkunden Sie den Next.js 13-Spielplatz.",
  "deploy": "Bereiten Sie Ihre Next.js-Website sofort für die Bereitstellung auf
}
```

- `dictionaries/cn.json`

```
{
  "get_started": "通过编辑开始",
  "doc": "查找关于 Next.js 功能和 API 的深入信息。",
  "learn": "通过互动课程学习 Next.js，包括测验！",
  "template": "探索 Next.js 13 的示范环境。",
  "deploy": "使用 Vercel 立即部署您的 Next.js 网站到可共享的 URL。"
}
```

Loading translations

Create a function `getDictionary` to load the translations for the requested locale.

```
import 'server-only'

export type Locale = keyof typeof dictionaries

const dictionaries = {
  en: () => import('./dictionaries/en.json').then((module) => module.default),
  de: () => import('./dictionaries/de.json').then((module) => module.default),
  cn: () => import('./dictionaries/cn.json').then((module) => module.default),
}
```

```
export const getDictionary = async (locale: Locale) => dictionaries[locale]()
```

Using Translations in Components

You can now fetch the dictionary inside a layout or page to display translated content.

```
import SwitchLang from '@/components/SwitchLang/SwitchLang'
import Image from 'next/image'
import { Locale, getDictionary } from './dictionaries'
import styles from './page.module.css'

type Props = {
  params: {
    lang: Locale
  }
}
^^^^^^^^^^^^^^^

export default async function Home({ params: { lang } }: Props) {
                                  ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  const intl = await getDictionary(lang)
  ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

  return (
    <main className={styles.main}>
      <div className={styles.description}>
        <p>
          {intl.get_started}&nbsp;
          ^^^^^^^^^^^^^^^^^^
          <code className={styles.code}>src/app/page.tsx</code>
        </p>
          <SwitchLang />
        <div>
          <a
            href="https://vercel.com?utm_source=create-next-app&utm_medium=appdi
            target="_blank"
            rel="noopener noreferrer"
          >
            By{' '}
            <Image
              src="/vercel.svg"
              alt="Vercel Logo"
              className={styles.vercelLogo}
              width={100}
              height={24}
              priority
            />
          </a>
        </div>
      </div>

      <div className={styles.center}>
        <Image
          className={styles.logo}
          src="/next.svg"
          alt="Next.js Logo"
          width={180}
          height={37}
          priority
        />
      </div>

      <div className={styles.grid}>
        <a
          href="https://nextjs.org/docs?utm_source=create-next-app&utm_medium=ap
          className={styles.card}
          target="_blank"
          rel="noopener noreferrer"
        >
          <h2>
            Docs <span>-&gt;</span>
          </h2>
          <p>{intl.doc}</p>
             ^^^^^^^^^^
        </a>

        <a
          href="https://nextjs.org/learn?utm_source=create-next-app&utm_medium=a
          className={styles.card}
          target="_blank"
          rel="noopener noreferrer"
        >
          <h2>
            Learn <span>-&gt;</span>
          </h2>
          <p>{intl.learn}</p>
             ^^^^^^^^^^^^
        </a>
        <a
          href="https://vercel.com/templates?framework=next.js&utm_source=create
          className={styles.card}
          target="_blank"
          rel="noopener noreferrer"
        >
          <h2>
            Templates <span>-&gt;</span>
          </h2>
          <p>{intl.template}</p>
             ^^^^^^^^^^^^^^^
        </a>

        <a
          href="https://vercel.com/new?utm_source=create-next-app&utm_medium=app
          className={styles.card}
          target="_blank"
          rel="noopener noreferrer"
        >
          <h2>
            Deploy <span>-&gt;</span>
          </h2>
          <p>
            {intl.deploy}
            ^^^^^^^^^^^^^
          </p>
        </a>
      </div>
    </main>
  )
}
```

Static Generation for Multiple locales

To generate static routes for a set of locales, use `generateStaticParams` in any page or layout. This can be set globally, such as in the root layout:

```
// app/[lang]/layout.ts
export async function generateStaticParams() {
  return [{ lang: 'en' }, { lang: 'de' }, { lang: 'cn' }]
}
```

```
export default function Root({ children, params }) {
  return (
    <html lang={params.lang}>
      <body>{children}</body>
    </html>
  )
}
```

Conclusion

Implementing internationalization in Next.js provides a seamless experience for users from different linguistic backgrounds. By adapting routing.

Full Source Code

For the complete source code, including all files and dependencies, you can visit the GitHub Repository. Feel free to explore, fork, or clone the repository to delve deeper into the project.

Written by Darwin Keeley

3 Followers

Follow

Passionate Full Stack Developer | Building seamless web experiences | Expertise in front-end and back-end technologies.

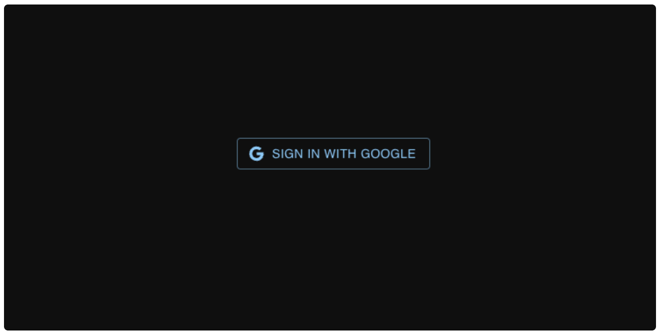
Recommended from Medium

Tushar Patel in Stackademic

How to Generate Dynamic Sitemap in Next.js?

Implement a sitemap using the Next-sitemap plugin, follow these step:-

Mar 15 154



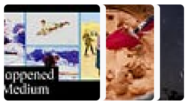
Derick Jang

Next.js App Router Internationalization

Next.js App Router has a built-in internationalization support since version...

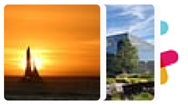
May 4 3

Lists



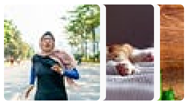
Staff Picks

711 stories · 1203 saves



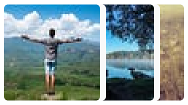
Stories to Help You Level-Up at Work

19 stories · 733 saves



Self-Improvement 101

20 stories · 2504 saves



Productivity 101

20 stories · 2178 saves

Alex Khomenko in Level Up Coding

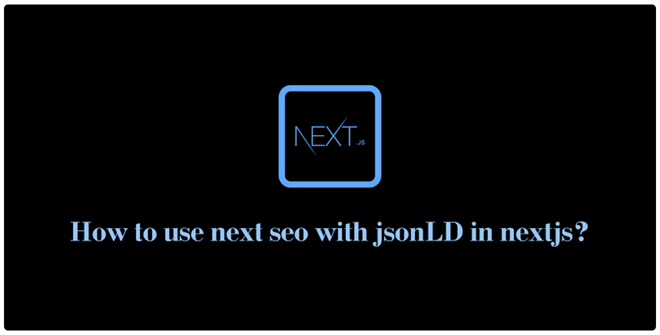
Sviat Kuzhelev

Open in app ↗

Medium Search Write

Apr 18, 2023 242 3

Jun 17 6.8K 2



Tushar Patel in Stackademic

How to Use Next SEO with jsonLD in Next.js?

In Next.js, SEO (Search Engine Optimization) plays a crucial role in ensuring your website...

Mar 7 201 1

Xiuer Old in JavaScript in Plain English

12 Github Repositories to Learn NextJS Quickly 🚀

NextJS is a popular JavaScript framework that allows us to build dynamic sites and...

Feb 22 234 2

See more recommendations